

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

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
Наталія АУШЕВА
“ ” _____ 2025 р.

Магістерська дисертація

на здобуття ступеня другого (магістерського) рівня вищої освіти
за освітньою програмою «Цифрові технології в енергетиці»
зі спеціальності 122 «Комп’ютерні науки»

на тему «Методи реалізації машинного перекладу для вивчення іншомовної лексики»

Виконав: студент 2 курсу, групи ТР-43мп

Ольховик Тарас Олександрович

(прізвище, ім’я, по батькові)

(підпис)

Науковий керівник к.ф.-м.н., доцент Андрій ДОНЕЦЬ

(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ)

(підпис)

Рецензент доцент каф. ТАЕ, к.т.н, доцент Олександр СІРИЙ

(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ)

(підпис)

Н.контроль асистент Володимир РУДИК

(посада, ім’я, ПРІЗВИЩЕ)

(підпис)

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ — 2025

**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 другий (магістерський)

спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Цифрові технології в енергетиці»

ЗАТВЕРДЖУЮ

Завідувач кафедри ЦТЕ

Наталія АУШЕВА

«__» _____ 2025 р.

**ЗАВДАННЯ
на магістерську дисертацію студенту**

Ольховику Тарасу Олександровичу

(прізвище, ім'я, по батькові)

1. Тема дисертації: «Методи реалізації машинного перекладу для вивчення іншомовної лексики»

керівник роботи Донець Андрій Георгійович, к.ф.-м.н., доцент

затверджені наказом по університету від «03» листопада 2025 р. № 4779-с

2. Термін подання студентом дисертації: 06.12.2025

3. Об'єкт дослідження: процес автоматизованого вивчення іншомовної лексики на основі методів машинного перекладу та алгоритмів розподілених повторень.

4. Вихідні дані: мова програмування Python 3.10, фреймворк Django 5.1.5, СУБД SQLite3, DeepL API для машинного перекладу, бібліотека requests для HTTP-запитів.

5. Перелік завдань: 1) проаналізувати існуючі програмні рішення для вивчення іншомовної лексики, виявити їхні переваги і недоліки;

2) дослідити провідні сервіси машинного перекладу та обґрунтувати вибір оптимального рішення для інтеграції у вебзастосунок;

3) провести аналіз методів машинного перекладу та алгоритмів розподілених повторень;

4) розробити структуру бази даних та архітектуру системи;

5) розробити програмне забезпечення з використанням обраних технологій та інструментів, виконати тестування розробленої системи, виправити знайдені недоліки.

6. Орієнтований перелік ілюстративного матеріалу: таблиці порівняння програмних рішень для вивчення лексики та сервісів машинного перекладу, схеми архітектур методів машинного перекладу, архітектура вебзастосунку, діаграма прецедентів системи, логічна модель даних та діаграма сутностей, діаграма класів, діаграма послідовності процесу авторизації, блок-схема алгоритму розподілених повторень, інтерфейси користувача, таблиці стартап-проекту.

7. Орієнтовний перелік публікацій: 1. Підходи до персоналізованої методики вивчення іншомовної лексики у вебзастосунках. Збірник наукових тез з матеріалами III Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення» MoSE-2025, КПІ, 2025.

8. Дата видачі завдання «16» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів підготовки магістерської дисертації	Термін виконання	Примітка
1	Аналіз методів реалізації машинного перекладу та аналогічних застосунків	18.09.2025	виконано
2	Вибір засобів реалізації системи та сервісу машинного перекладу	21.09.2025	виконано
3	Розроблення структури бази даних та архітектури системи	26.09.2025	виконано
4	Розроблення програмного забезпечення	13.10.2025	виконано
5	Тестування розробленої системи, виправлення недоліків	19.10.2025	виконано
6	Захист створеного програмного забезпечення	20.10.2025	виконано
7	Оформлення магістерської дисертації	25.11.2025	виконано
8	Передзахист	27.11.25	виконано
9	Подання матеріалів магістерської дисертації на кафедру	04.12.25	виконано
10	Захист	11.12.25	виконано

Студент

(підпис)

Тарас ОЛЬХОВИК

(ім'я, ПРИЗВИЩЕ)

Керівник роботи

(підпис)

Андрій ДОНЕЦЬ

(ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

Актуальність. Сучасні методи вивчення іноземних мов потребують ефективних інструментів для автоматизації процесу засвоєння лексики. Традиційні застосунки для вивчення мов мають суттєві обмеження: відсутність якісного автоматичного перекладу, складність налаштування алгоритмів повторення, необхідність ручного введення перекладів. Інтеграція сучасних технологій нейронного машинного перекладу з науково обґрунтованими методами розподілених повторень дозволяє створити систему, що значно підвищує ефективність процесу запам'ятовування іншомовної лексики та адаптується до індивідуальних потреб користувача.

Метою роботи є розроблення вебзастосунку для вивчення іншомовної лексики, який інтегрує сервіс машинного перекладу з алгоритмами розподілених повторень та забезпечує створення персоналізованих словників.

Для досягнення цієї мети були вирішені наступні **завдання**:

- проаналізувати аналогічні програмні рішення для вивчення іншомовної лексики, виявити їхні переваги і недоліки;
- дослідити провідні сервіси машинного перекладу та обґрунтувати вибір оптимального рішення для інтеграції у вебзастосунок;
- провести аналіз методів машинного перекладу та алгоритмів розподілених повторень;
- розробити структуру бази даних та архітектуру системи;
- розробити програмне забезпечення з використанням обраних технологій та інструментів, виконати тестування розробленої системи, виправити знайдені недоліки.

Об'єктом дослідження є процес автоматизованого вивчення іншомовної лексики на основі методів машинного перекладу та алгоритмів розподілених повторень.

Предметом дослідження є інтеграція машинного перекладу та методу розподілених повторень у вебсистему вивчення лексики.

Методи дослідження: машинний переклад, модифікований алгоритм розподілених повторень SuperMemo SM-2, нейронний машинний переклад на основі архітектури трансформер, DeepL Translator.

Вихідні дані: мова програмування Python 3.13.0, фреймворк Django 5.1.5, СУБД SQLite3, DeepL API для машинного перекладу, бібліотека requests для HTTP-запитів.

Практичне значення отриманих результатів: розроблений вебзастосунок автоматизує процес створення навчальних карток завдяки інтеграції високоякісного машинного перекладу. Система реалізує чотири спеціалізовані режими навчання, що дозволяє адаптувати процес вивчення під конкретні цілі користувача. Проєкт готовий до впровадження як стартап на ринку освітніх технологій України з низькими технологічними бар'єрами входу.

Апробація результатів дисертації. Основні положення даної роботи доповідались та обговорювались на 3-й міжнародно-науковій конференції «Сучасні аспекти інженерії програмного забезпечення» (НТУУ «КПІ ім.Ігоря Сікорського, 12-13 листопада 2025 р.).

Структура та обсяг магістерської дисертації: вступ, 5 розділів, висновки, список використаних джерел і додатки. Робота містить 38 рисунків, 34 таблиці, 72 посилання у списку джерел та 2 додатки. Основний вміст подано на 137 аркушах (кількість сторінок до загальних висновків включно), а загальний обсяг складає 165 сторінок.

Ключові слова: машинний переклад, вебзастосунок, DeepL API, вивчення іншомовної лексики, розподілені повторення, алгоритм SM-2, персоналізовані словники, адаптивне навчання.

ABSTRACT

Relevance of the topic. Modern methods of foreign language learning require effective tools for automating vocabulary acquisition. Traditional language learning applications have significant limitations: the lack of high-quality automatic translation, the complexity of configuring spaced repetition algorithms, and the need to enter translations manually. Integrating modern neural machine translation technologies with scientifically grounded spaced repetition methods makes it possible to create a system that significantly increases the efficiency of memorising foreign language vocabulary and adapts to the individual needs of the user.

The aim of the thesis is to develop a web application for learning foreign language vocabulary that integrates a machine translation service with spaced repetition algorithms and supports the creation of personalised dictionaries.

To achieve this aim, the following **tasks** were set and solved:

- to analyse similar software solutions for learning foreign-language vocabulary and identify their advantages and disadvantages;
- to investigate the leading machine translation services and justify the choice of the optimal solution for integration into the web application;
- to analyse machine translation methods and spaced-repetition algorithms;
- to develop the database structure and the system architecture;
- to develop the software using the selected technologies and tools, test the developed system, and correct any identified shortcomings.

The object of the research is the process of automated foreign language vocabulary learning based on machine translation methods and spaced repetition algorithms.

The subject of the research is the integration of machine translation and the spaced repetition method into a web-based vocabulary learning system.

Research methods: machine translation, a modified SuperMemo SM-2 spaced repetition algorithm, neural machine translation based on the transformer architecture, DeepL Translator.

Technologies used: Python 3.13.0 programming language, Django 5.1.5 framework, SQLite3 DBMS, DeepL API for machine translation, the requests library for HTTP requests.

Practical significance of the obtained results. The developed web application automates the creation of learning flashcards through the integration of high-quality machine translation. The system implements four specialised learning modes, allowing the learning process to be adapted to the specific goals of the user. The project is ready for implementation as a startup in the Ukrainian educational technology market with low technological barriers to entry.

Approbation of the results of the dissertation. The main provisions of this work were reported and discussed at the 3rd international scientific conference «Modern Aspects of Software Engineering».

Structure and volume of the thesis are as follows: an introduction, five chapters, conclusions, a list of references, and appendices. The thesis contains 38 pictures, 34 tables, 72 references in the reference list, and 2 appendices. The main text occupies 137 pages (from the introduction up to and including the general conclusions), and the total volume is 165 pages.

Keywords: machine translation, web application, DeepL API, foreign language vocabulary learning, spaced repetition, SM-2 algorithm, personalised dictionaries, adaptive learning.

ЗМІСТ

ВСТУП.....	10
1 ПОСТАНОВКА ЗАДАЧІ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ З МАШИННИМ ПЕРЕКЛАДОМ ТА АНАЛІЗ НАЯВНИХ РІШЕНЬ	12
1.1 Постановка задачі розроблення вебзастосунку для вивчення іноземної лексики.....	13
1.2 Аналіз існуючих систем для вивчення іноземної лексики.....	14
1.2.1 Система Duolingo	14
1.2.2 Система Memrise	17
1.2.3 Система Anki	19
1.3 Огляд сучасних рішень машинного перекладу.....	22
1.3.1 Сервіс Google Translate	22
1.3.2 Сервіс DeepL Translator	23
1.3.3 Сервіс Microsoft Translator	25
1.3.4 Сервіс Amazon Translate	26
Висновки до розділу 1	28
2 МЕТОДИ МАШИННОГО ПЕРЕКЛАДУ ТА ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ, ЗАСОБИ ЇХ РЕАЛІЗАЦІЇ.....	29
2.1 Огляд методів машинного перекладу	29
2.1.1 Класичні підходи до машинного перекладу.....	30
2.1.2 Нейронний машинний переклад.....	32
2.1.3 Сучасні нейронмережеві архітектури	35
2.2 DeepL як основний сервіс машинного перекладу.....	40
2.3 Методи вивчення іноземної лексики.....	43
2.4 Алгоритм розподілених повторень	47
2.5 Засоби розроблення вебзастосунку	50
2.5.1 Python як основна мова програмування	51
2.5.2 Фреймворк Django для розроблення серверної частини	52

	8
2.5.3 База даних SQLite.....	55
2.5.4 Технології клієнтської частини.....	56
Висновки до розділу 2	58
3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ З МАШИННИМ ПЕРЕКЛАДОМ	61
3.1 Архітектура вебзастосунку	61
3.2 Взаємодія користувача з системою	64
3.3 Структура бази даних системи	67
3.3.1 Таблиця User	70
3.3.2 Таблиця Dictionary	70
3.3.3 Таблиця Word	71
3.3.4 Таблиця WordAttempt	73
3.3.5 Таблиця LearningSession	74
3.4 Структура класів вебзастосунку	75
3.5 Модуль реєстрації та авторизації користувачів	76
3.6 Модуль інтеграції з DeepL API.....	79
3.7 Модуль реалізації алгоритму розподілених повторень.....	81
3.8 Модуль керування словниками та навчальними картками.....	86
Висновки до розділу 3	89
4 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ	91
4.1 Вимоги до програмного забезпечення та інсталяція	91
4.2 Використання системи	93
4.2.1 Реєстрація.....	94
4.2.2 Авторизація.....	97
4.2.3 Створення словників та додавання слів	100
4.2.4 Процес вивчення лексики за методом розподілених повторень	103
4.2.5 Моніторинг прогресу вивчення	110
Висновки до розділу 4	113
5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ	115

5.1 Опис ідеї стартап-проєкту	115
5.2 Технологічний аудит ідеї стартап-проєкту	118
5.3 Аналіз ринкових можливостей запуску стартап-проєкту	120
5.4 Розроблення ринкової стратегії стартап-проєкту	126
5.5 Розроблення маркетингової програми стартап-проєкту	129
Висновки до розділу 5	133
ВИСНОВКИ	135
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	138
ДОДАТОК А	145
ДОДАТОК Б	162

ВСТУП

У сучасних умовах глобалізації та цифровізації володіння іноземними мовами стає необхідною умовою професійної діяльності та міжкультурної комунікації. Для опанування будь-якої іноземної мови критично важливо сформувати достатній словниковий запас, оскільки лексика становить основу всіх мовних компетенцій: аудіювання, говоріння, читання та письма. Ефективність засвоєння іншомовної лексики залежить від раціонального розкладу повторень матеріалу.

Актуальність теми зумовлена необхідністю вдосконалення методів вивчення іншомовної лексики на основі науково обґрунтованих підходів до організації навчального процесу. Метод розподілених повторень, що базується на принципах забування, описаних Германом Еббінгаузом, значно підвищує ефективність довготривалого запам'ятовування. Розвиток технологій машинного перекладу, зокрема нейронного машинного перекладу на основі архітектури трансформер, відкриває нові можливості для створення інтелектуальних систем підтримки вивчення мов.

Аналіз популярних програмних рішень для вивчення лексики виявив ряд обмежень. Система Duolingo забезпечує гейміфікований підхід та адаптивні повторення, однак не підтримує створення персоналізованих словників і не надає повноцінного сервісу машинного перекладу для довільно введених користувачем лексичних одиниць. Платформа Memrise орієнтована на візуальні асоціації та мнемотехніки, проте також не інтегрує якісний машинний переклад. Програмне забезпечення Anki пропонує гнучкий алгоритм розподілених повторень з високою прозорістю параметрів, але має складний інтерфейс і не містить вбудованого сервісу машинного перекладу. Таким чином, існуючі системи не забезпечують комплексного підходу до інтеграції високоякісного машинного перекладу з адаптивними алгоритмами розподілених повторень.

Метою магістерської роботи є розроблення вебзастосунку для вивчення іншомовної лексики, який інтегрує сервіс машинного перекладу з алгоритмами

розподілених повторень та забезпечує створення персоналізованих словників. Завданням дослідження є аналіз існуючих програмних рішень для вивчення лексики та сучасних сервісів машинного перекладу, розроблення структури бази даних та архітектури системи, програмна реалізація вебзастосунку з використанням обраних технологій, а також тестування розробленої системи. Для досягнення мети використовуються методи аналізу програмних інтерфейсів машинного перекладу, алгоритми адаптивного планування повторень на основі модифікованого методу розподілених повторень, технології веб-програмування та проєктування баз даних.

Практичне значення отриманих результатів полягає у можливості використання розробленої системи студентами та викладачами іноземних мов для підвищення ефективності засвоєння нової лексики завдяки поєднанню якісного автоматичного перекладу з науково обґрунтованими методами організації повторень.

Магістерська дисертація складається зі вступу, п'яти розділів, висновків та списку використаних джерел. Перший розділ присвячений постановці задачі, аналізу існуючих систем вивчення лексики та сервісів машинного перекладу. У другому розділі розглянуто теоретичні основи машинного перекладу та алгоритмів розподілених повторень. Третій розділ зосереджується на розробленні архітектури системи та реалізації інтеграції з API машинного перекладу. Четвертий розділ описує користувацьку взаємодію з розробленим вебзастосунком. П'ятий розділ представляє стартап-аналіз проєкту.

1 ПОСТАНОВКА ЗАДАЧІ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ З МАШИННИМ ПЕРЕКЛАДОМ ТА АНАЛІЗ НАЯВНИХ РІШЕНЬ

У сучасному світі, внаслідок цифровізації та глобалізації, володіння іноземними мовами стає дедалі необхіднішим для професійної діяльності чи комунікації. Для засвоєння будь-якої іноземної мови необхідно освоїти, окрім граматичних правил та конструкцій, її лексику.

Практика та розклад повторень є одними з ключових факторів, які впливають на результат запам'ятовування іноземних слів. Проте традиційні методи запам'ятовування недостатньо враховують механізм людської пам'яті, тому часто виявляються малоефективними.

З розвитком вебтехнологій та технологій машинного перекладу з'явилися нові можливості для вивчення іноземних слів, що дозволяє ефективніше формувати словниковий запас. Сучасні онлайн-рішення для вивчення мов часто не забезпечують достатньої узгодженості між функціоналом перекладу та повторенням слів для вивчення. Користувачі змушені або вручну вводити переклади з використанням зовнішніх сервісів, або обмежуватися готовими курсами з фіксованим набором лексики.

Метою цієї роботи є реалізація вебзастосунку для вивчення іноземної лексики на основі підходу розподілених повторень з використанням інтегрованого машинного перекладу.

У цьому розділі здійснюється постановка задачі розроблення вебзастосунку, проводиться огляд існуючих програмних рішень для вивчення іноземної лексики та машинного перекладу.

1.1 Постановка задачі розроблення вебзастосунку для вивчення іншомовної лексики

Вивчення іноземної мови традиційно включає формування чотирьох основних навичок, а саме: аудіювання, говоріння, читання та письма. Для кожної з цих компетенцій критично важливим є словниковий запас.

За результатами багатьох досліджень, таких як метааналіз 2022 року Су Кьон Кім та Стюарта Вебба, стає зрозуміло, що ефективність довготривалого запам'ятовування іншомовних слів значно підвищується при використанні методу розподілених повторень [1]. Цей підхід базується на принципі забування, вперше описаному Г. Еббінгаузом, згідно з яким повторення матеріалу через стратегічно визначені інтервали часу запобігає втраті інформації з довготривалої пам'яті [2; 3].

Сучасні онлайн-рішення для вивчення мов часто не забезпечують достатньої узгодженості між функціоналом перекладу та повторенням слів для вивчення. Користувачі змушені або вручну вводити переклади з використанням зовнішніх сервісів, або обмежуватися готовими курсами з фіксованим набором лексики. Ця проблема є особливо актуальною для студентів та самоуків, які прагнуть створювати персоналізовані словники для специфічних галузей знань чи професійних потреб.

Метою даної роботи є розроблення вебзастосунку для вивчення іншомовної лексики з використанням сучасного методу машинного перекладу та адаптивного алгоритму розподілених повторень.

Для реалізації поставленої мети необхідно розв'язати наступні завдання:

- проаналізувати існуючі програмні рішення для вивчення іншомовної лексики, виявити їхні переваги і недоліки;
- дослідити провідні сервіси машинного перекладу та обґрунтувати вибір оптимального рішення для інтеграції у вебзастосунок;
- розробити структуру бази даних та архітектуру системи;

- розробити програмне забезпечення з використанням обраних технологій та інструментів;

- виконати тестування розробленої системи, виправити знайдені недоліки.

Розроблений вебзастосунок повинен мати наступний функціонал:

- реєстрація та авторизація користувачів;
- створення словників, можливість заповнення словами поштучно та масово;
- автоматичне заповнення перекладу введених лексем через інтегрований машинний переклад;
- вибір режиму вивчення іншомовних слів з адаптивними стратегіями планування повторень для кращого запам'ятовування;
- відстеження статистики вивчення з візуалізацією прогресу;
- адаптація інтервалів повторень в залежності від результатів попередніх спроб.

1.2 Аналіз існуючих систем для вивчення іншомовної лексики

На даний момент існує чимало варіантів програмного забезпечення, розробленого саме для опанування іноземних мов та збагачення словникового запасу. Детальний аналіз сильних та слабких сторін провідних (за технічною складовою) та популярних платформ дозволить краще розуміти предметну область.

1.2.1 Система Duolingo

Перш за все варто розглянути Duolingo, оскільки це популярне та провідне рішення у контексті вивчення іноземної мови [4]. Це кросплатформенний сервіс, щоденна аудиторія якого перевищує 50 мільйонів користувачів, а щомісячна становить більше 135 мільйонів.

Платформа базується на принципах гейміфікації навчального процесу. Учні послідовно опрацьовують структуровані модулі, виконуючи завдання різних типів, таких як переклад текстів, сприйняття слів на слух чи конструювання слів. Для покращення засвоєння матеріалу Duolingo застосовує модифіковану версію алгоритму розподілених повторень. Інтерфейс вебсайту зображено на рисунку 1.1.

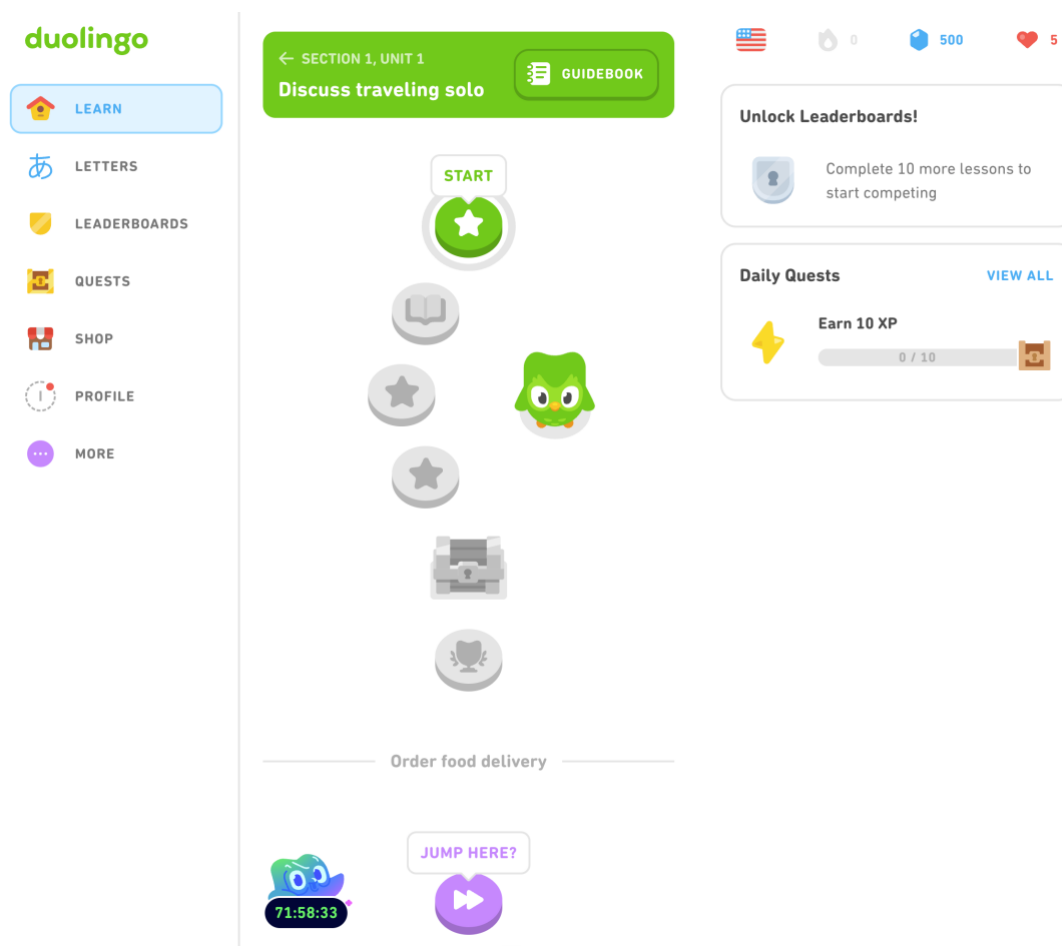


Рисунок 1.1 — Інтерфейс вебсайту Duolingo

Таблиця 1.1 демонструє особливості Duolingo у контексті вивчення лексики та інтегрованого машинного перекладу. Для кожного критерію наведено оцінку та детальний опис функціональних можливостей платформи, що дозволяє виявити як переваги, так і обмеження цієї системи. Особлива увага приділена аналізу якості перекладу, прозорості алгоритмів навчання та можливостям персоналізації процесу вивчення лексики. Результати аналізу представлено нижче.

Таблиця 1.1 — Особливості Duolingo

Критерій	Оцінка	Особливості
Інтегрований нейронний машинний переклад для введених слів	Частково	Переклад відбувається під час вправ лише у межах уроків, таких як: підказки, переклад окремих слів і речень. Відсутній повноцінний сервіс машинного перекладу, як у Google Translate чи DeepL. Для повноцінного професійного перекладу Duolingo не розглядається як достатній інструмент [4].
Прозорість алгоритму повторень	Недостатня	Система використовує власний алгоритм розподілених повторень, — Birdbrain. Проте в інтерфейсі не відображаються інтервали, ймовірності забування слів чи інші параметри.
Складність для користувача	Низька	Інтерфейс інтуїтивно легкий, гейміфікований, але обмежений контроль користувача над вивченням матеріалу.
Створення власних словників	Ні	Відсутня можливість створювати свої словники. Можливе лише вивчення окремих списків слів у деяких курсах.
Безкоштовний функціонал розподілених повторень	Так	Система надає безкоштовний базовий доступ до курсів, розподілених повторень і адаптивних вправ.
Підтримка масового додавання слів	Ні	Відсутня можливість створити власний список слів для вивчення.
Адаптивні інтервали повторень	Так	Модель машинного навчання Birdbrain моделює знання користувача і адаптивно підбирає час повторення слів [5].

Отже, Duolingo є гейміфікованим застосунком з адаптивними розподіленими повтореннями, але має суттєві обмеження, такі як неможливість створення власних словників чи недостатня якість функцій перекладу.

1.2.2 Система Memrise

На відміну від Duolingo, Memrise [6] є платформою для вивчення мов з акцентом на запам'ятовування лексики завдяки візуальним асоціаціям та мнемотехніці.

Алгоритм системи використовує метод розподілених повторень для планування повторення слів. Платформа надає доступ як до курсів від платформи, так і до курсів, створених користувачами. Інтерфейс системи представлено на рисунку 1.2.

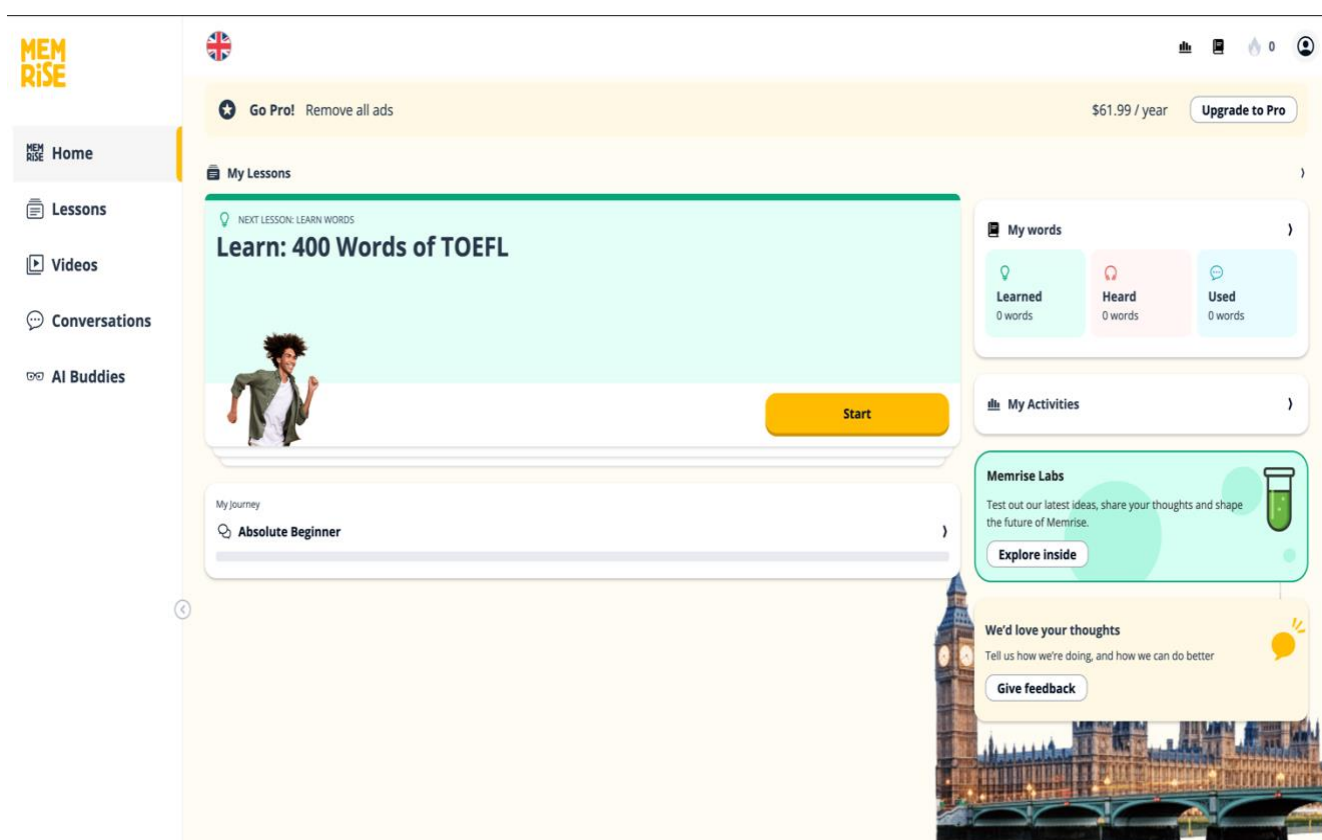


Рисунок 1.2 — Інтерфейс вебсайту Memrise

Основні особливості, переваги та недоліки платформи Memrise зведено у таблиці 1.2, де подано відповідні критерії оцінювання, присвоєні їм оцінки та короткий опис специфічних характеристик сервісу за кожною категорією.

Таблиця 1.2 — Особливості Memrise

Критерій	Оцінка	Особливості
Інтегрований нейронний машинний переклад для введених слів	Ні	Відсутній сервіс для повноцінного машинного перекладу введених слів. Переклад реалізовано лише у межах курсів.
Прозорість алгоритму повторень	Низька	Алгоритм розподілених повторень з адаптивним підбором слів реалізовано, але користувачу не демонструються ні інтервали, ні параметри моделі.
Складність для користувача	Низька	Інтерфейс інтуїтивний і гейміфікований. Користувач проходить короткі сесії з повторенням з аудіо й відео від носіїв мови. Проте розділення курсів офіційних та користувацьких ускладнює навігацію.
Створення власних словників	Частково	Історично це була одна з ключових переваг Memrise. На даний момент цей функціонал винесений на окрему платформу.
Безкоштовний функціонал розподілених повторень	З обмеженнями	Повний набір можливостей розподілених повторень у безкоштовній версії обмежений, але базові уроки та повторення є доступними.
Підтримка масового додавання слів	Так	У редакторі користувацьких курсів можна вставляти довгі списки слів або таблиці зі словника, хоча це стосується лише курсів, створених користувачем.
Адаптивні інтервали повторень	Так	Застосовується алгоритм розподілених повторень, який адаптує частоту показу карток до успішності відповідей. Додатково використовуються візуальні асоціації, гейміфікація та мультимедійні матеріали від носіїв мови.

Отже, Memrise є гейміфікованою платформою для запам'ятовування лексики. Вона поєднує розподілені повторення, мнемоніки та матеріали від носіїв мови, але відсутній повноцінний сервіс машинного перекладу та є обмеження створення власних словників.

1.2.3 Система Anki

На відміну від попередніх рішень, Anki є повністю безкоштовним і відкритим програмним забезпеченням. Це програма з вбудованим алгоритмом розподілених повторень, яка дозволяє створювати та використовувати картки для запам'ятовування. Також вона дозволяє користувачам повністю контролювати процес навчання та налаштовувати параметри алгоритму повторень [7]. Інтерфейс програми Anki зображено на рисунку 1.3.

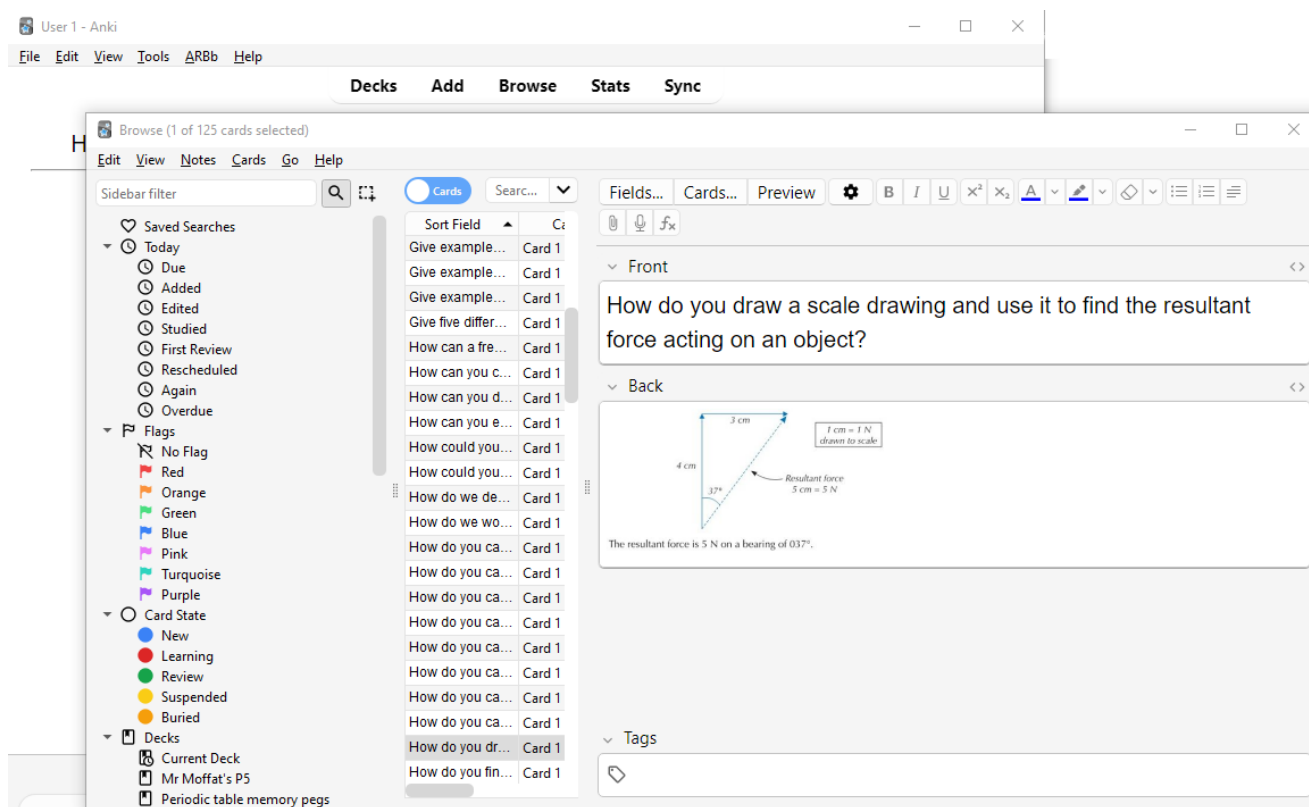


Рисунок 1.3 — Інтерфейс програми Anki

У таблиці 1.3 наведено оцінку Anki за ключовими критеріями.

Таблиця 1.3 — Особливості програми Anki

Критерій	Оцінка	Особливості
Інтегрований нейронний машинний переклад для введених слів	Ні	Користувачі змушені самостійно вводити переклади або використовувати сторонні доповнення.
Прозорість алгоритму повторень	Висока	Алгоритм планування повторень реалізований на модифікованому алгоритмі SM-2 з оптимізацією часу повторення. Робота алгоритму, формули і параметри детально описані в довідці.
Складність для користувача	Висока	Інтерфейс виглядає застарілим порівняно з сучасними застосунками. Надлишкова складність інтрефейсу, забагато опцій. Початок навчання занадто складний для нових користувачів в аспекті надмірних інтелектуальних навантажень.
Створення власних словників	Так	Користувач створює власні набори слів, обирає структуру нотаток та шаблони карток, може їх імпортувати чи експортувати.
Безкоштовний функціонал розподілених повторень	Так	Anki є безкоштовним, з відкритим кодом, синхронізація через AnkiWeb також безоплатна. Платною є лише офіційна iOS-версія.
Підтримка масового додавання слів	Так	Користувач може імпортувати чи експортувати списки слів чи нотатки.
Адаптивні інтервали повторень	Так	Інтервали повторень динамічно змінюються залежно від обраної користувачем складності для кожного слова.

Отже, Anki є якісним, прозорим та адаптивним інструментом вивчення іншомовної лексики за методом розподілених повторень. Застосунок підтримує створення власних колод (списків для вивчення), мультимедіа та масове додавання

слів. Проте платформа має заскладний інтерфейс і має високий поріг входу у зв'язку з надмірними навантаженнями на користувача від алгоритму. Це ускладнює використання. У порівнянні з іншими платформами, Anki є максимально гнучкою, але менш дружньою до нового користувача.

Для систематизації результатів аналізу було створено порівняльну таблицю ключових особливостей розглянутих систем. Результати порівняння наведено у таблиці 1.4.

Таблиця 1.4 — Результат порівняння розглянутих систем

Критерій	Duolingo	Memrise	Anki
Інтегрований нейронний машинний переклад для введених слів	Частково	Ні	Ні
Прозорість алгоритму повторень	Недостатня	Низька	Висока
Складність для користувача	Низька	Низька	Висока
Створення власних словників	Ні	Частково	Так
Безкоштовний функціонал розподілених повторень	Так	З обмеженнями	Так
Підтримка масового додавання слів	Ні	Так	Так
Адаптивні інтервали повторень	Так	Так	Так

На основі виконаного аналізу провідних рішень для вивчення іншомовної лексики, можна зауважити, що кожна з розглянутих систем має певні сильні сторони, але жодна з них не задовольняє всі вимоги. Системи Duolingo і Memrise орієнтовані на готові курси з обмеженою гнучкістю, не містять сучасний нейронний машинний переклад і не інформують користувачів щодо параметрів алгоритмів повторень. Програма Anki, не зважаючи на повний контроль над процесом навчання, є надто складною для пересічного користувача, не містить вбудованого машинного перекладу.

Отже, розроблення вебзастосунку, який поєднував би адаптивний алгоритм розподілених повторень, зручний інтерфейс та містив би якісний нейронний

машинний переклад, є актуальним завданням. Користувачі зможуть ефективно вивчати ті слова, які захочуть, без необхідності ручного введення перекладів чи використання сторонніх сервісів.

1.3 Огляд сучасних рішень машинного перекладу

В контексті оброблення природної мови машинний переклад досліджується лише протягом останніх десятиліть. Сучасні провідні системи машинного перекладу базуються переважно на нейронних мережах, насамперед на архітектурі глибокого навчання, яку називають трансформер [8; 9]. Нейронний машинний переклад надає значно вищу якість перекладу порівняно з попередніми статистичними методами. Це відбувається завдяки здатності моделювати складні контекстуальні залежності між словами та фразами [8].

Архітектура трансформер, запропонована Ашишом Васвані та співавторами у 2017 році, зробила якісний стрибок у сфері машинного перекладу завдяки використанню багатоголового механізму уваги. Це дозволило моделі одночасно враховувати різні аспекти контексту при генерації перекладу [9]. Згадана архітектура стала фундаментом для більшості сучасних систем машинного перекладу та великих мовних моделей.

Проведемо детальніший огляд провідних рішень машинного перекладу, які можуть бути інтегровані у вебзастосунки.

1.3.1 Сервіс Google Translate

Google Translate [10] є найпопулярнішим у світі перекладачем. Це сервіс з використанням машинного перекладу, розроблений компанією Google. Станом на 2025 рік, Google Translate підтримує понад 249 мов та діалектів. Завдяки цьому сервіс виступає відправною точкою для порівняння інших рішень. У даному програмному забезпеченні використовується нейронний машинний переклад на

основі згаданої архітектури трансформер з надвеликими обсягами навчальних даних.

У таблиці 1.5 наведено узагальнену оцінку Google Translate за ключовими критеріями для заданої предметної області.

Таблиця 1.5 — Оцінка Google Translate за основними критеріями

Критерій	Оцінка
Кількість мов	Понад 249 мов
Якість перекладу для європейських мов	Висока
Підтримка контексту	Висока, але переклади буквальні
Ціна API	Середня (\$20 за 1 млн символів)
Безкоштовний тариф	500000 символів/міс
Швидкість перекладу	Найвища
Додаткові функції	Переклад зображень, голосу і вебсторінок

Отже, Google Translate забезпечує вкрай високе охоплення мов і є незамінним вибором, коли необхідний переклад рідкісних мов. Для європейських мов якість перекладів в цілому висока. Проте, у складних чи стилістично чутливих текстах переклади є занадто буквальними, порівняно з DeepL.

Вартість системи є середньою, присутній ліміт на безкоштовне використання. Додаткові можливості, такі як переклад зображень, голосу і вебсторінок, роблять Google Translate універсальним інструментом. Але, для задач точного перекладу європейської лексики, він не є передовим рішенням.

1.3.2 Сервіс DeepL Translator

Іншим популярним рішенням є DeepL Translator [11]. Це сервіс машинного перекладу, запущений у 2017 році компанією DeepL SE, який спеціалізується на високоякісному перекладі європейських мов. Рішення базується на власній нейромережевій архітектурі з використанням трансформер-підходів та навчається

на високоякісних даних з бази Linguee. Перевагами перекладача є точність перекладу та збереження стилю.

У професійному середовищі перекладачів сервіс вважається основним інструментом, оскільки він зменшує обсяг постредагування після перекладу (зокрема за рахунок збереження стилю). Проведені дослідження підтверджують перевагу DeepL над конкурентами [12; 13]. Так, згідно з опитуванням Асоціації мовних компаній 2024 року, 82% мовних сервісних компаній використовують DeepL для перекладів, і лише 46% — Google Translate. Експерти оцінюють переклади DeepL як точніші за Google Translate, особливо для таких мовних пар, як: англійська-німецька, англійська-французька і англійська-іспанська [12]. У таблиці 1.6 проведено оцінку DeepL Translator за основними критеріями для заданої предметної області.

Таблиця 1.6 — Оцінка DeepL Translator за основними критеріями

Критерій	Оцінка
Кількість мов	Понад 100 мов
Якість перекладу для європейських мов	Найвища
Підтримка контексту	Найвища, точно передає стиль тексту
Ціна API	Висока (приблизно \$25 за 1 млн символів)
Безкоштовний тариф	500000 символів/міс
Швидкість перекладу	Висока
Додаткові функції	Глосарії, налаштування формальності, якісний переклад документів

Станом на 2025 рік машинний переклад DeepL Translator забезпечує найкращу з існуючих систем якість перекладу для європейських мов. Це особливо корисно в контексті вивчення лексики, де точність перекладу і врахування стилю є ключовими.

Кількість підтримуваних мов менша, ніж у Google Translate, але доступність для перекладу більше 100 мов є достатньою в контексті поставленої задачі.

Вартість системи є високою, але наявність безкоштовного тарифу та якість перекладу роблять DeepL якісним вибором для інтеграції у розроблюване програмне забезпечення.

1.3.3 Сервіс Microsoft Translator

Ще одним рішенням є Microsoft Translator [14]. Цей сервіс машинного перекладу інтегрований в екосистему продуктів Microsoft, таких як Office чи Teams. Підтримується понад 100 мов. Як і в інших рішеннях, використовується нейронний машинний переклад на основі архітектури трансформер.

Microsoft Translator є частиною екосистеми Azure AI і тісно інтегрується з продуктами Microsoft 365. У бізнес-середовищі сервіс має сильні позиції завдяки помірній ціні та інтеграції в бізнес-процеси.

У таблиці 1.7 наведено оцінку Microsoft Translator за основними критеріями для заданої предметної області.

Таблиця 1.7 — Оцінка Microsoft Translator за основними критеріями

Критерій	Оцінка
Кількість мов	Понад 100 мов
Якість перекладу для європейських мов	Середня
Підтримка контексту	Середня
Ціна API	Низька (від \$10 за 1 млн символів)
Безкоштовний тариф	2 млн символів/міс
Швидкість перекладу	Висока
Додаткові функції	Персональний перекладач, інтеграція з різним ПЗ від Microsoft, розмовний режим

Отже, Microsoft Translator є незамінним вибором, коли є необхідною інтеграція з Microsoft 365 чи Azure. Іншою перевагою є низька вартість, що критично важливо при надвеликих обсягах перекладів.

У порівнянні з DeepL та Google, якість перекладу гірша, частіше вимагає додаткового редагування, особливо у спеціалізованих текстах. Для розроблюваного вебзастосунку, перевагою якого має бути якісне засвоєння лексики, Microsoft Translator не є оптимальним вибором.

1.3.4 Сервіс Amazon Translate

Наприкінці 2017 року компанія Amazon Web Services (AWS) представила Amazon Translate — сервіс машинного перекладу, який використовує нейронний переклад для багатьох мов [15]. Цей перекладач орієнтований насамперед на інтеграцію з хмарною платформою AWS. Він використовується переважно для автоматичного перекладу великих масивів контенту у корпоративних рішеннях.

У таблиці 1.8 наведено оцінку Amazon Translate за основними критеріями для заданої предметної області.

Таблиця 1.8 — Оцінка Amazon Translate за основними критеріями

Критерій	Оцінка
Кількість мов	Понад 75 мов
Якість перекладу для європейських мов	Середня
Підтримка контексту	Середня
Ціна API	Середня (\$15 за 1 млн символів)
Безкоштовний тариф	2 млн символів/міс
Швидкість перекладу	Висока
Додаткові функції	Власна термінологія, робота з паралельними даними, інтеграція з AWS

Підсумовуючи таблицю, Amazon Translate є оптимальним рішенням для великих інфраструктур на базі AWS. Якість перекладу є прийнятною, але, як правило, вона гірша ніж у DeepL чи Google. Проте для розроблюваного вебзастосунку інтеграція з AWS може виявитися надто складною. З огляду на необхідності у якісному опрацюванні лексики, Amazon Translate не є оптимальним рішенням на початковій стадії розроблення програмного забезпечення.

У результуючій таблиці 1.9 наведено ключові особливості порівнюваних систем.

Таблиця 1.9 — Порівняння сервісів машинного перекладу

Критерій	Google Translate	DeepL Translator	Microsoft Translator	Amazon Translate
Кількість мов	Найвища	Висока	Висока	Середня
Якість перекладу для європейських мов	Висока	Найвища	Середня	Середня
Підтримка контексту	Висока	Найвища	Середня	Середня
Ціна API	Середня	Висока	Низька	Середня
Безкоштовний тариф	500000 символів/міс	500000 символів/міс	2 млн символів/міс	2 млн символів/міс
Швидкість перекладу	Найвища	Висока	Висока	Висока
Додаткові функції	Переклад зображень, голосу і вебсторінок	Глосарії, налаштування формальності, якісний переклад документів	Персональний перекладач, інтеграція з різним ПЗ від Microsoft, розмовний режим	Власна термінологія, робота з паралельними даними, інтеграція з AWS

Проведений аналіз показав, що для завдань розроблюваного вебзастосунку оптимальним вибором є інтеграція машинного перекладу DeepL Translator. Найвища якість перекладу для європейських мов, передове розуміння контексту, наявність безкоштовного тарифу та відсутність суттєвих недоліків зумовлюють цей вибір.

Висновки до розділу 1

У даному розділі було аргументовано актуальність розроблення вебзастосунку для вивчення іншомовної лексики з інтегрованим машинним перекладом. Проведений аналіз предметної області продемонстрував, що існуючі системи вивчення лексики не інтегрують сучасні методи нейронного машинного перекладу, мають непрозорі методи розподілених повторень, а також не мають функціоналу створення власних словників з бажаними словами для вивчення.

Було сформульовано мету роботи та визначено основні завдання розроблення вебзастосунку з інтегрованим машинним перекладом. Система повинна забезпечувати високоякісний переклад довільно введених лексичних одиниць з використанням DeepL Translator та покращувати їх запам'ятовування завдяки адаптивному алгоритму розподілених повторень.

Огляд наявних рішень для вивчення лексики показав їх обмеженість у контексті певних характеристик. Саме це аргументувало доцільність розроблення системи, яка б поєднувала переваги проаналізованих рішень без їхніх недоліків.

Аналіз сучасних рішень машинного перекладу показав, що DeepL Translator є оптимальним вибором для інтеграції у вебзастосунок для вивчення іншомовної лексики. До основних переваг можна віднести найвищу якість перекладу та краще розуміння контексту. При цьому сервіс DeepL не має значних недоліків та є передовим рішенням на момент проведення дослідження.

На основі даного аналізу було встановлено вимоги до функціонування системи та обґрунтовано вибір сервісу машинного перекладу. У наступному розділі буде розглянуто проєктування вебзастосунку з врахуванням аналізу, проведеного у першому розділі.

2 МЕТОДИ МАШИННОГО ПЕРЕКЛАДУ ТА ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ, ЗАСОБИ ЇХ РЕАЛІЗАЦІЇ

Розроблення застосунків для вивчення іноземних мов розпочинається з підбору технологій і методів, які визначають архітектуру системи та можливості її практичної реалізації. Правильно обрані інструменти не лише полегшують процес розроблення, а й забезпечують високу якість застосунку та його масштабованість.

У цьому розділі детально розкриваються теоретичні основи машинного перекладу та сучасні методи вивчення іншомовної лексики. Проводиться ґрунтовний аналіз як теоретичних підходів до машинного перекладу та запам'ятовування лексики, так і практичних інструментів для реалізації вебзастосунків.

Послідовно розглядаються ключові аспекти розроблення: огляд еволюції методів машинного перекладу від класичних підходів до сучасних нейромережових архітектур, аналіз провідних сервісів машинного перекладу, дослідження методів вивчення іншомовної лексики та алгоритмів розподілених повторень, а також детальний опис засобів розроблення вебзастосунку, що включає мову програмування Python, фреймворк Django, базу даних SQLite та технології клієнтської частини.

2.1 Огляд методів машинного перекладу

Машинний переклад є однією з найбільш складних та актуальних технологій у галузі обробки природної мови. Протягом останніх десятиліть у цій сфері відбулися значні зміни: від перших систем, що базувалися на лінгвістичних правилах, до сучасних нейромережових моделей, які демонструють рівень якості, близький до людського перекладу.

Розглянемо основні етапи розвитку машинного перекладу: класичні підходи, що включають системи на основі правил та статистичні методи, нейронний машинний переклад та сучасні архітектури на базі механізмів уваги.

2.1.1 Класичні підходи до машинного перекладу

До класичних підходів до машинного перекладу належать методи, що активно використовувалися з середини XX століття до початку 2010-х років. Історично першими були системи, що базувалися на експертно сформульованих лінгвістичних правилах, а пізніше з'явилися статистичні підходи, що використовували великі паралельні корпуси текстів для побудови ймовірнісних моделей перекладу. Кожен із цих методів мав свої переваги та обмеження, що зрештою призвело до пошуку принципово нових рішень.

Перший підхід, відомий як машинний переклад на основі правил (Rule-Based Machine Translation, RBMT), ґрунтується на використанні лінгвістичних знань, закодованих експертами у вигляді формальних правил [16]. Принцип роботи RBMT можна побачити на рисунку 2.1.

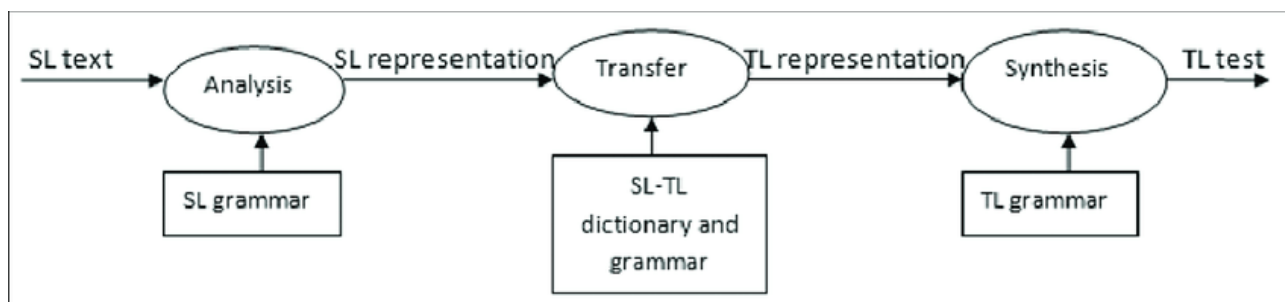


Рисунок 2.1 — Принцип роботи машинного перекладу на основі правил

На рисунку 2.1 продемонстровано, що система RBMT зазвичай складається з трьох основних компонентів: модуля аналізу вихідного тексту, модуля трансферу між мовами та модуля генерації тексту цільовою мовою.

Модуль аналізу виконує морфологічний, синтаксичний та семантичний розбір речення, визначаючи його граматичну структуру та значення.

Модуль трансферу застосовує набір правил для перетворення структури з мови-джерела на структуру цільової мови, враховуючи відмінності в граматиці, порядку слів та ідіоматичних виразах.

Нарешті, модуль генерації створює фінальний текст цільовою мовою на основі отриманої структури [17].

Основною перевагою RBMT є можливість точного контролю за процесом перекладу та забезпечення граматичної правильності результату. Системи на основі правил добре працюють у вузькоспеціалізованих доменах, де термінологія та структура речень є відносно стандартизованими, наприклад, у технічній документації або юридичних текстах.

Проте цей підхід має суттєві обмеження. По-перше, створення повного набору правил для двох мов вимагає величезних зусиль висококваліфікованих лінгвістів і може тривати роками.

По-друге, така система часто є негнучкою і погано справляється з неоднозначністю природної мови, колоквіалізмами та неочікуваними конструкціями [17].

Статистичний машинний переклад (Statistical Machine Translation, SMT) з'явився як альтернативний підхід, що використовує великі паралельні корпуси текстів для автоматичного навчання моделей перекладу [18]. Ідея полягає в тому, що замість кодування експертних правил система аналізує мільйони пар речень у двох мовах і будує ймовірнісні відповідності між сегментами тексту.

Найпоширенішою реалізацією цього підходу є фразовий статистичний переклад (Phrase-Based SMT), який розбиває речення на фрази та знаходить найбільш ймовірні їхні відповідники в цільовій мові з використанням моделі мови для забезпечення природності результату [19].

Принцип роботи фразового статистичного перекладу продемонстровано на рисунку 2.2.

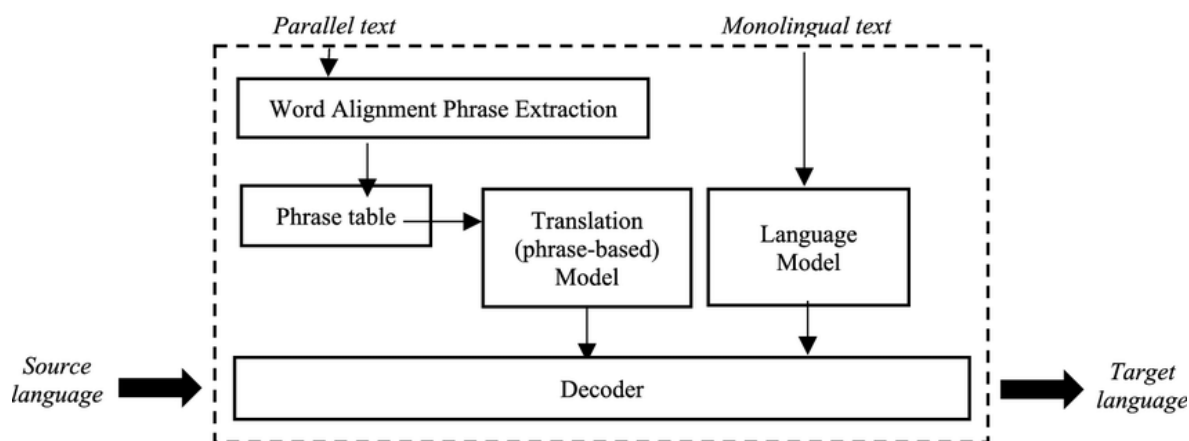


Рисунок 2.2 — Принцип роботи фразового статистичного перекладу

Статистичні методи продемонстрували кращу гнучкість порівняно з RBMT і стали домінуючим підходом у 2000-х роках, особливо після створення великих паралельних корпусів та розвитку обчислювальних потужностей. Однак і вони мають певні недоліки. Основною проблемою SMT є фрагментарність перекладу: система працює з невеликими фразами і не завжди здатна враховувати довготривалі залежності в реченні та ширший контекст. Крім того, якість статистичного перекладу критично залежить від обсягу та якості паралельного корпусу, що робить його менш ефективним для рідкісних мовних пар [20].

Обмеження класичних підходів, такі як складність створення правил для RBMT чи фрагментарність SMT, стимулювали дослідників до пошуку нових методів, здатних краще моделювати складні залежності в мові та враховувати ширший контекст. Це призвело до появи нейронного машинного перекладу, що став наступним етапом еволюції в цій галузі.

2.1.2 Нейронний машинний переклад

Для вирішення виявлених недоліків класичних підходів, зокрема фрагментарності перекладу та неефективної обробки довготривалих залежностей, з'явилися методи, засновані на використанні глибинного навчання. Нейронний машинний переклад (Neural Machine Translation, NMT) є підходом до автоматичного перекладу, що базується на застосуванні штучних нейронних мереж

для побудови наскрізної моделі перетворення речень з мови-джерела на мову-призначення [21]. На відміну від попередніх методів, які розбивали процес на окремі етапи, NMT використовує єдину нейромережеву архітектуру, що навчається перекладати речення цілком.

Ключовою ідеєю нейронного машинного перекладу є використання архітектури енкодер-декодер (encoder-decoder), також відомої як послідовнісна модель (sequence-to-sequence model) [22]. Принцип роботи цієї архітектури представлено на рисунку 2.3.

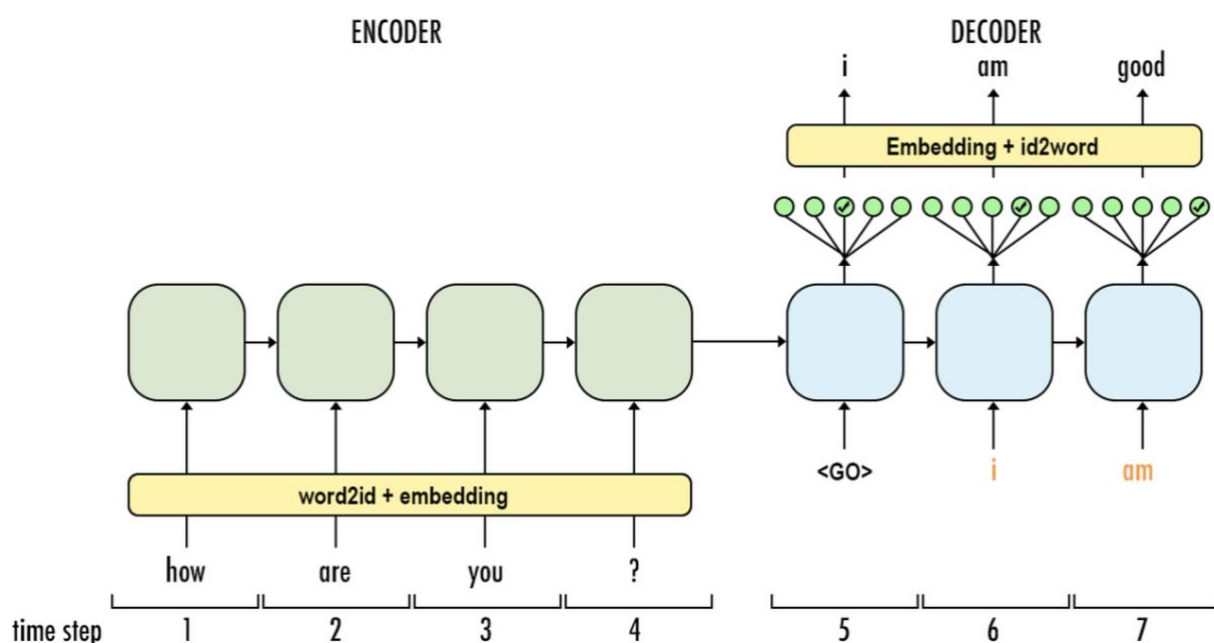


Рисунок 2.3 — Архітектура енкодер-декодер

Енкодер обробляє вхідне речення мовою-джерелом і перетворює його на векторне подання фіксованої довжини, яке містить усю семантичну інформацію про речення. Це векторне подання часто називають контекстним вектором або прихованим станом. Декодер приймає цей контекстний вектор і поступово генерує речення цільовою мовою, слово за словом, використовуючи інформацію з вектора та раніше згенерованих слів.

Перші успішні реалізації NMT використовували рекурентні нейронні мережі (Recurrent Neural Networks, RNN) як для енкодера, так і для декодера. На рисунку

2.4 представлена послідовнісна модель енкодер-декодер для рекурентної нейронної мережі.

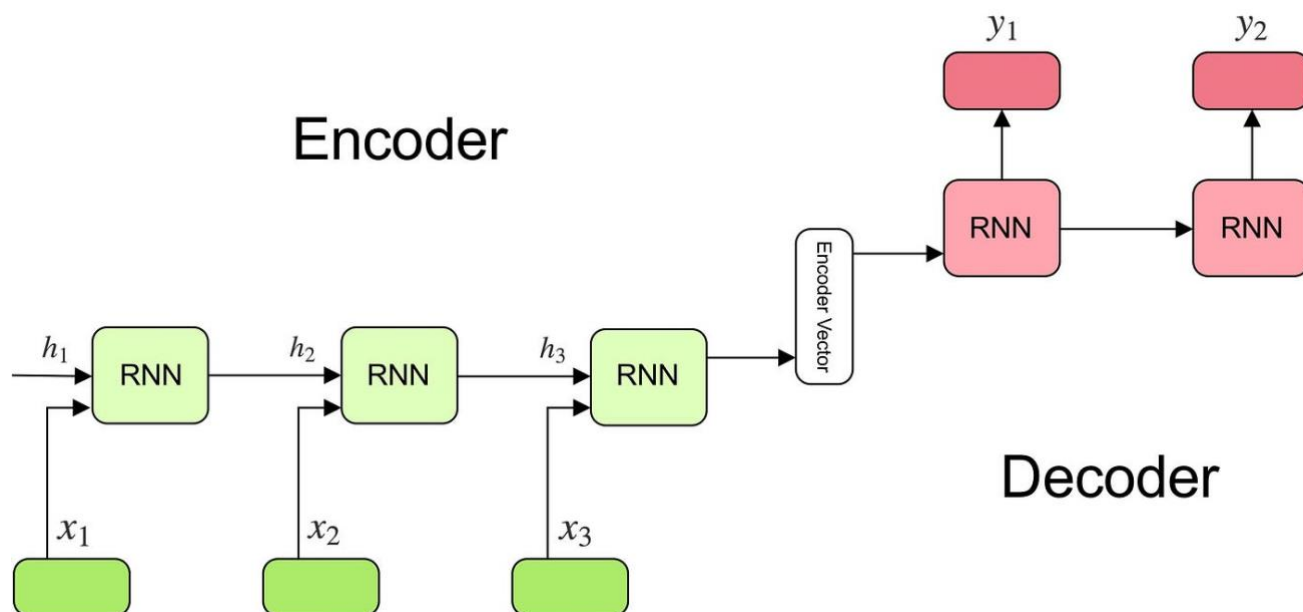


Рисунок 2.4 — Послідовнісна модель енкодер-декодер для рекурентної нейронної мережі

RNN мають здатність обробляти послідовності довільної довжини завдяки механізму рекурентних зв'язків, що дозволяє їм «запам'ятовувати» інформацію з попередніх кроків [23]. Однак звичайні RNN страждають від проблеми зникаючого градієнта, що ускладнює навчання на довгих послідовностях. Для вирішення цієї проблеми були розроблені більш досконалі архітектури, зокрема довга короткочасна пам'ять (Long Short-Term Memory, LSTM) та керовані рекурентні блоки, які містять спеціальні механізми для збереження інформації на довгих відстанях [24].

Введення векторних подань слів стало ще одним важливим кроком у розвитку NMT. Замість представлення слів як дискретних символів нейронні мережі використовують щільні векторні подання, де кожне слово кодується вектором дійсних чисел у багатовимірному просторі. Це дозволяє моделі виявляти семантичні та синтаксичні подібності між словами і краще узагальнювати знання на нові контексти [25]. Техніки, такі як, Word2Vec та GloVe дозволили отримувати

високоякісні векторні подання слів, що значно покращило якість нейронного перекладу.

Основними перевагами нейронного машинного перекладу над класичними методами є здатність краще враховувати контекст та моделювати складні залежності між словами, що призводить до більш природного та плавного перекладу. NMT-системи демонструють особливо високі результати для мовних пар із складними морфологічними системами та гнучким порядком слів. Крім того, наскрізний підхід до навчання дозволяє моделі самостійно виявляти релевантні лінгвістичні закономірності без необхідності ручного формулювання правил [26].

Водночас ранні архітектури нейронного перекладу на базі RNN мали труднощі з обробкою дуже довгих речень. Використання фіксованого контекстного вектора створювало «вузьке місце» в архітектурі, оскільки вся інформація про вхідне речення мала бути стиснута в один вектор обмеженої розмірності. Це призводило до втрати деталей, особливо для довгих та складних речень. Крім того, рекурентна природа RNN ускладнювала паралелізацію обчислень під час навчання, що уповільнювало процес підготовки моделі на великих наборах даних [27].

Ці виклики стали поштовхом до розроблення нових архітектур, зокрема механізмів уваги [28] та моделі Transformer, які дозволили подолати обмеження ранніх NMT-систем.

2.1.3 Сучасні нейромережеві архітектури

Подальшим розвитком нейронного машинного перекладу стала архітектура Transformer, яка кардинально змінила підхід до обробки послідовностей і стала основою для більшості сучасних систем NMT [9]. Ключовою інновацією Transformer є відмова від рекурентних структур на користь механізму уваги, що дозволяє моделі безпосередньо оцінювати важливість кожного слова вхідного речення для генерації кожного слова в перекладі. Цей підхід вирішив проблему «вузького місця» фіксованого контекстного вектора та дозволив ефективно паралелізувати обчислення. Архітектуру нейронного машинного перекладу

Transformer, запропоновану Ашишом Васвані та співавторами у 2017 році, продемонстровано на рисунку 2.5.

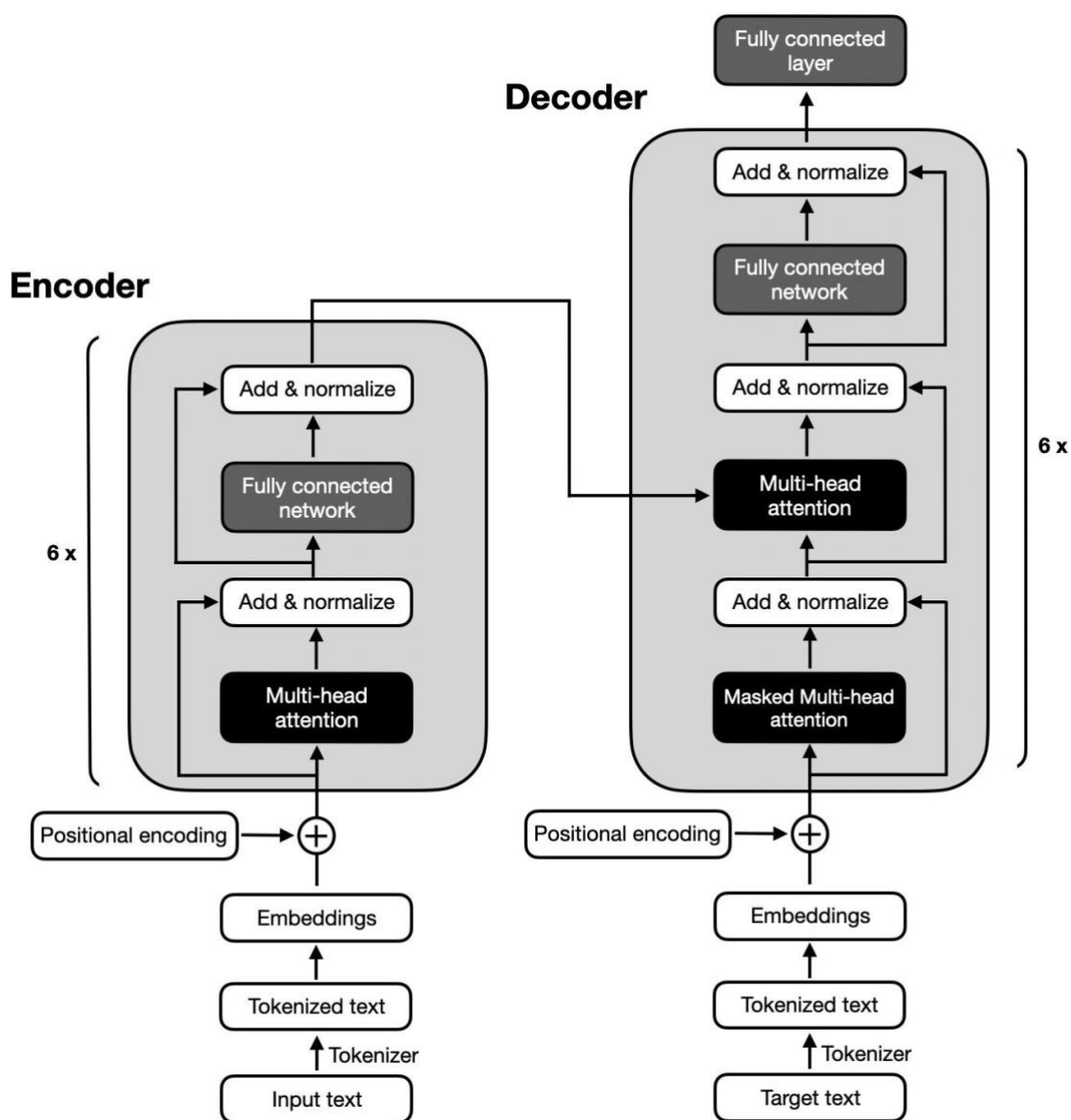


Рисунок 2.5 — Архітектура трансформера, запропонована Ашишом Васвані

Механізм уваги є центральним компонентом архітектури Transformer і полягає у здатності моделі динамічно зважувати важливість різних елементів вхідної послідовності при обробці кожного елемента вихідної послідовності. Замість того, щоб стискати всю інформацію про речення в один вектор, механізм уваги дозволяє декодеру на кожному кроці генерації «звертатися» до всіх слів

вхідного речення та визначати, які з них найбільш релевантні для поточного контексту [28]. Це досягається шляхом обчислення векторів запиту, ключів та значень для кожного елемента послідовності, а потім визначення ваг уваги як скалярного добутку запиту з ключами, нормованого за допомогою функції softmax.

Архітектура Transformer використовує не звичайний механізм уваги, а його вдосконалену версію — багатоголовий механізм самоуваги [9]. Цей механізм показано на рисунку 2.6.

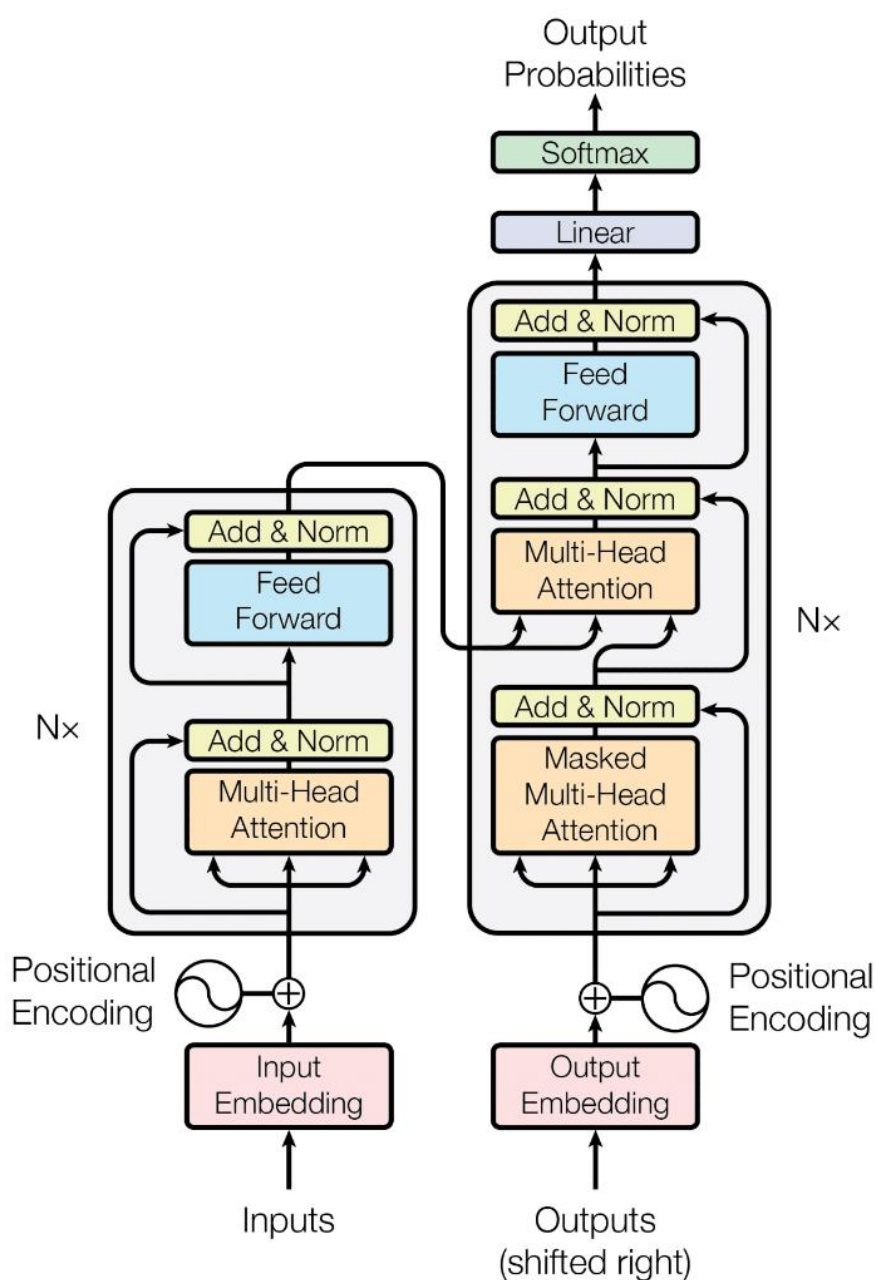


Рисунок 2.6 — Механізм самоуваги

Самоувага означає, що модель обчислює увагу між елементами однієї і тієї ж послідовності, що дозволяє кожному слову враховувати контекст інших слів у реченні. Багатоголовість передбачає застосування механізму уваги кілька разів паралельно з різними наборами параметрів, що дозволяє моделі одночасно фокусуватися на різних аспектах контексту — наприклад, на синтаксичних зв'язках, семантичних відношеннях або локальних залежностях.

Загальна структура моделі Transformer складається з двох основних блоків: стека шарів енкодера та стека шарів декодера. Кожен шар енкодера містить два підшари: шар багатоголового механізму самоуваги та позиційно-залежну повнозв'язну мережу. Кожен підшар оточений залишковим з'єднанням і шаром нормалізації, що полегшує навчання глибоких моделей. Декодер має аналогічну структуру, але містить додатковий підшар багатоголової уваги, що дозволяє йому враховувати виходи енкодера при генерації перекладу.

Оскільки архітектура Transformer не містить рекурентних зв'язків, вона не має вбудованого уявлення про порядок слів у послідовності. Для врахування позиційної інформації використовуються позиційні кодування, що додаються до векторних подань слів на вході енкодера та декодера. Ці кодування можуть бути як фіксованими, що обчислюються за тригонометричними функціями, так і навчальними параметрами моделі [9].

Схематично архітектура Transformer зображена на рисунку 2.7, де показано основні компоненти моделі: стек енкодера зліва, що обробляє вхідне речення, та стек декодера справа, що генерує переклад. Стрілки вказують напрямок потоку даних, а блоки механізму уваги забезпечують зв'язок між різними частинами моделі.

Важливою перевагою Transformer є можливість ефективної паралелізації обчислень. На відміну від RNN, які обробляють послідовність крок за кроком, усі елементи послідовності в Transformer можуть оброблятися одночасно завдяки механізму уваги. Це значно прискорює як процес навчання моделі на великих корпусах, так і процес інференсу при виконанні перекладів [9]. Завдяки цій властивості та високій якості результатів архітектура Transformer швидко стала

стандартом де-факто для машинного перекладу та багатьох інших задач обробки природної мови (рис. 2.7).

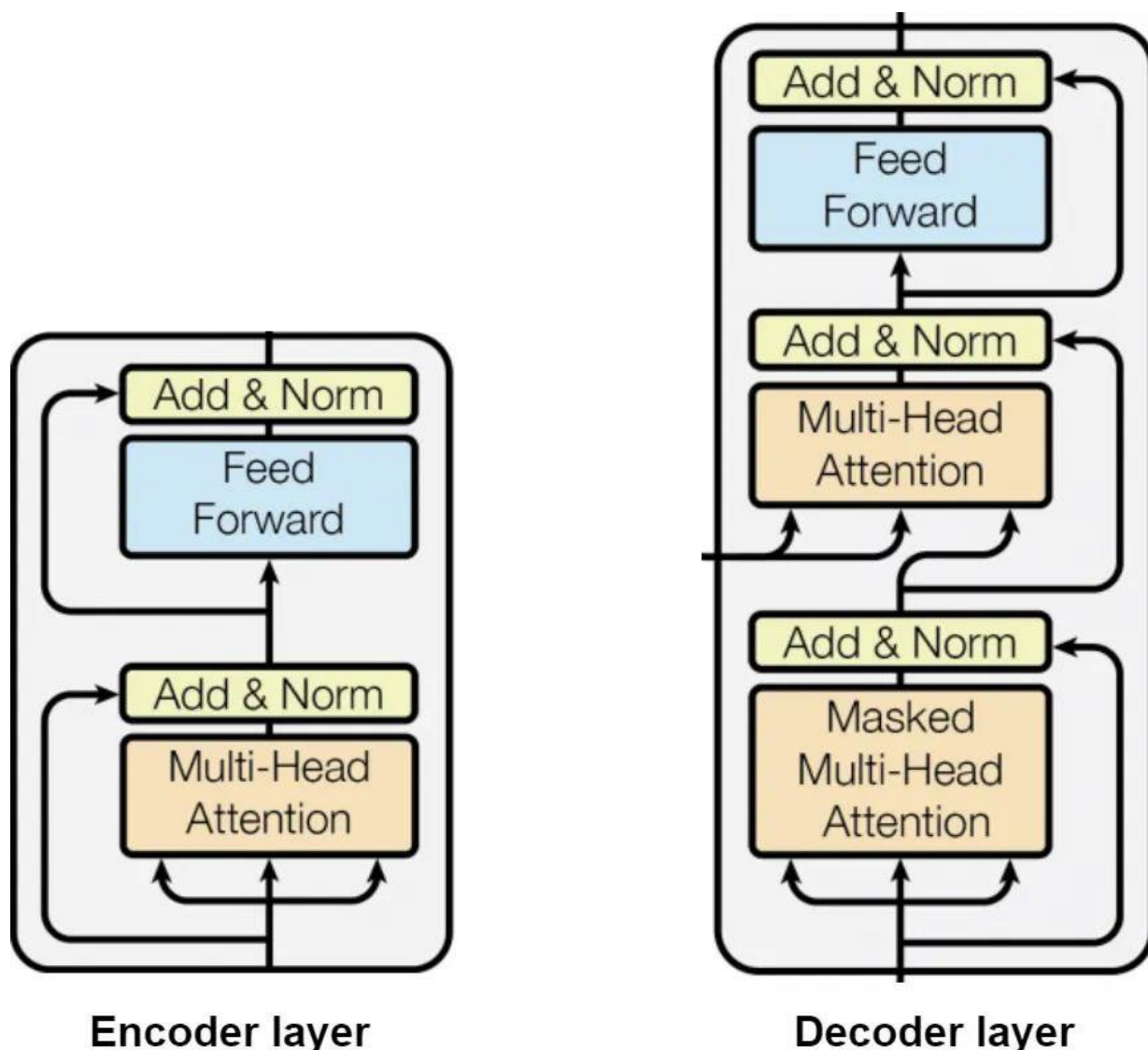


Рисунок 2.7 — Спрощена схема архітектури Transformer

На основі архітектури Transformer було розроблено численні варіанти та розширення, що адаптують базову модель для різних задач. BERT (Bidirectional Encoder Representations from Transformers) використовує лише енкодерну частину Transformer для створення контекстуалізованих подань слів, що виявилось корисним для задач розуміння тексту [29]. GPT (Generative Pre-trained Transformer) застосовує лише декодерну частину для генеративних задач і демонструє вражаючі можливості у створенні зв'язного тексту [30]. Ці та інші варіанти розширили сферу

застосування Transformer далеко за межі машинного перекладу, охопивши практично всі аспекти обробки природної мови.

Завдяки своїй гнучкості та ефективності, архітектура Transformer стала основою для побудови найсучасніших систем нейронного машинного перекладу, включаючи комерційні сервіси на кшталт Google Translate, DeepL, Microsoft Translator та інші. Саме ці архітектури та їхні модифікації лежать в основі сучасних сервісів машинного перекладу, забезпечуючи високу якість результатів, яку було б неможливо досягти класичними методами. У наступному підрозділі буде детальніше розглянуто конкретний сервіс DeepL, що був обраний для реалізації машинного перекладу.

2.2 DeepL як основний сервіс машинного перекладу

Для реалізації функції перекладу у вебзастосунку для вивчення іншомовної лексики необхідний інструмент, який забезпечує не лише високу якість перекладу, а й зручну інтеграцію через програмний інтерфейс (API), стабільність роботи та прийнятну вартість використання. Оскільки створення власної системи нейронного машинного перекладу вимагає значних обчислювальних ресурсів, великих паралельних корпусів текстів та тривалого часу на навчання моделі, доцільним рішенням є використання готового сервісу машинного перекладу.

На сучасному ринку існує кілька провідних сервісів машинного перекладу, що пропонують API для інтеграції в сторонні застосунки. Серед найбільш популярних можна виділити: Google Cloud Translation API, Microsoft Translator, Amazon Translate та DeepL API [10, 31, 32, 33]. Кожен із цих сервісів має свої особливості щодо якості перекладу, підтримуваних мовних пар, структури API та моделей ціноутворення. Проте для цілей даного проєкту було обрано саме DeepL як основний інструмент для реалізації функції машинного перекладу.

DeepL — це хмарний сервіс машинного перекладу, розроблений німецькою компанією DeepL SE, який використовує сучасні нейромережеві архітектури на

базі Transformer для забезпечення високої якості перекладу [33]. Компанія запустила свій сервіс у 2017 році, і з того часу DeepL неодноразово демонстрував вищу якість перекладу порівняно з конкурентами в незалежних порівняльних тестах, особливо для європейських мов. Сервіс використовує пропріетарну архітектуру нейронної мережі, натреновану на величезних обсягах перекладених текстів, що дозволяє йому генерувати переклади, близькі до природної мови [11].

Аналіз, проведений у розділі 1.3, показав, що DeepL демонструє найвищу якість перекладу для європейських мов. Це є критично важливим для застосунку вивчення лексики, де точність перекладу безпосередньо впливає на ефективність навчання. Хоча DeepL підтримує менше мов порівняно з конкурентами, його покриття включає всі основні європейські мови та декілька азійських, що є достатнім для даного проєкту [33].

Сервіс DeepL має чимало переваг, необхідних для реалізації машинного перекладу у вебзастосунку. Ключовою перевагою є висока якість перекладу. DeepL неодноразово демонстрував кращі результати порівняно з іншими сервісами в незалежних порівняльних тестах, особливо для мовних пар, що включають німецьку, французьку, іспанську та англійську мови. Переклади DeepL характеризуються більш природною інтонацією та кращим збереженням нюансів оригінального тексту [34].

Іншою перевагою є зручний та добре документований API. DeepL API надає простий REST-інтерфейс з чіткою документацією та прикладами коду для різних мов програмування, включаючи Python. Структура запитів та відповідей є інтуїтивною та добре продуманою, що спрощує інтеграцію в застосунок [33].

Сервіс має безкоштовний тариф, який можна використати для розроблення. DeepL пропонує безкоштовний план DeepL API Free, який дозволяє перекладати до 500000 символів на місяць без оплати. Цього більш ніж достатньо для розроблення, тестування та використання в рамках магістерської роботи, а також для невеликої кількості користувачів на початкових етапах експлуатації застосунку [33].

Важливою функціональною можливістю є контроль рівня формальності. DeepL API дозволяє вказувати бажаний рівень формальності перекладу

(формальний або неформальний), що може бути корисним для адаптації словникових карток під різні контексти використання лексики [33; 35].

Також варто відзначити швидкість обробки запитів. Сервіс забезпечує швидку обробку запитів, що є важливим для інтерактивного застосунку, де користувачі очікують на миттєву реакцію системи при додаванні нових слів до словника.

Нарешті, DeepL характеризується надійністю та стабільністю роботи. DeepL є стабільним комерційним сервісом із високою доступністю (uptime) та підтримкою, що мінімізує ризики збоїв у роботі застосунку.

Схематично процес взаємодії застосунку з DeepL API зображено на рисунку 2.8.

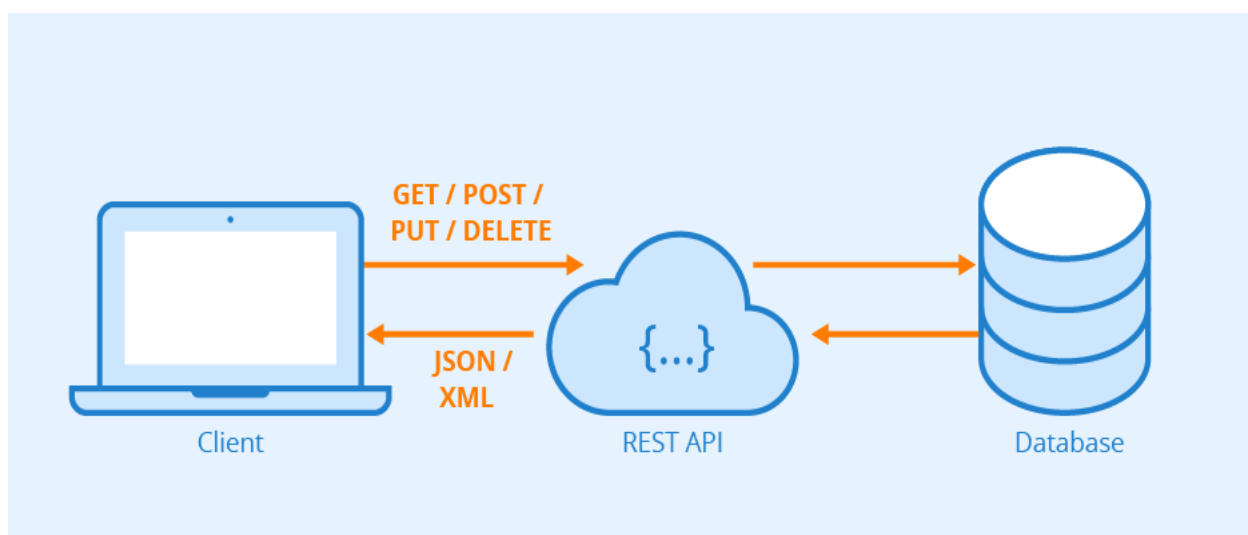


Рисунок 2.8 — Схема взаємодії вебзастосунку з DeepL API

Коли користувач вводить слово або фразу для додавання до словника, вебзастосунок формує HTTP-запит до DeepL API з текстом, мовою-джерелом та цільовою мовою. Сервіс DeepL обробляє запит за допомогою своїх неймережевих моделей і повертає переклад у форматі JSON. Застосунок отримує переклад, зберігає його в базі даних разом з оригінальним словом і відображає користувачеві. Вся взаємодія відбувається через захищене HTTPS-з'єднання з використанням API-ключа для автентифікації.

Технічно інтеграція DeepL API в застосунок на Python є простою завдяки наявності офіційної бібліотеки `deepl-python`, яка надає зручний інтерфейс для роботи з API без необхідності вручну формувати HTTP-запити [36]. Бібліотека підтримує як синхронні, так і асинхронні виклики, що дозволяє ефективно інтегрувати переклад у різні частини застосунку без блокування основного потоку виконання.

Враховуючи представлені переваги: високу якість перекладу, зручність інтеграції, наявність безкоштовного тарифу та додаткові можливості контролю над результатом, — для реалізації функції машинного перекладу в межах даного вебзастосунку було обрано сервіс DeepL як основний інструмент. Це рішення забезпечує оптимальний баланс між якістю перекладу, простотою реалізації та вартістю використання, що є критично важливим для магістерського проєкту з обмеженим бюджетом та часовими рамками.

2.3 Методи вивчення іншомовної лексики

Вивчення іншомовної лексики є ключовим компонентом процесу оволодіння іноземною мовою та однією з найбільш трудомістких задач для учнів. Дослідження в галузі психолінгвістики та педагогіки показують, що успішне засвоєння словникового запасу вимагає не лише первинного запам'ятовування нових слів, а й регулярного повторення матеріалу для його закріплення у довготривалій пам'яті [37]. Проблема полягає в тому, що без системного підходу до повторення новозасвоєна лексика швидко забувається, що підтверджується добре відомою кривою забування Германа Еббінгауза [38].

Розглянемо основні методи вивчення іншомовної лексики, їхні теоретичні основи та практичну ефективність. Розуміння цих методів є необхідним для обґрунтування вибору підходу до організації навчального процесу у вебзастосунку, зокрема для інтеграції алгоритму розподілених повторень.

Існує велика кількість методів вивчення лексики, які можна класифікувати за різними критеріями. За ступенем активності учня методи поділяються на пасивні (читання, прослуховування) та активні (говоріння, письмо, вправи). За способом презентації матеріалу виділяють контекстуальні методи, де слова вивчаються в контексті речень або текстів, та деконтекстні, де слова подаються у вигляді списків або карток [39]. За використанням технологій розрізняють традиційні паперові методи та цифрові інструменти, включаючи мобільні застосунки та вебплатформи.

До традиційних методів вивчення лексики належить механічне запам'ятовування за допомогою списків слів (зубріння), яке полягає в багаторазовому читанні пар слів у двох мовах до їх запам'ятовування [40]. Цей метод є одним із найдавніших і досі широко використовується завдяки своїй простоті. Однак дослідження показують, що механічне зубріння є малоефективним для довготривалого запам'ятовування, оскільки воно не створює міцних асоціативних зв'язків у пам'яті та не забезпечує глибокого розуміння значення слова [41].

Контекстуальний метод вивчення лексики передбачає засвоєння нових слів через їх використання в реченнях, діалогах або текстах [42]. Ідея полягає в тому, що слово, вивчене в контексті, краще запам'ятовується завдяки створенню асоціацій із ситуацією та іншими словами в реченні. Цей підхід є більш природним і відповідає процесу засвоєння рідної мови. Проте контекстуальне навчання потребує більше часу на опрацювання кожного слова та може бути менш ефективним на початкових етапах вивчення мови, коли учень ще не володіє достатнім базовим словниковим запасом для розуміння контексту.

Метод словникових карток є компромісом між механічним зубрінням і контекстуальним навчанням [43]. Кожна картка містить слово на одній стороні та його переклад або визначення на іншій. Учень переглядає картки, намагається згадати переклад і перевіряє себе, перегортаючи картку. Цей метод дозволяє концентруватися на окремих словах, забезпечуючи активне відтворення з пам'яті, що є важливим для запам'ятовування. Сучасні цифрові реалізації цього методу

додають можливості автоматизованого відстеження прогресу та адаптивного підбору матеріалу для повторення.

Мнемотехніки є додатковим інструментом для полегшення запам'ятовування важких слів [44]. Вони включають створення асоціацій, використання акронімів, рими, візуальних образів або історій, що пов'язують нове слово з вже відомою інформацією. Наприклад, для запам'ятовування англійського слова «procrastinate» (відкладати) можна створити історію про «про-кота-що-стинає», який постійно відкладає справи. Хоча мнемотехніки можуть бути дуже ефективними для окремих важких слів, вони потребують креативності та часу на створення асоціацій, що робить їх менш придатними для масового засвоєння великої кількості лексики.

Гейміфікація вивчення лексики передбачає використання ігрових механік для підвищення мотивації та залученості учнів [45]. Сучасні мовні застосунки часто включають елементи ігор: бали, рівні, досягнення, змагання з іншими користувачами.

Дослідження показують, що гейміфікація може значно підвищити регулярність занять і загальну кількість вивченого матеріалу, особливо на початкових етапах [46]. Однак надмірний акцент на ігрових елементах може відволікати від основної мети навчання та створювати поверхневе розуміння матеріалу.

Переваги та недоліки цих методів наведено у таблиці 2.1. Кожен метод має свої сильні та слабкі сторони. Ключовою проблемою процесу вивчення лексики залишається швидке забування нового матеріалу, що описується кривою забування Еббінгауза. Згідно з цією теорією, без повторення людина забуває близько 50% нової інформації протягом першої години та до 70% протягом перших 24 годин після вивчення [38].

Для подолання цієї проблеми необхідний системний підхід до планування повторень, який враховує часовий фактор та індивідуальні особливості запам'ятовування кожного елемента.

Таблиця 2.1 — Порівняння методів вивчення іншомовної лексики

Метод	Переваги	Недоліки	Ефективність довготривалого запам'ятовування
Механічне зубріння	Простота, не потребує додаткових ресурсів	Монотонність, низька ефективність	Низька
Контекстуальне навчання	Глибоке розуміння, природність	Потребує часу, залежність від рівня	Висока при регулярності
Словникові картки	Фокусування на окремих словах, мобільність	Відсутність контексту	Середня–висока
Мнемотехніки	Ефективність для важких слів	Потребує креативності, трудомісткість	Висока для обраних слів
Гейміфікація	Висока мотивація, регулярність	Можлива поверхневність	Середня

Особливу увагу в дослідженнях ефективності вивчення лексики приділяють оптимізації інтервалів між повтореннями матеріалу. Ефективні методи мають забезпечувати повторення матеріалу з інтервалами, що поступово збільшуються: перше повторення має відбутися невдовзі після початкового вивчення, наступні — через дедалі більші проміжки часу [47]. Цей принцип лежить в основі методу розподілених повторень, який довів свою високу ефективність у численних дослідженнях і став основою для багатьох сучасних застосунків для вивчення мов.

Отже, аналіз методів вивчення іншомовної лексики показує, що найбільш ефективними є підходи, що поєднують активне відтворення з пам'яті, систематичність повторень і врахування індивідуальних особливостей запам'ятовування. Метод розподілених повторень відповідає цим вимогам і є науково обґрунтованим підходом до організації навчального процесу.

2.4 Алгоритм розподілених повторень

Алгоритм розподілених повторень є науково обґрунтованим методом організації навчання, який базується на оптимальному розподілі повторень матеріалу в часі для максимізації довготривалого запам'ятовування [48].

Цей підхід виник як відповідь на проблему швидкого забування нової інформації та ґрунтується на емпіричних даних про те, як працює людська пам'ять. Замість рівномірних або випадкових повторень алгоритм динамічно визначає оптимальні інтервали для кожного елемента навчального матеріалу на основі того, наскільки добре учень його запам'ятав.

В основі методу розподілених повторень лежить крива забування, яку вперше описав німецький психолог Герман Еббінгауз у 1885 році на основі експериментів з власною пам'яттю [38]. Криву забування Германа Еббінгауза представлено на рисунку 2.9.

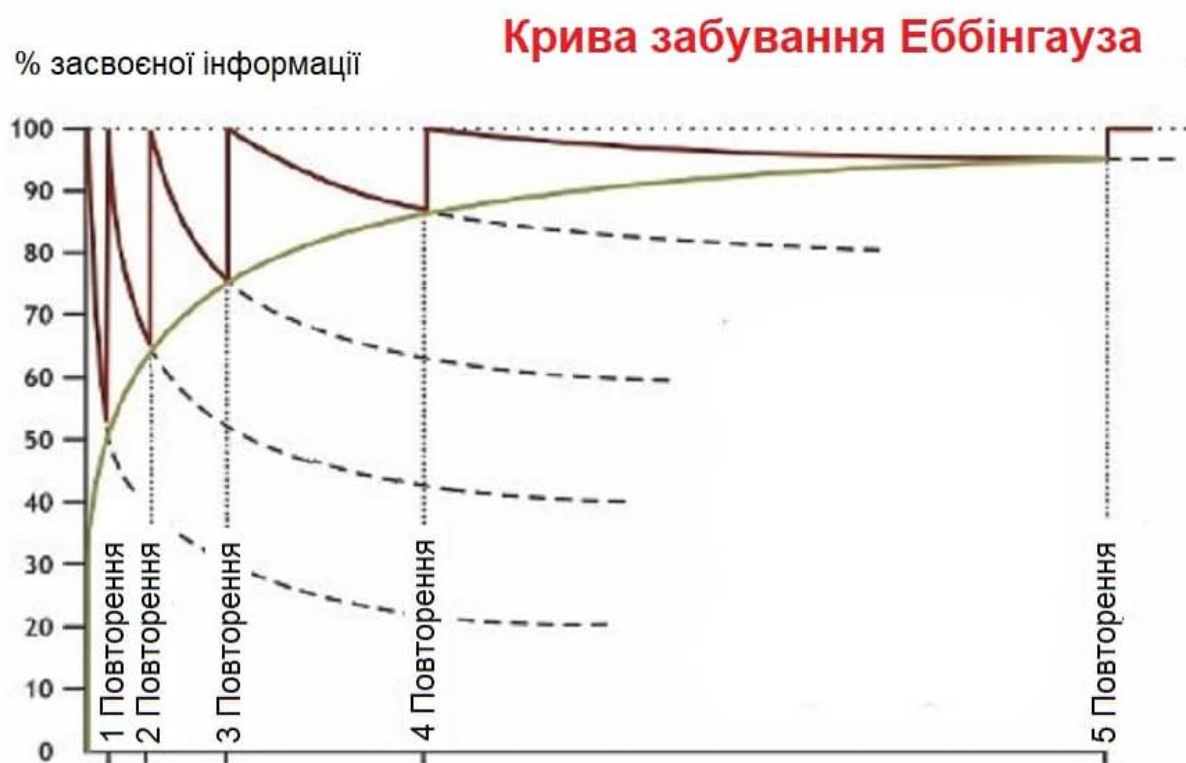


Рисунок 2.9 — Крива забування та вплив повторень на збереження інформації в пам'яті

На рисунку 2.9 зображено криву забування Еббінгауза та продемонстровано ефект повторень. Червона лінія показує природний спад здатності відтворити інформацію без повторень. Зелена лінія демонструє, як кожне повторення підіймає рівень запам'ятовування (% засвоєної інформації). Оптимальні моменти для повторення — одразу перед тим, як інформація буде забута, що максимізує ефективність навчання.

Крива забування показує, що ймовірність відтворення інформації з пам'яті експоненційно зменшується з часом після її вивчення. Без повторень більшість нової інформації забувається протягом перших днів або тижнів. Однак кожне успішне повторення зміцнює слід у пам'яті та уповільнює процес забування, що дозволяє поступово збільшувати інтервал до наступного повторення [49].

Один із найвідоміших і найбільш впливових алгоритмів розподілених повторень — це метод SuperMemo, розроблений Петром Возняком у 1980-х роках [50]. Його алгоритм SM-2 (SuperMemo-2), опублікований у 1988 році, став основою для більшості сучасних систем розподілених повторень завдяки простоті реалізації та доведеній ефективності. Алгоритм SM-2 використовує концепцію «коефіцієнта легкості» для кожного елемента, який відображає, наскільки легко цей елемент запам'ятовується користувачем, та динамічно розраховує інтервали між повтореннями [50, 51].

Коли користувач вивчає новий елемент (наприклад, словникову картку), йому призначається початковий коефіцієнт легкості, рівний 2.5. Після кожного повторення користувач оцінює якість свого відтворення інформації за шкалою від 0 до 5, де 0 означає повну відсутність спогаду, а 5 — легке та швидке відтворення. На основі цієї оцінки алгоритм оновлює коефіцієнт легкості та розраховує інтервал до наступного повторення.

Алгоритм SM-2 застосовується для планування повторень кожного окремого елемента. Для нового елемента задають коефіцієнт легкості, лічильник вдалих повторень та початковий інтервал. Далі система при кожному показі елемента фіксує оцінку якості відтворення за шкалою.

Якщо відповідь вважається вдалою, лічильник вдалих повторень збільшують на одиницю. Для першого вдалого повторення інтервал залишають рівним одному дню, для другого — встановлюють 6 днів. Починаючи з третього вдалого повторення новий інтервал обчислюють як попередній інтервал, помножений на коефіцієнт легкості, зазвичай із округленням до цілої кількості днів. Після цього коефіцієнт легкості коригують залежно від оцінки за формулою.

Якщо відповідь невдала, вважається, що користувач фактично «забув» елемент: лічильник вдалих повторень скидають до нуля, наступний показ призначають знову через один день, а коефіцієнт легкості знову перераховують. У підсумку, коли користувач добре відтворює матеріал, інтервали між повтореннями швидко зростають, а за частих помилок система повертає елемент до коротких інтервалів і зменшеного коефіцієнта легкості.

Залежно від оцінки алгоритм або збільшує інтервал до наступного повторення (при успішному відтворенні), або скидає лічильник повторень (при невдачі). Цикл триває до тих пір, поки елемент не буде надійно закріплений у довготривалій пам'яті.

Метод розподілених повторень має кілька ключових переваг порівняно з традиційними підходами до вивчення. По-перше, він значно економить час завдяки адаптивному плануванню: учень витрачає більше часу на важкий матеріал і менше — на легкий. По-друге, метод забезпечує довготривале запам'ятовування завдяки оптимізації моментів повторення. По-третє, системний підхід мотивує до регулярних занять, оскільки учень бачить конкретний прогрес у вигляді зростаючих інтервалів між повтореннями добре засвоєного матеріалу [52].

Дослідження ефективності розподілених повторень показують значні переваги цього методу над традиційним навчанням. Метааналіз Ніколаса Сепеди з колегами, проведений у 2006 році, продемонстрував, що учні, які використовували розподілені повторення, через 8–30 днів після навчання зберігали в пам'яті приблизно вдвічі більше матеріалу, порівняно з групою, що використовувала масоване повторення [53]. Інше дослідження Джефрі Карпіке та Генрі Родігера, проведене у 2008 році, показало, що активне відтворення з пам'яті в поєднанні з

розподіленими повтореннями є найбільш ефективним методом навчання порівняно з простим перечитуванням матеріалу [54].

Алгоритм SM-2, незважаючи на свою ефективність, має деякі обмеження. Зокрема, фіксовані початкові інтервали можуть бути не оптимальними для всіх типів матеріалу та учнів. Пізніші версії алгоритму SuperMemo та інші системи розподілених повторень запропонували більш складні моделі, що враховують додаткові фактори, такі як складність матеріалу, час, витрачений на вивчення, та історію повторень інших користувачів [50]. Однак базовий алгоритм SM-2 залишається найбільш популярним завдяки своїй простоті та достатній ефективності для більшості застосувань.

У даному вебзастосунку буде реалізовано модифіковану версію алгоритму SM-2, адаптовану до вивчення іншомовної лексики. Модифікація передбачає можливість налаштування початкових інтервалів відповідно до дати, на коли потрібно вивчити слово.

Загалом метод розподілених повторень має значні переваги та є доцільним для використання в системах вивчення лексики. Його реалізація в рамках даного проєкту дозволить забезпечити ефективне довготривале запам'ятовування іншомовної лексики та оптимізувати навчальний процес для користувачів вебзастосунку.

2.5 Засоби розроблення вебзастосунку

Розроблення програмних застосунків розпочинається з підбору технологій, які визначають архітектуру системи. Для вебзастосунку необхідно обрати технології як для серверної частини (backend), що відповідає за логіку, роботу з базою даних та інтеграцію з API машинного перекладу, так і для клієнтської частини (frontend), що забезпечує користувацький інтерфейс.

Технології мають бути достатньо простими для реалізації проєкту в рамках магістерської роботи, але водночас надавати всі необхідні можливості для створення якісного застосунку.

2.5.1 Python як основна мова програмування

Python — це високорівнева мова програмування загального призначення з динамічною типізацією та автоматичним управлінням пам'яттю, яка підтримує різні парадигми програмування, включаючи об'єктно-орієнтовану, процедурну та функціональну [55]. Створена Гвідо ван Россумом у 1991 році, Python здобув широку популярність завдяки своєму простому та зрозумілому синтаксису, що робить код легким для читання та розроблення. Філософія мови підкреслює важливість читабельності коду та принцип «має бути один — і бажано лише один — очевидний спосіб зробити щось» [56].

Для розроблення серверної частини вебзастосунку було обрано Python версії 3.13.0 — останню стабільну версію мови станом на момент початку розроблення. Це рішення обумовлене низкою ключових переваг Python для веброзроблення та інтеграції з API сторонніх сервісів.

Дана мова програмування має одну з найбільших екосистем бібліотек, що охоплюють практично всі аспекти програмування. Для веброзроблення доступні потужні фреймворки Django та Flask, для роботи з HTTP-запитами — бібліотека requests, для інтеграції з DeepL API — офіційна бібліотека deepl-python. Це значно пришвидшує розробку, оскільки більшість типових задач вже має готові рішення [55].

Для роботи зі сторонніми API Python є ідеальним вибором завдяки зручній обробці JSON, простоті виконання HTTP-запитів та можливості легко обробляти асинхронні операції. Інтеграція DeepL API в застосунок на Python вимагає лише кількох рядків коду.

Високу швидкість розроблення забезпечує простий синтаксис Python, який дозволяє швидко реалізовувати функціональність та вносити зміни. Відсутність необхідності компіляції та гнучка система типів прискорюють цикл розроблення та

тестування. Це особливо важливо для магістерських проєктів з обмеженими часовими рамками.

Python є кросплатформеною мовою програмування, код якої працює на всіх основних операційних системах (Windows, macOS, Linux) без модифікацій, що забезпечує гнучкість у виборі середовища розроблення та розгортання [55].

Для веброзроблення Python має два потужні зрілі фреймворки: Django для великих застосунків з багатою функціональністю «з коробки» та Flask для легших проєктів. Обидва фреймворки є добре документованими та мають велику спільноту розробників.

Python характеризується однією з найбільших спільнот програмістів у світі, що забезпечує доступність навчальних матеріалів, оперативну допомогу на форумах та постійний розвиток екосистеми бібліотек [57].

Підтримка сучасних практик розроблення є ще однією перевагою: Python підтримує типізацію підказок з версії 3.5, що дозволяє використовувати статичні аналізатори коду для виявлення помилок на етапі розроблення. Також мова має вбудовані інструменти для модульного тестування, документування та пакетування застосунків.

Ключові переваги Python: велика екосистема бібліотек, простота інтеграції з API, висока швидкість розроблення та підтримка сучасних вебфреймворків. Це зумовлює вибір даної мови в якості основного засобу реалізації серверної частини вебзастосунку для вивчення іншомовної лексики.

2.5.2 Фреймворк Django для розроблення серверної частини

Для розроблення серверної частини вебзастосунку доцільно використовувати вебфреймворк, який оптимізує створення типової функціональності вебсайтів, зокрема обробку HTTP-запитів, маршрутизацію URL, роботу з базою даних, автентифікацію користувачів та забезпечення безпеки. Використання вебфреймворку замість написання всієї логіки від самого початку значно пришвидшує розроблення та зменшує кількість помилок завдяки застосуванню якісних, перевірених практик.

Django — це високорівневий вебфреймворк для мови програмування Python, який реалізує архітектурний патерн Model–View–Template (MVT) і забезпечує швидку розробку безпечних та масштабованих вебзастосунків [58]. Створений у 2005 році для управління новинними сайтами, Django з часом еволюціонував у один з найпопулярніших вебфреймворків для Python. Філософія фреймворку підкреслює принцип DRY (Don't Repeat Yourself) — уникнення дублювання коду, та «батареї включені» (batteries included) — надання багатої стандартної функціональності без необхідності встановлення сторонніх пакетів [58].

Для реалізації серверної частини вебзастосунку було обрано фреймворк Django версії 5.1.5. Це рішення обґрунтовується наступними ключовими особливостями та перевагами фреймворку.

Django надає потужну вбудовану систему ORM (Object-Relational Mapping), що дозволяє працювати з базою даних через об'єкти Python замість написання SQL-запитів вручну. ORM автоматично генерує SQL-код, забезпечує міграції схеми бази даних та захист від SQL-ін'єкцій. Це значно спрощує роботу з даними та робить код більш читабельним та підтримуваним [58].

Автоматично генерована панель адміністратора є ще однією важливою перевагою фреймворку. Django автоматично створює повнофункціональний вебінтерфейс для управління вмістом бази даних на основі визначених моделей. Це дозволяє швидко додавати, редагувати та видаляти записи без написання додаткового коду, що є особливо корисним під час розроблення та тестування [58].

Для визначення URL-маршрутів Django використовує просту та гнучку систему маршрутизації, що дозволяє зв'язувати URL-адреси з відповідними функціями обробки (views). Це забезпечує чистоту структури проєкту та зручність підтримки [58].

Вбудована система шаблонів Django дозволяє відділити логіку від презентації, використовуючи спеціальний синтаксис для вставки динамічних даних у HTML-сторінки. Шаблони підтримують успадкування, що дозволяє уникати дублювання коду в різних сторінках застосунку [58].

Django автоматично забезпечує безпеку «з коробки», захищаючи від багатьох типових вразливостей вебзастосунків, включаючи XSS (міжсайтовий скриптинг), CSRF (підробка міжсайтових запитів), SQL-ін'єкції та clickjacking. Фреймворк також має вбудовану систему хешування паролів з використанням сучасних алгоритмів [58].

Вбудована система автентифікації надає повноцінне управління користувачами, що включає реєстрацію, вхід, вихід, зміну паролів та керування правами доступу. Це дозволяє швидко реалізувати функціональність облікових записів користувачів без необхідності створення власної системи [58].

Django має вбудовані засоби для підтримки інтернаціоналізації, що дозволяють перекладати інтерфейс застосунку на різні мови. Це може бути корисним для подальшого розширення функціональності вебзастосунку [58].

Архітектурний патерн MVT, який використовується в Django, схематично зображено на рисунку 2.10.

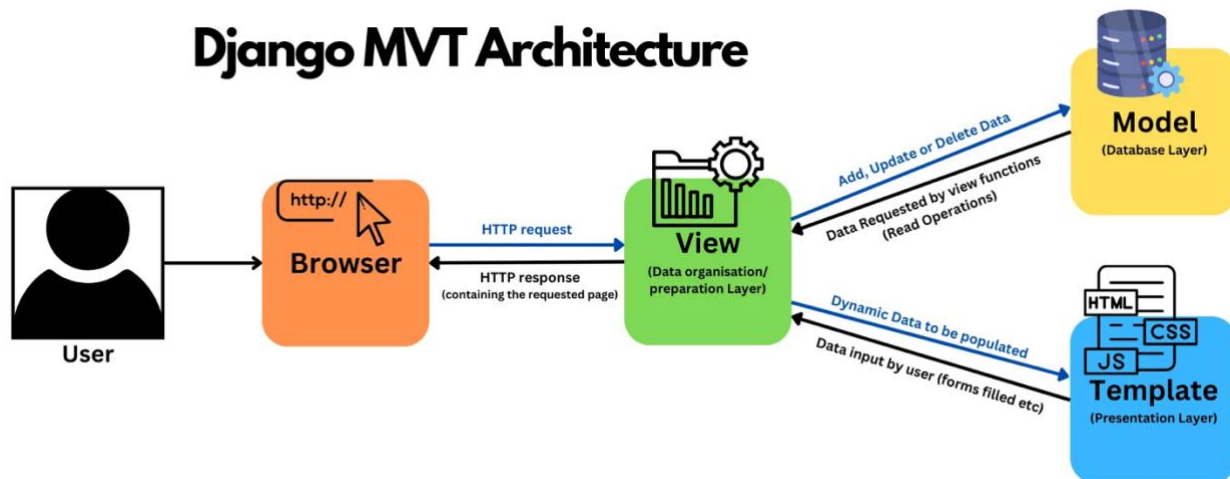


Рисунок 2.10 — Схема архітектурного патерну MVT у фреймворку Django

У цьому патерні компонент Model відповідає за роботу з даними та бізнес-логіку, View обробляє запити користувача та визначає, які дані відобразити, а Template формує HTML-сторінку з динамічними даними. Користувач надсилає HTTP-запит, який обробляється відповідним View, що взаємодіє з Model для

отримання необхідних даних та передає їх у Template для формування фінальної сторінки, яка повертається користувачеві.

Розвинена екосистема додатків Django дозволяє легко розширювати функціональність застосунку. Існують готові пакети для REST API (Django REST Framework), складних форм, обробки зображень, інтеграції з платіжними системами та багато інших. Для даного проєкту можуть бути корисними пакети для роботи з зовнішніми API та для реалізації асинхронних задач.

Враховуючи наявні засоби роботи з базами даних, зручну систему маршрутизації, вбудовану панель адміністратора та комплексні механізми забезпечення безпеки, для реалізації серверної частини вебзастосунку було обрано фреймворк Django. Цей вибір забезпечує швидку розробку, високу якість коду та можливість ефективної підтримки та розширення застосунку в майбутньому.

2.5.3 База даних SQLite

Розроблюваний вебзастосунок містить дані про користувачів, словникові одиниці з перекладами, історію повторень та статистику навчання, що потребують надійного та структурованого зберігання. Для цього необхідна система управління базами даних (СУБД), яка забезпечить збереження, ефективний доступ та маніпулювання даними. Вибір СУБД залежить від масштабу проєкту, очікуваного навантаження, вимог до продуктивності та складності розгортання [59].

Оскільки проєкт не передбачає високих навантажень, горизонтального масштабування на кілька серверів та одночасної роботи сотень тисяч користувачів, було прийнято рішення використати вбудовану реляційну СУБД SQLite замість клієнт-серверних систем.

SQLite — це реляційна база даних, що зберігає всі дані в одному файлі на диску та не вимагає окремого серверного процесу для роботи [60]. Це робить її ідеальним вибором для невеликих та середніх застосунків, навчальних проєктів.

До основних переваг використання SQLite для даного вебзастосунку можна віднести низку важливих аспектів. Ключовою перевагою є відсутність серверного процесу. SQLite працює безпосередньо з файлом бази даних, що усуває

необхідність налаштування, запуску та підтримки окремого серверу СУБД. Це значно спрощує розгортання застосунку, особливо в середовищі розроблення [61].

SQLite підтримує більшість стандартних можливостей SQL, що забезпечує достатню функціональність для потреб застосунку. До підтримуваних можливостей належать транзакції, тригери, представлення та індекси. Для потреб застосунку вивчення лексики цієї функціональності є цілком достатньо [61].

Портативність є ще однією важливою перевагою цієї системи управління базами даних. Файл бази даних SQLite є кросплатформеним і може бути відкритий на будь-якій операційній системі без конвертації. Це полегшує розробку на одній платформі та розгортання на іншій [61].

Для завдань розроблюваного вебзастосунку, функціональності SQLite є цілком достатньо. У разі необхідності масштабування застосунку в майбутньому, Django дозволяє легко мігрувати на більш потужні СУБД, такі як PostgreSQL, MySQL, та інші [58].

Структура бази даних застосунку включатиме основні таблиці для зберігання інформації про користувачів, словникові одиниці (слова з їх перекладами), історію повторень та параметри алгоритму розподілених повторень для кожної словникової одиниці. Використання реляційної моделі забезпечує цілісність даних та підтримку складних запитів для відображення статистики навчання користувача.

Таким чином, вибір SQLite як СУБД для даного вебзастосунку є обґрунтованим рішенням, що забезпечує простоту розроблення, розгортання та підтримки при достатній функціональності для реалізації усіх необхідних можливостей системи вивчення іншомовної лексики.

2.5.4 Технології клієнтської частини

Сучасні вебзастосунки розробляються з використанням технологій HTML, CSS та JavaScript, які разом забезпечують структуру, оформлення та інтерактивність користувацького інтерфейсу. Ці три технології є фундаментальними для веброботелення та підтримуються всіма сучасними

веббраузерами [62]. Для створення клієнтської частини вебзастосунку використовуються стандартні версії цих технологій — HTML5, CSS3 та JavaScript.

HTML (HyperText Markup Language) — це мова гіпертекстової розмітки, яка слугує для створення структури вебсторінки [62]. HTML5 додала нові семантичні елементи (header, nav, article, section, footer), що покращують структуру документа та його доступність для асистивних технологій. HTML-документ визначає ієрархію елементів сторінки: заголовки, параграфи, списки, форми, зображення та інші компоненти. У контексті даного проєкту HTML використовується для створення форм додавання нових слів, відображення словникових карток під час повторення та структурування інформаційних сторінок застосунку.

CSS (Cascading Style Sheets) — це мова каскадних таблиць стилів, що використовується для опису зовнішнього вигляду вебсторінок [63]. CSS дозволяє визначати кольори, шрифти, розміри, відступи, розташування елементів та інші візуальні параметри. CSS3, остання основна версія стандарту, додала підтримку анімацій, градієнтів, тіней, заокруглених кутів та адаптивного дизайну через медіа-запити. Адаптивний дизайн є особливо важливим для застосунку вивчення мов, оскільки користувачі можуть працювати з ним як на комп'ютерах, так і на мобільних пристроях. CSS у проєкті забезпечує привабливий та зручний інтерфейс, що сприяє кращому досвіду користування [64].

JavaScript — це мова програмування, яка дозволяє додавати інтерактивність до вебсторінок, обробляти події користувача (кліки, введення тексту, прокручування) та взаємодіяти з сервером без повного перезавантаження сторінки [65]. JavaScript виконується безпосередньо у веббраузері користувача та має доступ до Document Object Model (DOM) — об'єктного представлення структури сторінки, що дозволяє динамічно змінювати вміст та стиль елементів. У застосунку JavaScript використовується для відображення та перевертання словникових карток, валідації форм, відправки асинхронних запитів до сервера (AJAX) для отримання перекладів без перезавантаження сторінки, та відображення інтерактивних елементів статистики навчання [65].

Для забезпечення сумісності та покращення досвіду розроблення можуть бути використані додаткові інструменти та бібліотеки. Зокрема, фреймворк Bootstrap надає готові компоненти інтерфейсу та сітку для адаптивного дизайну, що пришвидшує створення естетично привабливого та зручного інтерфейсу [66]. Бібліотека jQuery може спростити роботу з DOM та обробку подій, хоча в сучасній веброботі її роль зменшується на користь нативних JavaScript API.

У даній роботі ці технології використовуються для реалізації клієнтської частини вебзастосунку, забезпечуючи зручний інтерфейс для користувача. HTML формує структуру сторінок, CSS відповідає за візуальне оформлення та адаптивність під різні розміри екранів, а JavaScript забезпечує динамічну взаємодію з користувачем та асинхронний обмін даними з сервером. Комбінація цих стандартних вебтехнологій разом із серверною частиною на Django створює повноцінний вебзастосунок для ефективного вивчення іншомовної лексики з використанням машинного перекладу та алгоритму розподілених повторень.

Висновки до розділу 2

У другому розділі було проведено ґрунтовний аналіз методів машинного перекладу, сучасних сервісів МП, підходів до вивчення іншомовної лексики та засобів розроблення вебзастосунків. Цей аналіз дозволив сформулювати теоретичну основу та обґрунтувати вибір технологій для практичної реалізації проєкту.

Огляд методів машинного перекладу показав еволюцію від класичних підходів на основі правил та статистичних методів до сучасних нейромережових архітектур. Класичні методи — RBMT та SMT — мають значні обмеження у вигляді складності створення правил та фрагментарності перекладу відповідно. Нейронний машинний переклад на базі архітектури енкодер-декодер з використанням RNN та LSTM значно покращив якість перекладу завдяки здатності моделювати довготривалі залежності. Остаточним проривом стала архітектура Transformer з механізмом багатоголової самоуваги, яка дозволила досягти якості

перекладу, близької до людської, та стала основою для всіх сучасних сервісів МП. Було обрано DeepL як основний інструмент для реалізації функції МП у вебзастосунку. Ключовими факторами вибору стали висока якість перекладу, особливо для європейських мов, зручний та добре документований API, наявність безкоштовного тарифу для розроблення та тестування, а також додаткові можливості контролю рівня формальності перекладу.

Аналіз методів вивчення іншомовної лексики виявив, що найбільш ефективними є підходи, що поєднують активне відтворення з пам'яті та систематичність повторень з урахуванням індивідуальних особливостей запам'ятовування. Традиційні методи (механічне зубріння, списки слів) мають низьку ефективність для довготривалого запам'ятовування. Контекстуальні методи та словникові картки демонструють кращі результати. Проблема швидкого забування нового матеріалу, описана кривою забування Еббінгауза, вирішується шляхом застосування алгоритму розподілених повторень.

Детальний розгляд алгоритму SM-2 показав його ефективність для організації навчального процесу. Алгоритм використовує адаптивне планування повторень на основі оцінки якості відтворення користувачем, що забезпечує оптимальні інтервали між повтореннями та максимізує довготривале запам'ятовування при мінімальних витратах часу. Дослідження підтверджують переваги розподілених повторень над традиційними методами навчання. Для даного проєкту буде реалізовано модифіковану версію алгоритму SM-2, адаптовану до специфіки вивчення іншомовної лексики.

Для розроблення вебзастосунку було обрано комплексний стек технологій. Мова програмування Python обрана як основа для серверної частини завдяки великій екосистемі бібліотек, простоті інтеграції з API, високій швидкості розроблення та вебфреймворку Django. Цей фреймворк забезпечує всі необхідні інструменти для швидкого розроблення безпечного вебзастосунку, включаючи ORM для роботи з базою даних, автоматичну панель адміністратора, систему маршрутизації та шаблонів, а також вбудовані механізми безпеки та автентифікації.

База даних SQLite обрана для зберігання даних користувачів та словників завдяки вбудованості в Python, простоті розгортання та достатній функціональності для потреб даного проєкту. Для клієнтської частини використовуються стандартні вебтехнології: HTML5 для структури сторінок, CSS3 для візуального оформлення та адаптивності, JavaScript для інтерактивності та асинхронної взаємодії з сервером.

Результати цього аналізу закладають основу для подальшого розроблення, що буде детальніше розглянуто в наступному розділі.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ З МАШИННИМ ПЕРЕКЛАДОМ

Створення ефективної та масштабованої архітектури, бази даних, реалізація чітких алгоритмів, є важливими кроками для розроблення будь-якого застосунку.

У цьому розділі детально описано програмну реалізацію вебзастосунку вивчення іншомовної лексики з інтегрованим машинним перекладом та алгоритмом розподілених повторень. Описано архітектуру системи, представлено діаграми прецедентів і класів, структуру бази даних тощо. Окремо розглянуто реалізацію ключових функціональних модулів системи.

3.1 Архітектура вебзастосунку

Архітектурою програмної системи називають план побудови розроблюваної системи, ідея якого полягає в описі взаємодії компонентів на високому рівні абстракції. Під час проєктування архітектури ставиться за мету розробити її масштабованою, надійною та логічною, щоб забезпечити ефективну підтримку системи на всіх етапах життєвого циклу. Даний етап є важливим, оскільки зміни в архітектуру після початку реалізації вносити набагато складніше, ніж у процесі первинного проєктування.

Для вебзастосунку було прийнято рішення використовувати багаторівневу клієнт-серверну архітектуру з патерном MVT (Model-View-Template), що є адаптованою версією класичного підходу MVC (Model-View-Controller) для фреймворка Django. Вибір даної архітектури обумовлений необхідністю забезпечити чітке розділення відповідальностей між компонентами системи, що спрощує як розробку нових функцій, так і підтримку існуючого коду.

Окрім цього, багаторівнева організація дозволяє ізолювати бізнес-логіку від представлення даних, що критично важливо для системи вивчення лексики, де алгоритми розподілених повторень мають працювати незалежно від інтерфейсу користувача.

Архітектура системи складається з кількох основних рівнів, кожен з яких відповідає за специфічний аспект функціонування застосунку. Клієнтська частина реалізована на основі стандартних веб-технологій: HTML5 для структурування контенту, CSS3 для стилізації інтерфейсу та JavaScript для забезпечення інтерактивності.

Серверна частина побудована з використанням мови програмування Python версії 3.13.0 та фреймворка Django версії 5.1.5, що надає якісні інструменти для обробки HTTP-запитів, маршрутизації URL-адрес та взаємодії з базою даних через об'єктно-реляційне відображення.

Для збереження даних використовується реляційна система керування базами даних SQLite3, яка у контексті даного проєкту забезпечує достатню продуктивність та зручність розгортання.

Взаємодія між клієнтською та серверною частинами відбувається за допомогою протоколу HTTP з використанням REST-принципів для організації кінцевих точок. Обмін даними здійснюється переважно у форматі JSON для асинхронних запитів, що дозволяє оновлювати окремі частини інтерфейсу без перезавантаження всієї сторінки.

Узагальнена схема архітектури вебзастосунку наведена на рисунку 3.1.

Як видно зі схеми, система реалізує чітке розділення між клієнтською та серверною частинами, що дозволяє незалежно розвивати кожен компонент. Запит від користувача проходить через браузер до серверної частини, де Django-view функції координують обробку запиту, взаємодіючи з моделями даних через ORM та викликаючи необхідні модулі бізнес-логіки. Результати обробки повертаються клієнту у вигляді відрендерених HTML-шаблонів або JSON-відповідей для асинхронних операцій.

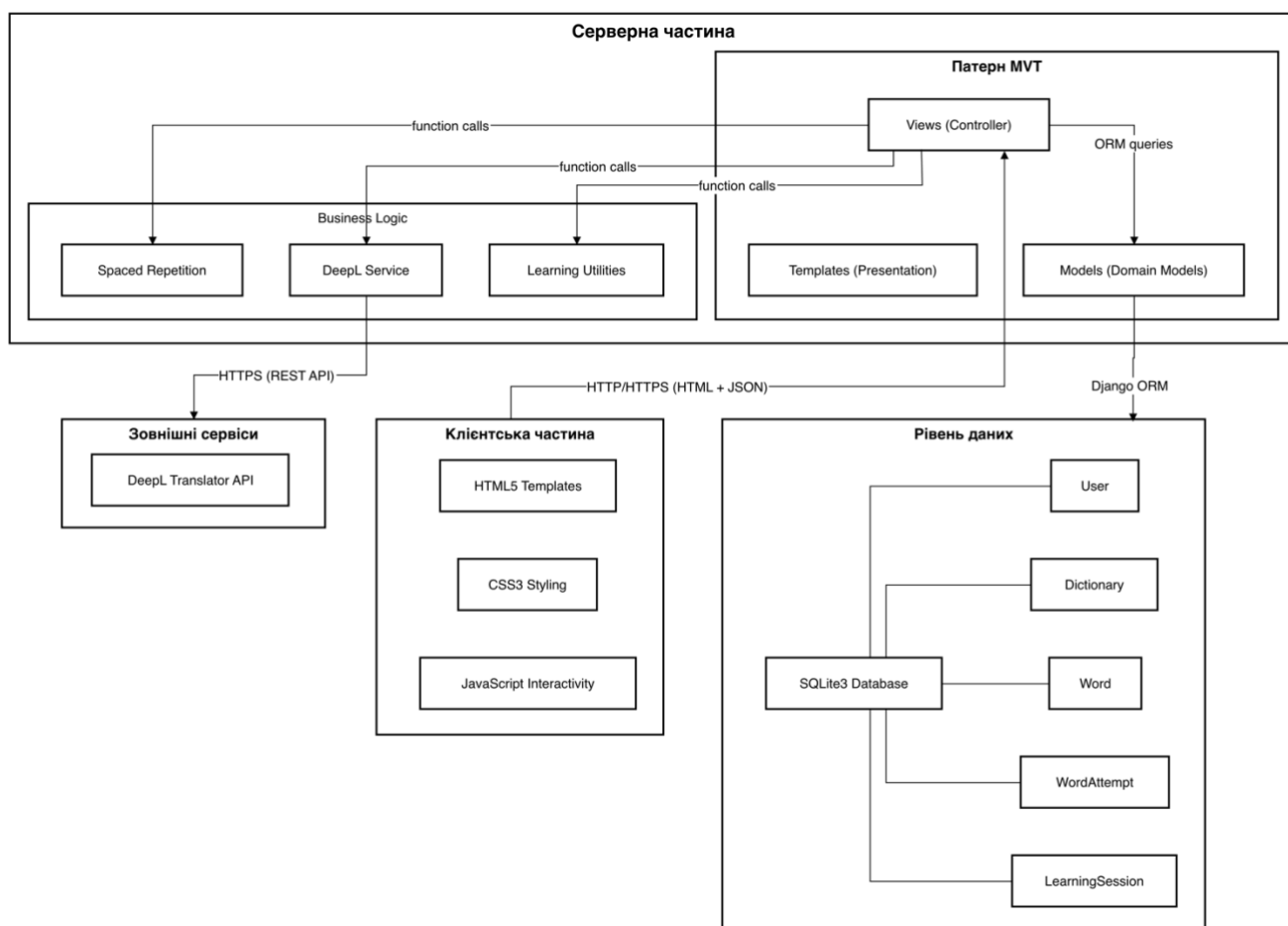


Рисунок 3.1 — Архітектура вебзастосунку

Особливістю даної архітектури є виділення окремих модулів для складної бізнес-логіки, зокрема модуля реалізації алгоритму розподілених повторень та модуля інтеграції з DeepL API. Такий підхід забезпечує модульність системи — кожен компонент може розвиватися та тестуватися незалежно, що критично важливо для підтримки складної логіки планування повторень з чотирма різними режимами навчання. Модуль інтеграції з зовнішнім сервісом перекладу реалізовано як окремий клас, що дозволяє за потреби замінити DeepL на інший сервіс машинного перекладу без модифікації основної логіки застосунку.

Рівень представлення організовано через Django-шаблони, які динамічно генерують HTML-контент на основі даних, переданих з view-функцій. Це дозволяє забезпечити серверний рендеринг сторінок, що покращує індексацію пошуковими системами та зменшує навантаження на клієнтську частину. JavaScript

використовується вибірково — для асинхронних операцій, таких як автоматичний переклад при введенні слова, перевірка відповідей користувача без перезавантаження сторінки та динамічна зміна інтерфейсу в залежності від контексту.

Використання описаних архітектурних рішень дало можливість організувати компоненти системи ефективно, чітко визначивши межі відповідальності кожного рівня. Така архітектура забезпечує масштабованість системи — додавання нових режимів навчання або інтеграція додаткових сервісів машинного перекладу не вимагає значних змін у базовій структурі.

Окрім цього, модульна організація спрощує тестування окремих компонентів та локалізацію помилок, що особливо важливо для підтримки якості складного навчального застосунку.

3.2 Взаємодія користувача з системою

Визначення головних можливостей та типів користувачів системи є обов'язковим етапом перед реалізацією будь-якого програмного продукту.

Для вирішення даної задачі прийнято використовувати діаграму прецедентів, що дозволяє відобразити акторів, тобто користувачів, які зможуть взаємодіяти з системою, а також прецеденти — підтримувані дії, які кожен із визначених акторів зможе виконувати в програмному застосунку. За рахунок такого підходу можна чітко визначити функціональні вимоги до системи та межі відповідальності різних ролей користувачів.

У системі вивчення іншомовної лексики передбачені три типи акторів: гість, авторизований користувач та адміністратор. Між цими акторами існує зв'язок узагальнення, що дозволяє одному актору успадковувати властивості іншого.

Усі випадки використання базового актора стають доступними його нащадкам, які, своєю чергою, можуть мати власні, специфічні варіанти використання. У даній моделі базовим актором є гість, тоді як авторизований

користувач виступає його нащадком з розширеним набором функціональних можливостей.

Гість має обмежений доступ до функціоналу системи — він може переглядати загальну інформацію про проєкт на сторінці «Про систему», ознайомлюватися з описом алгоритму розподілених повторень на інформаційній сторінці «Алгоритм», виконувати реєстрацію нового облікового запису та авторизуватися в системі, використовуючи існуючі облікові дані.

Після успішної авторизації гість стає авторизованим користувачем і отримує повний доступ до можливостей програмного продукту, зокрема до створення словників, додавання навчальних карток, проведення навчальних сесій та перегляду статистики прогресу.

Тож на етапі планування розроблення системи вивчення іншомовної лексики було розроблено діаграму прецедентів, що дозволила відобразити ключових акторів системи, а також їх можливі варіанти взаємодії з нею. Результат зображений на рисунку 3.2.

З діаграми прецедентів видно ключові функціональні можливості системи та їх розподіл між типами користувачів. Рисунок також демонструє зв'язки включення між прецедентами, що відображають залежності між різними функціями системи. Наприклад, прецедент «Додавання слів групою» включає прецедент «Автоматичний переклад слова», оскільки переклад є невід'ємною частиною процесу додавання нових навчальних карток. Аналогічно, прецедент «Почати навчальну сесію» включає прецедент «Відібрати слова», що реалізується через алгоритм розподілених повторень.

Неавторизовані користувачі автоматично перенаправляються на сторінку входу при спробі доступу до захищених ресурсів.

Основний функціонал системи: створення словників, додавання слів з автоматичним перекладом і проведення навчальних сесій з адаптивним алгоритмом повторень. Це доступно виключно авторизованим користувачам.

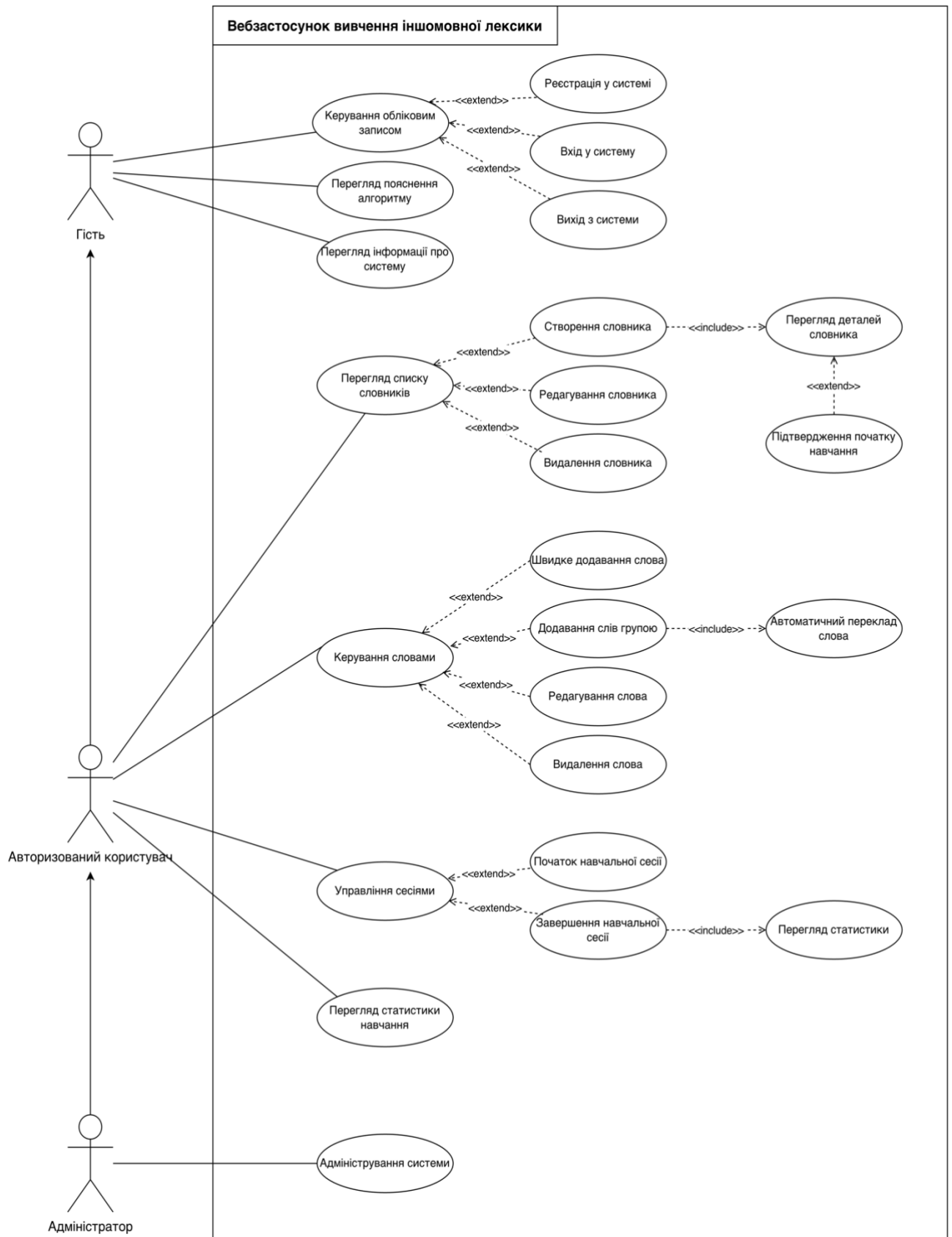


Рисунок 3.2 — Діаграма прецедентів системи вивчення іншомовної лексики

Така організація ролей забезпечує як захист персоналізованих навчальних матеріалів, так і можливість надання публічної інформації про систему потенційним користувачам без необхідності реєстрації.

3.3 Структура бази даних системи

На етапі проєктування системи було встановлено дані, з якими мають взаємодіяти розроблювані компоненти. До основних сутностей належать облікові записи користувачів, словники для групування навчальних карток, самі навчальні картки з параметрами алгоритму розподілених повторень, історичні записи спроб відповідей користувачів та метадані навчальних сесій.

Для збереження та швидкого доступу до постійних даних системи було прийнято рішення використовувати реляційну базу даних SQLite3, що забезпечує достатню продуктивність для роботи з персоналізованими навчальними матеріалами користувачів.

Для визначення властивостей та зв'язків між сутностями було побудовано логічну модель даних, яка відображає структуру збереження інформації та взаємозв'язки між різними типами об'єктів системи. Логічна модель даних зображена на рисунку 3.3.

Згідно з діаграмою, в системі виділено п'ять основних сутностей, між якими встановлено зв'язки типу один-до-багатьох для забезпечення цілісності даних (слово, словник, користувач, навчальна сесія та спроба відповіді).

Користувач може мати необмежену кількість словників, кожен словник містить багато навчальних карток, для кожної картки зберігається історія всіх спроб відповідей (рис. 3.3).

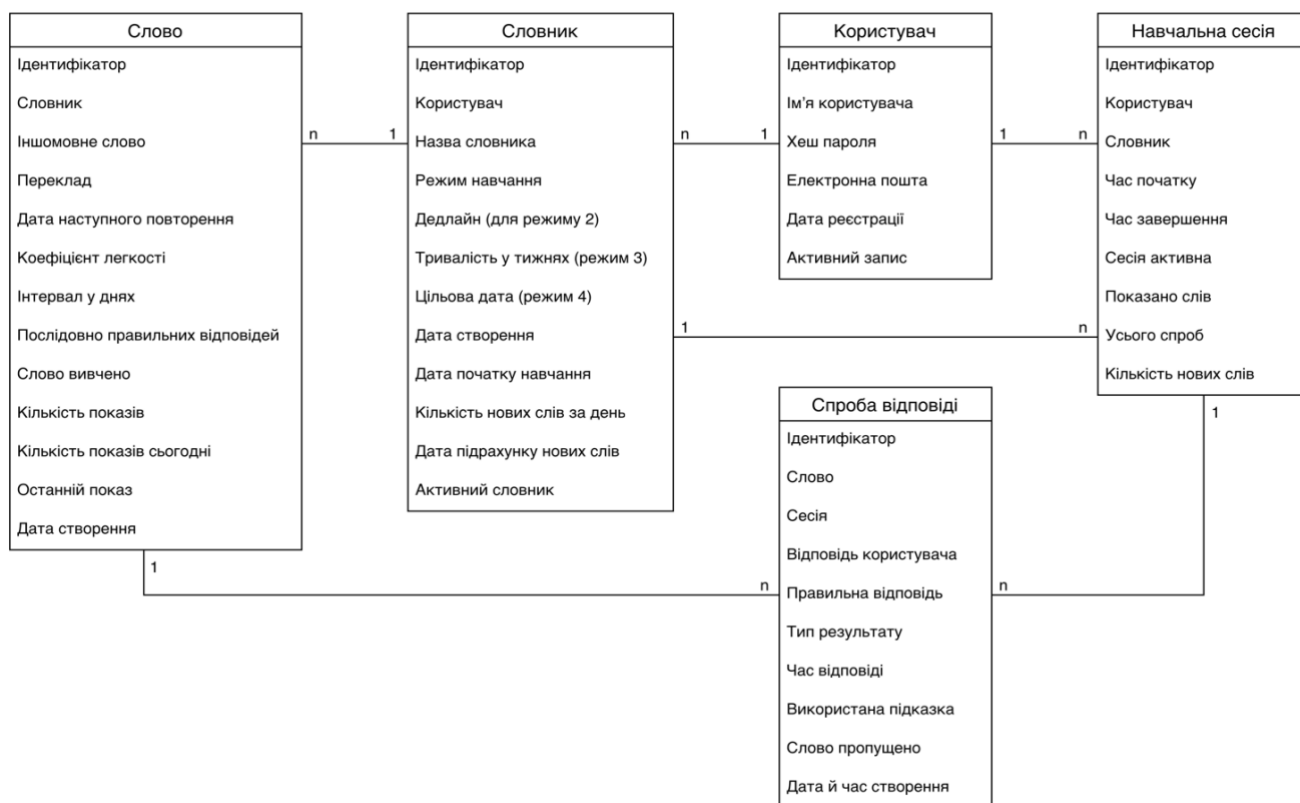


Рисунок 3.3 — Логічна модель даних предметної області

Навчальні сесії пов'язані як з користувачем, так і зі словником, що дозволяє аналізувати активність користувача в контексті конкретних навчальних матеріалів. Сутності представлені детальніше на рисунку 3.4.

Згідно з діаграмою, в системі виділено п'ять основних сутностей, між якими встановлено зв'язки типу один-до-багатьох для забезпечення цілісності даних. Центальною сутністю виступає User, від якої походять всі персоналізовані дані системи. Користувач може мати необмежену кількість словників Dictionary, кожен з яких містить багато навчальних карток Word. Для кожної картки система зберігає детальну історію взаємодій через записи WordAttempt, що дозволяє відстежувати прогрес та застосовувати адаптивний алгоритм повторень. Сутність LearningSession об'єднує спроби відповідей у межах одного навчального періоду для аналітики.

Структура даних побудована з дотриманням принципів нормалізації реляційних баз даних. Кожна сутність представляє окремий об'єкт предметної області з чітко визначеною відповідальністю. Зв'язки реалізовані через зовнішні ключі з каскадним видаленням.



Рисунок 3.4 — Діаграма сутностей

Розглянемо властивості кожної з таблиць детальніше, починаючи з центральної сутності системи — облікового запису користувача.

3.3.1 Таблиця User

Для зберігання облікових записів користувачів система використовує стандартну модель Django `django.contrib.auth.models.User`, що надає базовий функціонал автентифікації та авторизації. Таблиця містить такі ключові поля:

- `id` — числовий ідентифікатор, генерується автоматично, виступає первинним ключем;

- `username` — унікальне ім'я користувача для входу у систему, тип `VARCHAR`, обов'язкове для заповнення, до нього застосовується обмеження унікальності;

- `password` — хеш пароля, згенерований за алгоритмом PBKDF2, що забезпечує безпечне зберігання облікових даних без можливості відновлення оригінального пароля;

- `email` — електронна адреса користувача, тип `VARCHAR`;

- `date_joined` — дата та час реєстрації в системі, встановлюється автоматично при створенні запису;

- `is_active` — булевий прапорець активності облікового запису, дозволяє тимчасово блокувати доступ без видалення даних;

- додаткові службові поля включають `first_name`, `last_name` для збереження персональної інформації та `last_login` для відстеження останнього входу користувача.

3.3.2 Таблиця Dictionary

Таблиця `Dictionary` призначена для зберігання словників користувачів, що виступають контейнерами для групування навчальних карток за тематикою чи цільовим призначенням. Вона складається з наступних полів:

- `id` — первинний ключ, автоінкрементний ідентифікатор типу `INTEGER`;

— `user_id` — зовнішній ключ для зв'язку з таблицею `User`, визначає власника словника, при видаленні користувача всі його словники видаляються каскадно;

— `name` — назва словника, тип `VARCHAR` з обмеженням довжини 200 символів, обов'язкове поле;

— `learning_mode` — режим навчання, тип `INTEGER` з можливими значеннями 1, 2, 3, 4, що відповідають режимам «Назавжди», «До дати іспиту», «На кілька тижнів» та «Швидке запам'ятовування» відповідно;

— `deadline` — дата та час іспиту для режиму 2, тип `DATETIME`, опціональне поле;

— `weeks_duration` — тривалість навчання у тижнях для режиму 3, тип `INTEGER` з валідацією значень від 2 до 4;

— `target_date_mode4` — цільова дата завершення для режиму 4, тип `DATETIME`;

— `started_at` — мітка часу початку навчання словника, встановлюється при першій навчальній сесії;

— `new_words_shown_today` — лічильник нових слів, показаних сьогодні, використовується для обмеження 40 нових слів на день;

— `new_words_date` — дата для відстеження лічильника нових слів, дозволяє скидати лічильник щоденно;

— `is_active` — статус активності словника, за замовчуванням `TRUE`;

— `created_at` — дата створення запису, встановлюється автоматично.

3.3.3 Таблиця `Word`

Таблиця `Word` є центральною сутністю системи, що зберігає навчальні картки з усіма параметрами, необхідними для роботи алгоритму розподілених повторень. Структура таблиці включає такі властивості:

— `id` — унікальний ідентифікатор картки, автоінкрементний первинний ключ;

— `dictionary_id` — зовнішній ключ для зв'язку з таблицею `Dictionary`, визначає приналежність картки до конкретного словника;

— `foreign_word` — іншомовне слово або фраза, що підлягає вивченню, тип `VARCHAR` до 200 символів;

— `translation` — переклад слова українською мовою, отриманий від DeepL API або введений користувачем вручну;

— `next_review_date` — дата та час наступного планового показу картки, тип `DATETIME` з підтримкою `NULL` для позначення нових слів, на це поле накладено індекс для оптимізації вибірки слів за датою;

— `ease_factor` — коефіцієнт легкості слова, тип `FLOAT` з обмеженням значень від 1.3 до 2.5, за замовчуванням 2.0, використовується для розрахунку інтервалів повторення;

— `interval_days` — поточний інтервал повторення у днях, тип `INTEGER`, за замовчуванням 0 для нових слів;

— `consecutive_correct` — кількість правильних відповідей підряд, тип `INTEGER`, скидається до 0 при помилці;

— `is_learned` — булевий прапорець статусу вивченості, слова з `TRUE` не показуються у навчальних сесіях;

— `times_shown` — загальна кількість показів картки користувачу, тип `INTEGER`, інкрементується при кожному показі;

— `times_shown_today` — кількість показів за поточний день, використовується для обмеження 4-5 показів на день;

— `times_correct` — загальна кількість правильних відповідей типу «Добре», статистичний показник;

— `last_shown_at` — мітка часу останнього показу картки;

— `last_shown_date` — дата останнього показу, використовується для скидання денних лічильників;

— `created_at` — дата додавання картки до словника.

Для оптимізації продуктивності на поля `next_review_date` та комбінацію (`dictionary_id`, `next_review_date`) накладено індекси, що прискорює вибірку слів для навчальних сесій. Обмеження для поля `ease_factor` забезпечує коректність обчислень алгоритму розподілених повторень, не дозволяючи коефіцієнту

виходити за межі діапазону, визначеного оригінальним алгоритмом SuperMemo SM-2.

3.3.4 Таблиця WordAttempt

Таблиця WordAttempt призначена для збереження детальної історії всіх спроб відповідей користувача на навчальні картки, що дозволяє аналізувати прогрес навчання та коригувати алгоритм повторень. Структура таблиці така:

- `id` — унікальний ідентифікатор спроби;
- `word_id` — зовнішній ключ для зв'язку з таблицею Word, вказує на картку, для якої здійснено спробу;
- `session_id` — ідентифікатор навчальної сесії, тип VARCHAR до 100 символів, індексоване поле для швидкого групування спроб за сесіями;
- `user_answer` — відповідь користувача, тип TEXT для підтримки довгих фраз;
- `is_correct` — булевий прапорець правильності відповіді після нормалізованого порівняння;
- `result_type` — класифікація результату, тип VARCHAR з можливими значеннями "good" (добре), "hard" (важко), "error" (помилка), визначає вплив на параметри алгоритму;
- `response_time` — час відповіді у секундах, тип FLOAT, використовується для статистики;
- `used_hint` — булевий прапорець використання підказки, за замовчуванням FALSE;
- `was_skipped` — булевий прапорець пропуску слова, за замовчуванням FALSE;
- `created_at` — мітка часу здійснення спроби, індексоване поле для сортування історії.

Між таблицями WordAttempt та Word встановлено зв'язок один-до-багатьох з каскадним видаленням, що забезпечує автоматичне видалення всіх спроб при

видаленні картки. Композитний індекс на полях (word_id, created_at) оптимізує запити історії спроб для конкретної картки з сортуванням за часом.

3.3.5 Таблиця LearningSession

Таблиця LearningSession об'єднує спроби відповідей у межах одного навчального періоду для аналітики та відстеження активності користувача. Поля таблиці:

- id — первинний ключ сесії;
- user_id — зовнішній ключ для зв'язку з таблицею User;
- dictionary_id — зовнішній ключ для зв'язку з таблицею Dictionary, визначає словник, що вивчається;
- session_id — унікальний ідентифікатор сесії, тип VARCHAR, на поле накладено обмеження унікальності та індекс;
- started_at — мітка часу початку сесії;
- ended_at — мітка часу завершення сесії, може бути NULL для активних сесій;
- is_active — булевий прапорець активності сесії, за замовчуванням TRUE;
- words_shown — кількість показаних слів у сесії;
- total_attempts — загальна кількість спроб відповідей;
- new_words_count — кількість нових слів, введених у навчання протягом сесії.

За даним принципом були побудовані всі таблиці системи, забезпечуючи нормалізовану структуру бази даних з підтримкою цілісності через зовнішні ключі та каскадні операції. Встановлені індекси на критичних полях дозволяють ефективно виконувати складні запити для вибірки слів на навчання з урахуванням множини критеріїв: дати наступного показу, режиму навчання словника, денних лімітів та прострочених слів. Така організація даних забезпечує як швидкий доступ до навчальних матеріалів, так і можливість детального аналізу прогресу користувача через історичні записи спроб.

3.4 Структура класів вебзастосунку

Архітектура вебзастосунку відображає верхній рівень абстракції, проте окрім нього в системі існує нижній рівень, що складається з конкретних класів Python та Django, які реалізують функціональність системи.

Для компонування та виділення компонентів на низькому рівні було використано діаграму класів. Нею називають схематичне представлення, на якому зображені структурні елементи — класи, що використовуються в програмному забезпеченні, їхні атрибути, методи та взаємозв'язки.

Для кращого представлення класів на нижньому рівні було побудовано діаграму, що зображена на рисунку 3.5.

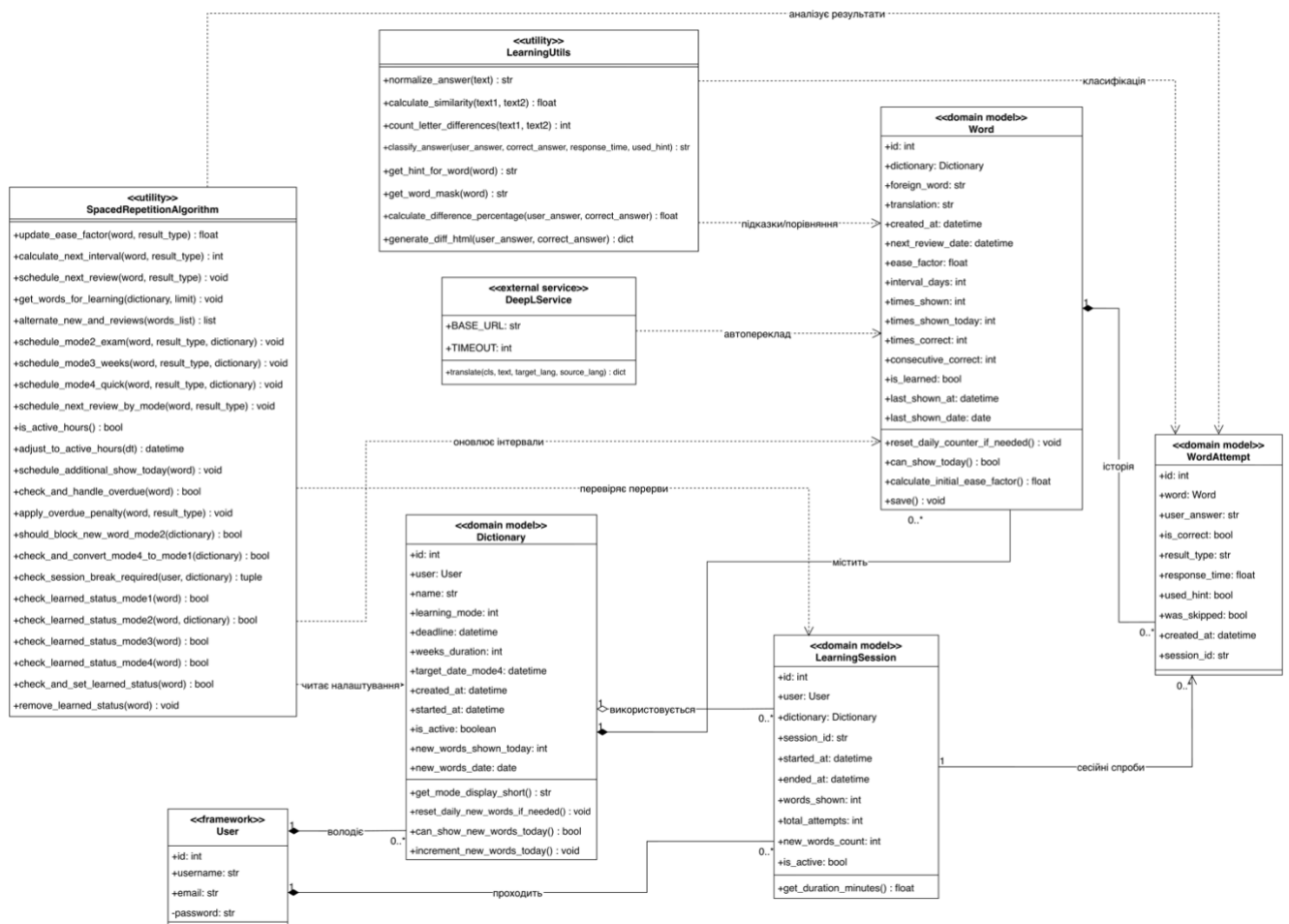


Рисунок 3.5 — Діаграма класів системи вивчення іншомовної лексики

З діаграми 3.5 видно ключові класи системи, їхні атрибути, методи та взаємозв'язки. Центральними сутностями виступають моделі Django, що представляють об'єкти предметної області та інкапсулюють логіку роботи з даними. Навколо моделі Dictionary будується основна логіка системи — від неї залежать навчальні картки Word, які, у свою чергу, пов'язані з історичними записами WordAttempt. Модель User виступає кореневою сутністю, від якої залежать як словники, так і навчальні сесії.

Діаграма класів відображає чітку ієрархію та організацію системи вивчення іншомовної лексики, де кожен клас виконує свою специфічну функцію, взаємодіючи з іншими компонентами. Моделі інкапсулюють логіку роботи з даними та складні обчислення параметрів розподілених повторень. Форми забезпечують надійну валідацію вхідних даних. Сервісні класи ізолюють зовнішні залежності, що робить систему стійкою до змін зовнішніх API.

3.5 Модуль реєстрації та авторизації користувачів

Захист системи реалізований з використанням вбудованих механізмів фреймворка Django, що надає перевірені та безпечні методи автентифікації користувачів. Модуль авторизації відіграє ключову роль у безпеці системи, забезпечуючи розмежування доступу між різними користувачами та захист персоналізованих навчальних матеріалів від несанкціонованого доступу. Для забезпечення процесу автентифікації було використано стандартну систему Django Authentication, що включає безпечне хешування паролів за алгоритмом PBKDF2 та керування користувацькими сесіями через session middleware.

Процес автентифікації ініціюється користувачем, який вводить свої облікові дані на сторінці входу в систему. Клієнтська частина надсилає POST-запит до серверної частини з іменем користувача та паролем. Сервер отримує запит та здійснює перевірку валідності наданої інформації, звертаючись до бази даних для пошуку відповідного облікового запису. Django використовує функцію

authenticate(), яка порівнює хеш введеного пароля з хешем, збереженим у базі даних, без можливості отримання оригінального пароля.

Якщо перевірка пройшла успішно, то користувач отримує токен сесії, який зберігається у браузері у вигляді cookie та використовується для ідентифікації у подальших запитах. Система викликає функцію auth_login(), що створює запис сесії в базі даних та встановлює cookie з ідентифікатором сесії у відповіді браузеру. Після успішної автентифікації користувач перенаправляється на головну сторінку застосунку, де йому стає доступний повний функціонал системи вивчення лексики. Механізм авторизації зображено на діаграмі послідовності, що представлена на рисунку 3.6.

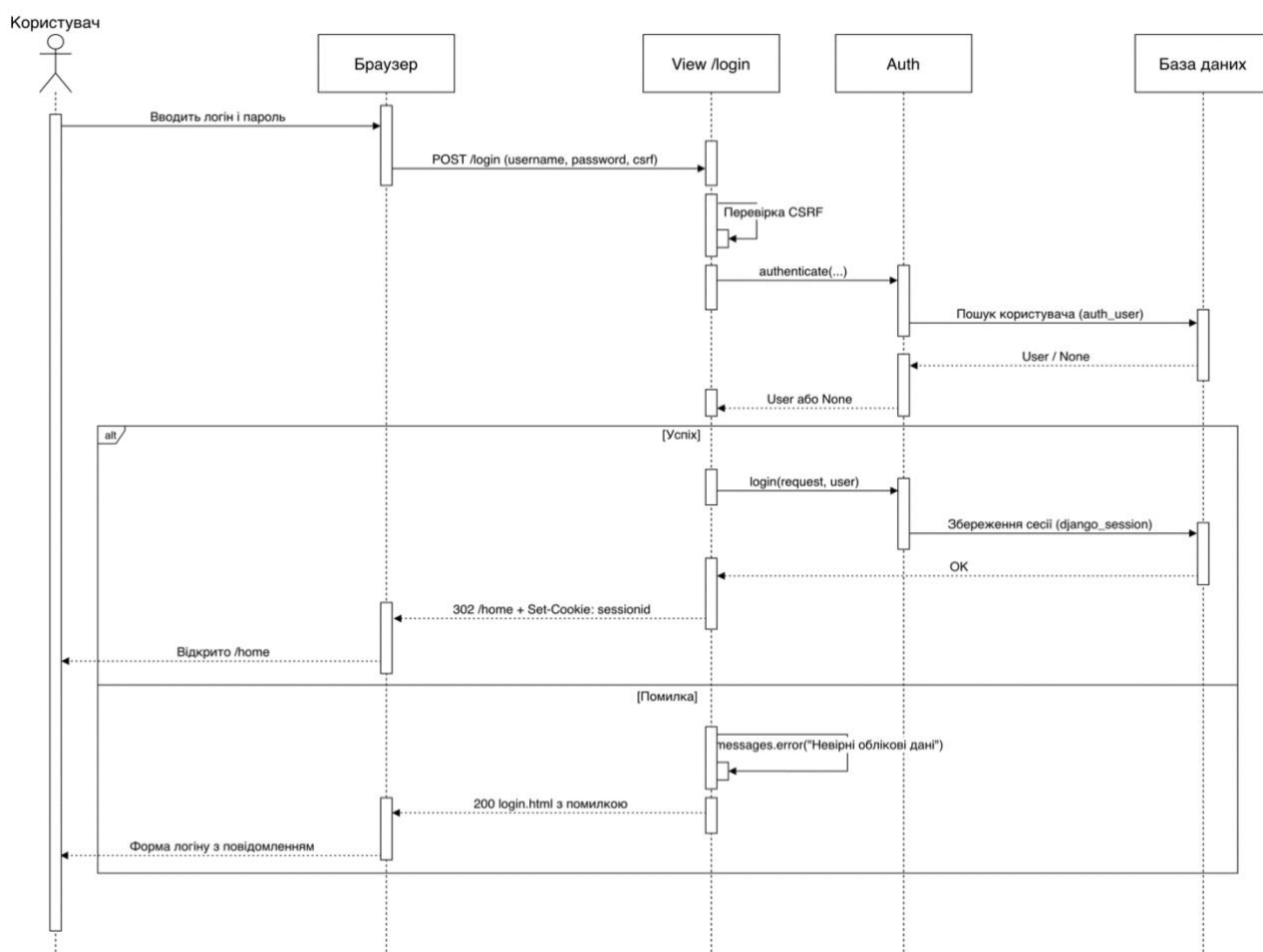


Рисунок 3.6 — Діаграма послідовності процесу авторизації користувача

Діаграма демонструє послідовність взаємодії компонентів системи під час авторизації користувача. Спочатку браузер надсилає облікові дані на сервер, після чого Django викликає систему автентифікації для перевірки коректності даних. Система автентифікації звертається до бази даних, витягує запис користувача та порівнює хеші паролів. У випадку співпадіння створюється запис сесії, а браузер отримує cookie з ідентифікатором, що дозволяє підтримувати автентифікований стан користувача протягом усього часу роботи з системою.

У разі невдалої автентифікації — коли ім'я користувача не знайдено в базі даних або хеш пароля не співпадає — функція `authenticate()` повертає значення `None`. Система обробляє цей випадок, відображаючи повідомлення про помилку через механізм Django messages framework та повторно показуючи форму входу користувачеві. Користувач має можливість повторити спробу входу з коректними обліковими даними або перейти на сторінку реєстрації для створення нового облікового запису.

Окрім процесу входу, модуль забезпечує функціональність реєстрації нових користувачів та виходу з системи. При реєстрації система перевіряє унікальність обраного імені користувача через запит до бази даних. Якщо користувач з таким іменем вже існує, система повертає відповідне повідомлення про помилку.

При успішній реєстрації створюється новий запис у таблиці `User` з хешованим паролем, після чого користувач автоматично авторизується в системі та перенаправляється на сторінку успішної реєстрації. Процес виходу реалізовано через функцію `auth_logout()`, що видаляє запис сесії з бази даних та очищає cookie у браузері, завершуючи автентифікований сеанс роботи користувача.

Додатковий рівень захисту забезпечується через використання декоратора `@login_required`, який застосовується до всіх view-функцій, що потребують автентифікації. Декоратор перевіряє наявність активної сесії користувача перед виконанням функції, перенаправляючи неавтентифікованих користувачів на сторінку входу. Це гарантує, що доступ до персоналізованих навчальних матеріалів — словників, карток, статистики — можливий лише для автентифікованих користувачів.

Окрім цього, всі view-функції додатково перевіряють власника ресурсу через фільтрацію за `user=request.user`, що запобігає доступу одного користувача до даних іншого навіть при знанні ідентифікаторів ресурсів.

3.6 Модуль інтеграції з DeepL API

Для забезпечення функціональності машинного перекладу було інтегровано зовнішній сервіс DeepL Translator, що являє собою нейронну систему машинного перекладу, побудовану на архітектурі Transformer з механізмами уваги. Модуль відповідає за взаємодію з RESTful API сервісу DeepL, що дозволяє виконувати переклад текстів між широким спектром мовних пар з високою якістю, наближеною до людського перекладу. Інтеграція реалізована з використанням бібліотеки `requests` для виконання HTTP-запитів та інкапсульована в окремий клас `DeepLService`, що забезпечує модульність та можливість легкої заміни сервісу перекладу у майбутньому.

Модуль викликається у контексті додавання нових слів до словника, коли користувач вводить іншомовний текст у форму. JavaScript на клієнтській стороні відстежує введення тексту та автоматично ініціює асинхронний запит до серверної частини для отримання перекладу. View-функція `translate_word_ajax()` обробляє такі запити, викликаючи метод `DeepLService.translate()` з текстом для перекладу та цільовою мовою. Система завжди використовує українську мову як цільову (код UK), тоді як вихідна мова визначається автоматично сервісом DeepL на основі аналізу вхідного тексту.

Взаємодія з API DeepL відбувається за допомогою HTTP-запитів методом POST до кінцевої точки `https://api-free.deepl.com/v2/translate`. Для автентифікації використовується API-ключ, який передається у параметрах запиту разом з текстом для перекладу та кодом цільової мови. Система використовує тарифний план DeepL API Free, що надає обмежену квоту символів для перекладу та обмеження

на кількість запитів за одиницю часу. Структура параметрів запиту до API представлена в таблиці 3.1.

Таблиця 3.1 — Параметри запиту до DeepL API

Параметр	Тип даних	Опис	Обов'язковий
auth_key	string	API-ключ для автентифікації у DeepL	Так
text	string	Текст для перекладу	Так
target_lang	string	Код цільової мови	Так
source_lang	string	Код вихідної мови	Ні

Після надсилання запиту модуль очікує відповідь від сервера DeepL протягом встановленого терміну, який становить 10 секунд. Отриманий від сервера результат має формат JSON та містить масив об'єктів перекладів. Модуль витягує перекладений текст з поля `translations[0].text` JSON-відповіді та інформацію про автоматично визначену вихідну мову з поля `translations[0].detected_source_language`. Ці дані обробляються модулем, упаковуються у словник Python з ключами `success`, `translation` та `detected_language`, після чого повертаються `view`-функції для подальшої передачі клієнтській частині у форматі JSON-відповіді.

У випадку виникнення помилок під час взаємодії з API модуль реалізує детальну обробку різних типів помилок. При отриманні HTTP-статусу 403 система ідентифікує невірний API-ключ та логує відповідне попередження. Статус 429 вказує на перевищення ліміту запитів, а статус 456 сигналізує про вичерпання квоти символів для перекладу.

Окрім HTTP-помилки, модуль обробляє винятки мережевого рівня — таймаути з'єднання через клас `requests.exceptions.Timeout` та помилки з'єднання через `requests.exceptions.ConnectionError`. При виникненні будь-якої помилки модуль повертає словник з прапорцем `success=False` та описовим повідомленням про тип помилки, що дозволяє `view`-функції інформувати користувача про проблему та запропонувати ввести переклад вручну.

Оброблення відповідей включає валідацію отриманих даних та їх перетворення у внутрішній формат системи. View-функція `translate_word_ajax()` перевіряє поле `success` у відповіді від `DeepLService` та формує JSON-відповідь для клієнта. При успішному перекладі JavaScript автоматично заповнює поле «Переклад» у формі додавання слів, пропонуючи користувачеві готовий варіант перекладу.

Користувач має можливість прийняти запропонований переклад або відредагувати його вручну перед збереженням картки до словника. Така організація взаємодії забезпечує баланс між автоматизацією процесу та контролем користувача над якістю навчальних матеріалів.

Реалізована інтеграція забезпечує надійне та швидке виконання перекладу текстів, мінімізуючи ручну роботу користувача при створенні навчальних карток. Модульна структура класу `DeepLService` дозволяє легко модифікувати параметри запитів, додавати обробку додаткових типів помилок або замінювати сервіс `DeepL` на альтернативний сервіс машинного перекладу без необхідності змін у інших частинах системи.

3.7 Модуль реалізації алгоритму розподілених повторень

Ключовою складовою системи вивчення лексики є алгоритм розподілених повторень, що визначає оптимальні інтервали між показами навчальних карток на основі індивідуальних характеристик запам'ятовування кожного слова користувачем. Для реалізації механізму повторень було обрано модифікований алгоритм `SuperMemo SM-2`, адаптований для підтримки чотирьох різних режимів навчання з різними стратегіями планування. Алгоритм визначає дату наступного показу слова на основі результату попередньої відповіді користувача, історії взаємодії з картою та параметрів обраного режиму навчання словника.

Алгоритм працює наступним чином: при кожній відповіді користувача на слово система класифікує результат як «Добре», «Важко» або «Помилка» на основі

порівняння введеної відповіді з еталонним перекладом та часу відповіді. Спочатку система оновлює коефіцієнт легкості слова — при результаті «Добре» коефіцієнт збільшується на 0.20, при результаті «Важко» зменшується на 0.15, при результаті «Помилка» зменшується на 0.30. Коефіцієнт обмежується діапазоном від 1.3 до 2.5, що забезпечує стабільність обчислень та відповідає рекомендаціям оригінального алгоритму SuperMemo SM-2. Після оновлення коефіцієнта система модифікує лічильник послідовних правильних відповідей — інкрементує його при результатах «Добре» або «Важко» та скидає до нуля при результаті «Помилка».

Розрахунок інтервалу до наступного показу залежить від режиму навчання словника. Для режиму «Назавжди», призначеного для довготривалого запам'ятовування, система використовує базовий графік: перша правильна відповідь призначає показ на наступний день, друга правильна відповідь призначає показ через шість днів, подальші інтервали розраховуються множенням попереднього інтервалу на коефіцієнт легкості з округленням до цілого числа днів. Якщо користувач відповів правильно, то інтервал зростає експоненціально, дозволяючи добре засвоєним словам показуватися все рідше, що оптимізує розподіл навчального навантаження.

Для режиму «До дати іспиту» алгоритм адаптує графік повторень під конкретну дату екзамену або перевірки знань. Система визначає контрольні точки — за сім днів до іспиту та за один день до іспиту, якщо часу достатньо, або за три дні та за один день для коротших періодів. Слова плануються таким чином, щоб обов'язково показатися у ці контрольні точки, забезпечуючи підкріплення запам'ятовування перед важливою подією. Між контрольними точками інтервали розраховуються випадково в діапазоні від трьох до п'яти днів, не перевищуючи відстань до наступної контрольної точки. Особливістю режиму є заборона показу нових слів за три дні до іспиту, що дозволяє сконцентруватися на повторенні вже вивчених матеріалів.

Режим «На кілька тижнів» реалізує фіксований ритм повторень для навчання протягом обмеженого періоду від двох до чотирьох тижнів. Після двох правильних відповідей слово переходить на тижневий ритм показів. За два тижні до завершення

періоду призначається перший фінальний показ, а за один-два дні до завершення — другий фінальний показ, що забезпечує закріплення матеріалу перед завершенням навчання. Такий підхід оптимізований для вивчення лексики у контексті короткострокових курсів або тематичних інтенсивів.

Режим «Швидке запам'ятовування» призначений для дуже коротких періодів від одного до семи днів та реалізує інтенсивний графік показів. Слова показуються наступного дня після першої правильної відповіді, потім на третій або четвертий день, і обов'язково у день завершення або за день до нього. Після досягнення цільової дати словник автоматично конвертується у режим «Назавжди» для підтримки довготривалого запам'ятовування вивчених слів. Для детального та наочного представлення цього механізму було створено блок-схему алгоритму розрахунку інтервалу, що зображена на рисунку 3.7.

Згідно з блок-схемою, процес обробки відповіді включає послідовність чітко визначених кроків, кожен з яких модифікує параметри навчальної картки. Після класифікації результату та оновлення коефіцієнта легкості алгоритм розгалужується залежно від режиму навчання, застосовуючи відповідну стратегію планування. Усі режими сходяться до загального фінального етапу — встановлення дати наступного показу та збереження змін у базі даних.

Програмна реалізація алгоритму виконана у вигляді набору функцій в окремому модулі `spaced_repetition.py`, що забезпечує ізоляцію складної логіки планування від інших частин системи. Основною точкою входу є функція `schedule_next_review_by_mode()`, яка виступає диспетчером для вибору режим-специфічної функції планування на основі значення поля `learning_mode` словника. Функція приймає об'єкт навчальної картки, тип результату відповіді та об'єкт HTTP-запиту для доступу до параметрів часу, після чого викликає відповідну функцію — `schedule_next_review()` для режиму 1, `schedule_mode2_exam()` для режиму 2, `schedule_mode3_weeks()` для режиму 3 або `schedule_mode4_quick()` для режиму 4.

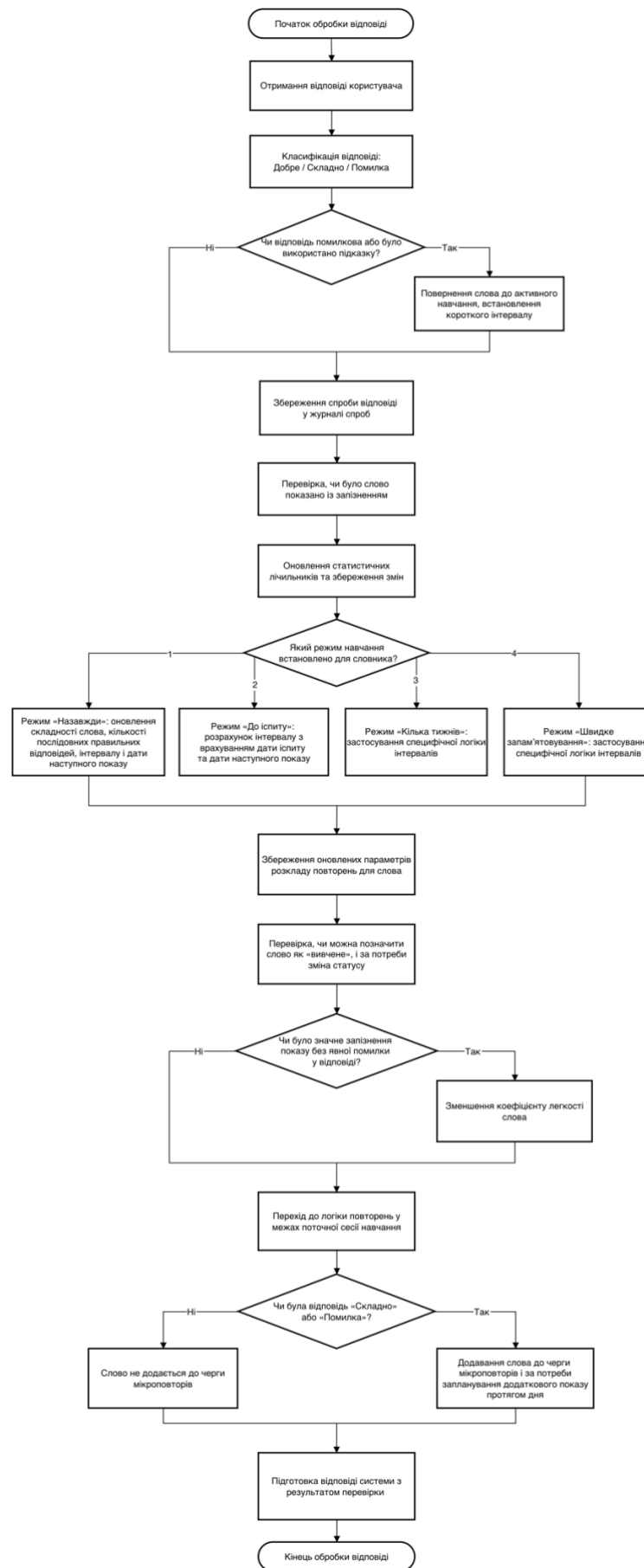


Рисунок 3.7 — Блок-схема алгоритму розподілених повторень

Кожна режим-специфічна функція реалізує свою логіку розрахунку інтервалів, але всі вони виконують спільний набір операцій — оновлюють коефіцієнт легкості через функцію `update_ease_factor()`, розраховують новий інтервал відповідно до правил режиму, коригують час показу на активні години (з 08:00 до 22:00) через функцію `adjust_to_active_hours()`, оновлюють статистичні поля картки та перевіряють виконання критеріїв статусу «вивчено» через функцію `check_and_set_learned_status()`. Усі зміни зберігаються в базі даних через виклик методу `save()` моделі `Word` з явним зазначенням оновлюваних полів для оптимізації продуктивності.

Окрім функцій планування, модуль містить функцію `get_words_for_learning()`, що відповідає за відбір слів для навчальної сесії. Функція виконує складну вибірку з бази даних, застосовуючи множину критеріїв — фільтрацію невивчених слів, групування за терміновістю (прострочені, на сьогодні, нові), застосування денних лімітів показів, обмеження нових слів до 40 за сесію, перемішування слів всередині груп для уникнення передбачуваних паттернів. Відібрані слова зберігаються у сесії Django для підтримки стану навчального процесу між запитами.

Реалізований алгоритм забезпечує ефективне запам'ятовування лексики за рахунок оптимізації інтервалів повторення відповідно до індивідуальних характеристик запам'ятовування кожного слова користувачем.

Адаптація класичного підходу SuperMemo SM-2 для підтримки множинних режимів навчання дозволяє системі задовольняти різні освітні сценарії — від довготривалого вивчення мови до підготовки до конкретного іспиту.

Модульна організація коду забезпечує можливість тестування окремих функцій та легкого додавання нових режимів навчання у майбутньому без модифікації існуючої логіки.

Завдяки цьому всі елементи системи узгоджено працюють під час формування розкладу повторень.

3.8 Модуль керування словниками та навчальними картками

Модуль забезпечує користувачам можливість керувати власними словниками та навчальними картками, реалізуючи повний спектр операцій для організації навчальних матеріалів. Основна функціональність модуля включає створення словників з різними режимами навчання, додавання навчальних карток з автоматичним перекладом, редагування існуючих записів, видалення непотрібних матеріалів та організацію карток для ефективного доступу. Для зручності користувачів реалізовано можливості пошуку карток за текстом слова або перекладу, фільтрації за статусом вивченості та сортування за різними критеріями.

Створення словника починається з заповнення форми, де користувач вводить назву словника та обирає один з чотирьох режимів навчання. Залежно від обраного режиму форма динамічно показує додаткові поля — для режиму «До дати іспиту» з'являється поле вибору дати та часу екзамену, для режиму «На кілька тижнів» — поле вибору тривалості від двох до чотирьох тижнів, для режиму «Швидке запам'ятовування» — поле цільової дати завершення. Валідація форми забезпечує обов'язковість заповнення режим-специфічних параметрів та коректність введених дат. Після успішного створення словника користувач може одразу перейти до додавання слів або розпочати навчання пізніше.

Додавання слів до словника реалізовано через `formset` — структуру, що дозволяє вводити до двадцяти слів одночасно на одній сторінці. Користувач вводить іншомовне слово у перше поле рядка, після чого JavaScript автоматично ініціює запит до модуля інтеграції з DeepL API. Отриманий переклад заповнює друге поле рядка, надаючи користувачеві готовий варіант перекладу. Користувач має можливість прийняти запропонований переклад без змін або відредагувати його для досягнення більшої точності чи врахування контексту вживання слова. Після заповнення необхідної кількості рядків користувач зберігає `formset`, при цьому система перевіряє кожен рядок на наявність дублікатів у словнику через порівняння іншомовного слова та перекладу без урахування регістру.

При створенні нової навчальної картки система автоматично розраховує початковий коефіцієнт легкості на основі довжини слова. Короткі слова отримують вищий коефіцієнт 2.3, що відображає їх потенційно більшу легкість запам'ятовування, тоді як довгі складні слова отримують нижчий коефіцієнт 1.7. Слова середньої довжини отримують стандартний коефіцієнт 2.0. Окрім коефіцієнта легкості, всі параметри алгоритму розподілених повторень ініціалізуються значеннями за замовчуванням — `next_review_date` встановлюється у `None` для позначення нового слова, `interval_days` у нуль, `consecutive_correct` у нуль, всі статистичні лічильники обнуляються. Картка стає доступною для вивчення після збереження у базу даних.

Модуль тісно пов'язаний з модулем інтеграції DeepL API для автоматичного отримання перекладів при додаванні карток. Виклик модуля перекладу відбувається асинхронно через AJAX-запит до view-функції `translate_word_ajax()`, яка валідує доступ користувача до словника та викликає метод `DeepLService.translate()`. Отриманий переклад повертається клієнту у форматі JSON та автоматично заповнює відповідне поле формсету. У разі помилок перекладу — недоступності API, вичерпання квоти, мережових проблем — користувач отримує повідомлення про необхідність ввести переклад вручну, що забезпечує можливість продовження роботи навіть при тимчасових проблемах із зовнішнім сервісом.

Створені картки передаються модулю розподілених повторень для організації навчального процесу. Зв'язок реалізовано через спільне використання моделі `Word` — модуль керування створює та модифікує записи карток, встановлюючи їх початкові параметри, тоді як модуль алгоритму розподілених повторень читає ці параметри для відбору карток на навчання та оновлює їх після кожної відповіді користувача. Функція `get_words_for_learning()` з модуля `SRS` фільтрує картки словника за критеріями дати наступного показу, статусу вивченості, денних лімітів та режиму навчання, формуючи персоналізовану навчальну сесію. Після відповідей користувача функція `schedule_next_review_by_mode()` оновлює всі параметри картки, після чого вони стають доступними для наступних вибірок з урахуванням нового стану.

На сторінці деталей словника користувач має доступ до повного переліку карток з можливістю їх фільтрації та пошуку. Пошук виконується за іншомовним словом або перекладом через Django ORM з використанням Q об'єктів для комбінування умов. Фільтрація за статусом дозволяє показувати лише вивчені картки, лише ті що на вивченні, лише нові або всі разом. Сортювання доступне за алфавітом іншомовного слова, за прогресом вивчення (кількість правильних відповідей) або за датою додавання до словника. Для кожної картки відображається статистична інформація — кількість показів, кількість правильних відповідей, дата наступного планового показу, що дозволяє користувачеві відстежувати прогрес вивчення кожного слова окремо.

Модуль підтримує редагування існуючих карток через окрему форму, доступну зі сторінки деталей словника. Користувач може змінити іншомовний текст або переклад для виправлення помилок чи уточнення значень. Параметри алгоритму розподілених повторень не редагуються вручну, оскільки вони розраховуються автоматично на основі історії відповідей та можуть відображатися лише для інформації. Видалення картки вимагає підтвердження через окрему сторінку для запобігання випадковим діям. При видаленні картки автоматично видаляються всі пов'язані записи спроб відповідей через каскадне видалення, визначене зовнішнім ключем у таблиці WordAttempt.

Статистична інформація про словник агрегується та відображається на головній сторінці та сторінці деталей словника. Система розраховує загальну кількість карток, кількість вивчених карток, кількість карток на вивченні, кількість нових карток та кількість прострочених карток, які мали бути показані раніше але не були переглянуті користувачем. Окрім цього, розраховується середній коефіцієнт легкості всіх карток словника та прогрес у відсотках як співвідношення вивчених карток до загальної кількості. Ці дані надають користувачеві комплексне розуміння стану вивчення словника та мотивують до продовження навчання.

Реалізований функціонал дозволяє користувачам ефективно організовувати власні навчальні матеріали з мінімальними витратами часу на рутинні операції. Інтеграція автоматичного перекладу значно прискорює процес створення карток,

тоді як гнучкі можливості пошуку та фільтрації забезпечують швидкий доступ до потрібних карток у великих словниках. Модуль забезпечує зручне керування словниками та картками, що є основою для подальшого ефективного вивчення лексики через систему розподілених повторень з адаптивним алгоритмом планування.

Висновки до розділу 3

У третьому розділі роботи представлено детальний опис програмної реалізації вебзастосунку для вивчення іншомовної лексики з інтегрованим машинним перекладом та алгоритмом розподілених повторень. Розроблена система базується на багаторівневій клієнт-серверній архітектурі з використанням фреймворка Django, що забезпечує чітке розділення відповідальностей між компонентами та спрощує подальшу підтримку і розширення функціоналу.

Структура бази даних системи організована як нормалізована реляційна модель з сутностями, між якими встановлено зв'язки типу один-до-багатьох для забезпечення цілісності даних. Застосування індексів на критичних полях дозволяє ефективно виконувати складні запити для вибірки слів на навчання з урахуванням множини критеріїв.

Модуль автентифікації та авторизації використовує вбудовані механізми Django для забезпечення безпечного доступу користувачів до персоналізованих навчальних матеріалів.

Інтеграція з DeepL Translator API забезпечує автоматичний переклад іншомовних слів при створенні навчальних карток, значно прискорюючи процес наповнення словників. Модуль реалізовано як окремий клас.

Реалізація алгоритму розподілених повторень базується на адаптації SuperMemo SM-2 з підтримкою чотирьох різних режимів навчання. Кожен режим використовує специфічну стратегію планування повторень, адаптовану до конкретних освітніх сценаріїв — від довготривалого вивчення мови до підготовки

до іспиту. Модульна організація коду алгоритму забезпечує можливість тестування та легкого розширення функціоналу.

Модуль керування словниками та навчальними картками надає повний спектр операцій для організації навчальних матеріалів з інтуїтивним інтерфейсом та потужними можливостями пошуку і фільтрації. Взаємодія всіх модулів системи забезпечує цілісний користувацький досвід та ефективне вивчення іншомовної лексики.

4 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ

Робота користувача з вебзастосунком складається з декількох послідовних кроків, необхідних для забезпечення ефективного вивчення іншомовної лексики.

У даному розділі представлено базовий сценарій роботи з системою: реєстрацію та авторизацію користувача, створення словників з різними режимами навчання, додавання слів з використанням автоматичного перекладу через DeepL API, проведення навчальних сесій за алгоритмом розподілених повторень та моніторинг прогресу вивчення.

4.1 Вимоги до програмного забезпечення та інсталяція

Розроблена система є вебзастосунком, побудованим на базі фреймворку Django та призначеним для вивчення іншомовної лексики за методом розподілених повторень. Для коректної роботи програмного продукту необхідно забезпечити відповідність апаратних та програмних характеристик робочого середовища встановленим вимогам.

Мінімальні апаратні вимоги включають наявність сучасного процесора архітектури x86-64 або ARM, не менше 2 ГБ оперативної пам'яті та приблизно 200 МБ вільного дискового простору для встановлення системи і залежностей. Рекомендований обсяг оперативної пам'яті становить 4 ГБ для комфортної роботи з декількома словниками одночасно.

Програмні вимоги до системи представлені в таблиці 4.1. Система підтримує роботу на операційних системах Windows 10/11, macOS версії 10.15 або новішої, а також Linux-дистрибутивах на базі Ubuntu 20.04 і вище.

Таблиця 4.1 — Системні вимоги для роботи з програмним продуктом

Компонент	Вимога
Операційна система	Windows 10/11, macOS 10.15+, Linux (Ubuntu 20.04+)
Веб-браузер	Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
Оперативна пам'ять	мінімум 2 ГБ, рекомендовано 4 ГБ
Дисковий простір	мінімум 200 МБ
Інтернет-з'єднання	необхідне для DeepL API
Компоненти, необхідні за умови відсутності розгортання застосунку на сервері	Python 3.10 або вища, Django 5.1.5, SQLite3.

База даних реалізована засобами SQLite3, що є вбудованим рішенням і не потребує окремої інсталяції серверу СУБД. Для взаємодії з системою користувачеві необхідний сучасний веб-браузер останніх версій — інтерфейс оптимізовано для роботи в Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. Важливою умовою є наявність стабільного інтернет-з'єднання, оскільки система інтегрована з DeepL Translator API для надання автоматичних підказок перекладу під час додавання нових слів до словників.

У випадку, якщо система не буде розгорнута на сервері (відсутня можливість використання без інсталяції), необхідно покроково встановити потрібні технології. Процес інсталяції системи починається зі встановлення необхідних залежностей Python. У кореневій директорії проєкту міститься файл requirements.txt, який містить перелік усіх необхідних бібліотек. Встановлення виконується командою `pip install -r requirements.txt` у терміналі операційної системи. Після успішної установки залежностей необхідно застосувати міграції бази даних командою `python manage.py migrate`, що створить структуру таблиць SQLite3.

Для запуску розробницького сервера Django використовується команда `python manage.py runserver`. У разі успішного розгортання в терміналі з'явиться повідомлення про запуск сервера за адресою `http://127.0.0.1:8000/`. Після переходу за цією адресою у веб-браузері користувач потрапляє на головну сторінку системи, що свідчить про готовність застосунку до використання. На рисунку 4.1 зображено інтерфейс головної інформаційної сторінки для неавторизованих користувачів.

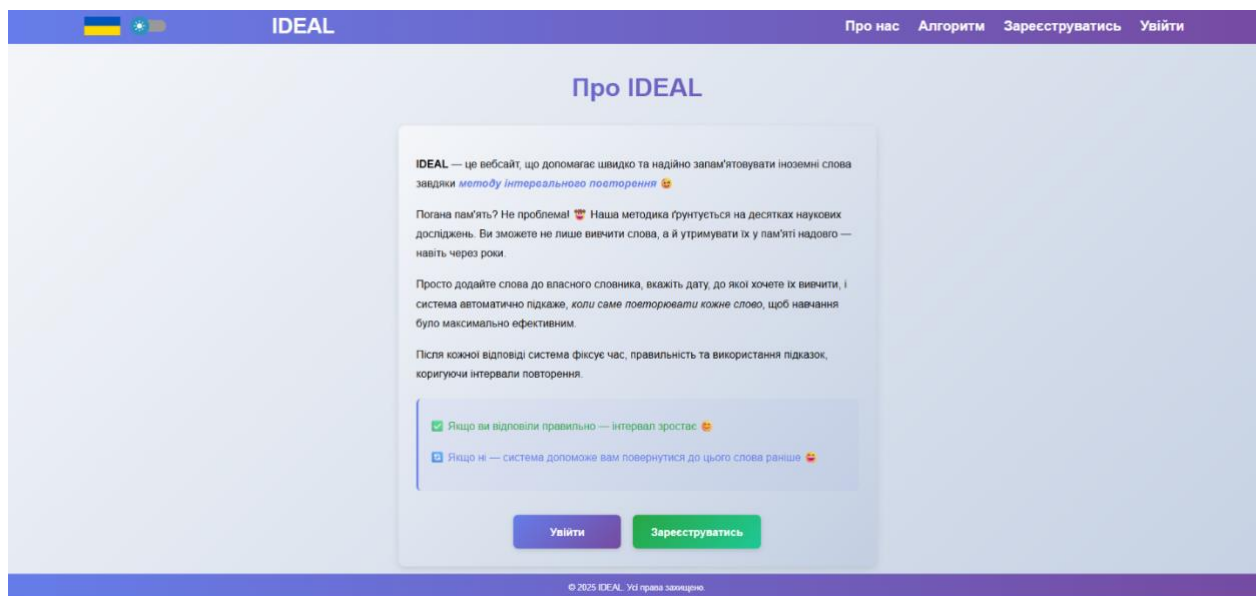


Рисунок 4.1 — Головна інформаційна сторінка системи

Для продуктивної роботи системи рекомендується налаштувати змінні оточення, зокрема API-ключ сервісу DeepL, який використовується для автоматичного перекладу слів. Після виконання всіх описаних кроків система готова до реєстрації користувачів та створення словників.

4.2 Використання системи

Система забезпечує інтуїтивно зрозумілий інтерфейс для вивчення іношомовної лексики з підтримкою машинного перекладу та алгоритмів розподілених повторень. Користувач має змогу взаємодіяти з системою за

допомогою веб-інтерфейсу, що адаптується до різних розмірів екрана та підтримує як світлу, так і темну теми оформлення.

Навігація в системі реалізована через горизонтальне меню у верхній частині сторінки, яке динамічно змінюється залежно від статусу авторизації користувача. Для неавторизованих користувачів доступні розділи «Про нас», «Алгоритм», «Зареєструватись» та «Увійти». Після успішної авторизації меню розширюється, надаючи доступ до основних функціональних модулів: «Поточні завдання», «Мої словники», «Статистика» та кнопки виходу з системи. У лівій частині хедера розміщені перемикачі мови інтерфейсу та теми оформлення для налаштування персоналізованого робочого середовища.

Після входу в систему користувач потрапляє на сторінку «Поточні завдання», яка є центральним дашбордом застосунку. Тут відображаються всі створені словники у вигляді інформативних карток, кожна з яких містить назву словника, мітку режиму навчання, статистику доступних для вивчення слів та прогрес-бар із відсотком засвоєння матеріалу. Картки оснащені кнопками швидкого доступу «Вчити зараз» для негайного початку навчальної сесії та «Деталі» для перегляду повної інформації про словник.

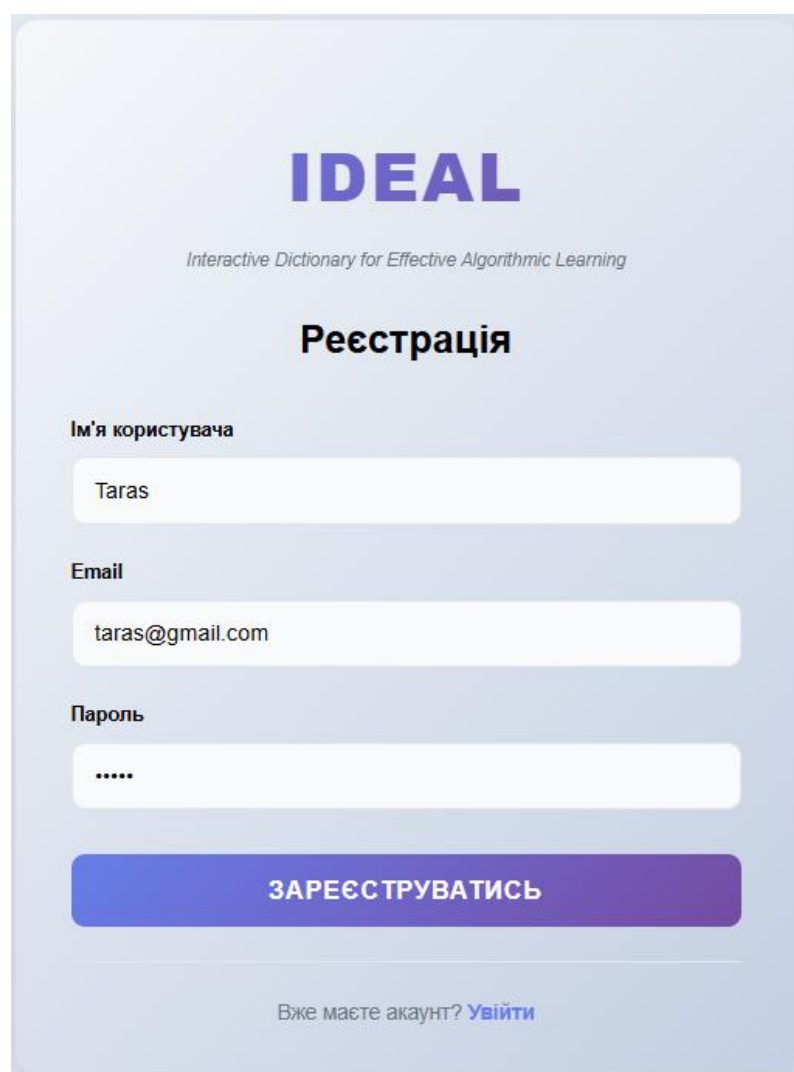
Основний робочий процес у системі включає послідовні етапи: створення словників із визначенням параметрів навчання, додавання іншомовних слів з використанням інтегрованого машинного перекладу, регулярне проходження навчальних сесій за алгоритмом розподілених повторень та моніторинг прогресу через детальну статистику. Система підтримує одночасну роботу з необмеженою кількістю словників різних мовних пар, що дозволяє організувати вивчення лексики відповідно до індивідуальних потреб користувача.

4.2.1 Реєстрація

Перш за все необхідно створити обліковий запис у системі для отримання доступу до функціоналу вивчення лексики. Процес реєстрації ініціюється натисканням посилання «Зареєструватись» у навігаційному меню головної

сторінки або на сторінці входу. Після переходу за посиланням відкривається спеціалізована форма реєстрації, доступна за адресою /signup/.

На екрані з’являється форма, яку необхідно заповнити трьома обов’язковими полями. Перше поле призначене для введення бажаного імені користувача, яке має бути унікальним у системі. Друге поле вимагає вказати адресу електронної пошти, а третє — створити надійний пароль для захисту облікового запису. Кожне поле оснащено текстовими підказками, що допомагають користувачеві правильно заповнити форму. Приклад інтерфейсу форми реєстрації зображено на рисунку 4.2.



The image shows a registration form for a system called IDEAL. At the top, the logo 'IDEAL' is displayed in large blue letters, with the subtitle 'Interactive Dictionary for Effective Algorithmic Learning' below it. The title 'Реєстрація' (Registration) is centered in bold black text. Below the title, there are three input fields: 'Ім'я користувача' (Username) with the value 'Taras', 'Email' with the value 'taras@gmail.com', and 'Пароль' (Password) with masked characters '.....'. A large blue button labeled 'ЗАРЕЄСТРУВАТИСЬ' (Register) is positioned below the fields. At the bottom, there is a link that says 'Вже маєте акаунт? Увійти' (Already have an account? Log in).

Рисунок 4.2 — Інтерфейс форми реєстрації нового користувача

Система здійснює автоматичну валідацію введених даних на етапі відправки форми. Якщо обране ім'я користувача вже існує в базі даних, форма оновлюється з

відображенням повідомлення про помилку червоним кольором у верхній частині картки. При цьому введені дані зберігаються в полях форми, окрім паролю з міркувань безпеки, що дозволяє швидко виправити помилку без повторного заповнення всієї форми.

У разі успішного заповнення форми коректними даними та натискання кнопки «Зареєструватись» система створює новий обліковий запис у базі даних та автоматично авторизує користувача. Відбувається переадресація на сторінку підтвердження успішної реєстрації, з якої можна перейти до головної сторінки для початку роботи зі створення словників. Навігаційне меню оновлюється, відображаючи пункти, доступні для авторизованих користувачів. На рисунку 4.3 продемонстровано приклад успішної реєстрації.

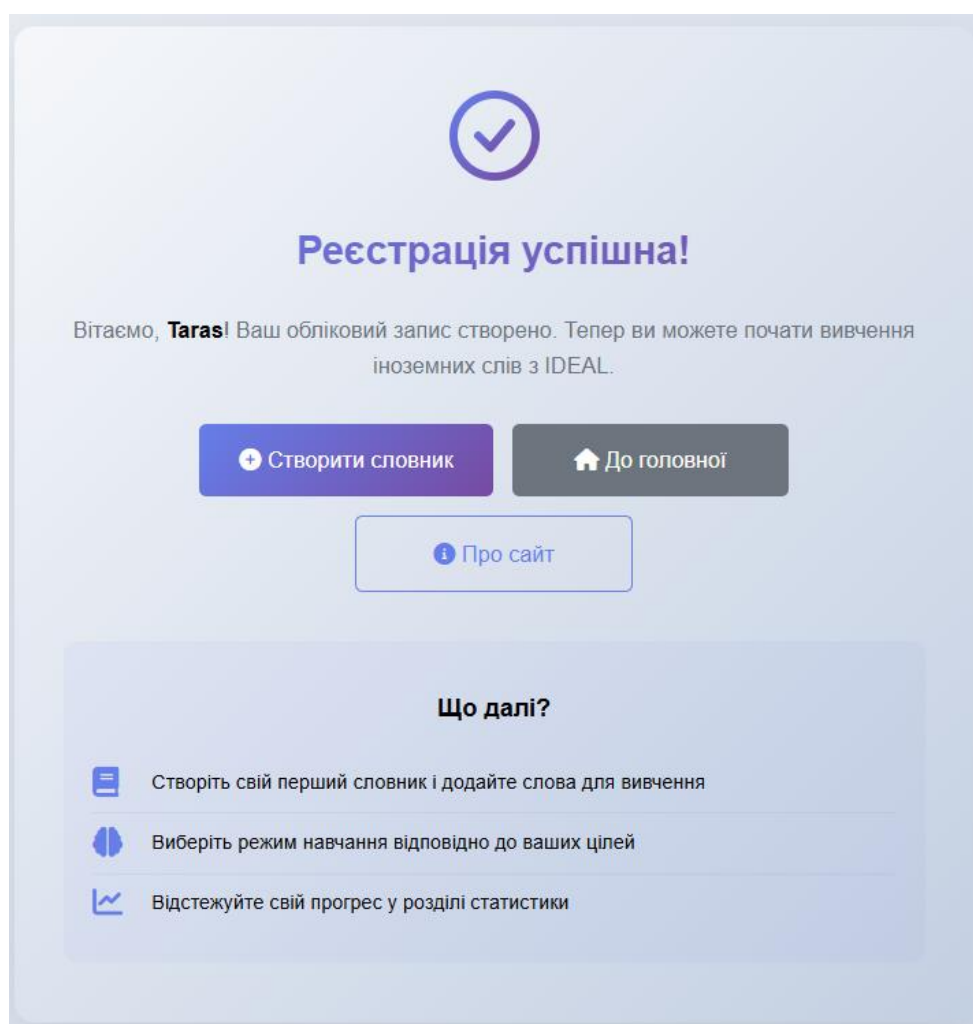
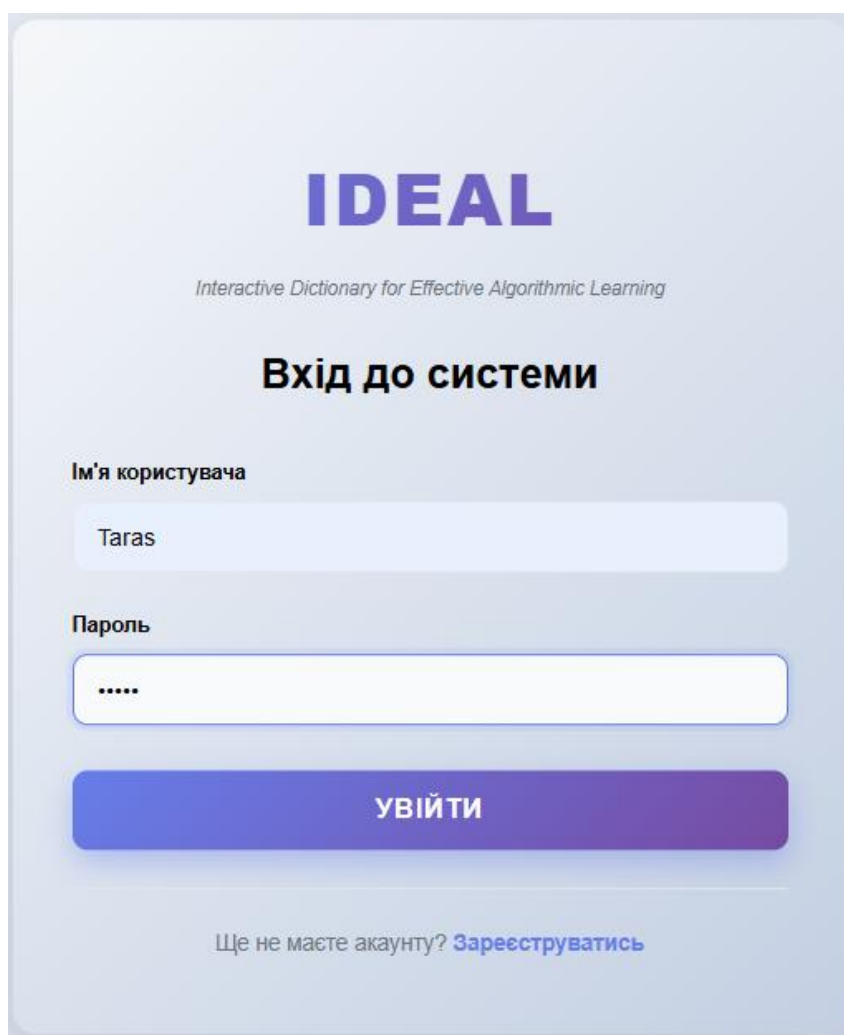


Рисунок 4.3 — Успішна реєстрація користувача

4.2.2 Авторизація

Кожна взаємодія з функціональними можливостями системи починається з авторизації користувача. Для входу необхідно перейти за посиланням «Увійти» у навігаційному меню або ввести безпосередньо URL-адресу /login/. Якщо користувач намагається отримати доступ до захищених розділів системи без попередньої авторизації, відбувається автоматична переадресація на сторінку входу.

Форма авторизації містить два поля для введення облікових даних: ім'я користувача та пароль. Інтерфейс форми виконаний у мінімалістичному стилі з чітко позначеними полями та підказками. Після заповнення обох полів користувач натискає кнопку «Увійти» для ініціації процесу автентифікації. На рисунку 4.4 зображено форму входу в систему.



The image shows a login interface for a system named IDEAL. At the top, the word "IDEAL" is displayed in large, bold, blue capital letters. Below it, the subtitle "Interactive Dictionary for Effective Algorithmic Learning" is written in a smaller, italicized font. The main heading "Вхід до системи" (Login to the system) is centered in bold black text. Below this, there are two input fields. The first is labeled "Ім'я користувача" (Username) and contains the text "Taras". The second is labeled "Пароль" (Password) and contains six dots, indicating a masked password. Below the password field is a large, rounded rectangular button with a blue-to-purple gradient, labeled "УВІЙТИ" (Login) in white capital letters. At the bottom of the form, there is a link that says "Ще не маєте акаунту? [Зареєструватись](#)" (Don't have an account? [Register](#)).

Рисунок 4.4 — Форма авторизації в системі

Система перевіряє відповідність введених даних записам у базі даних. У разі успішної перевірки активується механізм авторизації користувача. У випадку співпадіння облікових даних створюється сесія користувача, навігаційне меню оновлюється, відображаючи персоналізовані розділи системи, та відбувається переадресація на головну сторінку з поточними завданнями.

Після створення сесії система зберігає стан авторизованого користувача протягом усієї роботи у браузері. У межах активної сесії забезпечується доступ лише до персоналізованих розділів та даних, пов'язаних з поточними навчальними завданнями.

Результат введення коректних даних наведено на рисунку 4.5.

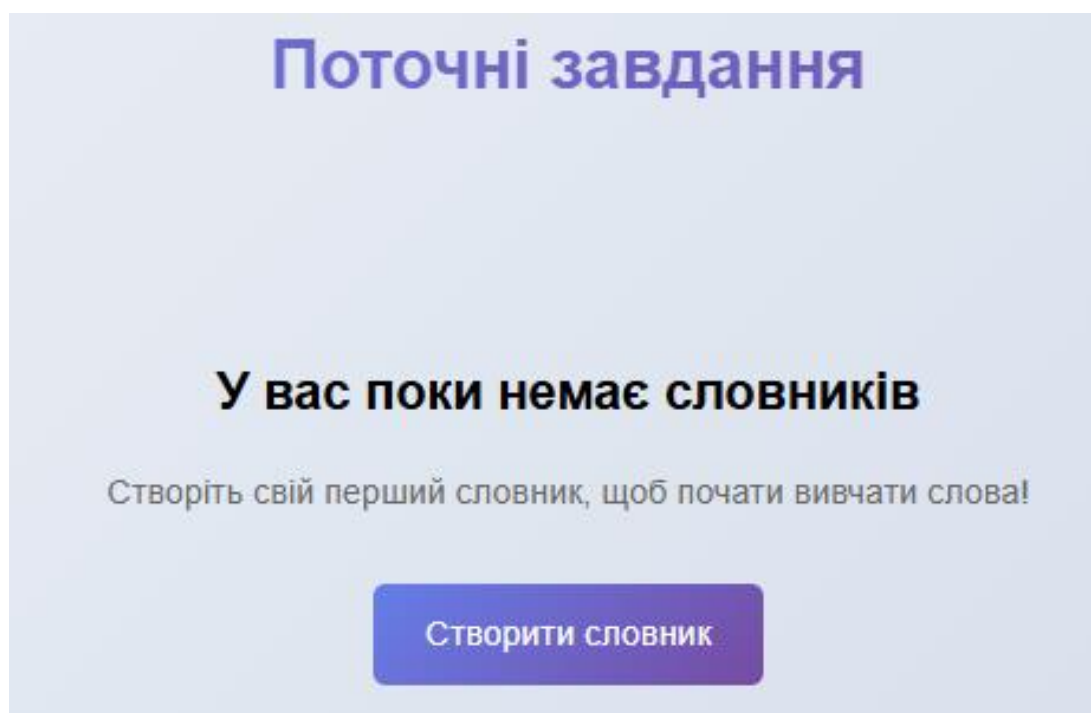


Рисунок 4.5 — Переадресація на сторінку «Поточні завдання»

Якщо введено некоректні облікові дані, форма оновлюється з відображенням повідомлення про помилку «Невірні облікові дані». Поля форми очищаються, дозволяючи користувачеві повторити спробу введення правильних даних.

Такий механізм забезпечує оперативне інформування користувача про помилку та підвищує загальний рівень безпеки доступу до системи.

На рисунку 4.6 наведено результат некоректно введених даних.

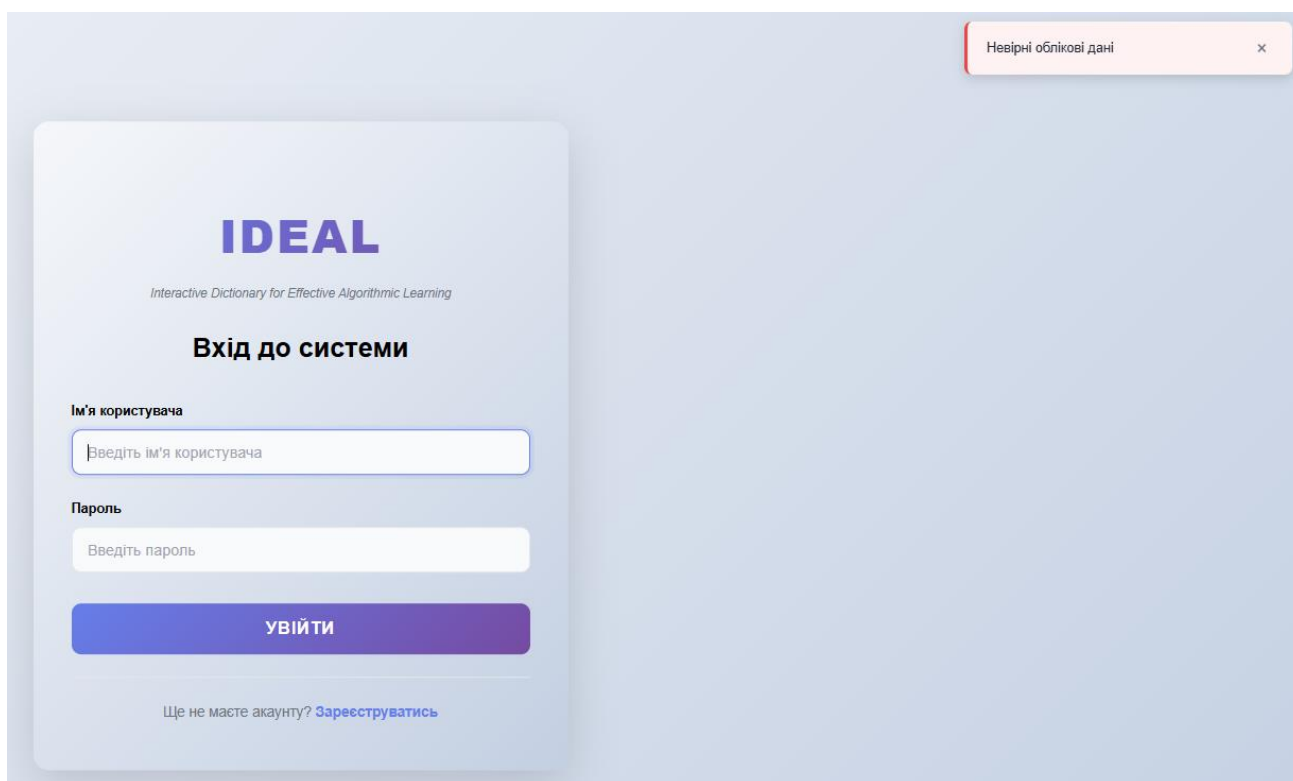


Рисунок 4.6 — Результат некоректно введених даних

Для виходу з системи авторизований користувач натискає «Вийти» у навігаційному меню. Розташування кнопки виходу зображено на рисунку 4.7.



Рисунок 4.7 — Розташування кнопки виходу

Система завершує сесію користувача, очищає дані авторизації та переадресовує на сторінку входу, повертаючи інтерфейс до стану для неавторизованих користувачів.

4.2.3 Створення словників та додавання слів

Робота з системою розпочинається зі створення словників, які є контейнерами для організації іншомовної лексики за тематичними або мовними категоріями. Для створення нового словника користувач натискає кнопку «Створити словник» на головній сторінці або в розділі «Мої словники». Після цього відкривається форма створення, доступна за адресою </dictionaries/create/>.

Форма створення словника містить обов'язкове поле назви з обмеженням у 25 символів. Під полем відображається динамічний лічильник введених символів, що допомагає користувачеві контролювати довжину назви.

Ключовим параметром є вибір режиму навчання зі спеціального випадаючого списку. Система пропонує чотири режими, кожен з яких оптимізовано для різних цілей вивчення лексики.

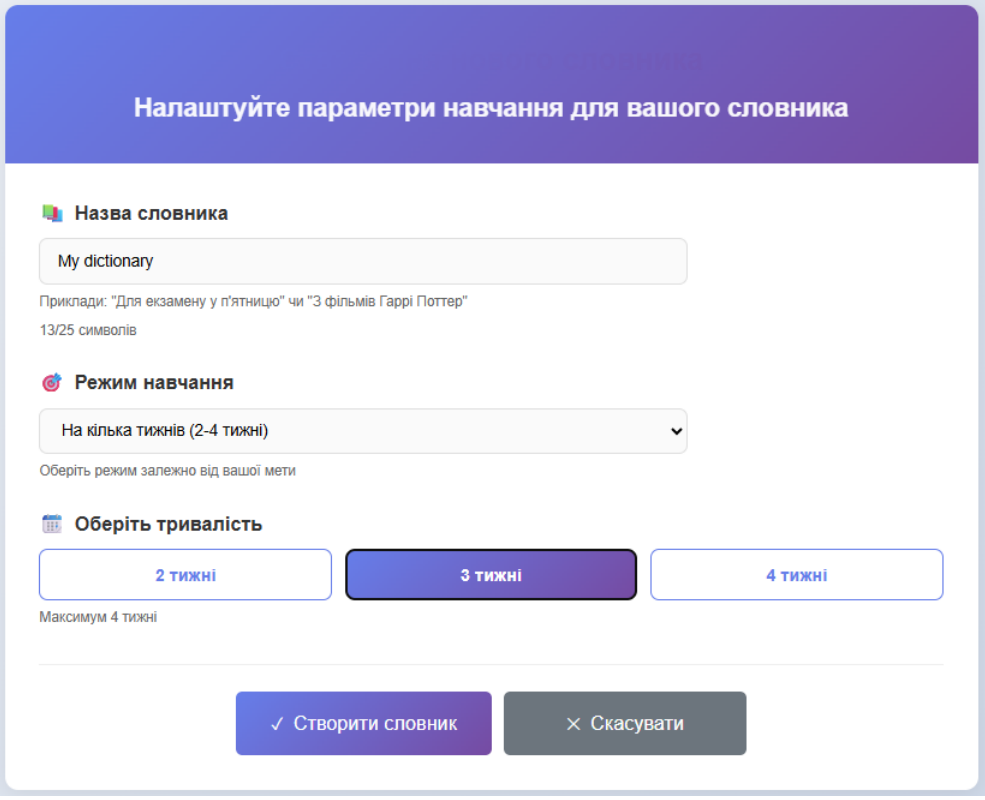
Режим «Назавжди» призначений для довготривалого запам'ятовування лексики з максимально розтягнутими інтервалами повторень. Режим «До дати іспиту» дозволяє вказати конкретний дедлайн, під який система підлаштовує інтенсивність повторень. Режим «На кілька тижнів» обмежує період вивчення терміном від двох до чотирьох тижнів для оперативного засвоєння матеріалу.

Режим «Швидке запам'ятовування» розрахований на інтенсивне вивчення протягом максимум семи днів.

Залежно від обраного режиму система динамічно відображає додаткові поля параметрів. Для режиму «До дати іспиту» з'являється поле вибору дати та часу іспиту.

Для режиму «На кілька тижнів» користувач обирає тривалість навчання за допомогою трьох кнопок, що відповідають двом, трьом або чотирьом тижням.

Режим «Швидке запам'ятовування» вимагає вказати цільову дату завершення, яка не може перевищувати семиденний термін від поточного дня. Приклад заповненої форми створення словника зображено на рисунку 4.8.



Налаштуйте параметри навчання для вашого словника

Назва словника

My dictionary

Приклади: "Для екзамєну у п'ятницю" чи "З фільмів Гаррі Поттер"

13/25 символів

Режим навчання

На кілька тижнів (2-4 тижні)

Оберіть режим залежно від вашої мети

Оберіть тривалість

2 тижні 3 тижні 4 тижні

Максимум 4 тижні

✓ Створити словник × Скасувати

Рисунок 4.8 — Форма створення словника з обраним режимом навчання

Після натискання кнопки «Створити словник» система валідує введені дані, створює запис у базі даних та відображає повідомлення про успішне створення. Автоматично відбувається переадресація на сторінку додавання слів для щойно створеного словника, що дозволяє одразу наповнити його лексичним матеріалом.

Система надає два способи додавання слів до словника: множинне додавання через табличну форму та швидке додавання окремого слова через модальне вікно. Для множинного додавання на сторінці `/dictionaries/<id>/add-words/` відображається таблична форма з двадцятьма рядками для одночасного введення. Кожен рядок містить поля для іншомовного слова та його перекладу українською мовою. Інтерфейс оптимізовано для ефективної роботи з клавіатурою — натискання клавіші Enter автоматично переміщує фокус до наступного поля.

Ключовою особливістю системи є інтеграція з DeepL Translator API для надання автоматичних підказок перекладу. Після введення іншомовного слова та паузи тривалістю три секунди система надсилає запит до API машинного перекладу. Під час обробки запиту в полі перекладу відображається індикатор

завантаження з текстом «Перекладаю...» та анімацією градієнта. По завершенні запиту поле автоматично заповнюється запропонованим варіантом перекладу з візуальною індикацією через зелену анімацію. Користувач має можливість прийняти автоматичний переклад або відредагувати його вручну відповідно до контексту використання слова. На рисунку 4.9 зображено процес автоматичного заповнення перекладу.

Додайте слова до свого словника
My dictionary

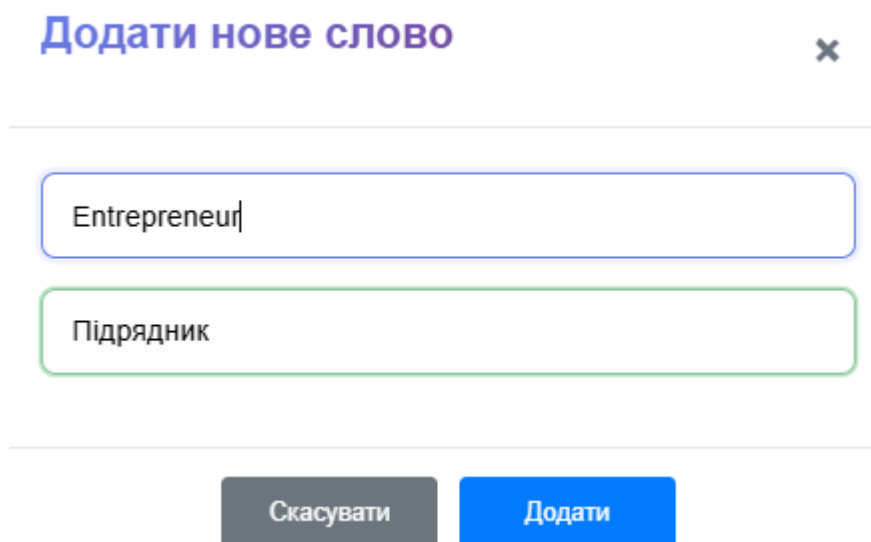
Додайте нові слова для вивчення. Можна додати до 20 слів за раз.

№	ІНШОМОВНЕ СЛОВО	ПЕРЕКЛАД	Дії
1	Prioritising	Визначення пріоритетів	
2	Іншомовне слово	Переклад українською	
3	Conscious	Свідомий	
4	Acquaintance	Знайомство	
5	Particularly	Перекладаю...	

Рисунок 4.9 — Автоматична підказка перекладу через DeepL API

Після заповнення необхідної кількості рядків користувач натискає кнопку «Зберегти слова». Система перевіряє всі заповнені рядки на коректність, виявляє потенційні дублікати у межах словника та зберігає нові слова в базу даних. Відображається повідомлення з підтвердженням кількості успішно доданих слів, після чого відбувається переадресація на сторінку деталей словника.

Для швидкого додавання одного слова без переходу на окрему сторінку на інтерфейсі деталей словника розміщена кнопка «Додати слово». Приклад додавання одного слова наведено на рисунку 4.10.



Додати нове слово

Entrepreneur

Підрядник

Скасувати Додати

Рисунок 4.10 — Додавання одного слова

При натисканні відкривається модальне вікно з компактною формою, що містить два поля вводу. Механізм автоматичного перекладу функціонує аналогічно табличній формі. Після збереження модальне вікно закривається, відображається сповіщення про успішне додавання, а сторінка оновлюється, показуючи нове слово у списку.

4.2.4 Процес вивчення лексики за методом розподілених повторень

Навчальний процес у системі реалізований через сесії вивчення, під час яких користувач послідовно повторює слова за алгоритмом розподілених повторень. Для запуску сесії користувач натискає кнопку «Вчити зараз» на головній сторінці або «Почати вчити» на сторінці деталей словника.

Перед початком сесії система автоматично перевіряє наявність слів, доступних для вивчення на поточний день, та формує оптимальну послідовність їх показу. На сторінці деталей словника користувач може переглянути повний список доданих слів, їх поточний статус вивчення, статистику показів та прогрес засвоєння матеріалу, що дозволяє прийняти обґрунтоване рішення про доцільність початку навчальної сесії. Сторінку деталей словника зображено на рисунку 4.11.

My dictionary На кілька тижнів (2-4 тижні)

Всього слів **5**

Вивчено **0**

Вивчається **0**

Нових **5**

Прострочені **0**

Доступно **0**

Додати слово
Додати багато слів
Редагувати словник
Почати вчити
Видалити словник

Слова у словнику (5)

Пошук слова...
Всі статуси
За алфавітом
Знайти

Іншомовне слово	Переклад	Показано	Статус	Дії
Acquaintance	Знайомство	0	Нове	
Conscious	Свідомий	0	Нове	
Entrepreneur	Підрядник	0	Нове	
Particularly	Зокрема	0	Нове	
Prioritising	Визначення пріоритетів	0	Нове	

[← Назад до списку словників](#)

Рисунок 4.11 — Сторінка деталей словника

Система перевіряє наявність слів, доступних для вивчення, та формує послідовність показу з урахуванням пріоритетів алгоритму.

Порядок відбору слів для навчальної сесії визначається трьома категоріями пріоритету. Спочатку система відбирає прострочені слова, які мали бути повторені раніше відповідно до запланованих дат. Далі включаються слова, заплановані для повторення на поточний день. Завершують список нові слова, які ще не показувались користувачеві. Алгоритм автоматично чергує нові слова з повтореннями у пропорції один до двох, що забезпечує ефективне запам'ятовування без перевантаження новою інформацією.

Система накладає обмеження на інтенсивність навчання для запобігання втомі та підтримання якості запам'ятовування. Максимальна кількість слів за одну

сесію становить сорок для стандартних режимів або шістдесят для режиму «До іспиту». Кількість нових слів обмежена двадцятьма за сесію та сорока за добу для кожного словника. Одне слово може показуватись максимум чотири рази на день, за винятком ситуацій наближення іспиту, коли ліміт збільшується до п'яти показів.

При показі першого слова сесії відображається модальне вікно з корисними порадами щодо ефективної роботи з системою. Користувач інформується про критерії оцінювання відповідей та рекомендації щодо оптимального темпу роботи. Вікно з корисними порадами продемонстровано на рисунку 4.12.

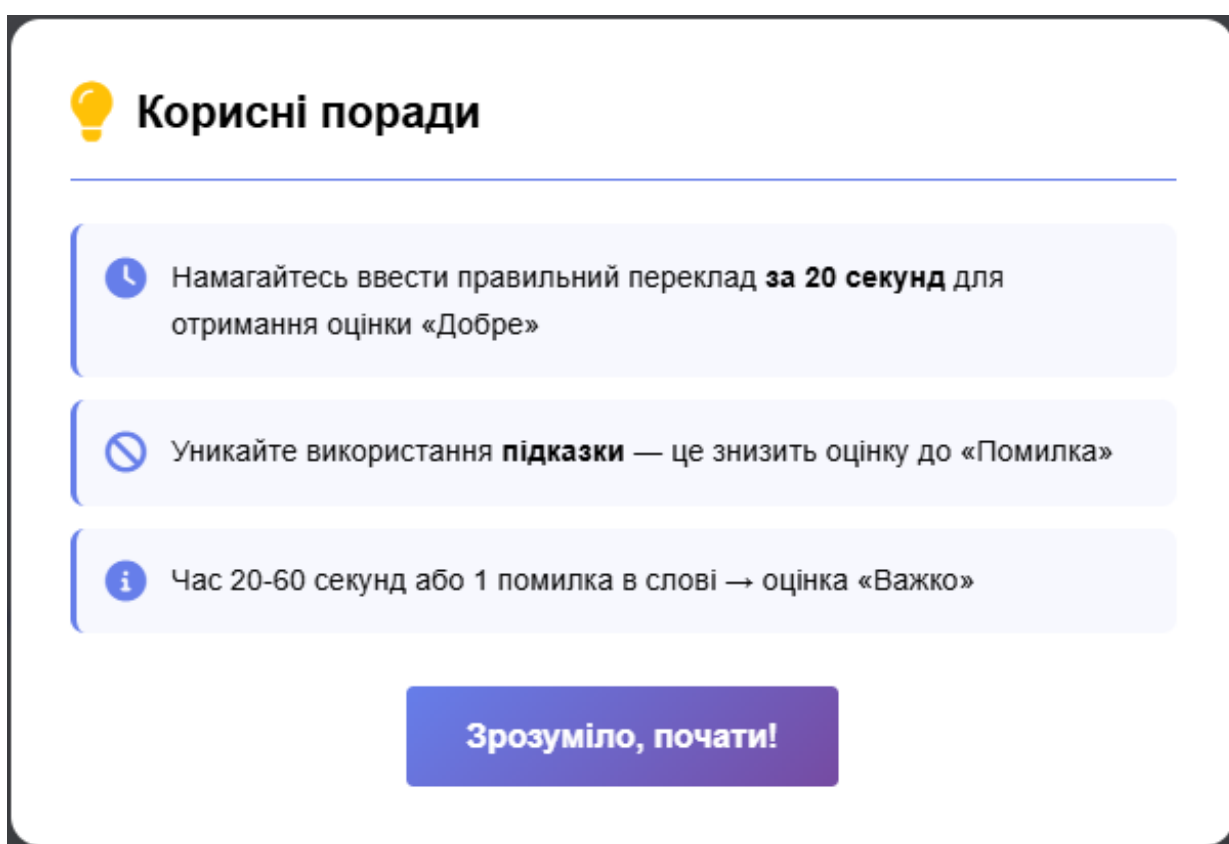


Рисунок 4.12 — Вигляд вікна з корисними порадами

Після закриття модального вікна розпочинається відлік таймера, і користувач бачить інтерфейс навчальної картки. У верхній частині екрана розміщено індикатор прогресу у форматі «Слово X/Y», що показує поточну позицію в сесії. Центральну частину займає іншомовне слово великим шрифтом, під яким знаходиться поле для введення перекладу. Інтерфейс навчальної картки зображено на рисунку 4.13.

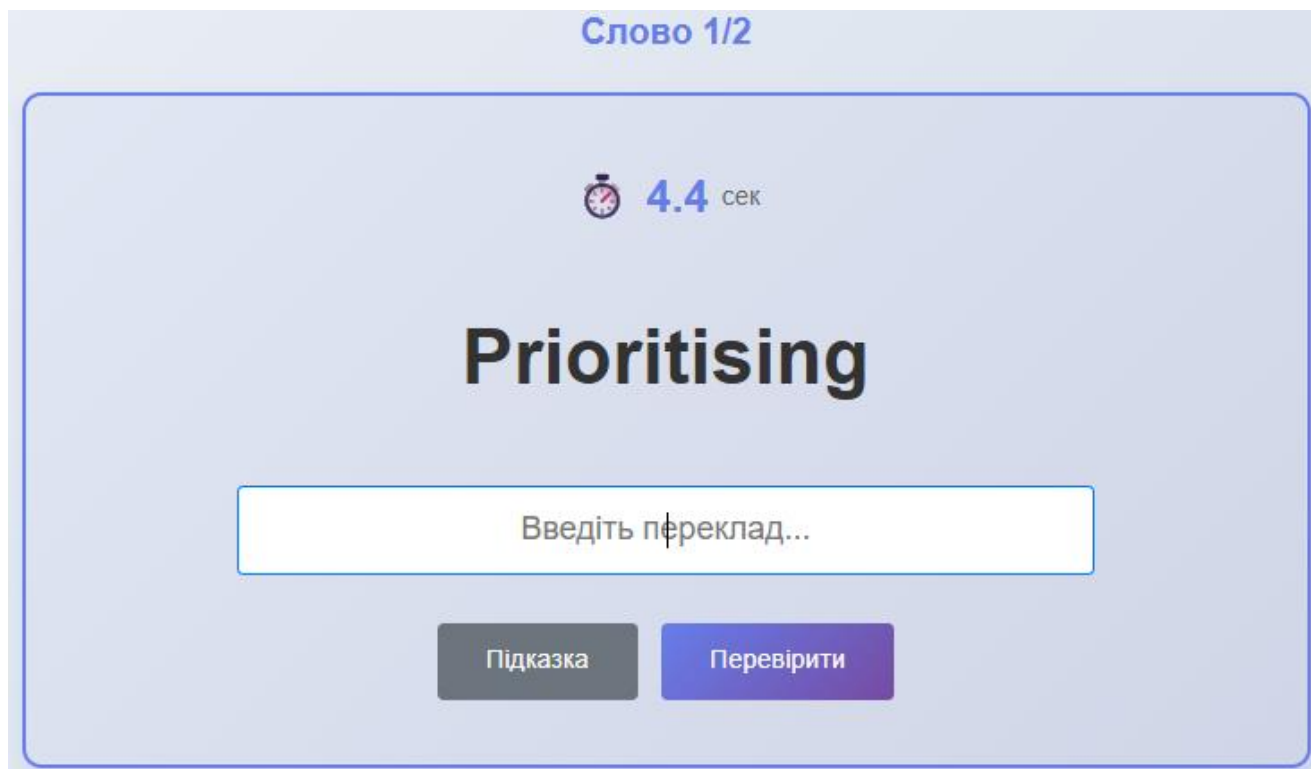


Рисунок 4.13 — Інтерфейс навчальної картки під час сесії вивчення

Користувач вводить переклад іншомовного слова у відповідне поле та натискає кнопку «Перевірити» або клавішу Enter. Система зупиняє таймер та класифікує відповідь за трьома категоріями залежно від точності перекладу та витраченого часу. Відповідь отримує оцінку «Добре», якщо переклад повністю правильний, час не перевищує двадцяти секунд, і користувач не звертався до підказки. Оцінка «Важко» присвоюється правильним відповідям, які потребували від двадцяти до шістдесяти секунд, або містять одну незначну помилку. Оцінка «Помилка» призначається при неправильному перекладі, перевищенні шістдесятисекундного ліміту або використанні функції підказки.

Після класифікації відповіді відображається секція результату з реакційним повідомленням та додатковою інформацією. Для правильних відповідей система виводить заохочувальне повідомлення. У випадку неправильної відповіді додатково показується правильний переклад та візуалізація відмінностей, якщо різниця не перевищує сорок відсотків від довжини слова. Компонент візуалізації підсвічує помилкові символи червоним кольором, а правильні позиції — зеленим,

що допомагає користувачеві зрозуміти характер помилки. Приклад відображення результату зображено на рисунку 4.14.

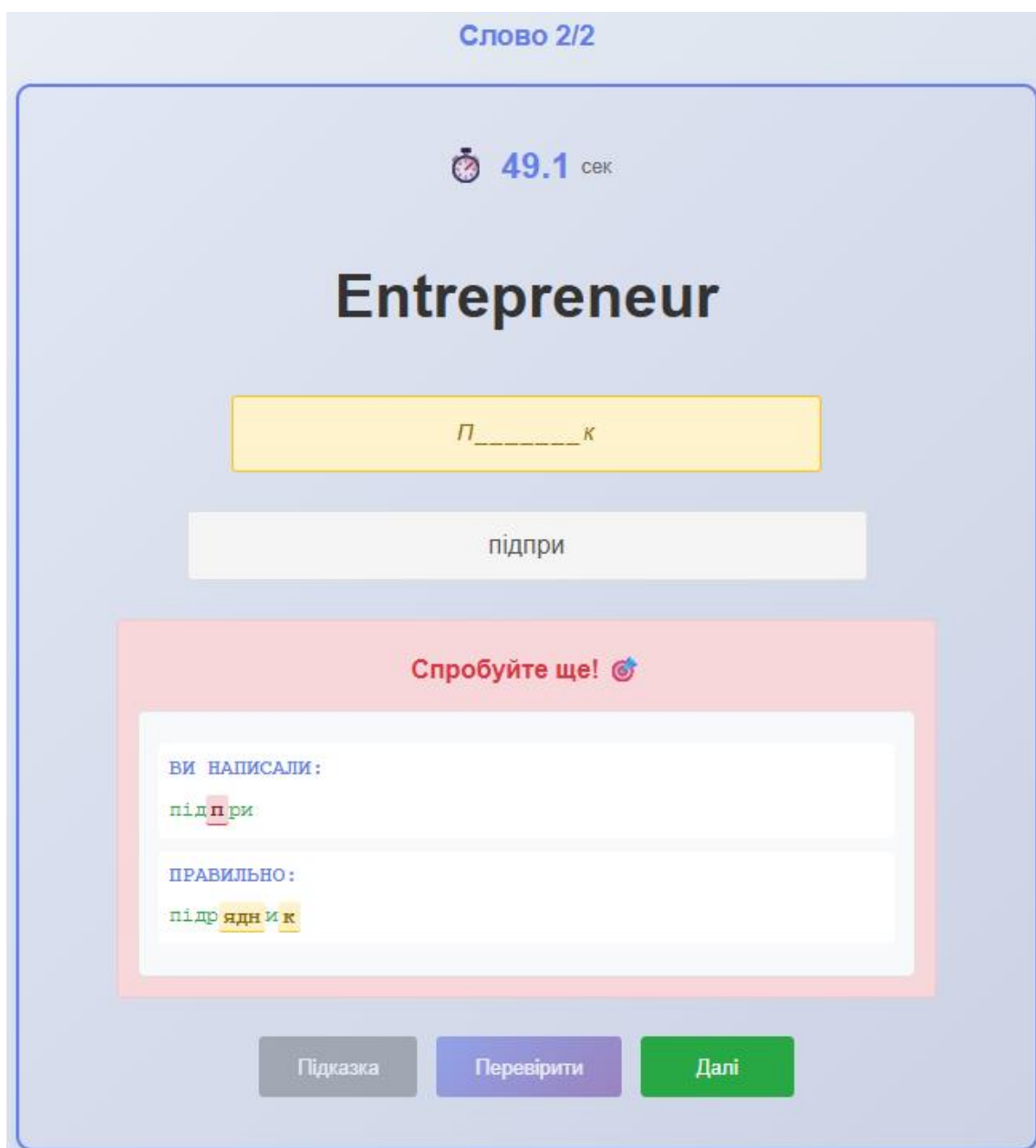


Рисунок 4.14 — Відображення результату перевірки відповіді

Ключовою особливістю алгоритму є механізм мікроповторів для слів з оцінками «Важко» та «Помилка». Такі слова автоматично додаються до спеціальної черги та повторно показуються в межах поточної сесії через три-сім інших слів. Система дозволяє максимум два мікроповтори на одне слово за сесію.

Якщо після двох спроб слово все ще не отримало оцінку «Добре», планується додатковий показ через дві-шість годин у межах поточного дня. На основі результату відповіді система автоматично планує наступний показ слова відповідно до обраного режиму навчання.

Для режиму «Назавжди» використовуються базові інтервали, що починаються з одного дня та поступово збільшуються до шести, дванадцяти, двадцяти чотирьох та сорока восьми днів.

Режим «До іспиту» адаптує інтервали під залишковий час до дедлайну, збільшуючи інтенсивність повторень у критичні періоди. Режими з обмеженим терміном використовують стиснуті інтервали для вкладання в задані часові рамки.

Інтервали коригуються індивідуальним коефіцієнтом легкості для кожного слова, який змінюється залежно від історії відповідей. Успішні відповіді збільшують коефіцієнт, що призводить до подовження інтервалів між повтореннями. Складні відповіді зменшують коефіцієнт та скорочують інтервали. Помилкові відповіді можуть скидати інтервал до початкового значення. Початкове значення коефіцієнта залежить від довжини слова — коротші слова отримують вищий коефіцієнт, припускаючи їх потенційно легше запам'ятовування.

Система фіксує всі навчальні дані у базі даних для подальшого аналізу та побудови статистики. Для кожної спроби відповіді створюється окремий запис з інформацією про результат класифікації, витрачений час, використання підказок та відповідний словник. Оновлюється загальна статистика слова, включаючи кількість показів, кількість правильних відповідей та поточне значення коефіцієнта легкості. Ці дані використовуються для формування графіків прогресу та звітів у розділі моніторингу.

По завершенні введення перекладу всіх слів сесії система автоматично переадресовує на підсумкову сторінку. Відображається детальна статистика з розподілом відповідей за категоріями, загальною кількістю спроб, витраченим

часом та середнім часом на одне слово. Ця інформація допомагає користувачеві оцінити ефективність сесії та визначити аспекти, що потребують покращення.

Приклад підсумкової сторінки зображено на рисунку 4.15.

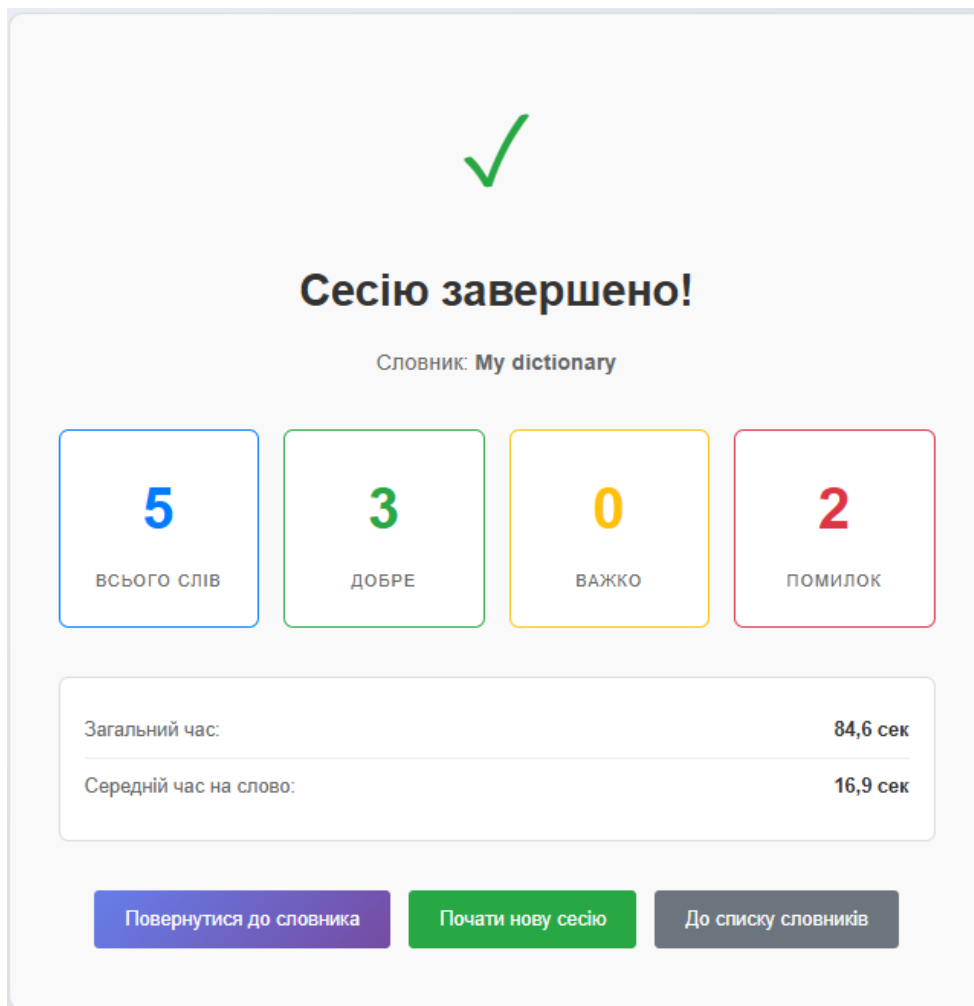


Рисунок 4.15 — Підсумкова статистика навчальної сесії

Система фіксує всі навчальні дані у базі даних для подальшого аналізу. Створюються записи про кожну спробу відповіді з інформацією про результат, витрачений час та використання підказок. Оновлюється статистика слів, включаючи загальну кількість показів, кількість правильних відповідей та коефіцієнт легкості. Ці дані використовуються для побудови детальної статистики прогресу вивчення, описаної у наступному підрозділі.

4.2.5 Моніторинг прогресу вивчення

Система забезпечує комплексний моніторинг прогресу через спеціалізований розділ статистики, доступний за посиланням «Статистика» у навігаційному меню. Сторінка агрегує дані з навчальних сесій та відповідей користувача за останні тридцять днів, представляючи їх у вигляді числових показників, графіків та діаграм.

У верхній частині сторінки розміщені п'ять інформаційних карток з основними показниками. Відображається загальна кількість створених словників, сумарна кількість слів у всіх словниках, кількість слів зі статусом «вивчено», кількість слів у процесі вивчення та загальний відсоток прогресу. Кожна картка оформлена з використанням кольорового кодування та іконок для швидкого візуального сприйняття інформації. Загальний вигляд статистичних карток зображено на рисунку 4.16.

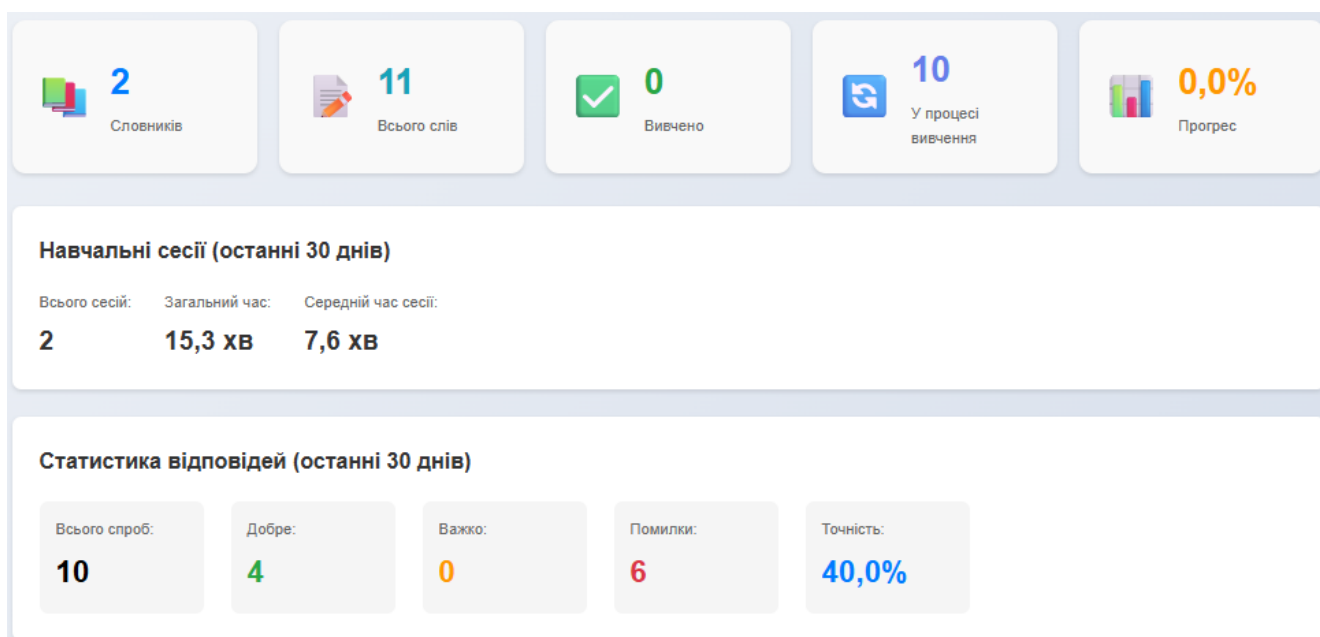


Рисунок 4.16 — Загальні показники прогресу вивчення

Секція статистики навчальних сесій містить інформацію про загальну кількість завершених сесій за місяць, сумарний витрачений час у хвилинах та середню тривалість однієї сесії. Ці показники дозволяють оцінити регулярність

занять та інтенсивність навчального процесу. Секція статистики відповідей деталізує результати роботи користувача, показуючи загальну кількість спроб та їх розподіл за категоріями «Добре», «Важко» та «Помилка». Додатково обчислюється показник точності як відсоток правильних відповідей від загальної кількості.

Графік прогресу вивчення реалізовано у вигляді лінійної діаграми з двома кривими. Зелена крива відображає кількість правильних відповідей за кожний день, червона крива — кількість неправильних відповідей, що включає категорії «Важко» та «Помилка». Горизонтальна вісь представляє дати у форматі день-місяць за останні тридцять днів, вертикальна вісь — кількість відповідей. Графік оснащено інтерактивними підказками, що відображаються при наведенні курсора на точки даних. Приклад графіка прогресу зображено на рисунку 4.17.

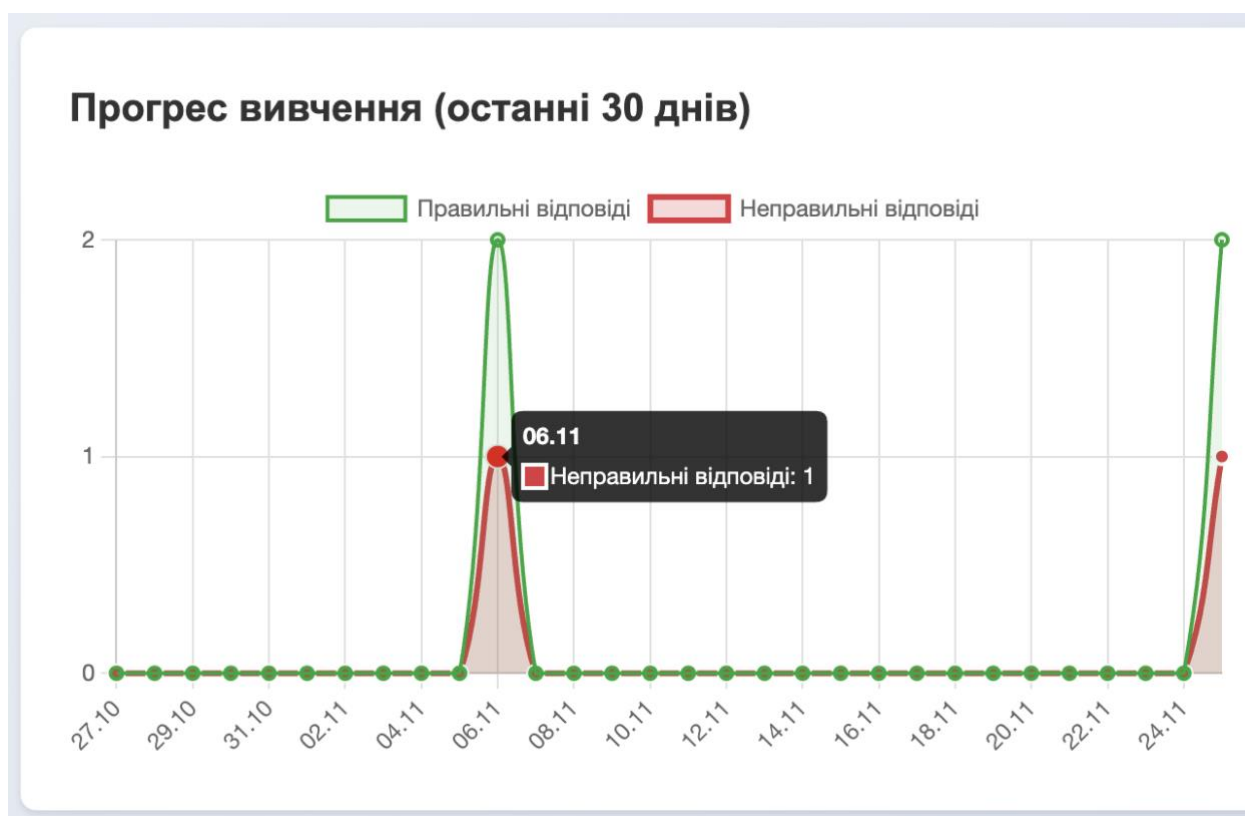


Рисунок 4.17 — Графік динаміки вивчення за останні 30 днів

Стовпчаста діаграма кількості сесій візуалізує регулярність занять, показуючи кількість навчальних сесій для кожного дня місяця. Відсутність

стовпців для певних днів вказує на перерви в навчанні, що допомагає користувачеві виявити періоди низької активності. Аналіз цієї діаграми у поєднанні з графіком прогресу надає комплексне уявлення про динаміку навчального процесу.

Секція статистики по окремих словниках представляє детальну інформацію про прогрес вивчення кожного словника. Для кожного словника відображається горизонтальний прогрес-бар з градієнтним заповненням зеленого кольору, що візуалізує відсоток вивчених слів. Поруч з прогрес-баром вказується числове співвідношення вивчених слів до загальної кількості та відсоток прогресу. Словники відсортовані за спаданням прогресу, що дозволяє швидко визначити найбільш та найменш засвоєні словники. Приклад відображення статистики по словниках зображено на рисунку 4.18.

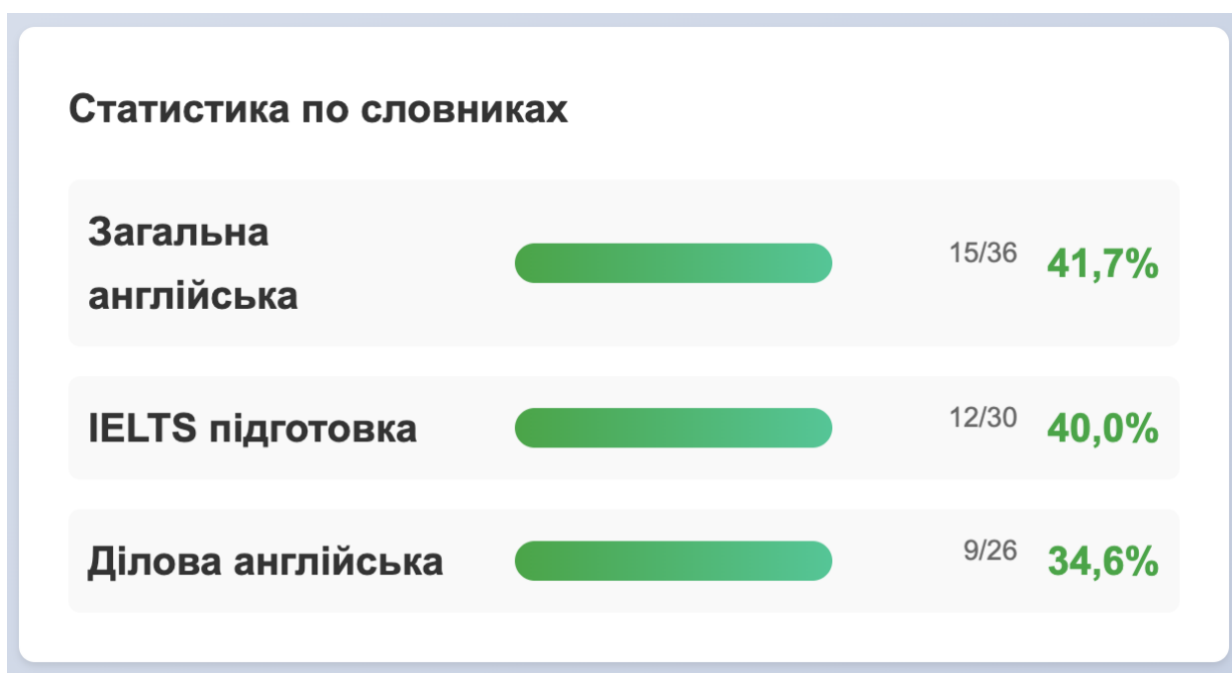


Рисунок 4.18 — Прогрес вивчення окремих словників

Усі графічні елементи статистики підтримують адаптацію до темної теми інтерфейсу. При перемиканні теми автоматично змінюються кольори сітки, тексту та фонових елементів для забезпечення комфортного сприйняття інформації у будь-яких умовах освітлення. Інтерактивність графіків дозволяє детально

аналізувати окремі періоди та виявляти тенденції покращення або погіршення результатів.

Таким чином система надає користувачеві повний набір інструментів для об'єктивної оцінки ефективності навчального процесу, виявлення проблемних аспектів та планування подальшої роботи над вивченням іншомовної лексики.

Висновки до розділу 4

У цьому розділі детально описано процес взаємодії користувача з розробленою системою вивчення іншомовної лексики за методом розподілених повторень. Розглянуто технічні вимоги до програмного забезпечення.

Продемонстровано повний цикл роботи користувача з системою, починаючи від реєстрації облікового запису та авторизації до створення словників з чотирма різними режимами навчання. Кожен режим оптимізовано під конкретні цілі вивчення: довготривале запам'ятовування, підготовка до іспиту, вивчення на кілька тижнів або швидке запам'ятовування протягом семи днів. Система динамічно відображає додаткові параметри залежно від обраного режиму, що забезпечує гнучкість налаштування під індивідуальні потреби.

Ключовою особливістю системи є інтеграція з DeepL Translator API для автоматичного надання підказок перекладу під час додавання слів до словників. Механізм працює асинхронно з тримісекундним таймером, відображаючи візуальні індикатори процесу перекладу та дозволяючи користувачеві коригувати запропоновані варіанти вручну. Це суттєво прискорює процес наповнення словників та знижує когнітивне навантаження на користувача.

Процес навчання реалізовано через сесії з послідовним показом слів за алгоритмом розподіленого повторення. Система автоматично класифікує відповіді користувача за трьома категоріями залежно від точності перекладу та витраченого часу, що визначає інтервал наступного повторення. Механізм мікроповторів забезпечує додаткове закріплення складних слів у межах однієї сесії, а система

обмежень запобігає перевантаженню користувача через денні ліміти на нові слова та кількість показів.

Розділ статистики надає комплексний моніторинг прогресу через інтерактивні графіки та діаграми, побудовані засобами Chart.js. Візуалізація динаміки правильних та неправильних відповідей за тридцять днів дозволяє користувачеві об'єктивно оцінити ефективність навчального процесу, виявити періоди низької активності та коригувати стратегію вивчення.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

Даний розділ містить інформацію про маркетинговий аналіз стартап-проєкту IDEAL (Interactive Dictionary for Effective Algorithmic Learning) — веб-застосунку для вивчення іншомовної лексики з інтегрованим машинним перекладом. Аналіз проводиться з метою встановлення оцінки перспективності впровадження інструменту для мовного навчання на українському ринку. Для впровадження готового програмного продукту визначаються можливі шляхи, щоб провести даний етап успішно та ефективно.

Детальний опис проєкту чітко надає відповідь на питання: яку задачу дозволяє вирішити розроблена система; хто є потенційними кінцевими користувачами програмного продукту; які можливості та переваги здатна запропонувати система, щоб зацікавити користувачів; які відмінності розробленого вебзастосунку від конкурентів. Встановлюється технологічна здійсненність проєкту на основі доступних рішень. Розглядаються можливі варіанти монетизації програмного продукту, оптимальна цінова політика порівняно з аналогами та шляхи збуту через прямі онлайн-канали.

Даний підхід надає можливість створити стратегії, які дозволяють бути конкурентоспроможним на ринку інструментів для вивчення іноземних мов, успішно розвиватися та просуватися серед цільової аудиторії студентів і молодих професіоналів, отримувати дохід від вебзастосунку.

5.1 Опис ідеї стартап-проєкту

У межах стартап-проєкту розробляється вебзастосунок під назвою IDEAL — Interactive Dictionary for Effective Algorithmic Learning. Це система для вивчення іншомовної лексики, що поєднує машинний переклад із алгоритмами

розподіленого повторення. Основна ідея — створення адаптивної системи, яка дозволяє ефективно запам'ятовувати іноземні слова завдяки індивідуалізованому плануванню повторень та автоматизації перекладу через DeepL API.

Система використовує модифікований алгоритм SuperMemo SM-2, динамічно визначаючи інтервали повторень на основі результатів перекладу користувача. Користувачі створюють персональні словники, додають слова з автоперекладом, проходять навчальні сесії та відстежують статистику прогресу.

Таблиця 5.1 містить опис ідеї стартап-проєкту з деталізацією змісту, напрямків застосування та вигод для користувачів.

Таблиця 5.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Вебзастосунок для вивчення іншомовної лексики з інтегрованим машинним перекладом DeepL та адаптивним алгоритмом розподіленого повторення. Система адаптує складність та інтервали повторень на основі індивідуальних результатів користувача	Самостійне вивчення іноземних мов студентами, професіоналами та ентузіастами. Підготовка до мовних іспитів, складання словникового запасу для подорожей, довгострокове опанування лексики	Автоматизація перекладу слів без перемикання між додатками. Науково обґрунтований підхід до запам'ятовування. Персоналізовані інтервали повторень. Чотири режими навчання для різних цілей. Детальна статистика прогресу та візуалізація результатів

Для аналізу сильних, слабких та нейтральних характеристик обрано Anki як один із найпоширеніших інструментів розподіленого повторення. Порівняльні характеристики наведено в таблиці 5.2.

Таблиця 5.2 — Визначення сильних, слабких та нейтральних характеристик

№ з/п	Характеристика	IDEAL	Anki	Позиція стартап-проекту
1	Інтеграція машинного перекладу	Вбудована інтеграція з DeepL API для автоматичного перекладу слів	Відсутня, потрібне ручне додавання перекладів або сторонні плагіни	Сильна
2	Складність налаштування	Інтуїтивний інтерфейс, готові режими навчання без необхідності конфігурації	Потребує детального налаштування параметрів алгоритму, що ускладнює освоєння для початківців	Сильна
3	Цільова спрямованість	Спеціалізація на вивченні лексики з оптимізованими сценаріями	Універсальний інструмент для запам'ятовування будь-якої інформації	Нейтральна
4	Вартість	Потенційно безкоштовна базова версія з можливістю преміум-підписки	Повністю безкоштовний (окрім iOS версії)	Нейтральна
5	Режими навчання	Чотири попередньо налаштовані режими для різних цілей	Один алгоритм, потребує ручного налаштування для різних сценаріїв	Сильна

Кінець таблиці 5.2

6	Статистика	Візуалізовані графіки за 30 днів, топ словників за прогресом	Детальна статистика з можливістю експорту даних	Нейтральна
7	Синхронізація	Обмежена функціональність, працює в межах браузера	Хмарна синхронізація AnkiWeb між усіма пристроями	Слабка
8	Гнучкість контенту	Обмеження текстового формату формат (слово-переклад)	Підтримка мультимедійних карток (аудіо, зображення, HTML)	Слабка
9	Мова інтерфейсу	Українська	Підтримка багатьох мов	Слабка

IDEAL перевершує Anki за інтеграцією машинного перекладу та простотою використання. Автоматизація через DeepL API усуває потребу ручного пошуку перекладів. Основною слабкою позицією є відсутність гнучкості контенту.

5.2 Технологічний аудит ідеї стартап-проєкту

Аудит технологій для реалізації стартап-проєкту зосереджено на визначенні наявності та доступності ключових технологій, необхідних для створення вебзастосунку IDEAL. Таблиця 5.3 містить інформацію про технології, що потрібні для здійснення стартап-проєкту, їхню наявність на ринку та доступність для розробників.

Таблиця 5.3 — Технологічна здійсненність ідеї проєкту

№ з/п	Ідея проєкту	Технології реалізації	Наявність технологій	Доступність технологій
1	Backend вебзастосунку	Python 3.13.0, Django 5.1.5	Наявна	Доступна безкоштовно (відкритий код)
2	База даних	SQLite3	Наявна	Доступна безкоштовно
3	Frontend інтерфейс	HTML5, CSS3, JavaScript	Наявна	Доступна безкоштовно
4	Машинний переклад	DeepL API Free (500000 символів/міс)	Наявна	Доступна безкоштовно з обмеженнями, комерційний план доступний
5	Алгоритм розподіленого повторення	Адаптація SuperMemo SM-2	Наявна	Доступна безкоштовно (алгоритм у відкритому доступі)
6	Система авторизації	Django Authentication System	Наявна	Доступна безкоштовно (вбудовано в Django)
7	Візуалізація статистики	Chart.js	Наявна	Доступна безкоштовно
8	HTTP-клієнт для API	Python requests library	Наявна	Доступна безкоштовно (відкритий код)
9	Валідація форм	Django Forms та validators	Наявна	Доступна безкоштовно (вбудовано в Django)
10	Хостинг	PythonAnywhere, Heroku, AWS, DigitalOcean	Наявна	Безкоштовні рівні доступні, платні плани для масштабування

Проект технологічно здійснений. Python, Django та DeepL API є доступними безкоштовно або з мінімальними витратами. DeepL Free надає 500000 символів/міс для тестування мінімально життєздатного продукту.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Аналіз ринкових можливостей проводиться з урахуванням характеристик потенційного ринку, визначення цільових груп клієнтів та факторів загроз і можливостей. Таблиця 5.4 містить характеристику потенційного ринку.

Таблиця 5.4 — Попередня характеристика потенційного ринку стартап-проекту

№ з/п	Показники стану ринку	Характеристика
1	Кількість основних гравців, од.	5, — універсальні системи Anki і Quizlet, мовні платформи Duolingo і Memrise, словникові додатки типу LinguaLeo
2	Загальний обсяг продажів, грн/ум.од.	Невідомо. Станом на 2024 рік глобальний ринок освітніх технологій становив більше \$250 млрд.
3	Динаміка ринку	Зростаюча. Цифровізація освіти прискорилося після пандемії COVID-19
4	Бар'єри входу	Помірні. Низькі технічні бар'єри, основна складність — залучення користувачів
5	Вимоги до сертифікації	Відсутні
6	Норма рентабельності, %	40-60% після самоокупності
7	Рівень ризику	Середній

Ринок є привабливим завдяки зростанню попиту на цифрові інструменти навчання та низьким технологічним бар'єрам входу. У таблиці 5.5 представлено аналіз потенційних клієнтів з урахуванням їхніх потреб та вимог до продукту.

Таблиця 5.5 — Характеристика потенційних клієнтів стартап-проєкту

№ з/п	Потреба	Цільова аудиторія	Відмінності у поведінці	Вимоги до товару
1	Запам'ятовування лексики для іспитів	Студенти ВНЗ (18-25 років)	Короткострокові дедлайни, структурований підхід	Режим підготовки до іспиту, статистика для контролю
2	Професійний словниковий запас	Професіонали (25-45 років)	Обмежений час, гнучкий графік	Режим довгострокового навчання, короткі сесії (10-15 хв)
3	Базова лексика для подорожей	Мандрівники (20-50 років)	Швидкий результат, практична лексика	Швидке запам'ятовування, тематичні словники

Після визначення цільових груп проведено аналіз факторів загроз (таблиця 5.6) для проєкту.

Таблиця 5.6 — Фактори загроз

№ з/п	Фактор	Зміст загрози	Можлива реакція
1	Конкуренція	Anki, Duolingo мають мільйони користувачів	Фокус на DeepL, оптимізація інтерфейсу для україномовних
2	Обмеження DeepL	500000 символів/міс	Моніторинг, перехід на DeepL Pro або альтернативи
3	Платоспроможність	Студенти шукають безкоштовні рішення	Freemium-модель, доступна ціна
4	Зміна трендів	Нові ШІ-інструменти	Моніторинг, інтеграція LLM

Далі необхідно проаналізувати фактори, що дозволять ефективно зайти на ринок, а також ті, на яких потрібно сконцентруватися для кращого просування на ринку. Фактори можливостей представлено у таблиці 5.7.

Таблиця 5.7 — Фактори можливостей

№ з/п	Фактор	Зміст можливості	Можлива реакція
1	Зростання освітніх технологій	Цифровізація освіти	Просування через освітні канали, партнерства
2	Локалізація	Мало україномовних інструментів	Розвиток українського контенту
3	LLM-інтеграція	ІІІ для генерації контексту	Впровадження ІІІ-помічника
4	Бізнес для бізнесу	Інтеграція у ЗВО та курси	Розроблення рекламних бізнес-пропозицій

Аналіз виявив ризики конкуренції та обмеження API. Є також і ключові можливості: зростання освітніх технологій, локалізація та інтеграція ІІІ.

Таблиця 5.8 демонструє ступеневий аналіз конкуренції на ринку.

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

№ з/п	Особливості	Прояв характеристики	Вплив
1	Тип конкуренції	Олігополія з диференціацією	Потреба чіткої диференціації
2	Рівень боротьби	Високий	Сильна маркетингова стратегія
3	Галузева ознака	Освітні технології, мовне навчання	Важливість доказової бази ефективності

Кінець таблиці 5.8

4	Конкуренція за типом	Товарно-видова	Можливість позиціонування як комплементарного інструменту
5	Характер переваг	Операційна ефективність	Фокус на технологічній досконалості

Таблиця 5.9 демонструє аналіз конкуренції в галузі за методикою Майкла Портера.

Таблиця 5.9 — Аналіз конкуренції в галузі за М. Портером

№ з/п	Складові	Прямі конкуренти	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
1	Характеристика	Anki, Quizlet, Memrise, LinguaLeo	Компанії освітніх технологій, ШІ-стартапи	DeerL API, хостинг-провайдери	Студенти, професіонали, мандрівники	Словники, підручники, курси, репетитори
2	Інтенсивність	Висока	Середня	Низька (альтернативи доступні)	Середня (легке переключення)	Середня

Аналіз за М. Портером підтверджує високу конкуренцію, але виявляє можливості диференціації.

На основі проведеного аналізу визначається перелік факторів конкурентоспроможності (таблиця 5.10). До них віднесено характеристики продукту, що впливають на сприйняття сервісу користувачами. Виокремлення і цілеспрямований розвиток цих факторів дозволяє сформувати стійкі конкурентні переваги IDEAL порівняно з наявними рішеннями на ринку.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№ з/п	Фактор	Обґрунтування
1	Інтеграція DeepL	Усуває потребу перемикання між застосунками
2	Режими навчання	Чотири режими без ручного налаштування
3	Україномовний інтерфейс	Повна локалізація для української аудиторії
4	Адаптивний алгоритм SM-2	Науково обґрунтований метод
5	Простота	Інтуїтивний інтерфейс
6	Статистика	Візуалізовані графіки за 30 днів
7	Доступна ціна	Freemium з доступною підпискою

Таблиця 5.11 містить порівняльний аналіз сильних та слабких сторін IDEAL відносно ключового конкурента Anki з використанням бальної оцінки.

Таблиця 5.11 — Порівняльна бальна оцінка конкурентних факторів

№ з/п	Фактор конкурентоспроможності	Бали IDEAL (1-20)	Бали Anki (1-20)	Обґрунтування
1	Інтеграція перекладу	18	8	IDEAL має вбудований DeepL, Anki потребує сторонніх плагінів
2	Простота налаштування	17	10	IDEAL пропонує готові режими, Anki вимагає ручної конфігурації
3	Спеціалізація на лексиці	16	12	IDEAL оптимізований для вивчення слів, Anki універсальний
4	Мультиплатформність	11	19	IDEAL обмежений вебверсією, Anki має нативні застосунки
5	Гнучкість контенту	10	18	Anki підтримує мультимедіа та HTML, IDEAL обмежений текстом

Кінець таблиці 5.11

6	Синхронізація	12	18	Anki має AnkiWeb, IDEAL має обмежену синхронізацію
7	Вартість	16	17	Обидва мають безкоштовні опції, але IDEAL планує доступну підписку
8	Статистика	15	16	Обидва надають детальну аналітику
Загальна сума		115	118	

За сумою балів Anki незначно випереджає IDEAL, але стартап-проект має переваги в критичних для початківців факторах: інтеграції перекладу та простоті налаштування.

SWOT-аналіз у таблиці 5.12 узагальнює сильні та слабкі сторони проекту.

Таблиця 5.12 — SWOT-аналіз стартап-проекту

№ з/п	Сильні сторони	Слабкі сторони
1	Інтеграція DeepL API	Відсутність мобільних застосунків
2	Алгоритм SM-2	Обмежений функціонал
3	Чотири режими навчання	Залежність від DeepL API
4	Інтуїтивний інтерфейс	Монолітна архітектура
5	Україномовна локалізація	Відсутність мультимедіа
6	Простий стек технологій	Обмежена синхронізація
7	Візуалізована статистика	Відсутність впізнаваності бренду
№ з/п	Можливості	Загрози
1	Зростання освітніх технологій	Висока конкуренція
2	Партнерства з ЗВО	Низька платоспроможність цільової аудиторії
3	Інтеграція LLM	Обмеження DeepL Free
4	Розширення на інші мови	Швидка зміна трендів
5	Мобільні застосунки	Копіювання ідеї

Кінець таблиці 5.13

6	Геміфікація	Залежність від API
7	Бізнес для бізнесу	

Таблиця 5.13 містить альтернативи ринкового впровадження з оцінкою ймовірності отримання ресурсів.

Таблиця 5.13 — Альтернативи ринкового впровадження стартап-проєкту

№ з/п	Альтернатива впровадження	Ймовірність ресурсів	Строки
1	Ринок бізнес для бізнесу України, freemium	Висока	6-9 міс
2	Партнерство з ЗВО	Середня	12-18 міс
3	Ринок бізнес для бізнесу корпорацій	Середня	9-15 міс
4	Міжнародний ринок	Низька	18-24 міс

Оптимальна альтернатива — український ринок бізнес для бізнесу з freemium-моделлю. Саме для неї висока ймовірність ресурсів, 6-9 міс до мінімально життєздатного продукту. Аналіз показав сприятливі умови завдяки зростаючому попиту та можливостям диференціації.

5.4 Розроблення ринкової стратегії стартап-проєкту

Розроблення ринкової стратегії вебзастосунку передбачає поетапне визначення та оцінювання ключових сегментів цільової аудиторії. Для кожної потенційної групи аналізуються профіль користувачів, готовність сприйняти продукт, орієнтовний обсяг попиту, інтенсивність конкуренції та простота входу в сегмент. Узагальнені результати такого сегментування подано у таблиці 5.14.

Таблиця 5.14 — Вибір цільових груп потенційних споживачів

№ з/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
1	Студенти ВНЗ (18-25 років), що готуються до мовних іспитів	Висока. Активно шукають ефективні інструменти підготовки	Високий. Велика кількість студентів щороку складають іспити з іноземних мов	Висока. Багато конкуруючих рішень	Середня. Потрібен маркетинг через освітні канали
2	Молоді професіонали (25-35 років) у міжнародних компаніях	Висока. Мають мотивацію для кар'єрного зростання	Середній. Обмежений час для навчання	Висока. Конкуренція за увагу зайнятих користувачів	Складна. Потрібен бізнес-орієнт. підхід або таргетована реклама
3	Мандрівники та туристи (20-50 років)	Середня. Шукають швидкі рішення перед подорожжю	Середній. Сезонний попит	Середня. Менше спеціалізованих рішень	Середня. Можливе просування через туристичні канали
Обрані цільові групи: студенти ВНЗ та молоді професіонали (висока готовність сприйняти продукт).					

Для обраних сегментів розробляється базова стратегія розвитку (таблиця 5.15). Вона враховує результати сегментування, рівень конкуренції в кожному з обраних ринкових сегментів та наявні ресурсні можливості команди.

У таблиці подано обрану альтернативу розвитку проєкту, стратегію охоплення ринку та ключові конкурентоспроможні позиції, які формують основу подальшої стратегії спеціалізації та диференціації.

Таблиця 5.15 — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Орієнтація на український ринок бізнес для бізнесу з freemium-моделлю	Стратегія диференційованого маркетингу: фокус на цільові групи студентів та молодих професіоналів	Інтеграція DeepL API, україномовний інтерфейс, попередньо налаштовані режими навчання, простота використання	Стратегія спеціалізації та диференціації

Стратегія конкурентної поведінки визначає позицію IDEAL відносно конкурентів (таблиця 5.16).

Таблиця 5.16 — Визначення стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових користувачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні. Ринок інструментів для вивчення мов є зрілим з усталеними гравцями	Комбінований підхід: залучення нових користувачів (студентів-початківців) та переманювання незадоволених користувачів конкурентів	Частково. Базовий алгоритм розподілених повторень є стандартом галузі, але інтеграція DeepL та режими навчання є унікальними	Стратегія зайняття конкурентної ніші

Стратегія позиціонування, представлена у таблиці 5.17, визначає, як вебзастосунок IDEAL сприйматиметься цільовою аудиторією.

Таблиця 5.17 — Визначення стратегії позиціонування

№ з/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати позицію
1	Простота використання, швидкий старт без складних налаштувань	Спеціалізація та диференціація	Інтеграція DeerpL, попередньо налаштовані режими	«Розумний словник для зайнятих людей»
2	Ефективність навчання на основі науково обґрунтованих методів	Спеціалізація та диференціація	Адаптований алгоритм SM-2	«Навчання, яке працює»
3	Україномовний інтерфейс	Спеціалізація та диференціація	Повна локалізація для української аудиторії	«Створено для українців»

Стратегія зайняття конкурентної ніші через автоматизацію перекладу та україномовну локалізацію дозволить ефективно конкурувати на ринку.

5.5 Розроблення маркетингової програми стартап-проєкту

Маркетингова програма формалізує переваги продукту та стратегію впровадження. Таблиця 5.18 узагальнює ключові переваги концепції вебзастосунку IDEAL.

Таблиця 5.18 — Визначення ключових переваг концепції потенційного товару

№ з/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Швидке створення словників без ручного пошуку перекладів	Вбудована інтеграція з DeepL API для автоматичного перекладу слів	Anki та Quizlet потребують ручного додавання перекладів або встановлення сторонніх плагінів
2	Ефективне запам'ятовування без надмірних витрат часу	Науково обґрунтований алгоритм з адаптивними інтервалами повторень	Система автоматично визначає оптимальний час для повторення кожного слова
3	Простота налаштування для користувачів без технічного досвіду	Чотири попередньо налаштовані режими навчання без потреби у конфігурації	Конкуренти часто вимагають ручного налаштування параметрів алгоритму
4	Україномовний інтерфейс	Повна локалізація інтерфейсу та документації українською	Більшість глобальних конкурентів не підтримують українську мову або мають неповну локалізацію
5	Відстеження прогресу та мотивація внаслідок цього	Детальна статистика з візуалізованими графіками за 30 днів	Демонстрація прогресу підвищує мотивацію користувачів

Таблиця 5.19, в свою чергу, відображає трирівневу модель товару.

Таблиця 5.19 — Опис трьох рівнів моделі товару IDEAL

Рівні товару	Сутність та складові
I. Товар за задумом	Вебзастосунок для вивчення лексики з розподіленим повторенням та автоперекладом
II. Товар у реальному виконанні	Властивості: DeepL API, 4 режими, статистика, мікроповтори Якість: стабільна робота, швидкість, точність перекладу Марка: IDEAL Упаковка: адаптивний веб-інтерфейс
III. Товар із підкріпленням	Захист: авторське право, комерційна таємниця Послуги: техпідтримка, навчальні матеріали, експорт статистики

Цінові межі визначено з урахуванням аналогів та доходів цільової аудиторії представлено у таблиці 5.20.

Таблиця 5.20 — Визначення меж встановлення ціни на товар

Товари-замінники	Товари-аналоги	Доходи цільової аудиторії	Межі ціни
Курси: 3000-8000 грн/міс, репетитори: 300-500 грн/год	Duolingo: 300 грн/міс, Memrise: 640 грн/міс, Anki: безкоштовний	Студенти: 3000-7000 грн/міс, професіонали: 15000-35000 грн/міс	Нижня: 0 грн (freemium) Верхня: 500-640 грн/міс або 6000-7680 грн/рік

Обрана цінова стратегія: freemium (базовий безкоштовний, преміум 149 грн/міс або 1290 грн/рік). Таблиця 5.21 визначає систему збуту.

Таблиця 5.21 — Формування системи збуту для стартап-проекту

Поведінка клієнтів	Функції збуту	Глибина каналу	Система збуту
Онлайн-реєстрація, пробний період	Безкоштовний доступ, реєстрація, платежі, підтримка	Нульовий рівень (прямий)	Власний сайт з LiqPay/Stripe

Таблиця 5.22 визначає концепцію маркетингових комунікацій.

Таблиця 5.22 — Концепція маркетингових комунікацій для стартап-проекту

Поведінка цільової аудиторії	Канали	Позиціонування	Завдання	Концепція
Соцмережі, Google, спільноти	Instagram, Facebook, Telegram, Google Ads, SEO, блог	«Розумний словник», «Навчання, яке працює», «Для українців»	Інформування, демонстрація переваг, спонукання до реєстрації	«Вивчайте слова ефективно з автоперекладом і адаптивним алгоритмом»

Маркетингова програма включає freemium-модель, прямий збут через власний сайт та просування через Google Ads, соцмережі та контент-маркетинг. Основні канали: контекстна реклама, таргетинг у Facebook/Instagram (18-35 років, освіта/саморозвиток), блог про методики запам'ятовування.

Аналіз показав сприятливі умови для входження. Стратегія зайняття ніші на українському бізнес для бізнесу ринку дозволить ефективно конкурувати та залучати клієнтів.

Висновки до розділу 5

У результаті проведеного аналізу стартап-проєкту можна зробити кілька ключових висновків. Вебзастосунок для вивчення іншомовної лексики з інтеграцією машинного перекладу та алгоритмів розподіленого повторення має значний потенціал на українському ринку освітніх технологій, зокрема через зростаючий попит на цифрові інструменти мовного навчання та низькі технологічні бар'єри входу. Визначені сильні сторони проєкту, такі як автоматична інтеграція DeepL API для перекладу, чотири попередньо налаштовані режими навчання, україномовний інтерфейс та науково обґрунтований адаптивний модифікований алгоритм SuperMemo SM-2, створюють передумови для успіху на конкурентному ринку.

Технологічний аудит підтвердив повну здійсненність проєкту, оскільки всі необхідні технології (Python, Django, DeepL API, SQLite) є доступними безкоштовно або з мінімальними витратами на етапі розробки мінімально життєздатного продукту. Обрані технології є зрілими та широко підтримуваними, що знижує технічні ризики впровадження.

Незважаючи на це, існують певні слабкі сторони, зокрема обмежена синхронізація між пристроями, залежність від стороннього API та відсутність впізнаваності бренду на ринку. Однак можливості для розвитку, зокрема через партнерства з освітніми установами, інтеграцію великих мовних моделей для генерації контекстних прикладів, впровадження моделі бізнес для бізнесу для корпоративних клієнтів та розширення на інші мови інтерфейсу, відкривають перспективи для подальшого масштабування стартапу.

Щодо загроз, важливими є висока конкуренція з боку усталених гравців, таких як Anki, Duolingo і Quizlet, обмеження DeepL Free API (500000 символів/міс), низька платоспроможність цільової аудиторії студентів та швидка зміна технологічних трендів у галузі освітніх технологій. Порівняльний аналіз показав, що IDEAL має значні переваги в критичних для початківців факторах — інтеграції перекладу та простоті налаштування.

Розроблена ринкова стратегія передбачає зайняття конкурентної ніші на українському ринку з фокусом на студентів вищих навчальних закладів та молодих професіоналів у міжнародних компаніях. Обрана freemium-модель монетизації (базовий безкоштовний доступ та преміум-підписка 149 грн/міс) є оптимальною для цільової аудиторії з урахуванням рівня доходів. Маркетингова програма включає прямий збут через власний вебсайт з інтегрованими платіжними системами та просування через контекстну рекламу Google Ads, таргетовану рекламу у соціальних мережах та контент-маркетинг.

Отже, для подальшого успіху проєкту важливо зосередитися на швидкому запуску мінімально життєздатного продукту для отримання зворотного зв'язку від користувачів, ефективному позиціонуванні через унікальні переваги автоматизації та локалізації, а також на моніторингу використання DeepL API для своєчасного переходу на комерційний план при зростанні бази користувачів.

ВИСНОВКИ

У результаті виконання магістерської дисертації розроблено вебзастосунок для вивчення іншомовної лексики, який інтегрує можливості машинного перекладу та науково обґрунтований алгоритм розподілених повторень. Система призначена для забезпечення ефективного запам'ятовування лексичних одиниць іноземних мов шляхом автоматизації процесу генерації перекладів та оптимізації графіка повторень навчального матеріалу.

Проведений аналіз існуючих систем вивчення іноземних мов показав, що популярні платформи, такі як Duolingo, Memrise та Anki, мають значні обмеження у контексті інтеграції машинного перекладу з алгоритмами розподілених повторень. Виявлено, що Duolingo та Memrise не надають користувачам можливості самостійно додавати власний контент з автоматичним перекладом, а Anki, незважаючи на гнучкість, потребує ручного введення перекладів або використання сторонніх розширень. Аналіз сервісів машинного перекладу підтвердив переваги DeepL для європейських мовних пар завдяки використанню нейронних мереж та контекстного аналізу, що забезпечує вищу якість перекладу порівняно з альтернативними рішеннями. Це обґрунтувало доцільність створення спеціалізованої системи, яка поєднує переваги DeepL API з науково підтвердженими методами розподілених повторень.

Розроблено архітектуру вебзастосунку на основі модифікованого алгоритму SuperMemo SM-2, який забезпечує адаптивне планування повторень на основі індивідуальних показників засвоєння матеріалу користувачем. Система реалізує автоматичний розрахунок інтервалів повторень з урахуванням коефіцієнта легкості для кожної лексичної одиниці, що дозволяє оптимізувати навантаження на довготривалу пам'ять та підвищити ефективність запам'ятовування. Інтеграція з DeepL API забезпечує отримання високоякісних перекладів у реальному часі з підтримкою контекстного аналізу, що особливо важливо для багатозначних слів та усталених словосполучень.

Здійснено програмну реалізацію системи з використанням мови програмування Python, фреймворку Django для серверної частини та бази даних SQLite3 для зберігання користувацьких даних. Архітектура побудована за принципом модульності, що включає окремі компоненти для роботи з перекладом, управління картками повторень, аутентифікації користувачів та відображення навчального матеріалу. Використання Django забезпечило швидкість розробки, надійність роботи з базою даних та вбудовані механізми безпеки. Клієнтська частина реалізована з використанням стандартних веб-технологій HTML, CSS та JavaScript, що гарантує сумісність з різними браузерами та пристроями.

Проведене тестування підтвердило працездатність та ефективність розробленої системи. Перевірка функціональності показала коректну роботу механізмів інтеграції з DeepL API, точність розрахунку інтервалів повторень за алгоритмом SM-2 та стабільність роботи системи аутентифікації користувачів. Тестування користувацького інтерфейсу продемонструвало зручність навігації та інтуїтивність процесу додавання нових лексичних одиниць і проходження сесій повторень. Результати підтвердили, що система забезпечує безперебійну роботу основних функціональних модулів та відповідає поставленим вимогам щодо продуктивності.

Розроблене програмне забезпечення має широкий потенціал для використання студентами вищих навчальних закладів, викладачами іноземних мов та особами, які здійснюють самостійне вивчення іноземних мов. Система може бути застосована для підготовки до міжнародних іспитів, професійного навчання спеціалізованої термінології, а також для підтримки формального навчального процесу як допоміжний інструмент. Проведений аналіз стартап-проєкту показав наявність ринкових можливостей у сегменті освітніх технологій, а також підтвердив економічну доцільність комерціалізації розробленого рішення.

Подальший розвиток системи доцільно спрямувати на розширення функціональних можливостей, зокрема впровадження різноманітних типів навчальних вправ, що виходять за межі класичних карток повторень, додавання

аудіопідтримки для тренування вимови та сприйняття на слух, реалізацію системи детальної аналітики прогресу навчання з візуалізацією статистичних показників засвоєння матеріалу, а також інтеграцію механізмів автоматичного генерування контекстних прикладів використання лексичних одиниць у реченнях.

Перспективним напрямом є також розроблення адаптивних алгоритмів, які враховують індивідуальні особливості навчання кожного користувача на основі методів машинного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kim S. K., Webb S. The Effects of Spaced Practice on Second Language Learning: A Meta-Analysis. *Language Learning*. 2022. Vol. 72, no. 1. P. 269–319. URL: <https://doi.org/10.1111/lang.12479>.
2. Ebbinghaus H. Memory: A contribution to experimental psychology. Dover, 1964. 123 p.
3. Spaced learning: A review on the use of spaced learning in language teaching and learning / N. M. Noor et al. *Journal of Language and Linguistic Studies*. 2021. Vol. 17, no. 2. P. 1023–1031. URL: <https://doi.org/10.52462/jlls.71>.
4. Gamification in mobile-assisted language learning: a systematic review of Duolingo literature from public release of 2012 to early 2020 / M. Shortt et al. *Computer Assisted Language Learning*. 2021. P. 1–38. URL: <https://doi.org/10.1080/09588221.2021.1933540>.
5. Bicknell K., Brust C., Settles B. How Duolingo's AI Learns what you Need to Learn: The language-learning app tries to emulate a great human tutor. *IEEE Spectrum*. 2023. Vol. 60, no. 3. P. 28–33. URL: <https://doi.org/10.1109/mspec.2023.1006163>
6. Aminatun D., Oktaviani L. Memrise: Promoting Students' Autonomous Learning Skill through Language Learning Application. *Metathesis: Journal of English Language, Literature, and Teaching*. 2019. Vol. 3, no. 2. P. 214. URL: <https://doi.org/10.31002/metathesis.v3i2.1982>.
7. Getting Started with Anki. *Introduction — Anki Manual*. URL: <https://docs.ankiweb.net>.
8. Neural machine translation: A review of methods, resources, and tools / Z. Tan et al. *AI Open*. 2020. Vol. 1. P. 5–21. URL: <https://doi.org/10.1016/j.aiopen.2020.11.001>.
9. Vaswani A., Shazeer N., Parmar N. *ma ih*. Attention Is All You Need // *Advances in Neural Information Processing Systems*. — 2017. — Vol. 30. — P. 5998–6008.
10. Cloud Translation documentation. *Google Cloud Documentation*. URL: <https://cloud.google.com/translate/docs>.
11. DeepL Translator: Official Website. *DeepL SE*. URL: <https://www.deepl.com>.

12. Sebo P., de Lucia S. Performance of machine translators in translating French medical research abstracts to English: A comparative study of DeepL, Google Translate, and CUBBITT. PLOS ONE. 2024. Vol. 19, no. 2. P. e0297183. URL: <https://doi.org/10.1371/journal.pone.0297183>.

13. Колумбет В.П., Донець А. Г., Ольховик Т.О. Підходи до персоналізованої методики вивчення іншомовної лексики у вебзастосунках. Збірник наукових тез з матеріалами III Міжнародної науково-практична конференції «Сучасні аспекти інженерії програмного забезпечення» MoSE-2025, КПІ, 2025. <https://ipzeconf.kpi.ua/mose>.

14. Junczys-Dowmunt M. Microsoft Translator at WMT 2019: Towards Large-Scale Document-Level Neural Machine Translation [Електронний ресурс] // arXiv preprint arXiv:1907.06170 [cs.CL]. — 2019. — Режим доступу: <https://doi.org/10.48550/arXiv.1907.06170>.

15. Amazon Translate — Developer Guide. *Amazon Web Services, Inc.* URL: <https://aws.amazon.com/translate/>.

16. Hutchins W. J. Machine Translation: A Brief History. *Concise History of the Language Sciences*. 1995. P. 431–445. URL: <https://doi.org/10.1016/b978-0-08-042580-1.50066-0>.

17. Machine Translation: An Introductory Guide (Cleary Business Studies) / D. Arnold et al. Blackwell Pub, 1994. 200 p.

18. A statistical approach to language translation / P. Brown et al. *the 12th conference*, Budapest, Hungary, 22–27 August 1988. Morristown, NJ, USA, 1988. URL: <https://doi.org/10.3115/991635.991651>.

19. Koehn P., Och F. J., Marcu D. Statistical phrase-based translation. *the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*, Edmonton, Canada, 27 May — 1 June 2003. Morristown, NJ, USA, 2003. URL: <https://doi.org/10.3115/1073445.1073462>.

20. Koehn P. Statistical Machine Translation. Cambridge University Press, 2010.

21. Koehn P. Neural Machine Translation. Cambridge University Press, 2020. 406 p.

22. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation / K. Cho et al. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar. Stroudsburg, PA, USA, 2014. URL: <https://doi.org/10.3115/v1/d14-1179>.

23. Sherstinsky A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*. 2020. Vol. 404. P. 132306. URL: <https://doi.org/10.1016/j.physd.2019.132306>.

24. Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural Computation*. 1997. Vol. 9, no. 8. P. 1735–1780. URL: <https://doi.org/10.1162/neco.1997.9.8.1735>.

25. Mikolov T., Sutskever I., Chen K., Corrado G., Dean J. Distributed Representations of Words and Phrases and their Compositionality [Електронний ресурс] // arXiv preprint arXiv:1310.4546 [cs.CL]. — 2013. Режим доступу: <https://doi.org/10.48550/arXiv.1310.4546>.

26. Огурцова О., Шевченко О. Нейронний машинний переклад: основні принципи, сильні та слабкі сторони // *European Science*. — 2023. — № 18(4). — С. 86–92. DOI: <https://doi.org/10.30890/2709-2313.2023-18-04-019>.

27. Kalchbrenner N., Blunsom P. Recurrent Continuous Translation Models. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA. Stroudsburg, PA, USA, 2013. P. 1700–1709. URL: <https://doi.org/10.18653/v1/d13-1176>.

28. Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate // *arXiv*. — 2014. — arXiv:1409.0473. — DOI: <https://doi.org/10.48550/arXiv.1409.0473>.

29. Rogers A., Kovaleva O., Rumshisky A. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*. 2020. Vol. 8. P. 842–866. URL: https://doi.org/10.1162/tacl_a_00349.

30. Language Models are Few-shot Multilingual Learners / G. I. Winata et al. *Proceedings of the 1st Workshop on Multilingual Representation Learning*, Punta Cana,

Dominican Republic. Stroudsburg, PA, USA, 2021. URL: <https://doi.org/10.18653/v1/2021.mrl-1.1>.

31. Azure Translator documentation — quickstarts, tutorials, API reference - Foundry Tools. *Microsoft Azure*. URL: <https://learn.microsoft.com/en-us/azure/ai-services/translator/>.

32. Amazon Translate Documentation. *Amazon Web Services, Inc.* URL: <https://aws.amazon.com/translate/>.

33. DeepL API Technical Documentation. *DeepL SE*. URL: <https://www.deepl.com/docs-api>.

34. Klubička F., Toral A., Sánchez-Cartagena V. M. Quantitative fine-grained human evaluation of machine translation systems: a case study on English to Croatian. *Machine Translation*. 2018. Vol. 32, no. 3. P. 195–215. URL: <https://doi.org/10.1007/s10590-018-9214-x>.

35. Sennrich R., Haddow B., Birch A. Controlling Politeness in Neural Machine Translation via Side Constraints. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California. Stroudsburg, PA, USA, 2016. URL: <https://doi.org/10.18653/v1/n16-1005>.

36. deepl-python: Official Python Client for DeepL API. *DeepL SE*. URL: <https://github.com/DeepLcom/deepl-python>.

37. Nation I. S. P. Learning vocabulary in another language. [Wellington, NZ] : School of Linguistics and Applied Language Studies, Victoria University of Wellington, 1999. 465 p.

38. Ebbinghaus H. Memory: A Contribution to Experimental Psychology. *Annals of Neurosciences*. 2013. Vol. 20, no. 4. URL: <https://doi.org/10.5214/ans.0972.7531.200408>.

39. Schmitt N., Schmitt D. Vocabulary in Language Teaching. Cambridge University Press, 2020. 300 p.

40. Nakata T. DOES REPEATED PRACTICE MAKE PERFECT? THE EFFECTS OF WITHIN-SESSION REPEATED RETRIEVAL ON SECOND

LANGUAGE VOCABULARY LEARNING. *Studies in Second Language Acquisition*. 2016. Vol. 39, no. 4. P. 653–679. URL: <https://doi.org/10.1017/s0272263116000280>.

41. Craik F. I. M., Lockhart R. S. Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior*. 1972. Vol. 11, no. 6. P. 671–684. URL: [https://doi.org/10.1016/s0022-5371\(72\)80001-x](https://doi.org/10.1016/s0022-5371(72)80001-x).

42. Laufer B. Incidental vocabulary acquisition in a second language: the construct of task-induced involvement. *Applied Linguistics*. 2001. Vol. 22, no. 1. P. 1–26. URL: <https://doi.org/10.1093/applin/22.1.1>.

43. Nakata T. Computer-assisted second language vocabulary learning in a paired-associate paradigm: a critical investigation of flashcard software. *Computer Assisted Language Learning*. 2011. Vol. 24, no. 1. P. 17–38. URL: <https://doi.org/10.1080/09588221.2010.520675>.

44. Atkinson R. C., Raugh M. R. An application of the mnemonic keyword method to the acquisition of a Russian vocabulary. *Journal of Experimental Psychology: Human Learning and Memory*. 1975. Vol. 1, no. 2. P. 126–133. URL: <https://doi.org/10.1037/0278-7393.1.2.126>.

45. From game design elements to gamefulness / S. Deterding et al. *the 15th International Academic MindTrek Conference*, Tampere, Finland, 28–30 September 2011. New York, New York, USA, 2011. URL: <https://doi.org/10.1145/2181037.2181040>.

46. Cunningham C., Zichermann G. Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. O'Reilly Media, Incorporated, 2011.

47. Enhancing human learning via spaced repetition optimization / B. Tabibian et al. *Proceedings of the National Academy of Sciences*. 2019. Vol. 116, no. 10. P. 3988–3993. URL: <https://doi.org/10.1073/pnas.1815156116>.

48. Kornell N., Bjork R. A. Learning Concepts and Categories. *Psychological Science*. 2008. Vol. 19, no. 6. P. 585–592. URL: <https://doi.org/10.1111/j.1467-9280.2008.02127.x>.

49. Murre J. M. J., Dros J. Replication and Analysis of Ebbinghaus' Forgetting Curve. *PLOS ONE*. 2015. Vol. 10, no. 7. P. e0120644. URL: <https://doi.org/10.1371/journal.pone.0120644>.
50. Woźniak P., Gorzelańczyk E. Optimization of repetition spacing in the practice of learning. *Acta Neurobiologiae Experimentalis*. 1994. Vol. 54, no. 1. P. 59–62. URL: <https://doi.org/10.55782/ane-1994-1003>.
51. Application of a computer to improve the results obtained in working with the SuperMemo method — SuperMemo. *SuperMemo*. URL: <https://www.supermemo.com/en/archives1990-2015/english/ol/sm2>.
52. Improving Students' Learning With Effective Learning Techniques / J. Dunlosky et al. *Psychological Science in the Public Interest*. 2013. Vol. 14, no. 1. P. 4–58. URL: <https://doi.org/10.1177/1529100612453266>.
53. Distributed practice in verbal recall tasks: A review and quantitative synthesis. / N. J. Cepeda et al. *Psychological Bulletin*. 2006. Vol. 132, no. 3. P. 354–380. URL: <https://doi.org/10.1037/0033-2909.132.3.354>.
54. Karpicke J. D., Roediger H. L. The Critical Importance of Retrieval for Learning. *Science*. 2008. Vol. 319, no. 5865. P. 966–968. URL: <https://doi.org/10.1126/science.1152408>.
55. Lutz M. Learning Python. O'Reilly Media, Incorporated, 2013.
56. Peters T. The Zen of Python. *Pro Python*. Berkeley, CA, 2010. P. 301–302. URL: https://doi.org/10.1007/978-1-4302-2758-8_14.
57. Stack Overflow Developer Survey 2023. *Stack Overflow*. URL: <https://survey.stackoverflow.co/2023/>.
58. Django documentation. *Django Software Foundation*. URL: <https://docs.djangoproject.com/>.
59. Begg C., Connolly T. Database Systems: A Practical Approach to Design, Implementation, and Management, Global Edition. Pearson Education, Limited, 2015. 1442 p.
60. Allen G., Owens M. Introducing SQLite. *The Definitive Guide to SQLite*. Berkeley, CA, 2010. P. 1–16. URL: https://doi.org/10.1007/978-1-4302-3226-1_1.

61. SQLite Documentation. *SQLite*. URL: <https://www.sqlite.org/docs.html>.
62. A Vocabulary and Associated APIs for HTML and XHTML. *World Wide Web Consortium (W3C)*. URL: <https://www.w3.org/TR/>.
63. Meyer E. A., Weyl E. CSS: The Definitive Guide. 4th ed. O'Reilly Media, 2017. 1090 p.
64. Marcotte E. Responsive web design. 2nd ed. New York: A Book Apart, 2014. 153 p.
65. Flanagan D. JavaScript: The Definitive Guide. O'Reilly Media, Incorporated, 2006.
66. Bootstrap Documentation. *Bootstrap*. URL: <https://getbootstrap.com/docs/>.

ДОДАТОК А

КОД ПРОГРАМНОГО ЗАСТОСУНКУ

МЕТОДИ РЕАЛІЗАЦІЇ МАШИННОГО ПЕРЕКЛАДУ
ДЛЯ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ

Київ — 2025

Файл ideal/settings.py

```

from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY =

DEBUG = False

ALLOWED_HOSTS = []

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "website",
]

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]

ROOT_URLCONF = "ideal.urls"

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "ideal.wsgi.application"

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.CommonPasswordValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.NumericPasswordValidator",
    },
]

LANGUAGE_CODE = "uk"

TIME_ZONE = "Europe/Kyiv"

```

```

USE_I18N = True

USE_TZ = True

STATIC_URL = "/static/"

STATICFILES_DIRS = [BASE_DIR / "website/static"]

STATIC_ROOT = BASE_DIR / "staticfiles"

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"

LOGIN_URL = '/login/'
LOGOUT_REDIRECT_URL = '/login/'
LOGIN_REDIRECT_URL = '/'

DEEPL_API_KEY =

```

Файл website/deepl_service.py

```

import logging
import requests
from django.conf import settings

logger = logging.getLogger(__name__)

class DeepLService:

    BASE_URL = "https://api-free.deepl.com/v2/translate"

    TIMEOUT = 10

    @classmethod
    def translate(cls, text, target_lang='UK', source_lang=None):

        api_key = getattr(settings, 'DEEPL_API_KEY', None)
        if not api_key:
            logger.error("DeepL API: API ключ не налаштований у settings.py")
            return {
                'success': False,
                'error': 'API ключ не налаштований'
            }

        if not text or not text.strip():
            logger.warning("DeepL API: Спроба перекласти порожній текст")
            return {
                'success': False,
                'error': 'Порожній текст'
            }

        text = text.strip()

        params = {
            'auth_key': api_key,
            'text': text,
            'target_lang': target_lang,
        }

        if source_lang:
            params['source_lang'] = source_lang

        try:
            logger.info("DeepL API: Надсилання запиту для перекладу")

            response = requests.post(
                cls.BASE_URL,
                data=params,
                timeout=cls.TIMEOUT
            )

            if response.status_code == 200:

```

```

data = response.json()

if 'translations' in data and len(data['translations']) > 0:
    translation = data['translations'][0]['text']
    detected_lang = data['translations'][0].get('detected_source_language', 'N/A')

    logger.info("DeepL API: Успішний переклад")

    return {
        'success': True,
        'translation': translation,
        'detected_language': detected_lang
    }
else:
    logger.error("DeepL API: Відповідь не містить перекладів")
    return {
        'success': False,
        'error': 'Некоректна відповідь від API'
    }

elif response.status_code == 400:
    logger.error("DeepL API: Невірний запит (400)")
    return {
        'success': False,
        'error': 'Невірний запит'
    }

elif response.status_code == 403:
    logger.error("DeepL API: Доступ заборонено (403)")
    return {
        'success': False,
        'error': 'Невірний API ключ'
    }

elif response.status_code == 413:
    logger.error("DeepL API: Текст занадто довгий (413)")
    return {
        'success': False,
        'error': 'Текст занадто довгий'
    }

elif response.status_code == 429:
    logger.warning("DeepL API: Перевищено ліміт запитів (429)")
    return {
        'success': False,
        'error': 'Перевищено ліміт запитів'
    }

elif response.status_code == 456:
    logger.error("DeepL API: Вичерпано квоту символів (456)")
    return {
        'success': False,
        'error': 'Вичерпано квоту символів'
    }

elif response.status_code == 503:
    logger.warning("DeepL API: Сервіс тимчасово недоступний (503)")
    return {
        'success': False,
        'error': 'Сервіс тимчасово недоступний'
    }

else:
    logger.error(f"DeepL API: Невідома помилка. Статус: {response.status_code}")
    return {
        'success': False,
        'error': f"Помилка API (код {response.status_code})"
    }

except requests.exceptions.Timeout:
    logger.error(f"DeepL API: Таймаут запиту ({cls.TIMEOUT}s)")
    return {
        'success': False,
        'error': 'Час очікування вичерпано'
    }

except requests.exceptions.ConnectionError:
    logger.error("DeepL API: Помилка з'єднання")
    return {

```

```

        'success': False,
        'error': 'Помилка з\'єднання'
    }

except requests.exceptions.RequestException as e:
    logger.error(f'DeepL API: Помилка запиту: {str(e)}")
    return {
        'success': False,
        'error': 'Помилка запиту'
    }

except Exception as e:
    logger.error(f'DeepL API: Несподівана помилка: {str(e)}")
    return {
        'success': False,
        'error': 'Несподівана помилка'
    }

```

Файл website/models.py

```

from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MinValueValidator, MaxValueValidator
from django.utils import timezone
from .time_utils import get_current_time

```

```
class Dictionary(models.Model):
```

```

    LEARNING_MODES = [
        (1, 'Назавжди (довготривало)'),
        (2, 'До дати іспиту'),
        (3, 'На кілька тижнів (2-4 тижні)'),
        (4, 'Швидко запам\'ятовування (1-7 днів)'),
    ]

```

```

    user = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name='dictionaries',
        verbose_name='Користувач'
    )
    name = models.CharField(
        max_length=200,
        verbose_name='Назва словника'
    )
    learning_mode = models.IntegerField(
        choices=LEARNING_MODES,
        default=1,
        verbose_name='Режим навчання'
    )
    deadline = models.DateTimeField(
        null=True,
        blank=True,
        verbose_name='Дедлайн (для режиму 2)'
    )

```

```

    weeks_duration = models.IntegerField(
        null=True,
        blank=True,
        validators=[MinValueValidator(2), MaxValueValidator(4)],
        verbose_name='Тривалість у тижнях (для режиму 3)'
    )

```

```

    target_date_mode4 = models.DateTimeField(
        null=True,
        blank=True,
        verbose_name='Дата завершення (для режиму 4)'
    )

```

```

    days_duration = models.IntegerField(
        null=True,
        blank=True,
        validators=[MinValueValidator(1), MaxValueValidator(7)],
        verbose_name='Тривалість у днях (для режиму 4)'
    )

```

```

created_at = models.DateTimeField(
    auto_now_add=True,
    verbose_name='Дата створення'
)
started_at = models.DateTimeField(
    null=True,
    blank=True,
    verbose_name='Дата початку навчання'
)
is_active = models.BooleanField(
    default=True,
    verbose_name='Активний'
)

new_words_shown_today = models.IntegerField(
    default=0,
    verbose_name='Нових слів показано сьогодні'
)
new_words_date = models.DateField(
    null=True,
    blank=True,
    verbose_name='Дата відстеження нових слів'
)

class Meta:
    verbose_name = 'Словник'
    verbose_name_plural = 'Словники'
    ordering = ['-created_at']

def __str__(self):
    return f'{self.name} ({self.user.username})'

def get_mode_display_short(self):
    mode_map = {
        1: 'Назавжди',
        2: 'До іспиту',
        3: 'Кілька тижнів',
        4: 'Швидко',
    }
    return mode_map.get(self.learning_mode, 'Невідомо')

def reset_daily_new_words_if_needed(self, request=None):
    today = get_current_time(request).date() if request else timezone.now().date()
    if self.new_words_date != today:
        self.new_words_shown_today = 0
        self.new_words_date = today
        self.save(update_fields=['new_words_shown_today', 'new_words_date'])

def can_show_new_words_today(self, request=None):
    self.reset_daily_new_words_if_needed(request)
    return self.new_words_shown_today < 40

def increment_new_words_today(self, request=None):
    self.reset_daily_new_words_if_needed(request)
    self.new_words_shown_today += 1
    self.save(update_fields=['new_words_shown_today'])

class Word(models.Model):

    dictionary = models.ForeignKey(
        Dictionary,
        on_delete=models.CASCADE,
        related_name='words',
        verbose_name='Словник'
    )
    foreign_word = models.CharField(
        max_length=200,
        verbose_name='Іншомовне слово'
    )
    translation = models.CharField(
        max_length=200,
        verbose_name='Переклад'
    )
    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name='Дата додавання'
    )

```

```

next_review_date = models.DateTimeField(
    null=True,
    blank=True,
    verbose_name='Дата наступного показу',
    db_index=True
)

ease_factor = models.FloatField(
    default=2.0,
    validators=[MinValueValidator(1.3), MaxValueValidator(2.5)],
    verbose_name='Легкість слова'
)

interval_days = models.IntegerField(
    default=0,
    verbose_name='Поточний інтервал (днів)'
)

times_shown = models.IntegerField(
    default=0,
    verbose_name='Показано разів (загалом)'
)

times_shown_today = models.IntegerField(
    default=0,
    verbose_name='Показано разів (сьогодні)'
)

times_correct = models.IntegerField(
    default=0,
    verbose_name='Правильних відповідей ("Добре")'
)

consecutive_correct = models.IntegerField(
    default=0,
    verbose_name='Правильних підряд'
)

is_learned = models.BooleanField(
    default=False,
    verbose_name='Вивчено'
)

last_shown_at = models.DateTimeField(
    null=True,
    blank=True,
    verbose_name='Час останнього показу'
)

last_shown_date = models.DateField(
    null=True,
    blank=True,
    verbose_name='Дата останнього показу'
)

class Meta:
    verbose_name = 'Слово'
    verbose_name_plural = 'Слова'
    ordering = ['foreign_word']
    indexes = [
        models.Index(fields=['next_review_date']),
        models.Index(fields=['dictionary', 'next_review_date']),
    ]

def __str__(self):
    return f'{self.foreign_word} - {self.translation}'

def reset_daily_counter_if_needed(self, request=None):
    today = get_current_time(request).date() if request else timezone.now().date()
    if self.last_shown_date != today:
        self.times_shown_today = 0
        self.last_shown_date = today
        self.save(update_fields=['times_shown_today', 'last_shown_date'])

def can_show_today(self, request=None):
    self.reset_daily_counter_if_needed(request)
    max_shows = 4
    if self.dictionary.learning_mode == 2 and self.dictionary.deadline:
        current_time = get_current_time(request) if request else timezone.now()
        hours_until_deadline = (self.dictionary.deadline - current_time).total_seconds() / 3600
        if hours_until_deadline <= 48:
            max_shows = 5
    return self.times_shown_today < max_shows

```

```

def calculate_initial_ease_factor(self):
    letter_count = sum(1 for c in self.foreign_word if c.isalpha())

    if letter_count <= 4:
        return 2.3
    elif letter_count >= 15:
        return 1.7
    else:
        return 2.0

def save(self, *args, **kwargs):
    is_new = not self.pk

    if is_new and self.ease_factor == 2.0:
        self.ease_factor = self.calculate_initial_ease_factor()

    if is_new and not self.next_review_date:
        self.next_review_date = None

    super().save(*args, **kwargs)

```

```
class WordAttempt(models.Model):
```

```

    RESULT_TYPES = [
        ('good', 'Добре'),
        ('hard', 'Важко'),
        ('error', 'Помилка'),
    ]

    word = models.ForeignKey(
        Word,
        on_delete=models.CASCADE,
        related_name='attempts',
        verbose_name='Слово'
    )
    user_answer = models.TextField(
        verbose_name='Відповідь користувача'
    )
    is_correct = models.BooleanField(
        verbose_name='Правильна відповідь'
    )
    result_type = models.CharField(
        max_length=10,
        choices=RESULT_TYPES,
        verbose_name='Тип результату'
    )
    response_time = models.FloatField(
        verbose_name='Час відповіді (секунд)'
    )
    used_hint = models.BooleanField(
        default=False,
        verbose_name='Використана підказка'
    )
    was_skipped = models.BooleanField(
        default=False,
        verbose_name='Пропущено'
    )
    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name='Час спроби',
        db_index=True
    )
    session_id = models.CharField(
        max_length=100,
        verbose_name='ID сесії',
        db_index=True
    )

    class Meta:
        verbose_name = 'Спроба відповіді'
        verbose_name_plural = 'Спроби відповідей'
        ordering = ['-created_at']
        indexes = [
            models.Index(fields=['word', '-created_at']),
            models.Index(fields=['session_id']),
        ]

    def __str__(self):

```

```

return f'{self.word.foreign_word}: {self.result_type} ({self.created_at.strftime("%Y-%m-%d %H:%M")})'

class LearningSession(models.Model):

    user = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name='learning_sessions',
        verbose_name='Користувач'
    )
    dictionary = models.ForeignKey(
        Dictionary,
        on_delete=models.CASCADE,
        related_name='sessions',
        verbose_name='Словник'
    )
    session_id = models.CharField(
        max_length=100,
        unique=True,
        db_index=True,
        verbose_name='ID сесії'
    )
    started_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name='Час початку'
    )
    ended_at = models.DateTimeField(
        null=True,
        blank=True,
        verbose_name='Час завершення'
    )
    words_shown = models.IntegerField(
        default=0,
        verbose_name='Показано слів'
    )
    total_attempts = models.IntegerField(
        default=0,
        verbose_name='Загальна кількість спроб'
    )
    new_words_count = models.IntegerField(
        default=0,
        verbose_name='Нових слів у сесії'
    )
    is_active = models.BooleanField(
        default=True,
        verbose_name='Активна сесія'
    )

    class Meta:
        verbose_name = 'Навчальна сесія'
        verbose_name_plural = 'Навчальні сесії'
        ordering = ['-started_at']
        indexes = [
            models.Index(fields=['user', '-started_at']),
            models.Index(fields=['session_id']),
        ]

    def __str__(self):
        return f'Сесія {self.dictionary.name} ({self.started_at.strftime("%Y-%m-%d %H:%M")})'

    def get_duration_minutes(self):
        if self.ended_at:
            duration = (self.ended_at - self.started_at).total_seconds() / 60
            return round(duration, 1)
        else:
            duration = (timezone.now() - self.started_at).total_seconds() / 60
            return round(duration, 1)

```

Файл website/learning_utils.py

```

import re
from difflib import SequenceMatcher

```

```

def normalize_answer(text):

    if not text:
        return ""

    text = text.strip().lower()

    text = re.sub(r'^\w\s\''-', ', text, flags=re.UNICODE)

    text = re.sub(r'\s+', ' ', text)

    return text.strip()

def calculate_similarity(text1, text2):

    return SequenceMatcher(None, text1, text2).ratio()

def count_letter_differences(text1, text2):

    matcher = SequenceMatcher(None, text1, text2)
    differences = 0

    for tag, i1, i2, j1, j2 in matcher.get_opcodes():
        if tag != 'equal':
            differences += max(i2 - i1, j2 - j1)

    return differences

def classify_answer(user_answer, correct_answer, response_time, used_hint):

    user_normalized = normalize_answer(user_answer)
    correct_normalized = normalize_answer(correct_answer)

    if used_hint:
        return 'error'

    if response_time > 60:
        return 'error'

    if user_normalized == correct_normalized:
        if response_time <= 20:
            return 'good'
        else:
            return 'hard'

    letter_count = sum(1 for c in correct_answer if c.isalpha())
    if letter_count >= 4:
        differences = count_letter_differences(user_normalized, correct_normalized)
        if differences == 1 and response_time <= 60:
            return 'hard'

    return 'error'

def get_hint_for_word(word):

    letter_count = sum(1 for c in word if c.isalpha())

    words = word.split()
    is_phrase = len(words) > 1

    if is_phrase and letter_count >= 6:
        hint_parts = []
        for w in words:
            if w:
                first_letter = next((c for c in w if c.isalpha()), '_')
                rest = ''.join(' ' if c.isalpha() else c for c in w[1:])
                hint_parts.append(first_letter + rest)
        return ''.join(hint_parts)

    elif letter_count <= 4:
        return word

    elif letter_count <= 8:

```

```

result = []
first_letter_shown = False
for c in word:
    if c.isalpha():
        if not first_letter_shown:
            result.append(c)
            first_letter_shown = True
        else:
            result.append('_')
    else:
        result.append(c)
return ''.join(result)

else:
    result = []
    letters_positions = [(i, c) for i, c in enumerate(word) if c.isalpha()]

    if letters_positions:
        first_pos = letters_positions[0][0]
        last_pos = letters_positions[-1][0]

        for i, c in enumerate(word):
            if c.isalpha():
                if i == first_pos or i == last_pos:
                    result.append(c)
                else:
                    result.append('_')
            else:
                result.append(c)
        else:
            result = list(word)

    return ''.join(result)

def get_word_mask(word):
    return ''.join('_' if c.isalpha() else c for c in word)

```

Файл `website/spaced_repetition.py`

```

from datetime import timedelta, time
from django.utils import timezone
import random
from .time_utils import get_current_time

def update_ease_factor(word, result_type):
    current_ease = word.ease_factor

    if result_type == 'good':
        new_ease = current_ease + 0.20
    elif result_type == 'hard':
        new_ease = current_ease - 0.15
    elif result_type == 'error':
        new_ease = current_ease - 0.30
    else:
        new_ease = current_ease

    new_ease = max(1.3, min(2.5, new_ease))

    return new_ease

def calculate_next_interval(word, result_type):
    consecutive_good = word.consecutive_correct

    if result_type == 'good':
        if consecutive_good == 0:
            return 1
        elif consecutive_good == 1:

```

```

        return 6
    else:
        next_interval = round(word.interval_days * word.ease_factor)
        return max(1, next_interval)

elif result_type == 'hard':

    if consecutive_good == 0:
        return 1
    elif consecutive_good == 1:
        return 6
    else:
        next_interval = round(word.interval_days * word.ease_factor)
        return max(1, next_interval)

elif result_type == 'error':
    return 1

return 1

def schedule_next_review(word, result_type, request=None):

    new_ease = update_ease_factor(word, result_type)
    word.ease_factor = new_ease

    if result_type == 'good':
        word.consecutive_correct += 1
    elif result_type == 'hard':
        word.consecutive_correct += 1
    elif result_type == 'error':
        word.consecutive_correct = 0

    next_interval_days = calculate_next_interval(word, result_type)
    word.interval_days = next_interval_days

    word.next_review_date = get_current_time(request) + timedelta(days=next_interval_days)

    word.save(update_fields=['ease_factor', 'interval_days', 'next_review_date', 'consecutive_correct'])

def get_words_for_learning(dictionary, request=None, limit=None):

    now = get_current_time(request)
    today_start = now.replace(hour=0, minute=0, second=0, microsecond=0)
    today_end = today_start + timedelta(days=1)

    all_words = dictionary.words.filter(is_learned=False)

    all_words = [w for w in all_words if w.can_show_today()]

    overdue = [w for w in all_words if w.next_review_date and w.next_review_date < today_start]

    due_today = [w for w in all_words if w.next_review_date and today_start <= w.next_review_date < today_end]

    new_words = [w for w in all_words if w.next_review_date is None]

    if should_block_new_words_mode2(dictionary, request):
        new_words = []

    elif not dictionary.can_show_new_words_today(request):
        new_words = []

    random.shuffle(overdue)
    random.shuffle(due_today)
    random.shuffle(new_words)

    words_to_learn = overdue + due_today + new_words

    if limit:
        words_to_learn = words_to_learn[:limit]

    return words_to_learn

def alternate_new_and_reviews(words_list):

    new_words = [w for w in words_list if w.times_shown == 0]
    reviews = [w for w in words_list if w.times_shown > 0]

```

```

if not new_words or not reviews:
    return words_list

result = []
max_len = max(len(new_words), len(reviews))

for i in range(max_len):
    if i < len(reviews):
        result.append(reviews[i])
    if i < len(new_words):
        result.append(new_words[i])

return result

def schedule_mode2_exam(word, result_type, dictionary, request=None):

    import random

    new_ease = update_ease_factor(word, result_type)
    word.ease_factor = new_ease

    if result_type in ['good', 'hard']:
        word.consecutive_correct += 1
    elif result_type == 'error':
        word.consecutive_correct = 0

    now = get_current_time(request)
    deadline = dictionary.deadline

    days_until_exam = (deadline - now).days

    if result_type == 'error':
        word.interval_days = 1
        word.next_review_date = now + timedelta(days=1)
    else:
        if days_until_exam >= 7:
            checkpoint1_days = days_until_exam - 7
            checkpoint2_days = days_until_exam - 1
        else:
            if days_until_exam >= 3:
                checkpoint1_days = days_until_exam - 3
                checkpoint2_days = days_until_exam - 1
            else:
                checkpoint1_days = max(1, days_until_exam - 1)
                checkpoint2_days = days_until_exam

    if word.consecutive_correct == 1:
        next_interval = 1
    elif word.consecutive_correct == 2:
        next_interval = min(checkpoint1_days, random.randint(3, 5))
    else:
        if checkpoint1_days > 5:
            next_interval = random.randint(3, 5)
        elif checkpoint1_days > 0:
            next_interval = checkpoint1_days
        elif checkpoint2_days > 0:
            next_interval = checkpoint2_days
        else:
            next_interval = 1

    word.interval_days = next_interval
    word.next_review_date = now + timedelta(days=next_interval)

    word.save(update_fields=['ease_factor', 'interval_days', 'next_review_date', 'consecutive_correct'])

def schedule_mode3_weeks(word, result_type, dictionary, request=None):

    new_ease = update_ease_factor(word, result_type)
    word.ease_factor = new_ease

    if result_type in ['good', 'hard']:
        word.consecutive_correct += 1
    elif result_type == 'error':
        word.consecutive_correct = 0

    now = get_current_time(request)

```

```

if dictionary.started_at and dictionary.weeks_duration:
    end_date = dictionary.started_at + timedelta(weeks=dictionary.weeks_duration)
    days_until_end = (end_date - now).days
else:
    days_until_end = 21

if result_type == 'error':
    word.interval_days = 1
    word.next_review_date = now + timedelta(days=1)
else:

    if word.consecutive_correct == 1:
        next_interval = 1
    elif word.consecutive_correct == 2:
        next_interval = 6
    else:
        if days_until_end > 14:
            next_interval = 7
        elif days_until_end > 7:
            next_interval = days_until_end - 7
        else:
            next_interval = max(1, days_until_end - 2)

    word.interval_days = next_interval
    word.next_review_date = now + timedelta(days=next_interval)

word.save(update_fields=['ease_factor', 'interval_days', 'next_review_date', 'consecutive_correct'])

def schedule_mode4_quick(word, result_type, dictionary, request=None):

    import random

    new_ease = update_ease_factor(word, result_type)
    word.ease_factor = new_ease

    if result_type in ['good', 'hard']:
        word.consecutive_correct += 1
    elif result_type == 'error':
        word.consecutive_correct = 0

    now = get_current_time(request)

    if dictionary.target_date_mode4:
        end_date = dictionary.target_date_mode4
        days_until_end = (end_date - now).days
    elif dictionary.started_at and dictionary.days_duration:
        end_date = dictionary.started_at + timedelta(days=dictionary.days_duration)
        days_until_end = (end_date - now).days
    else:
        days_until_end = 7

    if result_type == 'error':
        word.interval_days = 1
        word.next_review_date = now + timedelta(days=1)
    else:

        if word.consecutive_correct == 1:
            next_interval = 1
        elif word.consecutive_correct == 2 and days_until_end >= 5:
            next_interval = random.randint(3, 4)
        else:
            if days_until_end > 1:
                next_interval = max(1, days_until_end - 1)
            else:
                next_interval = days_until_end if days_until_end > 0 else 1

        word.interval_days = next_interval
        word.next_review_date = now + timedelta(days=next_interval)

    word.save(update_fields=['ease_factor', 'interval_days', 'next_review_date', 'consecutive_correct'])

def schedule_next_review_by_mode(word, result_type, request=None):

    dictionary = word.dictionary
    learning_mode = dictionary.learning_mode

```

```

if learning_mode == 1:
    schedule_next_review(word, result_type, request)
elif learning_mode == 2:
    schedule_mode2_exam(word, result_type, dictionary, request)
elif learning_mode == 3:
    schedule_mode3_weeks(word, result_type, dictionary, request)
elif learning_mode == 4:
    schedule_mode4_quick(word, result_type, dictionary, request)
else:
    schedule_next_review(word, result_type, request)

def should_block_new_words_mode2(dictionary, request=None):

    if dictionary.learning_mode != 2 or not dictionary.deadline:
        return False

    now = get_current_time(request)
    days_until_deadline = (dictionary.deadline - now).days

    return days_until_deadline < 3

def check_and_convert_mode4_to_mode1(dictionary, request=None):

    if dictionary.learning_mode != 4:
        return False

    if dictionary.target_date_mode4:
        end_date = dictionary.target_date_mode4
    elif dictionary.started_at and dictionary.days_duration:
        end_date = dictionary.started_at + timedelta(days=dictionary.days_duration)
    else:
        return False

    now = get_current_time(request)

    if now.date() >= end_date.date():
        dictionary.learning_mode = 1
        dictionary.save(update_fields=['learning_mode'])
        return True

    return False

```

Файл website/views.py (фрагменты)

```

from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.http import JsonResponse
from django.views.decorators.http import require_POST
import json
from .models import Dictionary, Word, WordAttempt
from .learning_utils import normalize_answer, classify_answer
from .spaced_repetition import schedule_next_review_by_mode, get_words_for_learning
from .deepl_service import DeepLService

@login_required
@require_POST
def learn_check_answer(request, dictionary_id, word_id):
    dictionary = get_object_or_404(Dictionary, id=dictionary_id, user=request.user)
    word = get_object_or_404(Word, id=word_id, dictionary=dictionary)

    user_answer = request.POST.get('answer', '').strip()
    response_time = float(request.POST.get('response_time', 0))
    used_hint = request.POST.get('used_hint', 'false') == 'true'
    session_id = request.POST.get('session_id', '')

    result_type = classify_answer(user_answer, word.translation, response_time, used_hint)

    user_normalized = normalize_answer(user_answer)
    correct_normalized = normalize_answer(word.translation)
    is_correct = user_normalized == correct_normalized

```

```

attempt = WordAttempt.objects.create(
    word=word,
    user_answer=user_answer,
    is_correct=is_correct,
    result_type=result_type,
    response_time=response_time,
    used_hint=used_hint,
    session_id=session_id,
)

was_new_word = (word.times_shown == 0)
if was_new_word:
    dictionary.increment_new_words_today(request)

word.times_shown += 1
if result_type == 'good':
    word.times_correct += 1
word.save(update_fields=['times_shown', 'times_correct'])

schedule_next_review_by_mode(word, result_type, request)

result_labels = {'good': 'Добре', 'hard': 'Важко', 'error': 'Помилка'}

return JsonResponse({
    'success': True,
    'is_correct': is_correct,
    'result_type': result_type,
    'result_label': result_labels.get(result_type, ""),
    'correct_answer': word.translation,
    'user_answer': user_answer,
})

@login_required
@require_POST
def translate_word_ajax(request, dictionary_id):
    dictionary = get_object_or_404(Dictionary, id=dictionary_id, user=request.user)

    try:
        data = json.loads(request.body)
        text = data.get('text', "").strip()

        if not text:
            return JsonResponse({'success': False, 'error': 'Порожній текст для перекладу'})

        result = DeepLService.translate(text, target_lang='UK')

        if result['success']:
            return JsonResponse({
                'success': True,
                'translation': result['translation'],
                'detected_language': result.get('detected_language', 'N/A')
            })
        else:
            return JsonResponse({
                'success': False,
                'error': f'Будь ласка, введіть переклад слова "{text}", оскільки перекладач не впорався.',
                'friendly_error': True
            })

    except json.JSONDecodeError:
        return JsonResponse({'success': False, 'error': 'Некоректний формат даних'})
    except Exception as e:
        return JsonResponse({'success': False, 'error': 'Виникла помилка при перекладі'})

@login_required
def learn_start(request, dictionary_id):
    dictionary = get_object_or_404(Dictionary, id=dictionary_id, user=request.user)

    total_words = dictionary.words.count()
    if total_words == 0:
        return redirect('dictionary_detail', dictionary_id=dictionary.id)

    if not dictionary.started_at:
        dictionary.started_at = get_current_time(request)
        dictionary.save(update_fields=['started_at'])

```

```

words_to_learn_objs = get_words_for_learning(dictionary, request)

if not words_to_learn_objs:
    return redirect('dictionary_detail', dictionary_id=dictionary.id)

max_words_per_session = 60 if dictionary.learning_mode == 2 else 40
words_to_learn_objs = words_to_learn_objs[:max_words_per_session]

new_words_count = sum(1 for w in words_to_learn_objs if w.times_shown == 0)
if new_words_count > 20:
    filtered_words = []
    new_count = 0
    for word in words_to_learn_objs:
        if word.times_shown == 0:
            if new_count < 20:
                filtered_words.append(word)
                new_count += 1
            else:
                filtered_words.append(word)
    words_to_learn_objs = filtered_words

words_to_learn = [word.id for word in words_to_learn_objs]
request.session['words_to_learn'] = words_to_learn
request.session['words_shown'] = []
request.session['new_words_count'] = sum(1 for w in words_to_learn_objs if w.times_shown == 0)

return redirect('learn_next_word', dictionary_id=dictionary.id)

```

ДОДАТОК Б

МЕТОДИ РЕАЛІЗАЦІЇ МАШИННОГО ПЕРЕКЛАДУ ДЛЯ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ

Аппробації наукових досліджень

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТР-43мп

Аркушів 3

Київ — 2025

