

Детектор Плагиата v. 2867 - Отчёт оригинальности: 15.06.2025 16:06:35

Проанализированный документ: Бак\_пояснювальна\_Голофаєв.docx Лицензия: ВОЛОДИМИР МАТІЄВСЬКИЙ

? Тип поиска: Поиск переписанного ? Язык: Uk

? Тип проверки: Интернет

ТЭЕ и кодировка: DocX n/a

Детальный анализ тела документа:







? Диаграмма соотношения частей:



? Граф распределения зон:







? Источники плагиата: 7

	→ 1%		70	1. <a href="https://grade.ua/uk/blog/prilozheniya-dlya-zapominaniya-angliyskikh-slov-podborka-programm/">https://grade.ua/uk/blog/prilozheniya-dlya-zapominaniya-angliyskikh-slov-podborka-programm/</a>
	→ 0,7%		50	2. <a href="https://cambridge.ua/uk/blog/7-best-vocabulary-apps/">https://cambridge.ua/uk/blog/7-best-vocabulary-apps/</a>
	→ 0,4%		30	3. <a href="https://uk.wikipedia.org/wiki/Anki">https://uk.wikipedia.org/wiki/Anki</a>

? Детали обработанных ресурсов: 185 - ОК / 7 - Ошибка

? Важные замечания:

Википедия:	Google Книги:	Сервисы платных работ:	Античит:
			
Обнаружена Wiki!	[не обнаружено]	[не обнаружено]	Обнаружено сокрытие!

? Античит-отчет UACE:

1. Статус: Анализатор <b>Включен</b> Нормализатор <b>Включен</b> сходство символов установлено на <b>100%</b>
2. Обнаруженный процент загрязнения UniCode: <b>11,1%</b> с лимитом: 4%
3. Процент нераспознанных символов после нормализации: <b>6,5%</b>
4. Все подозрительные символы будут отмечены фиолетовым цветом: <b>Abcd...</b>
5. Найдены невидимые символы: 0
Рекомендации по оценке: Особое внимание следует уделить анализу этого отчета! Предполагается, что этот документ содержит значительное количество символов, чуждых языку документа. Это прямое указание на то, что автор

документа использовал специальное программное обеспечение\онлайн-веб-сервис, чтобы эффективно скрыть текст в попытке избежать обнаружения потенциального плагиата. Настоятельно рекомендуется передать это дело на более высокий уровень! В случае сомнений обращайтесь: в службу поддержки Детектора плагиата!

Алфавитная статистика и анализ символов:

 Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

 Исключённые ресурсы:

URL не найдены

 Включённые ресурсы:

URL не найдены

## ? Детальный анализ документа:

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЗ

Цитування: 0,09%

id: 1

«ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Навчально-науковий інститут математики та інформаційних технологій (назва факультету, інституту) Кафедра інформаційних технологій та систем (назва кафедри) Пояснювальна записка до кваліфікаційної роботи за першим (бакалаврським) рівнем освіти на тему:

WINDOWS FORMS ЗАСТОСУНОК ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКИХ СЛІВ ЗА МЕТОДИКОЮ SPACED REPETITION Виконав: здобувач вищої освіти 4 курсу спеціальності 121

Цитування: 0,04%

id: 2

«Інженерія програмного забезпечення»

(шифр і назва напрямку підготовки, спеціальності) \_\_\_\_\_ Егор ГАЛАФАЄВ (прізвище та ініціали) Керівник \_\_\_\_\_ Микола СЕМЕНОВ (прізвище та ініціали) Рецензент \_\_\_\_\_ Владислав ІЩЕНКО (прізвище та ініціали) Полтава - 2025 ЗМІСТ ВСТУПЗ РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ МЕТОДИК ТА ІНСТРУМЕНТІВ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ ЛЕКСИКИ6 1.1. Аналіз традиційних та сучасних методик вивчення лексики англійської мови6 1.2 Теоретичні основи методики spaced repetition10 1.3 Функціональні вимоги до застосунку вивчення англійських слів18 РОЗДІЛ 2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ25 2.1 Архітектурна модель застосунку25 2.2 Вибір інструментів для розробки, реалізації та зберігання даних32 2.3 Опис модифікованого алгоритму SuperMemo для обчислення інтервалів повторення36 РОЗДІЛ 3. РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ТА ЙОГО АПРОБАЦІЯ38 3.1 Процес реалізації Windows-Forms застосунку38 3.2 Опис інтерфейсу застосунку46 3.3. Налаштування та тестування51 ЗАГАЛЬНІ ВИСНОВКИ54 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ56 ДОДАТОК А58 ДОДАТОК Б61 ДОДАТОК В63 ДОДАТОК Г66 ДОДАТОК Д73 ДОДАТОК Ж76 ВСТУП Сучасні тенденції глобалізації та активне поширення інформаційних технологій зумовлюють необхідність володіння англійською мовою як інструментом міжнародного спілкування, професійного розвитку та отримання інформації з іноземних джерел. Традиційні підходи до вивчення іноземної мови, які орієнтовані на механічне заучування слів, часто виявляються малоефективними. Водночас застосування інформаційних технологій дозволяє суттєво підвищити ефективність навчання завдяки впровадженню методик, які враховують особливості роботи людської пам'яті та сприйняття інформації. На сьогодні відомо низку методик вивчення англійської мови, серед яких: механічне заучування (метод повторення), метод карток (flashcards), інтерактивні тести, мнемотехніка, метод інтервальних повторень (spaced repetition) тощо. Саме методика spaced repetition, запропонована в межах теорії кривої забування Еббінгауза, посідає важливе місце серед сучасних підходів, оскільки враховує психологічні особливості пам'яті та забезпечує значне підвищення ефективності навчання завдяки оптимально підібраним інтервалам повторення інформації. Попри високу ефективність, ця методика недостатньо широко представлена в програмних продуктах, особливо серед настільних (desktop) додатків для персональних комп'ютерів під керуванням Windows. Більшість існуючих застосунків або орієнтовані на мобільні платформи, або не враховують специфіки українськомовного користувача. Таким чином, виникає актуальна проблема розробки зручного та ефективного застосунку на платформі Windows, що реалізує методику spaced repetition

Обнаружен Плагиат: 0,21% <https://grade.ua/uk/blog/prilozheniya...> + 2 ресурсов!

id: 3

для вивчення англійських слів. Мета роботи - розробити Windows Forms застосунок для вивчення англійських слів,

який використовує методику spaced repetition для оптимізації процесу запам'ятовування. Для досягнення поставленої мети визначено завдання: провести аналіз існуючих методик та програмних засобів для вивчення іноземних слів; дослідити сутність та переваги методики spaced repetition; визначити основні вимоги до застосунку; проєктувати загальну архітектуру та модель застосунку; здійснити вибір засобів реалізації (Windows Forms, мова програмування C#, система зберігання даних); реалізувати та протестувати застосунок відповідно до сформованих вимог. Об'єктом дослідження є процес вивчення англійських слів за допомогою спеціалізованих програмних засобів. Предметом дослідження є методика spaced repetition та її програмна реалізація у Windows Forms-застосунку. Методи

дослідження – теоретичний аналіз предметної області; порівняльний аналіз різних методик вивчення англійських слів; системний, вимірний підхід до розроблення, використання та супроводу програмного забезпечення; моделювання системи (архітектури програмного забезпечення); емпіричні методи оцінки надійності програмного забезпечення та інші методи інженерії програмного забезпечення. Бакалаврська робота складається з трьох розділів. У першому розділі здійснено аналіз сучасних методик вивчення англійської мови, зокрема розглянуто сутність, особливості та переваги методики [spaced repetition](#). Проведено порівняльний огляд наявних програмних засобів, які реалізують зазначену методику. У другому розділі наведено опис архітектури та моделі запропонованого застосунку, визначено функціональні вимоги до системи, представлено обґрунтування вибору засобів та інструментів розробки, таких як [Windows Forms](#), мова програмування [C#](#) та база даних [SQLite](#). Третій розділ присвячено безпосередньо розробці застосунку, описано процес реалізації ключових функцій, зокрема алгоритму [spaced repetition](#), створення графічного інтерфейсу та організації зберігання даних. Також у цьому розділі наведено результати тестування розробленого застосунку та аналіз його працездатності.

## РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ МЕТОДИК ТА ІНСТРУМЕНТІВ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ ЛЕКСИКИ

### 1.1. Аналіз традиційних та сучасних методик вивчення лексики англійської мови

Процес засвоєння іноземної лексики є одним із найскладніших аспектів при вивченні англійської мови. Існує низка традиційних та сучасних методик, кожна з яких має свої особливості, переваги та недоліки, зокрема з точки зору їх програмної реалізації. Однією з найдавніших методик є

Цитування: 0,03%

id: 4

«метод асоціацій»,

який полягає у встановленні зв'язку між новими словами та вже знайомими поняттями або образами. З програмної точки зору метод асоціацій реалізується через створення інтерактивних візуальних підказок або ілюстрацій. Приклади програм, які використовують цей метод, це [Rosetta Stone](#) (<https://eu.rosettastone.com/>) або [Drops](#) (рис. 1.2, <https://language Drops.com/>). Хоча метод асоціацій простий та наочний, його основний недолік полягає у значній суб'єктивності асоціацій, що робить уніфіковану програмну реалізацію складною та не завжди ефективною. Програма [Rosetta Stone](#) реалізує метод асоціацій через поєднання тексту, зображень і звукового супроводу, завдяки чому користувач засвоює лексику та граматику інтуїтивно — без потреби у словниках або перекладі. Система побудована таким чином, щоб навчання відбувалося природним способом, подібно до того, як діти вивчають рідну мову. Зі зростанням мовної компетентності користувача завдання поступово ускладнюються. Одним із прикладів є вправи, в яких на екрані відображаються чотири зображення (рис. 1.1), а користувач слухає або читає фразу, яка описує одне з них. Його завдання — обрати зображення, що найкраще відповідає цьому опису. В інших вправах необхідно самостійно завершити опис до зображення, орієнтуючись на контекст. Рис. 1.1. Вивчення англійських слів в [Rosetta Stone](#) [6] Під час виконання письмових завдань програма надає екранну клавіатуру, яка особливо корисна при введенні символів, відмінних від латиниці. За наявності мікрофона програма також аналізує вимову користувача. Існує можливість налаштування чутливості системи до вимови, що дозволяє поступово вдосконалювати мовлення та тренувати точність артикуляції. Рис. 1.2. Вивчення англійських слів в [Drops](#) [7]

Цитування: 0,03%

id: 5

«Метод мнемотехніки»

є іншим класичним підходом, що базується на спеціально підібраних прийомах для полегшення запам'ятовування слів. Прикладом програмної реалізації є застосунки, які створюють віршовані чи римовані фрази, або звукові асоціації. Незважаючи на ефективність для запам'ятовування окремих складних слів, цей метод погано масштабується на великі обсяги лексики, оскільки його ефективність залежить від творчих здібностей користувача або авторів контенту. Одним із відомих науковців, які досліджували мнемотехніки для вивчення іноземних мов, є Тоні Бьюзен ([Tony Buzan](#)). Він популяризував використання мнемотехнік через свої книги та лекції, зокрема, у контексті вивчення мов [1;2]. Серед сучасних досліджень звернемо увагу на роботу Дена Джейкоба Лонга та Девіда Карлсона ([Long, Dan Jacob and Carlson, David](#)) [4]. Також варто згадати дослідження українських науковців, таких як О.В. Крекотень, Л.І. Байдак та А.С. Чирва, які вивчали практичне застосування мнемотехнік у процесі вивчення іноземних мов у вищих

навчальних закладах [3]. Опишемо стисло етапи мнемотехніки. Перший етап — кодування. На цьому етапі слово перетворюється на уявний образ. Якщо мова йде про конкретні предмети, процес простий: наприклад, слово

Цитування: 0,01%

id: 6

«яблуко»

викликає чіткий зоровий образ самого яблука. Проте для абстрактних понять необхідно створити асоціацію самостійно. Наприклад, радість може асоціюватися з сонцем, яке світить яскраво; сум може асоціюватися з дощовою хмарою; любов може асоціюватися з серцем. Другий етап — утримання в пам'яті. На цьому етапі важливо кілька разів відтворити у свідомості створені образи й утримувати їх щонайменше 6 секунд. Третій етап — закріплення асоціативних зв'язків. Особливо важливим є вміння правильно поєднувати слово з його смисловим образом. Ще одним поширеним підходом є використання

Цитування: 0,04%

id: 7

«тематичних списків слів»,

які групують лексику за певними темами або категоріями. Подібна організація інформації є типовою для навчальних онлайн-платформ на зразок [Busuu](https://www.busuu.com/) (<https://www.busuu.com/>) або [Babbel](https://ua.babbel.com/) (<https://ua.babbel.com/>). Програмна реалізація цього методу достатньо проста, але має суттєвий недолік: лексика часто вивчається ізольовано від контексту, що ускладнює її подальше застосування у реальному спілкуванні. Цікавим прикладом тематичних списків слів є списки словника Кембриджського університету [5]. Більш сучасним методом є

Цитування: 0,04%

id: 8

«навчання за контекстом»,

який передбачає запам'ятовування слів у межах цілісних фраз або речень. Програмна реалізація методу контекстного навчання є більш складною, але й ефективнішою завдяки застосуванню адаптивних алгоритмів, наприклад, у застосунку [Lingvist](https://lingvist.com/) (<https://lingvist.com/>). Проте головним викликом залишається підбір адекватного та зрозумілого контексту для широкої аудиторії. Цікавим є той факт, що для вивчення при такій методиці достатньо лише аудіювання, що доводить метод

Цитування: 0,03%

id: 9

«[Effortless English](https://www.ajhogeclub.com/)»

(<https://www.ajhogeclub.com/>).

Цитування: 0,06%

id: 10

«Метод карток ([flashcards](#))»

отримав значне поширення завдяки простоті та ефективності повторення. Класичною реалізацією є застосунок [Anki](#), який дозволяє повторювати слова за спеціальними картками, визначаючи ступінь складності згадування. Попри свою популярність, метод [flashcards](#) часто не враховує індивідуальні особливості пам'яті користувача, оскільки інтервали повторень встановлюються за стандартними алгоритмами без глибокої адаптації до конкретної людини. Метод [flashcards](#) працює за наведеними нижче схемами. Створення карток: на одній стороні картки пишеться слово англійською мовою, а на іншій – його переклад або визначення рідною мовою. Можна також додати приклад речення або зображення для кращого запам'ятовування. Активне повторення: картки використовуються для регулярного повторення. Ви переглядаєте картки, намагаючись згадати значення слова або його переклад. Якщо ви правильно відповіли, картка відкладається в одну купу, якщо ні – в іншу. Система Лейтнера: це популярна техніка для організації повторення карток. Картки розподіляються по кількох коробках або відділеннях. Якщо ви правильно відповіли на картку, вона переміщується в наступну коробку, де її повторюють рідше. Якщо відповідь була неправильною, картка повертається в першу коробку для частішого повторення. Існує багато додатків для смартфонів, які автоматизують процес створення та повторення карток. Картки можна використовувати в групах або парах, де один учасник показує слово, а інший намагається його перекласти або пояснити. Це додає елемент гри та взаємодії. Таким чином, проведений аналіз свідчить, що більшість традиційних і навіть деяких сучасних методик вивчення англійської лексики або мають суттєві недоліки при програмній реалізації, або не враховують індивідуальні особливості пам'яті користувача, що знижує загальну ефективність навчання.

З огляду на це, найбільш перспективними є методики, що адаптуються до індивідуальних характеристик користувачів, зокрема методика [spaced repetition](#), яка дозволяє оптимізувати процес запам'ятовування відповідно до психологічних особливостей конкретної людини. 1.2 Теоретичні основи методики [spaced repetition](#) Метод [spaced repetition](#) (розподілених повторень) є потужним інструментом для оптимізації процесу запам'ятовування. Він базується на повторенні інформації з поступово збільшуваними інтервалами, що дозволяє закріпити знання в довготривалій пам'яті. Метод розподілених повторень ґрунтується на дослідженнях німецького психолога Германа Еббінгауза, який у 1880-х роках вивчав процес забування. Еббінгауз створив криву забування (рис. 1.3), яка показує, як швидко ми забуваємо нову інформацію [8]. Він виявив, що найбільше забування відбувається протягом перших годин після вивчення, а потім процес сповільнюється [9]. Однак, якщо інформацію повторювати перед тим, як вона буде забута, вона залишається в пам'яті довше [8]. Рис. 1.3. Крива Германа Еббінгауза [8] Алгоритми [SuperMemo](#) — це один із перших алгоритмів розподілених повторень, розроблений Пьотром Вожняком у 1980-х роках. Алгоритм [SuperMemo](#) використовує інформацію про те, наскільки добре ви запам'ятали матеріал, щоб визначити оптимальні моменти для повторення [10]. Наприклад, якщо ви добре пам'ятаєте слово, інтервал між повтореннями збільшується, якщо ні — зменшується [11]. Остання версія алгоритму, [SM-18](#), є найсучаснішою і використовується в усіх продуктах [SuperMemo](#) [12]. Алгоритми [Anki](#) — це популярна програма для створення карток з розподіленими повтореннями. Вона використовує модифікований алгоритм [SM2](#), який був розроблений для [SuperMemo](#) [13]. [Anki](#) дозволяє користувачам створювати картки з текстом, зображеннями, звуками та відео, а також автоматично визначає інтервали між повтореннями на основі успішності запам'ятовування [14].

 **Обнаружен Плагиат: 0,21%** <https://uk.wikipedia.org/wiki/Anki>

id: 11

У версії 23.10 було додано новий алгоритм [FSRS](#), який користувачі можуть ввімкнути у налаштуваннях

[13]. Переваги методу [spaced repetition](#): Покращення довготривалої пам'яті: метод розподілених повторень дозволяє закріпити інформацію в довготривалій пам'яті, що робить її доступною протягом тривалого часу. Ефективне використання часу: завдяки оптимізації інтервалів між повтореннями, ви витрачаєте менше часу на повторення вже відомої інформації і більше на те, що потребує закріплення. Персоналізація навчання: алгоритми, такі як [SuperMemo](#) та [Anki](#), адаптуються до індивідуальних потреб учня, що робить процес навчання більш ефективним. Зменшення забування: регулярні повторення перед тим, як інформація буде забута, значно зменшують швидкість забування. Метод [spaced repetition](#) є значно ефективнішим у порівнянні з традиційними методами, такими як зубріння або одноразове повторення, оскільки він враховує природні процеси забування і оптимізує час повторення для максимального запам'ятовування. Алгоритм [SM-18](#) є найновішою версією алгоритму розподілених повторень, розробленого для [SuperMemo](#) [15]. Він базується на моделі оцінки складності елементів, яка враховує зміни в складності під час навчання. Опишемо алгоритм [SM-18](#) за допомогою діаграми діяльності [UML](#), що зображена на рис. 1.4. Рис. 1.4. Представлення алгоритму [SM-18](#) діаграмою діяльності [UML](#) Як бачимо з рис. 1.4 алгоритм [SM-18](#) працює за схемою. Ініціалізація: встановлюються початкові параметри для інтервалу повторення та оцінки складності елемента. Повторення: оновлюється оцінка складності елемента. оцінюється запам'ятовуваність ([retrievability](#)), стабільність запам'ятовування та інтерференція. Оновлення інтервалу: якщо запам'ятовуваність висока, інтервал повторення збільшується, якщо ні — зменшується. Оновлення матриць пам'яті: оновлюються матриці пам'яті (матриця стабілізації [Slnc\[\]](#), матриця запам'ятовування [Recall\[\]](#)), які використовуються в обчисленнях. [Anki](#) використовує алгоритми розподілених повторень для оптимізації процесу запам'ятовування. Найпопулярнішими алгоритмами є [SM2](#) та [FSRS](#). Алгоритм [SM2](#) був розроблений для [SuperMemo](#) і використовується в [Anki](#) для визначення інтервалів повторення на основі успішності запам'ятовування. Його основні кроки: 1. Ініціалізація: - встановити початковий інтервал повторення. - встановити початкову оцінку складності елемента. 2. Повторення: - оцінити запам'ятовуваність ([retrievability](#)). - оновити інтервал повторення на основі успішності запам'ятовування. 3. Оновлення інтервалу: - якщо запам'ятовуваність висока, інтервал збільшується. - якщо запам'ятовуваність низька, інтервал зменшується. Діаграму діяльності для алгоритму [SM2](#) представлено на рис. 1.5. Рис. 1.5. Діаграма діяльності [UML](#) для алгоритму [SM2](#) Алгоритм [FSRS](#) ([Free Spaced Repetition](#))

**System**) — це новий алгоритм, який був доданий у версії **Anki** 23.10 (дивись рис. 1.6). Він використовує більш складні моделі для визначення оптимальних інтервалів повторення: 1. Ініціалізація: встановити початковий інтервал повторення. встановити початкову оцінку складності елемента. 2. Повторення: оцінити запам'ятовуваність (**retrievability**). оцінити стабільність запам'ятовування. оцінити інтерференцію. 3. Оновлення інтервалу: визначити оптимальний наступний інтервал повторення на основі кривої стабілізації. оновити матриці пам'яті. Рис. 1.6. Діаграма діяльності для алгоритму **FSRS** Для порівняння алгоритмів **SM-18**, **SM2** та **FSRS**, розглянемо такі критерії: Складність моделі: Наскільки складною є модель алгоритму. Адаптивність: Здатність алгоритму адаптуватися до індивідуальних потреб користувача. Ефективність запам'ятовування: Як добре алгоритм допомагає запам'ятовувати інформацію. Гнучкість налаштувань: Можливість налаштування параметрів алгоритму. Відкритість коду: Чи є алгоритм відкритим для модифікацій та досліджень. Підтримка та оновлення: Наявність підтримки та регулярних оновлень. За цими критеріями на основі досліджень [16;17; 18], сформуємо порівняльну таблицю 1.1. Таблиця 1.1 Порівняльна таблиця алгоритмів **SM-18**, **SM2** та **FSRS** Критерій **SM-18** **SM2** **FSRS**

Складність моделі	Висока	Низька	Середня
Адаптивність	Висока	Низька	Висока
Ефективність запам'ятовування	Дуже висока	Середня	Висока
Гнучкість налаштувань	Висока	Низька	Висока
Відкритість коду	Закритий	Відкритий	Відкритий
Підтримка та оновлення	Регулярні оновлення	Обмежена підтримка	Регулярні оновлення

На основі порівняння у табл. 1.1 зробимо такі проміжні висновки: **SM-18**: Це найсучасніший і найскладніший алгоритм, який забезпечує високу адаптивність та ефективність запам'ятовування. Він підходить для користувачів, які шукають найкращі результати і готові працювати з більш складними налаштуваннями. Однак, він є закритим і доступний лише в продуктах **SuperMemo** []. **SM2**: Це простий і надійний алгоритм, який використовується в **Anki**. Він менш адаптивний і має обмежену гнучкість налаштувань, але є відкритим і добре підтримується спільнотою. **FSRS**: Це сучасний алгоритм, який поєднує високу адаптивність і ефективність запам'ятовування з відкритістю коду. Він є хорошим вибором для користувачів, які хочуть мати можливість налаштовувати алгоритм під свої потреби і отримувати регулярні оновлення. Вибір алгоритму залежить від потреб і пріоритетів. Якщо потрібна максимальна ефективність але більш складне налаштування, **SM-18** може бути найкращим вибором. Якщо потрібен простий і надійний алгоритм, **SM2** буде хорошим варіантом. **FSRS** пропонує баланс між адаптивністю, ефективністю і відкритістю.

1.3 Функціональні вимоги до застосунку вивчення англійських слів Для побудови концептуальної моделі проєкту проаналізуємо основні поняття теми кваліфікаційної роботи. **Windows Forms** Визначення: технологія створення графічних інтерфейсів для настільних застосунків у середовищі **.NET**. Значення: основа для створення **UI** (вікон, кнопок, текстових полів) та логіки взаємодії з користувачем. Зв'язки: використовується для реалізації інтерфейсу навчальної системи та взаємодії з методикою **spaced repetition**.

2. Застосунок (**desktop application**) Визначення: автономна програма, яка виконується на персональному комп'ютері без потреби постійного підключення до Інтернету. Значення: надає користувачеві змогу вивчати лексику у зручному режимі. Зв'язки: реалізується за допомогою **Windows Forms**, містить усі навчальні та облікові функції.

3. Англійські слова (лексика) Визначення: одиниці мови (слова, фрази), які вивчаються користувачем. Значення: основний навчальний матеріал у програмі. Зв'язки: піддаються повторенню згідно з алгоритмами **spaced repetition**; відображаються в **UI**.

4. **Spaced repetition** (розподілене повторення) Визначення: методика повторення інформації з адаптивним інтервалом, що збільшується після кожного успішного запам'ятовування. Значення: основний навчальний механізм, що забезпечує ефективне довготривале запам'ятовування. Зв'язки: реалізується через алгоритм (наприклад, **SM-2**), впливає на логіку роботи застосунку та структуру бази даних.

5. Крива забування (**Ebbinghaus forgetting curve**) Визначення: модель, що описує, як швидко людина забуває інформацію без повторення. Значення: теоретичне обґрунтування для **spaced repetition**. Зв'язки: пояснює необхідність повторень у визначені моменти часу.

6. Словник користувача Визначення: набір слів, які користувач додав до системи для вивчення. Значення: персоналізований навчальний контент. Зв'язки: зберігається в базі даних; кожен запис пов'язаний із датами повторень та історією навчання.

7. Алгоритм **SuperMemo** / **Anki** Визначення: формалізовані моделі визначення оптимальних інтервалів повторення. Значення: визначають внутрішню логіку навчального процесу. Зв'язки: реалізуються у програмному коді для планування наступних сесій.

8. **SQLite** Визначення: легка локальна СУБД, яка не потребує серверного середовища. Значення: використовується для збереження даних про слова, повторення,

успішність користувача. Зв'язки: інтегрується з [Windows Forms](#) і забезпечує збереження інформації. Ґрунтуючись на результатах аналізу термінів побудуємо спрощену семантичну мережу для предметної області предмету дослідження (рис. 1.7). Рис. 1.7. Спрощена семантична мережа предмету дослідження На основі отриманої семантичної мережі сформулюємо загальні вимоги до функціональної поведінки системи. Система повинна забезпечувати користувача можливістю ефективного вивчення англійських слів, враховуючи інтервальну методику повторень ([spaced repetition](#)). Загальні вимоги охоплюють функціональні блоки, інтерфейсну поведінку та реакцію на дії користувача. Основні функціональні вимоги: 1.Реєстрація/ідентифікація користувача (не обов'язково) - локальне збереження прогресу користувача. - або режим без авторизації (у версії для однієї особи). 2.Керування словником: -Додавання нового слова (англійське слово, переклад, приклад, теги). -Редагування та видалення слів. -Імпорт/експорт словника ([JSON/CSV](#)). -Пошук і фільтрація за тегами або статусом. 3.Повторення слів: -Щоденне формування списку слів на повторення. -Візуалізація одного слова з варіантами відповіді або вільним введенням. -Фіксація результатів: правильно/неправильно. 4.Алгоритм [spaced repetition](#): -Використання модифікованого алгоритму [SM-2](#). -Розрахунок наступної дати повторення на основі відповіді. -Збереження дати останнього повторення, кількості правильних відповідей, рівня складності. 5.Статистика прогресу: -Кількість вивчених слів. -Графік повторень. -Відсоток правильних відповідей за останні дні. -Інтерфейс користувача: -Проста і зрозуміла навігація. -Українськомовний інтерфейс. -Підтримка масштабування шрифтів і теми оформлення. 7.Збереження даних: -Вся інформація має зберігатися локально ([SQLite](#)). -Автоматичне збереження прогресу. На рис. 1.8 зображено спрощену функціональну модель застосунку для вивчення англійських слів. Актор: користувач — взаємодіє з усіма функціями застосунку. Прецеденти: додавання, редагування, перегляд словника; проведення сесії повторення за [spaced repetition](#); збереження / завантаження словника; налаштування зовнішнього вигляду (мови, шрифтів тощо). Рис. 1.8. Спрощена діаграма прецедентів [UML](#) застосунку Додамо внутрішні прецеденти, функціональні зв'язки між діями користувача та підпроцесами системи (рис. 1.9). Блок

Цитування: 0,04%

id: 12

"Керування словником":

Прецеденти для додавання, редагування, видалення та перегляду слів; Відповідає формам і діям у головному вікні застосунку. Блок

Цитування: 0,04%

id: 13

"Повторення слів":

Сценарій запуску навчальної сесії; Внутрішні етапи: показ слова, реакція користувача, оцінка, оновлення інтервалу повторення. Блок

Цитування: 0,03%

id: 14

"Статистика":

Огляд загального прогресу та перегляд списку вже вивчених слів. Блок

Цитування: 0,04%

id: 15

"Налаштування інтерфейсу":

Можна доповнити деталізацією, якщо є функції зміни теми, мови тощо. Рис. 1.9. Діаграма прецедентів [UML](#) застосунку На основі розроблених діаграм сформулюємо основні можливості (що отримає користувач): персональний словник з підтримкою української мови. щоденні сесії повторення відповідно до індивідуального темпу. інтервальний алгоритм запам'ятовування. зручна навігація та перегляд прогресу. безпечне локальне збереження без підключення до Інтернету. Узгоджені технічні характеристики представлено в табл. 1.2 Таблиця 1.2 Узгоджені технічні характеристики: Компонент Вимоги Платформа [Windows](#) 10/11 Мова програмування [C#](#) ([.NET 6](#) або [.NET Framework ≥ 4.7.2](#)) Інтерфейс [Windows Forms](#) Зберігання даних [SQLite](#) (вбудована БД, без встановлення) Резервне копіювання [JSON](#)-експорт словника Останній крок на цьому етапі проектування – визначення етапів проектування та перелік документації для замовника. Етапи реалізації: Узгодження структури інтерфейсу та даних. Прототипування основних форм. Реалізація логіки [spaced repetition](#). Візуальне тестування. Налаштування і передача користувачеві з документацією. Додатки до документації (надаються замовнику): Презентація з інтерфейсами ([mockups](#) або [screenshot](#)). Інструкція користувача ([PDF](#)). Файл структури БД

([ERD](#) або [SQL](#)). Інструкція з резервного копіювання та відновлення словника. РОЗДІЛ 2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ 2.1 Архітектурна модель застосунку Опис архітектури [Windows Forms](#)-застосунку для вивчення англійських слів за методикою [spaced repetition](#) представимо у вигляді багаторівневої структури, яка забезпечує масштабованість, підтримку та зручність розробки. Система є клієнтським десктопним застосунком, реалізованим з використанням [Windows Forms](#) у середовищі [.NET](#). Вся логіка роботи та дані зберігаються локально, без залежності від зовнішніх [API](#) чи хмарних сервісів. Обираємо архітектурний шаблон Модель-Представлення-Презентер ([MVP](#)) — якій найкраще підходить для [WinForms](#). Загальну архітектуру опишемо за допомогою ключових компонентів системи, які розділені на логічні шари: [Data Layer](#) (Рівень даних) [SQLite](#) — локальна база даних для збереження слів, прогресу, статистики. [ORM](#): використання [Entity Framework Core](#) ([SQLite provider](#)) або ручна реалізація через [System.Data.SQLite](#). Таблиці: [Users](#) (опціонально); [Words](#) (англійське слово, переклад, приклад, теги); [Repetitions](#) (дата, результат, рівень складності, інтервал); [Settings](#) (тема, масштаб шрифтів тощо). [Business Logic Layer](#) (Логіка застосунку) [Spaced Repetition Engine](#): Реалізація модифікованого алгоритму [SM-2](#). Обчислення наступної дати повторення. Словник: Додавання, редагування, видалення, імпорт/експорт. Фільтрація та пошук: За тегами, статусом (нові, на повторення, вивчені). Статистика: Побудова графіків, підрахунок правильних відповідей. [Presentation Layer](#) (Інтерфейс користувача) [WinForms UI](#): Головне меню: Словник | Повторення | Статистика | Налаштування Форми: Додавання/редагування слова Список слів з фільтрами Сесія повторення (тестування) Статистика (графіки) Налаштування (тема, масштаб) Мова інтерфейсу: українська Підтримка тем: світла/темна Масштабування шрифтів: через [Font](#) у налаштуваннях 3. Збереження та синхронізація Автоматичне збереження після кожної дії. Локальне збереження у [SQLite](#). Імпорт/експорт словника у форматах [JSON/CSV](#). 4. Статистика Побудова графіків (наприклад, через бібліотеку [LiveCharts](#) або [OxyPlot](#)). Відображення: Кількість вивчених слів Відсоток правильних відповідей Динаміка повторень 5. Авторизація (опціонально) Простий режим: один користувач, без логіну. Розширений режим: локальна авторизація (логін/пароль), збереження прогресу окремо. На рис. 2.1 представлено узагальнену архітектуру проєкту у вигляді графу Рис. 2.1. Узагальнена архітектура проєкту Як бачим з рис. 2.1 умовні шари архітектури пов'язані між собою. [FormReview](#) пз шару інтерфейсу має зв'язок з шаром бізнес-логіки проєкту, використовує основний алгоритм [SM-2](#). Також шар [UI](#) безпосередньо пов'язано з шаром зберігання даних з базою даних англійських слів [Words](#). [Words](#) використовується в бізнес логіці для роботи зі словником, фільтрацією, пошуком та оброблення статистики. На рис. 2.2-2.4 деталізовано шари архітектури. Рис. 2.2. Шар [Data Layer](#) Рис. 2.3. Шар [Business Logic Layer](#) Рис. 2.4. Шар [UI](#) Як бачимо з рис. 2.2-2.4 архітектура містить: [Data Layer](#): [SQLite](#), [ORM](#), таблиці ([Users](#), [Words](#), [Repetitions](#), [Settings](#)); [Business Logic Layer](#): [Spaced Repetition Engine](#), Словник, Фільтрація, Статистика; [Presentation Layer](#): [WinForms UI](#), форми, підтримка тем, мови, масштабування. Спрогнозуємо архітектуру інтерфейсу додатку (рис. 2.5) Рис. 2.5. Клієнтський застосунок [Windows Forms](#) На основі діаграми класів [UML](#) опишемо ключовий клас системи [WordItem](#) ([Words](#)) (рис. 2.6). Рис. 2.6. Клас [WordItem](#) На рис. 2.6 поля описують таке: [Id](#)Унікальний ідентифікатор слова у базі даних. - [EnglishWord](#)англійське слово, яке користувач вивчає. - [Translation](#)переклад слова українською мовою. - [Example](#)приклад використання слова в реченні. - [LastReviewed](#)дата, коли слово востаннє повторювалося. - [NextReview](#)дата наступного запланованого повторення. - [RepetitionCount](#)кількість успішних повторень цього слова. Зростає при кожному правильному повторенні. Методи класу [WordItem](#) (рис. 2.6) здійснюють такі дії: [UpdateReview](#)([bool isCorrect](#))оновлює дату повторення залежно від того, чи правильно згадано слово. Якщо відповідь правильна — інтервал збільшується, якщо ні — скидається. [GetNextInterval](#)()обчислює кількість днів до наступного повторення залежно від кількості правильних відповідей ([RepetitionCount](#)). [ToString](#)()повертає короткий текстовий опис слова (наприклад, для відображення в списку: англійське слово + переклад). Архітектурну модель проєкту у вигляді діаграми класів представлено на рис. 2.7 Рис. 2.7 Діаграма класів моделі проєкту Структуру проєкту як візуалізацію бачення результату представлено на скані рис. 2.8. Бачимо три папки, які відповідають шарам архітектури: - [Business](#); - [UI](#); - [Utils](#) / [EnglishTrainer](#) | [Data](#) | [AppDbContext.cs](#) | [Models](#) ([Word.cs](#), [Repetition.cs](#), [User.cs](#)) | [Business](#) | [RepetitionScheduler.cs](#) | [DictionaryManager.cs](#) | [StatisticsService.cs](#) | [UI](#) | [MainForm.cs](#) | [WordEditorForm.cs](#) | [RepetitionForm.cs](#) | [StatisticsForm.cs](#) | [Utils](#) | [ImportExportService.cs](#) | [ThemeManager.cs](#) | [Program.cs](#) Рис. 2.8. Запланована структура проєкту Таким чином

архітектуру проєкту розроблено, представлено та описано. 2.2 Вибір інструментів для розробки, реалізації та зберігання даних На основі результатів аналізу, отриманих у першому розділі, було визначено, що ефективна реалізація застосунку для вивчення англійських слів за методикою [spaced repetition](#) потребує інструментів, які забезпечують простоту розробки графічного інтерфейсу користувача, підтримку локального зберігання даних та гнучкість реалізації алгоритму розподілених повторень. Тому, враховуючи вимоги автономності та зручності застосунку, було обрано технології [Visual Studio](#) з [Windows Forms](#) та [SQLite](#) з використанням [Entity Framework Core](#). [Visual Studio](#) – сучасне інтегроване середовище розробки (IDE) від компанії [Microsoft](#), яке дозволяє створювати різноманітні типи програмних продуктів: десктопні, веб-, мобільні та серверні застосунки. Середовище надає розвинуті засоби налагодження, тестування, роботи з кодом, а також розширену підтримку мов програмування, включаючи [C#](#). Важливу роль у [Visual Studio](#) відіграє технологія [Windows Forms](#), яка є одним із ключових інструментів для створення інтерфейсів настільних застосунків. [Windows Forms](#) – технологія для створення графічних користувацьких інтерфейсів у межах платформи [.NET](#). Вона базується на об'єктно-орієнтованому підході та забезпечує швидке та інтуїтивно зрозуміле створення форм, діалогових вікон, кнопок, списків та інших елементів управління. [Windows Forms](#) залишається актуальним вибором для простих десктопних застосунків завдяки простоті, швидкості розробки та сумісності з [.NET](#). Його переваги включають [WYSIWYG](#)-дизайнер, нативні елементи управління та низьке споживання ресурсів. У порівнянні з [WPF](#) та [UWP](#), [WinForms](#) краще підходить для швидкої розробки простих інтерфейсів, особливо коли не потрібна складна візуалізація або підтримка новітніх [API](#) [19]. Порівнюючи [Windows Forms](#) з альтернативними технологіями, такими як [WPF](#) ([Windows Presentation Foundation](#)) або [UWP](#) ([Universal Windows Platform](#)), можна виділити наступні переваги [Windows Forms](#): простота освоєння, широка підтримка спільноти розробників, а також стабільна продуктивність навіть на комп'ютерах із невеликою потужністю. Саме ці фактори роблять [Windows Forms](#) оптимальним рішенням для реалізації десктопного навчального застосунку, що відповідає основним вимогам до простоти, зручності та автономності використання. Рис. 2.9

Порівняння [WinForms](#), [WPF](#), [UPF](#). Цей стовпчиковий графік (рис. 2.9) демонструє умовні показники: [Startup Time](#) (час запуску) [Memory Usage](#) (споживання пам'яті) [Rendering Speed](#) (швидкість рендерингу) [WinForms](#) показує найкращі результати в усіх категоріях, що підтверджує його доцільність для простих застосунків. Для організації зберігання даних у розроблюваному застосунку було вибрано базу даних [SQLite](#) – вбудовану СУБД, яка не потребує додаткових налаштувань або встановлення серверного середовища. [SQLite](#) характеризується мінімальними ресурсними вимогами, високою швидкістю роботи, надійністю та простотою використання. Така СУБД дозволяє забезпечити локальне зберігання слів, результатів навчання та конфігурації програми, що повністю відповідає поставленим вимогам автономності [22]. Для інтеграції [SQLite](#) із застосунком на основі [Windows Forms](#) було вирішено використати [ORM](#) (об'єктно-реляційний мапер) – [Entity Framework Core](#) [20]. Цей фреймворк забезпечує зручну взаємодію між об'єктами у коді та таблицями бази даних, дозволяючи уникнути складного ручного написання [SQL](#)-запитів. [Entity Framework Core](#) автоматично здійснює перетворення моделей [C#](#) (таких як [WordItem](#)) на таблиці [SQLite](#) і навпаки, що значно спрощує розробку та підтримку проєкту. Використання [Entity Framework Core](#) у поєднанні з [Windows Forms](#) дозволяє знизити ймовірність помилок, скоротити час розробки та спростити подальший розвиток програмного забезпечення [21]. Рис. 2.10. Вплив [ORM](#) на ефективність розробки Ця кругова діаграма (рис. 2.10) показує, як [ORM](#) (зокрема [Entity Framework Core](#)) впливає на ефективність розробки: 30% — зниження кількості помилок; 25% — прискорення розробки; 25% — зменшення обсягу [SQL](#)-коду; 20% — покращення підтримки коду. Отже, вибір інструментів [Visual Studio](#) з [Windows Forms](#) та [SQLite](#) у поєднанні з [Entity Framework Core](#) є повністю обґрунтованим для реалізації автономного навчального застосунку, який відповідає ключовим критеріям простоти, ефективності, стабільності роботи та зручності підтримки у майбутньому. 2.3 Опис модифікованого алгоритму [SuperMemo](#) для обчислення інтервалів повторення Для реалізації ефективної методики [spaced repetition](#) було обрано модифікований варіант алгоритму [SuperMemo 2](#) ([SM-2](#)), який забезпечує адаптивне визначення оптимальних інтервалів для повторення інформації, враховуючи результати відповідей користувача. Оригінальний алгоритм [SuperMemo](#) запропонував польський науковець П'єotr Возняк, але для потреб розробленого застосунку його було дещо спрощено й адаптовано, щоб полегшити інтеграцію в інтерфейс [Windows Forms](#). Основна ідея алгоритму полягає в поступовому збільшенні інтервалів повторення у разі правильної

відповіді, а також скиданні інтервалу до мінімального у випадку неправильної відповіді. У модифікованій версії алгоритму кожне слово має такі параметри: [RepetitionCount](#) — кількість послідовних правильних відповідей; [LastReviewed](#) — дата останнього повторення; [NextReview](#) — дата наступного повторення. Алгоритм працює наступним чином: При першому повторенні слова інтервал встановлюється 1 день. Після кожної наступної правильної відповіді інтервал подвоюється (наприклад: 1, 2, 4, 8 днів і т.д.). У разі неправильної відповіді кількість повторень обнуляється, і слово повторюється на наступний день. Таким чином, цей алгоритм адаптивно реагує на індивідуальні особливості запам'ятовування конкретного користувача, автоматично підлаштовуючи графік повторень під його можливості. Діаграму діяльності для модифікованого алгоритму представлено на рис. 2.11. Рис. 2.11 Алгоритм [spaced repetition](#) (модифікований [SM-2](#))

Отже алгоритм для реалізації методу [spaced repetition](#) представлено. РОЗДІЛ 3. РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ТА ЙОГО АПРОБАЦІЯ 3.1 Процес реалізації [Windows-Forms](#) застосунку

Процес реалізації навчального застосунку з використанням технології [Windows Forms](#) складався з декількох основних етапів: створення інтерфейсу користувача, реалізація базових класів та алгоритмів, інтеграція з базою даних та підключення логіки [spaced repetition](#). Перший етап: проєктування інтерфейсу користувача. На початку реалізації було створено інтерфейс головної форми ([MainForm.cs](#)). Інтерфейс організований таким чином, що користувач отримує швидкий доступ до основних функцій: перегляду статистики, запуску повторення слів та редагування словника. Основними компонентами головної форми стали вкладки сторінок, які містять кнопки переходу до інших форм, до статистики ([StatisticForm.cs](#)) та сторінку з налаштуваннями. Після цього було реалізовано форми для управління словником ([Dict.cs](#) та [WordEditorForm.cs](#)) та для повторення слів ([RepetitionForm.cs](#)). Під час розробки інтерфейсу використовувались стандартні компоненти [Windows Forms](#): [Button](#), [Label](#), [TextBox](#), [DataGridView](#), [ComboBox](#) та [ProgressBar](#). Другий етап: розробка класів бізнес-логіки. На цьому етапі створено бізнес правила роботи зі словником [DictionaryManager](#). Також реалізовано клас для роботи алгоритму [spaced repetition](#), який забезпечує адаптивне визначення інтервалів повторення. Для цього було створено методи обчислення наступної дати повторення та обробки результатів відповідей користувача. Третій етап: інтеграція бази даних [SQLite](#) з [Entity Framework Core](#). Для локального зберігання даних застосунку було використано базу даних [SQLite](#). Для зручної інтеграції цієї бази даних у [Windows Forms](#)-застосунок було використано [Entity Framework Core](#). Було створено контекст даних ([ApplicationDbContext](#)), у якому визначено набір слів як [DbSet Word](#). Модель містить ключові властивості для збереження англійського слова, його перекладу, прикладу використання, дати останнього повторення, наступного повторення та лічильника повторень. Це дозволило виконувати [CRUD](#)-операції (додавання, редагування, видалення) без необхідності ручного написання [SQL](#)-запитів. Четвертий етап: інтеграція бізнес-логіки з інтерфейсом. На заключному етапі бізнес-логіка застосунку була інтегрована з графічним інтерфейсом. Для цього було реалізовано обробники подій ([event handlers](#)) кнопок та інших елементів інтерфейсу. Наприклад, при натисканні кнопки

Цитування: 0,03%

id: 16

"Почати повторення"

викликається метод, який формує список слів на повторення, після чого відкривається форма повторення, яка в циклі відображає слова для користувача та обробляє відповіді. Було також реалізовано можливість редагування словника у відповідній формі [WordEditorForm.cs](#), із забезпеченням збереження змін у базі даних. Вся інформація, яку вводить або змінює користувач, зберігається локально. Повністю код приведено у додатках (див Додатки А- Ж). Опишемо основні коди. На першому етапі створюємо модель нашої бази даних за допомогою [Entity Framework Core](#). Листинг 3.1 описує клас [AppDbContext](#), що успадковує [DbContext](#) і використовується для взаємодії з базою даних. Він містить три [DbSet](#) властивості: - [Words](#) – колекція слів ([Word](#)). - [Repetitions](#) – колекція повторень ([Repetition](#)), що зберігає дані про тренування слів. - [Users](#) – колекція користувачів ([User](#)). - Метод [OnConfiguring](#) встановлює підключення до бази даних [SQLite](#), зберігаючи її у файлі [english\\_trainer.db](#). Листинг 3.1 `public class AppDbContext : DbContext { public DbSet Word Words { get; set; } public DbSet Repetition Repetitions { get; set; } public DbSet User Users { get; set; } protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) { optionsBuilder.UseSqlite(`

Цитування: 0,04%

id: 17

"Data Source=english\_trainer.db"

); } } Далі описуємо таблиці бази [Word](#), [Repetition](#), [User](#), які представлені у лістингах 3.2-3.4. Лістинг 3.2 [public class Word](#) { [public int Id](#) { [get](#); [set](#); } [public string EnglishWord](#) { [get](#); [set](#); } = [string.Empty](#); [public string Translation](#) { [get](#); [set](#); } = [string.Empty](#); [public string Example](#) { [get](#); [set](#); } = [string.Empty](#); [public string Tags](#) { [get](#); [set](#); } = [string.Empty](#); // інші поля [public DateTime LastReviewed](#) { [get](#); [set](#); } [public DateTime NextReviewDate](#) { [get](#); [set](#); } [public int Interval](#) { [get](#); [set](#); } // інтервал у днях } [Word](#) – це модель, що представляє слово у базі даних: [Id](#) – унікальний ідентифікатор. [EnglishWord](#) – англійське слово. [Translation](#) – переклад. [Example](#) – приклад використання слова. [Tags](#) – список тегів для категоризації. [LastReviewed](#) – дата останнього повторення. [NextReviewDate](#) – дата наступного повторення. [Interval](#) – інтервал у днях між повтореннями. Лістинг 3.3 [public class Repetition](#) { [public int Id](#) { [get](#); [set](#); } [public int WordId](#) { [get](#); [set](#); } [public DateTime Date](#) { [get](#); [set](#); } [public bool Result](#) { [get](#); [set](#); } [public int DifficultyLevel](#) { [get](#); [set](#); } [public int Interval](#) { [get](#); [set](#); } } [Repetition](#) – ця модель містить дані про повторення слова: [Id](#) – унікальний ідентифікатор запису. [WordId](#) – ідентифікатор слова, яке повторюється. [Date](#) – дата повторення. [Result](#) – результат тренування (успішно чи ні). [DifficultyLevel](#) – рівень складності. [Interval](#) – інтервал до наступного повторення. Лістинг 3.4 [public class User](#) { [public int Id](#) { [get](#); [set](#); } [public string Username](#) { [get](#); [set](#); } = [string.Empty](#); [public string Password](#) { [get](#); [set](#); } = [string.Empty](#); } [User](#) – представляє дані користувача: [Id](#) – унікальний ідентифікатор. [Username](#) – ім'я користувача. [Password](#) – закований пароль. Для опису бізнес правил роботи із словником створено декілька методів класу [DictionaryManager](#): [AddWord\(Word word\)](#) – додає нове слово до бази даних. Використовується контекст [AppDbContext](#), щоб зберегти передане слово у таблиці [Words](#). Однак метод не перевіряє, чи слово вже існує, що може спричинити дублювання записів. [EditWord\(Word word\)](#) – оновлює інформацію про слово в базі даних. Спочатку шукає запис за [Id](#). Якщо слово знайдене, оновлює його англійське значення ([EnglishWord](#)), переклад ([Translation](#)), приклад використання ([Example](#)) та теги ([Tags](#)). Зміни зберігаються після виклику [SaveChanges\(\)](#). [DeleteWord\(int wordId\)](#) – видаляє слово з бази даних. Метод шукає слово за ідентифікатором ([wordId](#)). Якщо воно існує, то воно видаляється з таблиці [Words](#), а зміни зберігаються. [SearchWords\(string tag\)](#) – шукає слова за заданим тегом. Використовуючи [LINQ](#), метод фільтрує слова, які містять переданий тег у своєму списку тегів ([Tags](#)). Отримані результати повертаються у вигляді списку ([List Word](#)). Для прикладу приведемо код [EditWord](#), який приведено у лістингу 3.5. Лістинг 3.5 [public void EditWord\(Word word\)](#) { [using \(var db = new AppDbContext\(\)\)](#) { [var existingWord](#) = [db.Words.Find\(word.Id\)](#); [if \(existingWord != null\)](#) { [existingWord.EnglishWord](#) = [word.EnglishWord](#); [existingWord.Translation](#) = [word.Translation](#); [existingWord.Example](#) = [word.Example](#); [existingWord.Tags](#) = [word.Tags](#); [db.SaveChanges\(\)](#); } } } Цей код здійснює оновлення існуючого слова у базі даних за його ідентифікатором ([Id](#)). Основні етапи: 1. Створення контексту бази даних – використовується [using \(var db = new AppDbContext\(\)\)](#), що забезпечує правильне відкриття та закриття підключення до бази даних. 2. Пошук слова в базі – метод [db.Words.Find\(word.Id\)](#) шукає слово за переданим [Id](#). Якщо слово існує, його дані оновлюються. 3. Оновлення даних – якщо [existingWord != null](#) (тобто слово знайдене), оновлюються такі поля: - [EnglishWord](#) – оновлюється англійське слово. - [Translation](#) – оновлюється переклад. - [Example](#) – оновлюється приклад використання. - [Tags](#) – оновлюються теги. 4. Збереження змін – Викликається [db.SaveChanges\(\)](#), що зберігає оновлені дані в базі. Інші коди для [DictionaryManager](#) можна знайти в додатках. Основний метод повторення реалізується в класі [RepetitionScheduler](#) з такими методами: 1. [CalculateNextRepetitionDate\(Repetition repetition, bool isCorrect\)](#) – використовує просту реалізацію алгоритму [SM-2](#) для планування повторень. - якщо відповідь правильна, інтервал подвоюється. Якщо неправильна, встановлюється мінімальний (1 день). - оновлює [Interval](#) та обчислює нову дату повторення ([DateTime.Now.AddDays\(interval\)](#)). 2. [UpdateRepetition\(Word word, bool isCorrect\)](#) – оновлює інтервал повторення слова. - якщо користувач відповів правильно, інтервал подвоюється (або встановлюється 1, якщо він був 0). - якщо відповідь неправильна, інтервал скидається до 1 дня. - оновлюється дата наступного повторення ([NextReviewDate](#)) та дата останнього перегляду ([LastReviewed](#)). 3. [IsDueForReview\(Word word\)](#) – перевіряє, чи потрібно повторити слово. - якщо дата наступного повторення [NextReviewDate](#) дорівнює або менша за сьогоднішню ([DateTime.Today](#)), слово потрібно повторити. У якості прикладу реалізації приведемо код методу [UpdateRepetition](#) (лістинг 3.6). Лістинг 3.6 [public void UpdateRepetition\(Word word, bool isCorrect\)](#) { [if \(isCorrect\)](#) { // Якщо правильна відповідь — збільшуємо інтервал

```
word.Interval = word.Interval == 0 ? 1 : word.Interval * 2; } else { // Якщо помилка — скидаємо
інтервал word.Interval = 1; } // Оновлюємо дату наступного повторення
word.NextReviewDate = DateTime.Today.AddDays(word.Interval); word.LastReviewed =
DateTime.Today; } // Метод для перевірки, чи потрібно повторити слово public bool
IsDueForReview(Word word) { return word.NextReviewDate = DateTime.Today; } Метод
UpdateRepetition оновлює інтервал повторення слова залежно від правильності відповіді
користувача: 1. Перевірка правильності відповіді: якщо відповідь правильна (isCorrect ==
true), інтервал повторення подвоюється (word.Interval * 2). Якщо інтервал був 0, він
встановлюється як 1. якщо відповідь неправильна (isCorrect == false), інтервал скидається
до 1 дня. 2. Оновлення дати повторення: поле NextReviewDate встановлюється відповідно
до нового інтервалу (DateTime.Today.AddDays(word.Interval)), що визначає дату наступного
повторення слова. поле LastReviewed записує поточну дату (DateTime.Today), що означає,
що слово було повторене. 3.2 Опис інтерфейсу застосунку Інтерфейс розробленого
Windows Forms-застосунку виконаний відповідно до принципів простоти, доступності та
зручності користування. Він складається з трьох основних форм: головна форма
('MainForm'), форма редактора словника ('Dict'), редагування слова ('WordEditorForm') та
форма для повторення слів ('RepetitionForm'). Нижче наведено опис кожної з форм з
урахуванням основних елементів інтерфейсу. Головна форма застосунку (MainForm).
Головна форма є стартовим екраном програми, який відкривається після запуску
застосунку. Вона має мінімалістичний дизайн і містить такі основні елементи:
```

” Цитування: 0,04% id: 18

«Кнопка Розпочати тренування»

дозволяє користувачу розпочати навчальну сесію повторення слів. Кнопка

” Цитування: 0,04% id: 19

«Перейти в словник»

відкриває форму управління словником, де можна додавати, редагувати або видаляти слова.

” Цитування: 0,03% id: 20

«Панель статистики»

виводить інформацію про загальну кількість слів у словнику, кількість слів на повторення сьогодні, а також загальний прогрес навчання. Вкладка

” Цитування: 0,01% id: 21

«Налаштування»

забезпечує додатковий доступ до налаштувань та інших функцій. Зовнішній вигляд головної форми зображено на рис.3.1-3.3. Форма редактора словника Dict (рис. 3.4). Ця форма призначена для керування набором слів, які користувач вивчає. Інтерфейс форми забезпечує зручне додавання, редагування та видалення слів. Основні компоненти форми: Таблиця (DataGridView) відображає список усіх слів, які є у словнику, включаючи англійське слово, переклад та дату наступного повторення. Користувач може вибрати слово для редагування, просто клацнувши по ньому Рис. 3.1. Вкладка

” Цитування: 0,01% id: 22

«Словник»

головної форми Рис. 3.2. Вкладка

” Цитування: 0,33% id: 23

«Повторення (тренування) головної форми Рис. 3.3. Вкладка «Налаштування» головної форми Рис. 3.4. Форма редактора словника (Dict) Кнопки управління словником: «Додати слово»

– додає нове слово в словник.

” Цитування: 0,01% id: 24

«Редагувати»

– редагування слова.

” Цитування: 0,01% id: 25

«Видалити»

– видаляє слово зі словника.

” Цитування: 0,03% id: 26

«Експорт [JSON](#)»

– експортує словник в [JSON](#)

” Цитування: 0,03% id: 27

«Імпорт [JSON](#)»

– робить імпорт з [JSON](#)

” Цитування: 0,03% id: 28

«Поле пошуку»

забезпечує швидкий пошук слів. Форма [Dict](#) — це головне вікно для роботи зі словником англійських слів. Вона реалізована як [WinForms](#)-форма з підключенням до бази даних через [AppDbContext](#). Основна мета — перегляд, фільтрація, редагування, імпорт та експорт слів. При запуску форми викликається [LoadData\(\)](#), яка завантажує всі слова з бази. Всі дії з базою ([Add](#), [Edit](#), [Delete](#), [Filter](#), [Import](#), [Export](#)) виконуються через [AppDbContext](#). Для імпорту/експорту використовується сервіс [ImportExportService](#). Рис. 3.5. Форма [WordEditorForm](#) Форма [WordEditorForm](#) (рис. 3.5) має мінімалістичний інтерфейс для додавання або редагування запису в таблиці англійських слів. Основна форма для повторення слів [RepetitionForm](#) (рис. 3.6). Ця форма реалізує основний функціонал застосунку, пов'язаний з методикою [spaced repetition](#). Вона дозволяє користувачеві проводити навчальні сесії та оцінювати свій рівень запам'ятовування слів. Основні елементи інтерфейсу форми повторення: Поле відображення англійського слова На екрані показується англійське слово, яке користувач повинен пригадати або перекласти. Рис. 3.6. Форма [RepetitionForm](#) Поле введення відповіді Користувач може вводити переклад слова або вказати, що не пам'ятає слово. Кнопка відповіді натискається після введення відповіді. Індикатор прогресу сесії Відображає, скільки слів залишилося у поточній навчальній сесії та відсоток правильних відповідей. Область показу правильної відповіді Після відповіді користувача виводиться правильний переклад слова. Отже, інтерфейс застосунку забезпечує зрозумілу та комфортну роботу користувача із системою. Використання стандартних компонентів [Windows Forms](#) дозволило створити простий та функціональний інтерфейс, який відповідає ключовим вимогам до навчального програмного забезпечення. 3.3. Налаштування та тестування Після завершення розробки було проведено тестування застосунку в різних сценаріях: додавання нових слів, запуск повторення, відповіді на слова та редагування записів. У процесі тестування були виявлені дрібні помилки інтерфейсу та бізнес-логіки, які були оперативно усунуті. Для тестування використовувався [github](#). Вигляд репозиторію на рис. 3.7 Рис. 3.7. Репозиторій на [github](#) На рис. 3.8 Зображено послідовність комітів в процесі текстування та налагодження (в зворотному порядку). Опис комітів: 00db5c2 – [user](#) – [add .gitattributes](#) та [.gitignore](#). – 2025-02-19 початкове налаштування репозиторію. d550bcf – [user](#) – [add](#) – 2025-03-19 перше додавання основного коду. 24b531e – [Halafaiev](#) – [first bag](#) – 2025-04-21 початок роботи з функціоналом

” Цитування: 0,01% id: 29

"[bag](#)".

Рис. 3.8. Послідовність комітів при налагодженні та тестуванні 033987a – [Halafaiev](#) – [check bag with dbgrid](#) – 2025-04-21 - Перевірка

” Цитування: 0,01% id: 30

"[bag](#)"

з використанням [dbgrid](#). 32aed63 – [Halafaiev](#) – [check bag edit](#) – 2025-04-21 - Тестування редагування

” Цитування: 0,01% id: 31

"[bag](#)".

761ab5f – [Halafaiev](#) – 2 – 2025-04-26 - Можливо, тестовий або проміжний коміт. 9292189 – [Halafaiev](#) – [add filter](#) – 2025-05-01 - Додано фільтрацію. a131bef – [Halafaiev](#) – [fix Open and Save JSON](#) – 2025-05-02 - виправлення збереження/відкриття [JSON](#). 959de3c – [Halafaiev](#) – [fixed utf-8 json](#) – 2025-05-12 - виправлення кодування [JSON](#). edf5f81 – [Halafaiev](#) – [finf bag with repeat](#) – 2025-05-16 - Робота з повторюваними

”

Цитування: 0,01% id: 32

eb8a21d - Halafaiev - dont all repeat - 2025-05-18 - Обмеження повторів. 36b5311 - Halafaiev - dont all repeat2 - 2025-05-22 - Остаточне доопрацювання логіки повторів. У результаті тестування підтверджено стабільність та коректність роботи програми. Застосунок відповідає поставленим вимогам і готовий до експлуатації кінцевим користувачем. Таким чином, процес реалізації [Windows Forms](#)-застосунку був виконаний відповідно до затверджених вимог, що дозволило отримати завершений програмний продукт, готовий до практичного використання. ЗАГАЛЬНІ ВИСНОВКИ У результаті виконання бакалаврської роботи було розроблено десктопний застосунок на платформі [Windows Forms](#), призначений для ефективного вивчення англійських слів за методикою [spaced repetition](#). В ході виконання роботи було проведено аналіз існуючих методик вивчення лексики, здійснено вибір оптимальних технологій для реалізації програми, а також спроектовано та реалізовано програмний продукт. Під час аналізу сучасних методик вивчення англійської лексики було встановлено, що методика [spaced repetition](#) вирізняється високою ефективністю завдяки адаптивному підходу до процесу запам'ятовування. Вона базується на психологічних дослідженнях кривої забування Германа Еббінгауза, що дозволяє максимально ефективно використовувати можливості пам'яті людини. Для реалізації застосунку були обрані технології [Visual Studio](#) з [Windows Forms](#), що забезпечили простоту та зручність створення користувацького інтерфейсу, а також локальна база даних [SQLite](#) із застосуванням [Entity Framework Core](#), що дозволило зручно організувати зберігання та роботу з даними. Вибір цих інструментів обумовлений їхніми перевагами у створенні автономних, легких для підтримки та розширення програмних продуктів. В роботі також було розроблено та реалізовано модифікований алгоритм [SuperMemo](#), який адаптивно визначає інтервали повторень, враховуючи правильність відповідей користувача. Це дозволяє оптимізувати навчальний процес і забезпечити ефективне довготривале запам'ятовування лексики. Розроблений програмний продукт відповідає усім визначеним вимогам та успішно пройшов етап тестування. Користувач отримує зручний інструмент для систематичного та ефективного вивчення англійських слів у комфортному режимі, що сприяє кращому запам'ятовуванню та засвоєнню інформації. Подальшими напрямками розвитку застосунку можуть бути додавання функцій синхронізації словника з хмарними сервісами, реалізація мультимедійних можливостей, таких як аудіосупровід, та створення мобільної версії застосунку, що дозволить значно розширити коло потенційних користувачів.

Заявление об ограничении ответственности:

Этот отчет должен быть правильно истолкован и проанализирован квалифицированным специалистом, который несет ответственность за оценку!

Любая информация, представленная в этом отчете, не является окончательной и подлежит ручному просмотру и анализу. Пожалуйста, следуйте инструкциям: [Рекомендации по оценке](#)

Детектор Плагиата - Ваше право на оригинальность! ☐ SkyLine LLC