МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма: «Розробка програмного забезпечення»

Група: 4РП-08

Дипломний проєкт

здобувача освіти денної форми навчання
РП.08.18.000.ДП

СІЧКАРЯ
ОЛЕКСАНДРА ОЛЕГОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**

Освітньо-професійна програма: «**Розробка програмного забезпечення»**

Група: **4РП-08**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка програмного застосунку-помічника для вивчення іноземних мов

Проектний матеріал складається з пояснювальної записки на 71 сторінках та графічного (презентаційного) матеріалу на 12 аркушах (слайдах)

Дипломник _____ (Січкар О.О.)

Керівник _____ (Шувалова І.О.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділенням _____ (Краснокутська К.Г.)

Захист «26» 06 2025 р. Протокол ЕК № 9

Оцінка ЕК 3/70

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 Інженерія програмного забезпечення*

Освітньо-професійна програма *Розробка програмного забезпечення*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"12" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Січкарю Олександру Олеговичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка програмного застосунку-помічника для вивчення іноземних мов*

затверджена наказом по коледжу від "14" листопада 2024 р. № 246

2. Термін здачі закінченого проекту

3. Вихідні данні до проекту *1.Реалізувати роботу застосунку для вивчення іноземних. 2.Реалізувати можливість додавання, редагування, пошуку та видалення записів у словнику. 3.Реалізувати функціонал тренувань за картками 4.Реалізувати модуль аудіювання. 5.Передбачити запис статистики тренувань. 6.Передбачити можливість локального зберігання аудіофайлів. 7.Передбачити можливість повторного прослуховування та перевірки відповідей без переходів між сторінками..*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз та класифікація мовних навчальних застосунків; Аналітичний огляд інтерфейсних та функціональних рішень аналогів; Психологічне обґрунтування методів навчання; Огляд середовищ розробки для створення мобільних застосунків; Огляд архітектурних шаблонів для мобільних застосунків; Проєктування структури даних для словника та аудіювання; Розробка функціональних модулів; Реалізація локального сховища даних та механізмів обробки

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

Діаграма взаємодії для віртуального асистента; Діаграма процесу формування документів; Модель захисту віртуального асистента; Інтегрована архітектура віртуального асистента; Гексагональна архітектура віртуального асистента; ER-діаграма сутностей і зв'язків у БД віртуального асистента; Діаграма станів віртуального асистента; Блок-схема алгоритму забезпечення безпеки при формуванні текстових документів; Приклади роботи розробленого віртуального асистента для безпечного формування текстових документів

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Шувалова І.О.*		
Економічний розділ	*Канський М.Ю*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник *Шувалова І.О.* _____
(підпис)

Завдання прийняв до виконання *Січкар О.О* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка задачі проєктування*	03.04.2025	Виконано
2	*Аналіз технічного завдання та пошук літератури*	10.04.2025	Виконано
3	*Аналіз психологічних принципів вивчення лексики*	14.04.2025	Виконано
4	*Аналіз психологічних принципів вивчення лексики*	21.04.2025	Виконано
5	*Вибір архітектури застосунку*	28.04.2025	Виконано
6	*Розробка структури даних*	04.05.2025	Виконано
7	*Розробка словника: інтерфейс і логіка*	07.05.2025	Виконано
8	*Розробка модуля карток з логікою повторення*	10.05.2025	Виконано
9	*Розробка модуля аудіювання та обробка відповідей*	20.05.2025	Виконано
10	*асистента для формування текстових документів*	31.05.2025	Виконано
11	*Тестування функціональності, UI і бази даних*	10.05.2025	Виконано
12	*Оптимізація, відлагодження та усунення помилок*	11.05.2025	Виконано
13	*Виконання економічних розрахунків*	13.05.2025	Виконано
14	*Розробка питань з охорони праці та техніки безпеки*	14.05.2025	Виконано
15	*Підготовка мультимедійної презентації проекту*	13.05.2025	Виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 08. 18 000. 00 ДП	Дипломний проект		
А4			РП 08. 18 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 08. 18 000. 00 ДП		
Розробив		Січкар О.О.		14.06.23	Розробка програмного застосунку-помічника для вивчення іноземних мов	Літ.	Аркуш 4 / Аркушів 71
Перевірив		Шувалова І.О.		16.06.23		Н Д П	
Н. контр.		Петрашова В.І.		16.06.23	ВСП "ОТФК ОНТУ" гр.4РП-08		
Затверд.		Кривченко Ю.В.		20.06.			

ЗМІСТ

ВСТУП

Сучасний світ потребує гнучких та ефективних інструментів для вивчення іноземних мов, особливо у форматі мобільних додатків. У зв'язку з цим зростає попит на застосунки, які можуть адаптуватися до потреб користувача, пропонуючи не лише зручний інтерфейс, а й психологічно обґрунтовані підходи до навчання. Під час вивчення мови важливу роль відіграють регулярність, контекстність та індивідуальне повторення, що й лягли в основу розробленого програмного продукту.

Проєкт LanguageHelper було створено як практичне рішення для користувачів, які прагнуть зосередитись на особистому словниковому запасі та тренувати розуміння на слух без складних процедур і зайвого функціоналу. Реалізація на базі .NET MAUI дозволила забезпечити кросплатформеність і простоту використання, зберігаючи високу продуктивність і якість взаємодії.

У процесі розробки було проведено аналіз існуючих мовних застосунків, що дозволило виокремити їхні сильні та слабкі сторони й інтегрувати найкращі практики в власний продукт. Архітектурне рішення з використанням шаблону MVVM та локальної бази даних SQLlite гарантує надійність збереження і швидкий доступ до даних. Завдяки цьому користувачі отримують адаптивну систему повторення матеріалу, яка підлаштовується під індивідуальний темп і історію помилок.

Крім того, особливу увагу було приділено юзабіліті та ергономіці інтерфейсу: мінімалістичний дизайн і логічна навігація спрощують навчальний процес, дозволяючи зосередитися виключно на матеріалі. Досвід користувача посилюється завдяки вбудованому аудіо-модулю та інтуїтивним механікам карток, що робить LanguageHelper корисним інструментом як для початківців, так і для тих, хто прагне підтримувати рівень мови на високому рівні. Усе це стало основою для розробки дипломного проєкту, в якому детально описано процес створення та впровадження даного застосунку.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз існуючих мовних застосунків

На сучасному етапі розвитку мобільних технологій існує велика кількість додатків, спрямованих на допомогу у вивченні іноземних мов. Серед найпопулярніших — Duolingo, Memrise та Anki. У процесі підготовки та реалізації мого проєкту я вивчив особливості кожного з цих застосунків, що дозволило мені краще зрозуміти потреби користувача, виявити ефективні підходи до навчання та визначити слабкі сторони, яких варто уникнути в моїй розробці.

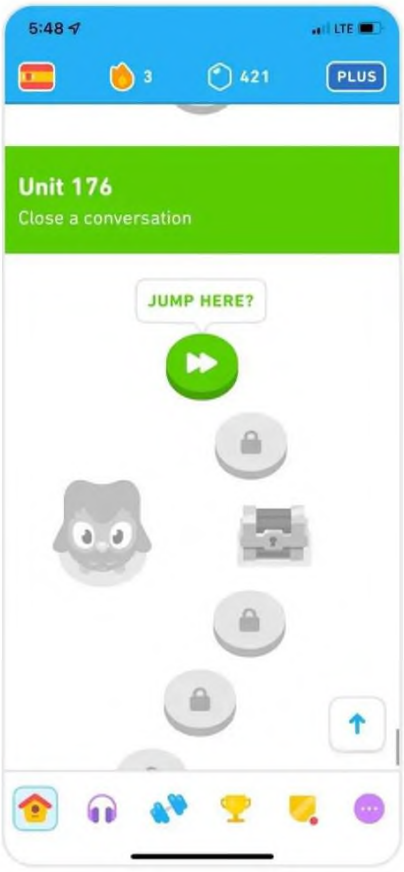


Рисунок 1.1 Головне меню Duolingo

Duolingo — це один із найвідоміших застосунків для вивчення мов, що використовує гейміфікований підхід. Його інтерфейс нагадує мобільну гру: користувачі проходять «рівні», отримують винагороди, накопичують очки. Основна форма навчання полягає у виконанні коротких вправ — переклад речень, вибір правильного варіанту, написання слів. Завдяки цьому Duolingo особливо

привабливий для початківців або тих, хто хоче проводити щодня декілька хвилин за вивченням мови. Водночас, у цьому підході відчутною є обмеженість — система не дозволяє користувачеві самому визначати темп навчання чи складати власний словник. Крім того, користувачі, які прагнуть більшої гнучкості або глибини у вивченні матеріалу, часто зіштовхуються з потребою оформлення підписки.

Однією з основних переваг застосунку є інтерфейс, виконаний у стилі гри, що значно підвищує зацікавленість користувача та сприяє регулярному навчанню. Гейміфікація дозволяє створити атмосферу змагання та досягнення, що мотивує користувача щоденно відкривати додаток і проходити нові вправи. Завдяки простому та інтуїтивно зрозумілому дизайну, платформа особливо добре підходить для початківців, які тільки починають знайомство з іноземною мовою.

Ще однією сильною стороною є підтримка великої кількості мов, що дає можливість користувачеві вивчати різні мовні групи з перекладом, аудіовимовою та тематичними вправами. Весь процес навчання супроводжується візуально привабливими елементами, звуковим супроводом і короткими уроками, що робить засвоєння матеріалу легким та ефективним.

Однак не обійшлося й без недоліків. Наприклад, користувач не має змоги створювати власні словники або змінювати вже надані теми, що обмежує індивідуалізацію навчання. Крім того, вправи з часом можуть здаватися занадто повторюваними або механічними, що знижує їхню ефективність на тривалому етапі вивчення. До того ж, значна частина функціоналу, зокрема офлайн-режим, розширена аналітика або тренування на час, доступна лише у версії Duolingo Plus, що вимагає оформлення платної підписки.

Memrise вирізняється тим, що основну увагу приділяє саме запам’ятовуванню лексики. В основі платформи лежить ідея повторення слів за допомогою мнемотехнік та відеофрагментів із носіями мови. Memrise частково дозволяє користувачам створювати власні набори слів, однак ця функція менш зручна в мобільному варіанті програми. Хоча загальна концепція дуже ефективна для засвоєння слів, проблема полягає в тому, що застосунок надмірно залежить

від готових курсів, які не завжди відповідають індивідуальним потребам користувача.

Серед основних переваг платформи слід відзначити використання відео з носіями мови, що створює ефект живого спілкування та наближає навчання до реальних ситуацій. Такий підхід сприяє кращому сприйняттю інтонацій, вимови та невербальних елементів комунікації. Навчання також базується на ефективному методі повторення з використанням мнемотехнік, що дозволяє закріпити нову лексику в довготривалій пам'яті. Додатковим плюсом є можливість частково створювати власні курси, що додає елемент персоналізації до стандартного навчального процесу.

Попри сильні сторони, система має й певні недоліки. Готові курси не завжди відповідають індивідуальним цілям і рівню підготовки користувача, що може знижувати ефективність навчання. Хоча функція створення власних наборів слів доступна, вона реалізована не надто зручно в мобільній версії, що ускладнює роботу зі смартфона. Крім того, значна частина навчального контенту, а також розширені можливості, залишаються заблокованими без оформлення платної підписки, що обмежує доступ до повного функціоналу.

Anki — це найгнучкіший, але і найменш дружній для початківців інструмент. Його головною перевагою є застосування алгоритму інтервального повторення (SRS), який забезпечує ефективне довготривале запам'ятовування. Anki дозволяє користувачам самостійно створювати картки, додавати зображення, аудіо, а також імпортувати готові колоди. Проте сам інтерфейс є застарілим, особливо у мобільній версії, а самостійне створення карток часто потребує додаткового часу та зусиль, що може відштовхнути менш досвідченого користувача

Однією з головних переваг платформи є потужна система інтервального повторення (SRS), яка базується на науково обґрунтованих принципах запам'ятовування. Цей механізм дозволяє ефективно засвоювати й утримувати великий обсяг інформації завдяки оптимально розподіленим повторенням. Особливо цінним є те, що користувач отримує повну свободу у створенні власних

карток — можна додавати зображення, аудіо, приклади вживання, теги та будь-який контент, який сприяє персоналізації процесу навчання. Окрім цього, існує велика бібліотека готових колод, які охоплюють різні теми й мови, що значно пришвидшує старт роботи.

Попри гнучкість і функціональність, інтерфейс застосунку часто виявляється занадто складним для новачків. Базова навігація та налаштування можуть здаватися незрозумілими, що ускладнює перші кроки у використанні. Мобільна версія також має обмеження: на iOS вона доступна лише за плату, а на Android хоча й безкоштовна, проте менш функціональна й часто оновлюється з запізненням. Крім того, ефективне використання платформи потребує часу для підготовки контенту, створення або адаптації колод, що не завжди зручно для користувачів, які шукають швидке та просте рішення.

Після аналізу цих трьох підходів стало зрозуміло, що кожен з них має власні переваги, однак і певні обмеження. Duolingo залучає ігровими елементами, але обмежує користувача в індивідуальному підході. Memrise пропонує автентичний контент, але залежить від заздалегідь створених курсів. Anki забезпечує гнучкість і науково обґрунтований підхід до повторення, але вимагає технічної підготовки.

У своєму застосунку я намагався об'єднати кращі риси цих сервісів, уникнувши їхніх недоліків. Основна ідея полягала у створенні простої у використанні програми, яка дозволяє швидко додавати нові слова, переглядати їх у форматі карток, тренувати запам’ятовування в інтерактивній формі, а також слухати фрази для розвитку навичок аудіювання. Усі ці функції повинні бути доступні локально, без обов’язкової реєстрації, складної навігації або платного доступу.

1.2 Психологічні аспекти вивчення слів

Успішне вивчення іноземної мови неможливе без запам’ятовування лексики. Саме тому важливо було не просто створити застосунок для відображення слів і перекладу, а реалізувати підхід, який базується на когнітивних особливостях людини — тобто враховує, як ми сприймаємо, обробляємо й запам’ятовуємо інформацію. У рамках дослідження та реалізації

мого застосунку я орієнтувався на три основні принципи: інтервальне повторення, мультимодальність сприйняття та ефект контексту.

Перший ключовий принцип — інтервальне повторення (Spaced Repetition). Згідно з дослідженнями у сфері психології пам’яті, люди краще запам’ятовують інформацію, якщо вона повторюється з поступовим збільшенням інтервалів часу між повтореннями. Іншими словами, чим довше ми пам’ятаємо слово — тим рідше потрібно його повторювати. Цей підхід лежить в основі алгоритмів таких систем, як Anki, але на практиці реалізується складно. У своєму проєкті я не створював повноцінну SRS-систему, однак структура застосунку дозволяє користувачу самостійно повертатися до слів на будь-якому етапі, не обмежуючи їх кількість або тематику. Це забезпечує гнучке індивідуальне повторення, яке може адаптуватися до потреб конкретного користувача.

Другий важливий фактор — мультимодальне сприйняття. Відомо, що люди краще запам’ятовують інформацію, яка надходить одночасно через кілька каналів: візуальний, аудіальний та моторний (наприклад, коли ми бачимо слово, чуємо його звучання та повторюємо вголос або натискаємо кнопку). Саме тому в застосунку реалізовано аудіофункціонал — можливість прослухати фразу на іноземній мові, паралельно читаючи її текстовий варіант. Це дозволяє активізувати одразу кілька типів пам’яті й покращити результат.

Окрему увагу я приділив простоті взаємодії з програмою. З точки зору когнітивного навантаження, користувач має отримувати мінімальну кількість інформації одночасно, щоб уникати перевантаження. Саме тому дизайн карток у моєму застосунку — мінімалістичний: користувач бачить лише одне слово з перекладом або аудіофразу з перекладом, а не списки чи тести. Такий формат дозволяє сфокусувати увагу лише на одному елементі, що сприяє кращому запам’ятовуванню.

Третій принцип — ефект контексту. Дослідження показують, що люди краще запам’ятовують інформацію, якщо вона пов’язана з контекстом або використовується в ситуації. Хоча у застосунку не реалізовано повноцінні діалоги чи сюжетні вставки, багато фраз, що використовуються для прослуховування, є

побутовими або життєвими — тобто тими, які реально можуть знадобитися у повсякденному спілкуванні. Це підсилює асоціативне запам’ятовування, навіть якщо користувач не вивчає граматику чи структуру речень.

Рисунок 1.2 Психологічні аспекти вивчення

Підсумовуючи, можна сказати, що під час створення мого застосунку я прагнув не просто реалізувати інтерфейс або технічну функцію, а закласти в основу науково обґрунтовані принципи психології навчання, які допомагають дійсно ефективно вивчати мову. Застосунок не замінює повноцінний курс, але є зручним інструментом-помічником для щоденного поповнення словникового запасу та тренування слуху.

1.3 Інтерфейсні рішення і функціонал аналогів застосунків-помічників

У сучасній екосистемі мобільного програмного забезпечення простота, інтуїтивність і швидкодія інтерфейсу відіграють не меншу роль, ніж функціонал. Особливо це стосується освітніх застосунків, де користувач очікує мінімального навантаження на увагу та максимально швидкого доступу до навчального матеріалу. У цьому підрозділі я розглянув типові інтерфейсні та функціональні рішення, реалізовані в популярних мовних застосунках, з метою порівняння їх із моїм застосунком LanguageHelper, а також для аналізу, що з цього досвіду можна адаптувати або, навпаки, уникнути.

Один із головних викликів у створенні мовного помічника — це баланс між кількістю функцій і простотою інтерфейсу. Наприклад, у Duolingo інтерфейс яскравий, з великою кількістю анімацій, кольорових індикаторів прогресу, повідомлень і досягнень. Це створює атмосферу гри, але іноді розсіює увагу.

Memrise, навпаки, має більш стриманий інтерфейс, із наголосом на короткі відео з носіями мови й текстові блоки. Anki майже повністю позбавлений графічного оформлення — користувач працює з простими картками, які відображаються послідовно, без зайвих візуальних елементів.

У своєму застосунку я обрав підхід, близький до Anki та Memrise: мінімалістичний інтерфейс, в якому на передньому плані не кнопки чи іконки, а навчальний матеріал. При відкритті сторінки користувача зустрічає лише те, що потрібно: кнопка для переходу до тренування, чітко структуровані блоки слів або фраз, просте меню з основними пунктами («Картки», «Аудіювання», «Словник»). Така концепція дозволяє уникнути перевантаження і сфокусувати увагу на навчанні, а не на навігації.

Ще однією важливою особливістю сучасних застосунків є адаптивність і логічна структура розділів. У багатьох продуктах інтерфейс збудовано навколо головного екрана з декількома ключовими модулями: навчання, статистика, профіль, магазин, налаштування. У LanguageHelper я реалізував лише основні функціональні розділи — ті, які безпосередньо пов’язані з навчальним процесом. Наприклад:

- Картки — дозволяють тренувати переклад слів у обох напрямках.
- Аудіювання — дає змогу слухати фрази та обирати відповідь.
- Словник — містить усі вивчені слова, з можливістю пошуку.

Також у деяких аналогічних програмах, зокрема Duolingo, інтерфейс побудований навколо так званої «навчальної траєкторії» — користувач повинен проходити теми послідовно. Я ж вирішив не обмежувати користувача черговістю — будь-який розділ мого застосунку доступний у будь-який момент. Це підходить для людей, які хочуть самостійно формувати темп навчання, самі вирішувати, що саме повторювати чи слухати.

Щодо реалізації аудіофункціоналу, то в Duolingo він здебільшого вбудований у вправи. У Memrise — виноситься окремим елементом з відео та прикладами вимови. У LanguageHelper я реалізував просту систему: кожна фраза містить кнопку прослуховування, за якою прив’язаний аудіофайл. При натисканні

аудіо відтворюється без переходу або завантаження нової сторінки, що прискорює навчання й не відволікає.

Ще один момент, який я врахував, — реакція інтерфейсу на дії користувача. У деяких застосунках, таких як Anki, немає майже ніякого зворотного зв'язку: натиснув — перейшов далі. Я ж додав візуальні підказки та анімації при переходах (наприклад, зміна картки, перемикання режимів), щоб зробити взаємодію більш приємною та зрозумілою.

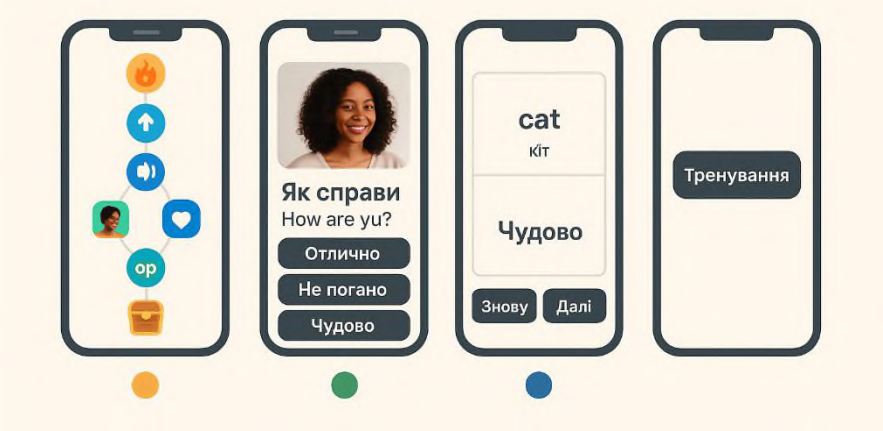


Рисунок 1.3 Інтерфейсні рішення у додатках вивчення мов

На завершення варто зазначити, що багато комерційних застосунків мають складну систему налаштувань, статистику, щоденні завдання, рівні тощо. Я навмисно виключив усі несуттєві функції, зосередившись лише на тому, що прямо пов'язане з вивченням слів і прослуховуванням. Таким чином, застосунок не перевантажений, легкий у запуску й адаптований до коротких сесій навчання — 2–3 хвилини в транспорті або між заняттями.

1.4 Огляд .NET MAUI як середовища розробки

Для реалізації програмного застосунку LanguageHelper було обрано .NET Multi-platform App UI (MAUI) — сучасну кросплатформову технологію від Microsoft, яка дозволяє створювати мобільні застосунки для Android, iOS, Windows і macOS, використовуючи одну базу коду на мові C#.

Головна перевага MAUI, яка й стала вирішальною під час вибору інструменту, — це можливість створювати нативні інтерфейси з високою продуктивністю без потреби у вивченні кількох мов програмування для різних

платформ. Тобто, завдяки MAUI, я міг реалізувати інтерфейс, логіку, базу даних та аудіофункціонал одночасно для Android і Windows, працюючи лише в межах єдиного проєкту.

Розробка застосунку здійснювалася у середовищі Visual Studio 2022, яке повністю підтримує .NET MAUI завдяки наявності офіційних шаблонів і розширень. Ця IDE надала всі необхідні інструменти для створення кросплатформеного інтерфейсу, управління ресурсами, налаштування навігації та тестування застосунку. Після створення нового проєкту на основі шаблону MAUI автоматично формується базова структура, яка відповідає архітектурному підходу MVVM і включає всі необхідні компоненти для початку роботи.

Серед ключових елементів структури варто виокремити файли App.xaml та AppShell.xaml. Перший відповідає за стилізацію застосунку, визначення ресурсів і тем оформлення, тоді як другий реалізує навігаційну оболонку, що дозволяє організувати маршрути між сторінками. Каталоги Pages та ViewModels розділяють візуальні компоненти й бізнес-логіку: кожна сторінка має відповідний ViewModel, що відповідає за взаємодію з даними та реакцію на події користувача. Нарешті, файл MauiProgram.cs виконує роль конфігураційного центру, в якому реєструються служби, моделі та ViewModel-и в DI-контейнері, забезпечуючи інверсію залежностей і масштабованість проєкту.

Оскільки MAUI підтримує XAML для опису користувацького інтерфейсу, я реалізував всі сторінки саме в цьому форматі. Це дозволило легко налаштувати зовнішній вигляд кнопок, списків, тексту, а також зв’язати їх із властивостями ViewModel через data binding. Такий підхід забезпечив чисту архітектуру, у якій UI і логіка розділені, а отже, зручні для підтримки й тестування.

Особливо зручним виявився вбудований механізм навігації через Shell — структура, яка дозволяє визначати всі переходи між сторінками в одному місці (AppShell.xaml). У моєму проєкті було реалізовано три основні сторінки: CardsPage, ListeningPage та DictionaryPage, які користувач може перемикати з головного екрана. Shell автоматично генерує нижню панель навігації, що спрощує сприйняття.

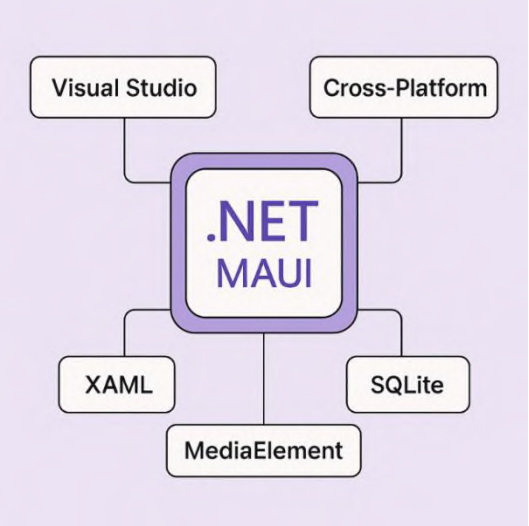


Рисунок 1.4 Структура .NET MAUI

Ще однією ключовою функцією, яку забезпечує MAUI — це локальне зберігання даних. Для цього я використав популярну бібліотеку SQLite-net, яка легко інтегрується в MAUI і дозволяє створювати та зчитувати базу даних без зайвого коду. Я створив клас-модель для слова (WordModel.cs) та сервіс роботи з базою (WordRepository.cs), які дозволяють додавати, видаляти та шукати записи. Це забезпечує повноцінний словник прямо на пристрої без потреби в інтернеті.

Крім цього, MAUI надає простий API для роботи з медіа — відтворення аудіо я реалізував через MediaElement, вбудований у платформу. При натисканні кнопки фраза відтворюється з локального файлу, і користувач може зосередитися на звучанні, не переходячи між екранами. Це зроблено без залучення сторонніх бібліотек — усе нативно, завдяки MAUI.

Під час роботи над проєктом я також оцінив зручність відладки на емуляторі Android та вбудованій підтримці hot reload, що дозволило вносити зміни в інтерфейс і бачити результат у реальному часі без повної перекомпіляції.

Таким чином, .NET MAUI став універсальним, надійним і зручним середовищем, яке дозволило реалізувати усі ключові функції застосунку LanguageHelper — від UI до локального сховища, від аудіо до навігації. При цьому код залишився читабельним, структурованим і готовим до розширення.

1.5 Концепція застосунку

Ідея створення застосунку LanguageHelper виникла з особистого досвіду регулярного вивчення англійської мови, коли постала потреба мати зручний, простий, але ефективний інструмент для повторення слів і тренування розуміння на слух, який би не вимагав постійного доступу до Інтернету, складної реєстрації або інтеграції з великими освітніми платформами. Більшість існуючих застосунків або надто гейміфіковані (що уповільнює навчальний процес), або перенасичені функціоналом, або обмежують свободу користувача в формуванні власної навчальної бази.

LanguageHelper було задумано як інструмент-помічник: не як повноцінний курс або платформа, а як мобільний додаток, який завжди під рукою й дозволяє працювати з власним контентом. Ціль — допомогти користувачу краще запам’ятовувати індивідуально обрані слова, концентруючи увагу лише на тих, які є складними або новими.

Рисунок 1.5 Переваги застосунку

1.5.1 Цільова аудиторія

Застосунок LanguageHelper орієнтований на широку аудиторію користувачів, які прагнуть ефективно вивчати та повторювати лексику іноземних

мов у зручному форматі. Його функціонал особливо стане у пригоді учням і студентам, яким потрібен швидкий доступ до повторення слів, вивчених під час занять. Завдяки простому інтерфейсу та зручному механізму тренувань, застосунок дозволяє зберігати власні слова й негайно розпочинати практику без необхідності додаткових налаштувань.

Для дорослих, які самостійно опановують нову мову, LanguageHelper може слугувати персональним помічником — без нав'язливої реклами, платних підписок і прив'язки до інтернету. Він дозволяє вивчати лексику у зручному темпі, будь-де та будь-коли, що важливо для людей із напруженим графіком. Особливо корисним застосунок стане для фахівців, яким необхідно вивчати вузькоспеціалізовану лексику, наприклад, юридичні, медичні або технічні терміни, адже дає змогу самостійно створювати власні набори слів і керувати ними відповідно до індивідуальних потреб.

Універсальність LanguageHelper полягає в тому, що він орієнтований не на готові теми, а на повну свободу користувача у формуванні словника. Це робить його особливо привабливим для тих, хто шукає спосіб тренуватись самостійно, офлайн, без обмежень у виборі лексичного матеріалу чи типу тренування.

1.5.2 Ключові функціональні елементи

Однією з ключових особливостей LanguageHelper є функціональність редагованого словника, яка надає користувачеві повний контроль над лексичним наповненням застосунку. Інтерфейс реалізований таким чином, що додавання нових слів або фраз відбувається швидко й інтуїтивно. Кожен запис містить оригінальне слово та його переклад, і при потребі користувач може внести правки або видалити непотрібний елемент. Крім цього, передбачено зручний механізм пошуку по словнику за ключовими словами, що значно полегшує навігацію навіть при великій кількості записів. Такий підхід дозволяє сформувати власну лексичну базу, яка повністю відповідає індивідуальним потребам, без зайвих або неактуальних тем.

Другим важливим елементом функціоналу є режим тренувань за допомогою карток, який ґрунтується на класичному методі асоціативного

запам’ятовування. У LanguageHelper цей режим реалізовано у двох варіантах подачі інформації. Перший — коли на екрані відображається переклад, а користувач має згадати відповідне слово іноземною мовою. Другий — зворотній, де показується слово, і потрібно пригадати його значення. Обидва підходи сприяють розвитку навичок активного та пасивного словникового запасу, забезпечуючи гнучкість у підході до навчання та можливість вибору оптимального варіанту для кожного конкретного користувача.

Після кожної відповіді користувач вказує, чи знав він слово. Якщо слово було вказане неправильно, воно автоматично позначається як «проблемне» й буде частіше з’являтись у наступних тренуваннях. Ця адаптивна логіка формує ефективну криву повторення — повторюються лише слабкі місця, а не всі слова поспіль.

Одним із розділів застосунку LanguageHelper є аудіювання — важливий елемент мовної практики, який спрямований на розвиток навичок сприйняття мови на слух. У цьому режимі користувач може прослуховувати короткі фрази, пов’язані з лексикою, що вивчається. Для кожного слова або словосполучення доступне аудіо: як попередньо вбудоване, так і додане вручну, що дає змогу адаптувати звучання до власних потреб або навіть використовувати голоси різних носіїв мови. Після прослуховування користувачеві пропонується обрати правильний переклад або впізнати слово. Система зберігає результати — правильні й неправильні відповіді фіксуються, і ті слова, які викликають труднощі, автоматично включаються до майбутніх повторень. Таким чином, створюється цикл адаптивного навчання, який підсилює запам’ятовування саме складних для користувача елементів.

Ще одним важливим аспектом застосунку є навігація та загальний користувацький досвід (UX). Інтерфейс побудований за принципом мінімалізму, щоб не відволікати від головного — навчання. Головний екран містить лише три основні кнопки: «Картки», «Аудіювання» та «Словник». Такий підхід дозволяє миттєво розпочати потрібний режим без зайвих дій. Усі переходи між розділами, додавання нових слів, вибір тренувань чи перевірка результатів реалізовані

інтуїтивно, з урахуванням зручності навіть для недосвідчених користувачів. Це робить LanguageHelper доступним і приємним у використанні навіть у щоденному режимі.

1.5.3 Адаптивність і навчальна логіка

Кожна взаємодія користувача з LanguageHelper має навчальну цінність і впливає на формування подальшої стратегії повторення матеріалу. У випадках, коли користувач дає неправильну відповідь у режимі карток або помиляється під час завдань на аудіювання, система автоматично фіксує цю помилку. Такі слова не вилучаються зі словника, а, навпаки, набувають спеціального статусу — вони вважаються проблемними і стають пріоритетними для наступних сесій. Це означає, що в майбутньому саме ці елементи з більшою ймовірністю з'являтимуться частіше, поки не будуть засвоєні належним чином.

Таким чином, у LanguageHelper реалізовано спрощений, але ефективний механізм інтервального повторення, який не потребує складних налаштувань або алгоритмів з боку користувача. Застосунок самостійно аналізує історію взаємодії — які слова викликали труднощі, коли вони востаннє тренувались, як часто на них припадали помилки — і на основі цього формує нову вибірку для кожного тренування. Такий підхід дозволяє зосередитися саме на тих елементах, що вимагають додаткової уваги, забезпечуючи індивідуальне, гнучке та ефективне засвоєння матеріалу.

1.5.4 Відмінності від аналогів

На відміну від Duolingo, де користувач прив'язаний до фіксованих тем і повинен проходити вправи у строго визначеному порядку, LanguageHelper надає повну свободу вибору — користувач може навчатися вибірково, швидко й ефективно, зосереджуючись лише на тому матеріалі, який актуальний саме в цей момент. Такий підхід особливо зручний для тих, хто має індивідуальні цілі або бажає повторити конкретні теми без зайвих кроків. А на відміну від Anki, інтерфейс LanguageHelper спеціально адаптований під мобільне використання: він не вимагає складного імпорту колод, не перевантажений технічними налаштуваннями та дозволяє миттєво додавати нові слова й фрази безпосередньо

з інтерфейсу, що значно пришвидшує і спрощує процес навчання. Така функціональна гнучкість і доступність стали одним із ключових напрямів у дипломному проєкті, присвяченому розробці цього застосунку.

1.6 Дизайн навчальних механік

1.6.1 Тренування за допомогою карток

Механізм карток у застосунку LanguageHelper є одним із ключових функціональних елементів, адже саме через нього реалізується головне завдання — закріплення словникового запасу користувача. Основна ідея — представити одне слово на екрані, змусити користувача його згадати, а потім самостійно оцінити, наскільки правильно він відповів. Таким чином формується самоконтрольований цикл навчання.

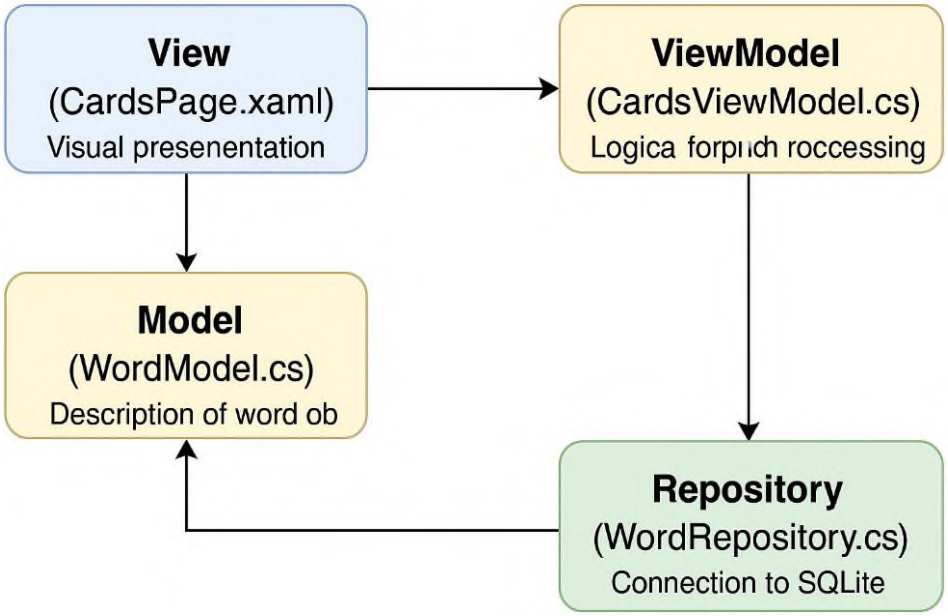


Рисунок 1.6 Логіка проєкту

Архітектурна побудова застосунку базується на шаблоні MVVM (Model–View–ViewModel), що є рекомендованим підходом для проєктів, розроблених у середовищі .NET MAUI. Така структура дозволяє розділити візуальну частину інтерфейсу, бізнес-логіку та роботу з даними, забезпечуючи зручність супроводу та масштабування проєкту.

Візуальний рівень реалізовано у файлі *CardsPage.xaml*, який відповідає за те, що бачить користувач під час тренування: відображення поточного слова або

перекладу, а також кнопки для оцінки відповіді — «Знав» і «Не знав». Уся логіка взаємодії з даними зосереджена у відповідній моделі представлення *CardsViewModel.cs*. Вона відповідає за вибір слів зі словника, обробку результатів взаємодії користувача, а також за переключення між словами та оновлення статистики. Таким чином, весь функціонал карток реалізований без прямої прив'язки до інтерфейсу, що полегшує тестування й повторне використання логіки.

Модель даних представлена класом *WordModel.cs*, який описує основні властивості кожного елемента словника — унікальний ідентифікатор, оригінал слова, його переклад, кількість допущених помилок, дату останнього оновлення та інші параметри. Робота з базою даних реалізована через окремий репозиторій *WordRepository.cs*, який виконує всі операції з SQLite: вибірку, оновлення, додавання та видалення записів. Такий підхід забезпечує чисту архітектуру та дозволяє вільно змінювати реалізацію сховища без порушення логіки на рівні ViewModel.

У структурі коду особливу роль відіграє файл *CardsViewModel.cs*, який відповідає за логіку взаємодії з даними під час тренування. Саме тут зберігається список слів, сформований для поточної сесії навчання. При ініціалізації сторінки викликається метод LoadWordsForTraining(), що здійснює фільтрацію слів зі словника. Відбір базується на двох основних критеріях: наявність помилок у минулому (значення поля MistakesCount перевищує певний поріг) та давність останнього використання слова. Таким чином, у вибірку потрапляють ті одиниці, які або складні для користувача, або ще не були повторені достатню кількість разів.

Візуальне відображення організоване через властивість CurrentWord, яка завжди містить одне поточне слово. Її значення змінюється двічі: під час початкового завантаження першого елемента та після кожної відповіді користувача. Це дозволяє плавно керувати перебігом тренування — перемикати слова без потреби у складних оновленнях інтерфейсу.

Кнопки взаємодії реалізовані через команди, що прив’язані до методів обробки відповідей. При натисканні кнопки «Знав» виконується команда MarkAsKnownCommand, яка перевіряє наявність помилок у поточного слова, за потреби зменшує значення MistakesCount, оновлює відповідний запис у базі даних через виклик WordRepository.UpdateWordAsync(word) і завантажує наступне слово у властивість CurrentWord. Аналогічно, при виборі «Не знав» активується команда MarkAsUnknownCommand, яка збільшує кількість помилок, зберігає зміни в базі та також переходить до наступного елемента. Завдяки цьому процес тренування не переривається, а користувач поступово зосереджується саме на складних для нього словах.

Рисунок 1.7 Діаграма виконання

Усі команди реалізовано з використанням CommunityToolkit.Mvvm, що дозволяє уникнути зайвого коду завдяки атрибутам [RelayCommand] і [ObservableProperty].

*SELECT * FROM Words*
WHERE MistakesCount > 0 OR LastTrained < (сьогодні - 2 дні)
ORDER BY MistakesCount DESC
LIMIT 10

Це дозволяє створити адаптивний список для повторення, який з часом сам очищається.

Візуальний інтерфейс (CardsPage.xaml)

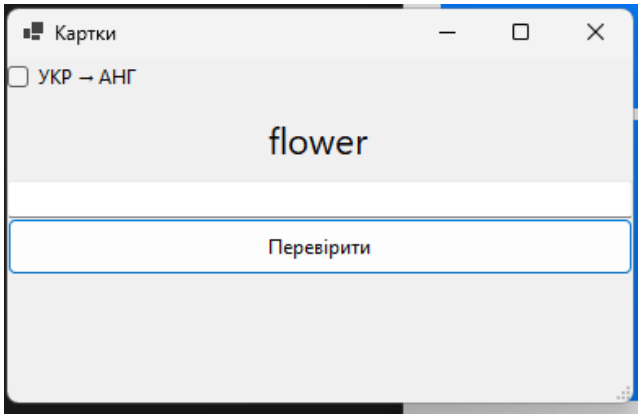


Рисунок 1.8 Інтерфейс модулю «картки»

Інтерфейс сторінки тренування у застосунку LanguageHelper побудований за принципом максимального спрощення, щоб зосередити увагу користувача виключно на навчальному матеріалі. У центрі екрана розміщується великий напис — Label, який відображає поточне слово або його переклад залежно від обраного режиму. Завдяки великому шрифту та контрастному оформленню інформація сприймається легко навіть при швидкому погляді, що важливо під час динамічного навчання.

Під словом, у нижній частині екрана, розташовані дві великі кнопки для оцінки відповіді. Зліва — червона кнопка з написом «Не знав», яка сигналізує про помилкову відповідь. Справа — зелена кнопка «Знав», яка фіксує правильну відповідь. Обидві кнопки мають значний розмір для зручного натискання, особливо на мобільних пристроях. У деяких варіантах інтерфейсу між цими кнопками або трохи нижче розміщується третя — «Пропустити». Вона є опціональною і дозволяє перейти до наступного слова без фіксації результату, якщо користувач не хоче відповідати на поточне завдання.

У верхній частині екрана знаходиться індикатор прогресу, який може відображатися у вигляді тексту типу «3 з 10» або просто числового лічильника. Це дозволяє користувачеві орієнтуватися в межах сесії, розуміючи, скільки слів уже пройдено й скільки залишилося. Такий підхід до побудови інтерфейсу

забезпечує чіткість, швидкість реакції та мінімум відволікаючих елементів, що робить навчання комфортним та ефективним.

Анімації можуть використовуватись при перемиканні карток: fade-out → fade-in, що реалізується через VisualStateManager або бібліотеки MAUI Animations Toolkit.

У ViewModel реалізовано логіку чергування напрямків запам'ятовування, що дозволяє чергувати тип завдання: з перекладу на слово або навпаки. Це досягається за допомогою внутрішньої змінної, наприклад isReversedDirection, яка визначає поточний напрямок відображення картки. Якщо змінна має значення true, користувач бачить переклад і має згадати оригінальне слово. Якщо false — спочатку показується слово, і потрібно вказати його переклад. Такий механізм дозволяє рівномірно тренувати як активне, так і пасивне знання лексики.

Щоб зробити процес навчання динамічнішим, передбачено автоматичну зміну напрямку через певний інтервал. Наприклад, після кожних п'яти карток змінна isReversedDirection інвертується. Це значення можна зберігати у налаштуваннях користувача, щоб дозволити змінювати частоту або повністю вимикати цю функцію. Зміна напрямку відбувається непомітно для користувача, проте вона урізноманітнює тренування й запобігає звиканню до однотипного формату.

Під час кожного оновлення інтерфейсу текст на екрані формується на основі цієї логіки. Властивість CurrentText, яка відповідає за відображення основного слова на картці, динамічно обирає, що саме показувати. Якщо активним є напрямок переклад → слово, то відображається CurrentWord.Translation. У зворотному випадку — CurrentWord.Original. Завдяки цьому реалізація залишається гнучкою, а код — чистим і масштабованим.

Робота з базою даних у застосунку LanguageHelper організована за допомогою бібліотеки SQLite-net, яка забезпечує просте й ефективне збереження локальних даних у мобільних застосунках .NET MAUI. Усі дані зберігаються у вигляді таблиці, де кожен рядок відповідає окремому слову, внесеному користувачем до словника. Структура таблиці чітко визначена: кожен запис має

унікальний ідентифікатор Id (тип int, первинний ключ), слово іноземною мовою (Original), його переклад українською (Translation), а також кількісні показники — MistakesCount (кількість неправильних відповідей під час тренувань) та LastTrained (дата останнього використання цього слова в картках). Додатково може зберігатися поле TimesShown, яке фіксує, скільки разів слово було показано під час сесій, що дозволяє аналізувати загальне навантаження на окремі одиниці лексики.

Під час підготовки до чергової сесії тренування ViewModel звертається до бази через відповідний репозиторій — клас, що інкапсулює всі запити до SQLite. Через метод, наприклад GetWordsForTrainingAsync(), здійснюється вибірка слів за визначеними критеріями: враховується значення MistakesCount, давність останнього повторення (LastTrained) та інші параметри, що дозволяють формувати індивідуальну та адаптивну вибірку. Репозиторій також відповідає за оновлення інформації після кожного тренування — при зміні кількості помилок або часу останнього показу відповідний запис оновлюється через UpdateWordAsync(word). Такий підхід забезпечує ефективне збереження прогресу користувача та гнучкість у формуванні нових сесій.

*SELECT * FROM Words*

WHERE MistakesCount > 0 OR LastTrained < (сьогодні - 2 дні)
ORDER BY MistakesCount DESC
LIMIT 10

Це дозволяє створити адаптивний список для повторення, який з часом сам очищається.

```
public class WordModel
{
    [PrimaryKey, AutoIncrement]
    public int Id { get; set; }
    public string Original { get; set; }         // слово іноземною
    public string Translation { get; set; }      // переклад українською
    public int MistakesCount { get; set; }       // кількість помилок
    public DateTime LastTrained { get; set; }    // останнє тренування
}
```

1.6.2 Розробка модулю аудіювання

Модуль аудіювання у LanguageHelper був спеціально створений для розвитку навичок розпізнавання іноземної мови на слух — одного з найбільш складних, але вкрай важливих аспектів мовного навчання. Його мета — допомогти користувачу не лише впізнавати знайомі слова, а й формувати асоціативний зв'язок між звучанням фраз та їх значенням. Для цього в застосунку використано стандартний елемент MediaElement, який є частиною .NET MAUI і не потребує додаткових бібліотек. Це дозволило реалізувати відтворення локальних аудіофайлів просто, стабільно й без втрати продуктивності.

Сценарій взаємодії з модулем аудіювання побудований так, щоб максимально повторювати реальні мовні ситуації. Після натискання кнопки «Прослухати» користувач чує іноземну фразу, яка може бути пов'язана з уже вивченими словами. Одразу після цього на екрані з'являються варіанти відповіді: можливі переклади або схожі фрази, серед яких треба обрати правильну. Після вибору система аналізує відповідь: якщо вона правильна, фраза вважається засвоєною; у разі помилки вона автоматично додається до черги для повторного тренування. Такий підхід дозволяє не лише покращити навичку аудіювання, а й інтегрувати її в загальну систему інтервального повторення.

Підбір фраз для аудіювання не є випадковим. Алгоритм формує сесію на основі попередніх помилок, зафіксованих у режимі карток, або вибирає ті фрази, які давно не використовувалися. Це забезпечує актуальність тренування й дозволяє зосередитися на складних для конкретного користувача прикладах. Уся інформація про фрази зберігається в моделі даних *PhraseModel.cs*, яка включає такі поля, як унікальний ідентифікатор, текст фрази, переклад, шлях до аудіофайлу, лічильник помилок та дату останнього прослуховування. Така структура дозволяє гнучко керувати контентом і забезпечує адаптивність навчання.

```
public class PhraseModel
{
    [PrimaryKey, AutoIncrement]
    public int Id { get; set; }
```

```
    public string Text { get; set; }
    public string Translation { get; set; }
    public string AudioFilePath { get; set; }
    public int MistakeCount { get; set; }
    public DateTime LastListened { get; set; }
}
```

Це клас-модель, яка описує структуру однієї фрази в модулі аудіювання. Він одночасно виступає як відображення таблиці в SQLite-базі. У ньому зберігається унікальний ідентифікатор Id, сам текст англійською (Text), його переклад (Translation), шлях до аудіофайлу (AudioFilePath), кількість помилок (MistakeCount) та дата останнього прослуховування (LastListened).

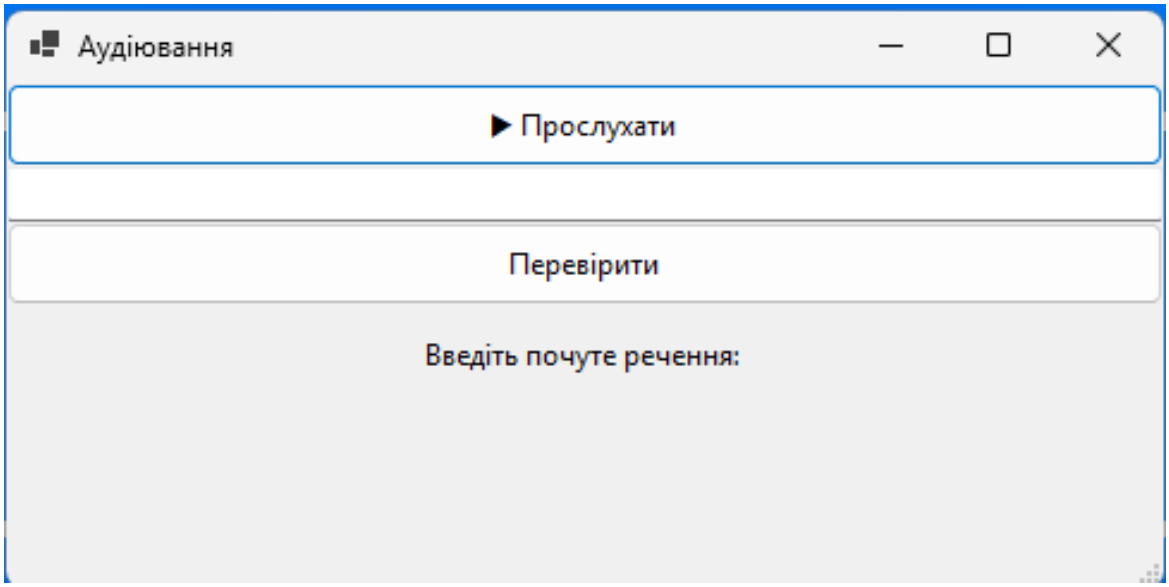


Рисунок 1.9 Інтерфейс модулю «Аудіювання»

Репозиторій — AudioRepository.cs/

```
public class AudioRepository
{
    private readonly SQLiteAsyncConnection _db;

    public AudioRepository(string dbPath)
    {
        _db = new SQLiteAsyncConnection(dbPath);
        _db.CreateTableAsync<PhraseModel>().Wait();
    }
    public Task<List<PhraseModel>> GetPhrasesForListeningAsync()
    {
```

```
        return _db.Table<PhraseModel>()
            .Where(p => p.MistakeCount > 0 || p.LastListened < DateTime.Now.AddDays(-3))
            .OrderByDescending(p => p.MistakeCount)
            .Take(5)
            .ToListAsync();
    }
    public Task UpdatePhraseAsync(PhraseModel phrase)
    {
        return _db.UpdateAsync(phrase);
    }
```

Цей клас відповідає за з'єднання з базою даних та витягування фраз, які мають бути використані в модулі аудіювання. Метод GetPhrasesForListeningAsync() повертає ті фрази, які користувач слухав давно або з якими у нього були помилки. Таким чином, система формує адаптивну вибірку — ми не показуємо випадкові фрази, а навпаки — ті, що справді потребують повторення.

Функція UpdatePhraseAsync() використовується для збереження результатів після кожного прослуховування: оновлюється кількість помилок або дата останнього тренування.

ViewModel — ListeningViewModel.cs

```
public partial class ListeningViewModel : ObservableObject
    {
        private readonly AudioRepository _repo;
        private List<PhraseModel> _phrases;
        private int _currentIndex = 0;
        [ObservableProperty]
        private PhraseModel currentPhrase;
        [ObservableProperty]
        private string userSelectedTranslation;
        [ObservableProperty]
        private bool isAnswerCorrect;
```

Це початкова частина ViewModel, яка визначає, які дані буде бачити і використовувати View (XAML-інтерфейс). Ми оголошуємо список фраз _phrases, індекс поточної фрази _currentIndex, а також публічні властивості CurrentPhrase

(фраза, що відтворюється), UserSelectedTranslation (вибір користувача) і IsAnswerCorrect (результат перевірки).

Ці властивості автоматично підтримують зв’язування (data binding) — тобто, коли змінюється CurrentPhrase, змінюється і відображення на екрані, без необхідності ручного оновлення інтерфейсу.

View (XAML) — ListeningPage.xaml

```
<MediaElement Source="{Binding CurrentPhrase.AudioFilePath}"
        AutoPlay="False"
        ShowsPlaybackControls="True"
        HeightRequest="60"
        HorizontalOptions="Center" />
```

Це елемент для відтворення аудіо. Ми використовуємо MediaElement, вбудований у .NET MAUI, і передаємо шлях до локального файлу через властивість CurrentPhrase.AudioFilePath. Вмикання та керування аудіо відбувається без потреби в сторонніх плагінах чи сервісах. Користувач може самостійно натиснути "play" або це зробить програма автоматично.

LanguageHelper має низку важливих особливостей, які роблять його зручним і функціональним інструментом для самостійного навчання. Однією з ключових переваг є підтримка локального зберігання аудіо, що означає — усі звукові файли завантажуються один раз і зберігаються безпосередньо на пристрої. Це дає змогу користувачу тренуватись без потреби в стабільному інтернет-з’єднанні, що особливо корисно у поїздках або в умовах обмеженого доступу до мережі.

Завдяки відсутності необхідності підключення до інтернету, застосунок працює швидко й автономно, що забезпечує приватність та постійну доступність функцій незалежно від зовнішніх умов. Усі дії — відтворення аудіо, вибір відповіді чи оновлення даних — виконуються локально.

Ще однією особливістю є миттєве реагування інтерфейсу. Перемикання між завданнями, перевірка відповідей і взаємодія з аудіо здійснюються без переходів між сторінками, що створює відчуття безперервності навчального процесу.

Замість перезавантаження вікон користувач отримує плавний досвід з миттєвою реакцією системи.

Крім того, реалізовано можливість повторного прослуховування фрази, що дозволяє повертатися до незрозумілого звучання стільки разів, скільки потрібно для впевненого розуміння. Це важливо для поступового формування слухової пам’яті та усунення мовного бар’єру, оскільки користувач сам контролює темп засвоєння матеріалу.

1.6.3 Розробка Словника

Словник у LanguageHelper виконує роль центрального елементу всієї системи — саме тут зосереджено основну мовну інформацію, з якою взаємодіє користувач. Усі слова, що проходять тренування в режимах карток та аудіювання, зберігаються в словнику після додавання або імпорту. Його структура побудована у вигляді зручного списку, що дозволяє швидко переглядати наявні записи, відстежувати прогрес та вносити зміни.

Користувач має змогу повноцінно керувати словником: додавати нові слова та переклади, редагувати існуючі записи, виправляти помилки або оновлювати формулювання. Для зручності реалізовано функцію пошуку, яка дозволяє знаходити потрібні слова за частиною тексту, незалежно від мови введення. Завдяки цьому навіть великі списки залишаються керованими, а користувач не втрачає часу на ручне перегортання. Такий підхід робить словник не просто сховищем слів, а гнучким інструментом для особистого навчання, що повністю адаптується під індивідуальні потреби.

Окрім базових функцій, словник у LanguageHelper інтегрується з іншими модулями програми, дозволяючи використовувати збережені слова в контексті інтерактивних вправ, тестів або ігор. Це значно підвищує ефективність засвоєння матеріалу, адже нові терміни автоматично підтягуються у відповідні тренувальні блоки. Завдяки цьому відбувається постійне повторення і закріплення лексики в різних форматах — як візуальних, так і слухових, що особливо корисно для користувачів із різними стилями навчання.

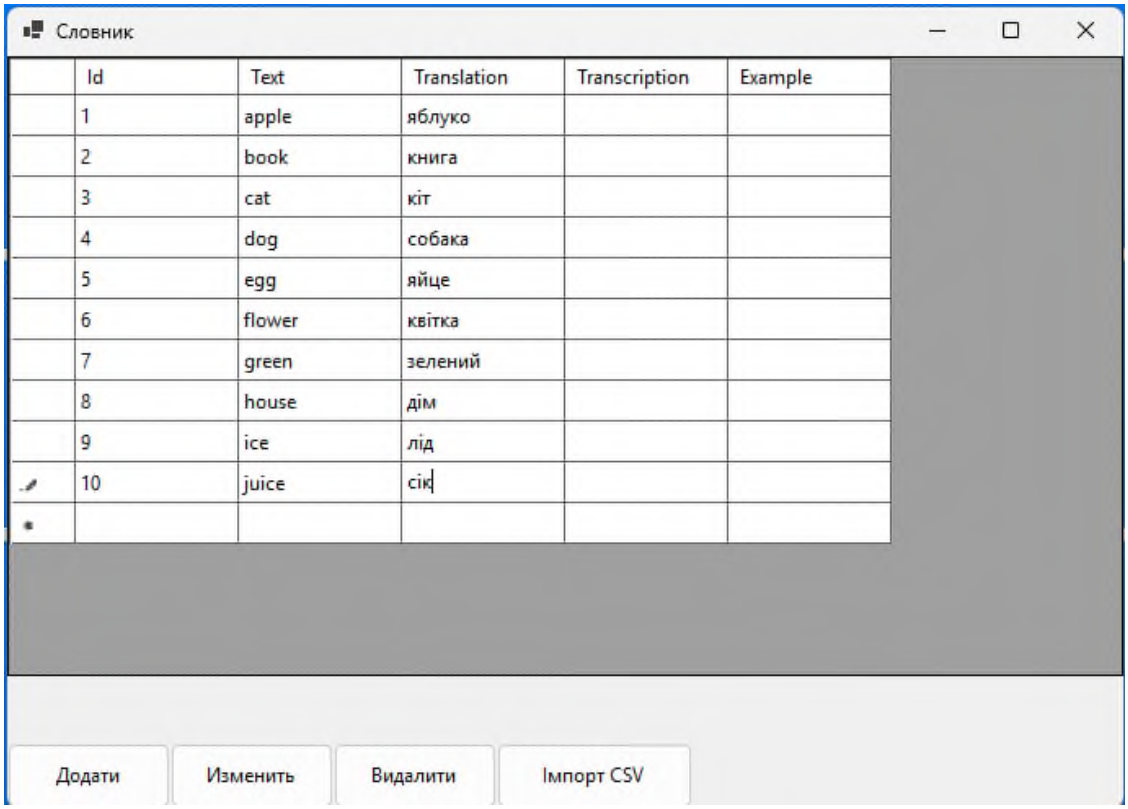


Рисунок 1.10 Інтерфейс модулю «Словник»

1.6.4 Огляд Основних функції словника

Функціональність керування словником у LanguageHelper побудована з урахуванням максимальної зручності для користувача, що дозволяє повністю адаптувати навчальний матеріал під індивідуальні потреби. Під час додавання нового слова користувач вводить оригінал іноземною мовою, його переклад українською, а за потреби може прикріпити власний аудіофайл — це відкриває можливість використовувати справжні голоси, записані вручну або взяті з зовнішніх джерел. Усі ці дані зберігаються в базі та автоматично стають доступними для тренування.

Механізм редагування записів повністю відкритий: кожен елемент словника можна змінити в будь-який момент. Це стосується як текстових полів — слова чи перекладу, так і шляху до аудіофайлу. Завдяки цьому користувач може коригувати записи при виявленні помилок або оновлювати їх за зміною власних навчальних цілей.

Для зручності реалізовано швидке видалення записів. Біля кожного рядка у списку словника розміщено іконку у вигляді кошика .Натискання на неї дозволяє миттєво вилучити непотрібне слово без зайвих підтверджень чи переходів.

Функція пошуку дозволяє оперативно знайти потрібне слово серед великої кількості записів. Для цього достатньо ввести слово повністю або частину його у спеціальне поле, після чого відображаються лише ті елементи, що відповідають запиту. Це значно пришвидшує роботу зі словником, особливо при повторному перегляді або редагуванні.

Крім того, реалізовано індикатор складності. Система використовує значення поля MistakesCount для візуального підсвічування записів. Слова, які викликають труднощі й мають високий показник помилок, автоматично виділяються певним кольором у списку, що допомагає користувачу вчасно звертати на них увагу та не ігнорувати в процесі навчання.

1.6.5 Розробка архітектури словника

Словник у LanguageHelper реалізовано за архітектурним шаблоном MVVM (Model–View–ViewModel), що забезпечує чіткий поділ відповідальностей і підтримує масштабованість коду. Кожен елемент структури відповідає окремому рівню взаємодії, дозволяючи легко супроводжувати, розширювати або модифікувати функціональність без втручання в інші частини програми.

Інтерфейс користувача реалізовано у файлі *DictionaryPage.xaml*. Тут розташовується список усіх доданих слів, зручно згрупований у вигляді таблиці або вертикального переліку. Над списком розміщується поле пошуку для фільтрації, а також кнопки для додавання нового запису та редагування існуючого. Завдяки лаконічному дизайну й зрозумілому розміщенню елементів інтерфейс залишається доступним навіть для недосвідчених користувачів.

Логіка взаємодії реалізована у *DictionaryViewModel.cs*. Саме тут обробляється логіка пошуку — введений запит передається у ViewModel, яка динамічно оновлює відображення списку відповідно до знайдених результатів. Крім того, ViewModel відповідає за оновлення словника при редагуванні запису,

ініціалізацію списку слів, а також фільтрацію за помилками або іншими критеріями, які можуть бути задані користувачем.

Модель слова описується в *WordModel.cs*. У цьому класі визначено структуру кожного елемента словника: Id, Original, Translation, MistakesCount, LastTrained, TimesShown, а також, за потреби, шлях до звукового файлу. Ця модель є основою для зберігання та передачі даних між шарами.

Для роботи з базою даних застосовується репозиторій *WordRepository.cs*, який виконує всі необхідні CRUD-операції (створення, читання, оновлення, видалення) за допомогою бібліотеки SQLite-net. Це дозволяє ViewModel зосередитися лише на логіці відображення, не займаючись технічними деталями збереження даних. Така архітектура забезпечує чистоту коду, простоту тестування та легку інтеграцію нових функцій.

```
public class WordModel
{
    [PrimaryKey, AutoIncrement]
    public int Id { get; set; }

    public string Original { get; set; }
    public string Translation { get; set; }
    public string AudioPath { get; set; }

    public int MistakesCount { get; set; }
    public DateTime LastUpdated { get; set; }
}
```

Кожен запис у словнику LanguageHelper структуровано як окремий об’єкт, що містить повний набір даних, необхідний для індивідуального контролю за вивченням слова. Центральним елементом є унікальний Id, який забезпечує ідентифікацію кожного запису в базі даних і дозволяє ефективно здійснювати операції оновлення чи видалення. Поле Original зберігає слово іноземною мовою, а Translation — його переклад українською, що становить базову мовну пару для тренувань.

Для кожного слова також зберігається шлях до звукового файлу у полі AudioPath. Це дозволяє пов’язати текст із відповідною аудіоінформацією, яку можна використовувати в режимі аудіювання. Важливим аналітичним

показником є MistakesCount — кількість помилок, допущених користувачем під час тренувань із цим словом. Значення цього поля впливає на частоту появи слова в майбутніх сесіях, формуючи адаптивний підхід до повторення.

Крім цього, у структурі запису передбачено поле LastUpdated, яке фіксує дату та час останнього редагування запису. Це дозволяє системі визначати, наскільки давно вносились зміни, а користувачеві — відстежувати актуальність словникового вмісту. Завдяки такій структурі кожен запис у словнику не лише виконує роль елемента тренування, а й стає одиницею статистики та персоналізованого аналізу навчального процесу.

```
public class WordRepository
{
    private readonly SQLiteAsyncConnection _db;
    public WordRepository(string path)
    {
        _db = new SQLiteAsyncConnection(path);
        _db.CreateTableAsync<WordModel>().Wait();
    }
    public Task<List<WordModel>> GetAllWordsAsync()
    {
        return _db.Table<WordModel>()
                .OrderBy(w => w.Original)
                .ToListAsync();
    }

    public Task<List<WordModel>> SearchWordsAsync(string query)
    {
        return _db.Table<WordModel>()
                .Where(w => w.Original.Contains(query) ||
w.Translation.Contains(query))
                .ToListAsync();
    }
    public Task<int> AddWordAsync(WordModel word)
    {
        return _db.InsertAsync(word);
    }
    public Task<int> UpdateWordAsync(WordModel word)
    {
        return _db.UpdateAsync(word);
    }
    public Task<int> DeleteWordAsync(WordModel word)
```

```
    {
        return _db.DeleteAsync(word);
    }
```

Цей клас забезпечує усі CRUD-операції для словника: отримання повного списку слів, пошук за шаблоном, додавання, оновлення та видалення. Його використовують всі інші ViewModel (включаючи картки та аудіювання), тому це єдине джерело правди в програмі.

```
public partial class DictionaryViewModel : ObservableObject
{
    private readonly WordRepository _repo;

    [ObservableProperty]
    private List<WordModel> words;

    [ObservableProperty]
    private string searchQuery;

    public DictionaryViewModel(WordRepository repo)
    {
        _repo = repo;
        LoadWords();
    }
    private async void LoadWords()
    {
        Words = await _repo.GetAllWordsAsync();
    }
    [RelayCommand]
    private async Task Search()
    {
        if (string.IsNullOrWhiteSpace(SearchQuery))
            LoadWords();
        else
            Words = await _repo.SearchWordsAsync(SearchQuery.Trim());
    }
    [RelayCommand]
    private async Task DeleteWord(WordModel word)
    {
        await _repo.DeleteWordAsync(word);
        LoadWords();
```

У цій ViewModel зберігається список слів, який оновлюється після кожної дії: додавання, пошуку або видалення. Пошук реалізовано через просту перевірку Contains(...), що дозволяє знаходити збіги за частиною слова.

1.7 Розробка дизайну інтерфейсу та візуального стилю

Інтерфейс застосунку LanguageHelper був спроєктований на основі трьох ключових принципів: мінімалізм, фокус на змісті та інтуїтивність. Кожне рішення візуального оформлення та організації елементів було спрямоване на те, щоб навчальний процес був простим, зосередженим і ефективним.

Мінімалізм проявляється у відсутності зайвих деталей — у застосунку немає складних меню, другорядних розділів або нав'язливих візуальних ефектів. Інтерфейс побудований так, щоб увага користувача була сконцентрована виключно на навчальному матеріалі: словах, перекладах, аудіофразах і результатах взаємодії. Основні елементи — це великі кнопки, чіткий текст, інтуїтивне розташування елементів, що спрощує використання навіть на невеликих екранах мобільних пристроїв.

Фокус на змісті означає, що в кожному режимі користувач бачить лише те, що необхідно для поточної дії: одне слово, одна фраза, одне завдання. Немає розсіювання уваги на кілька одночасних завдань або неважливу інформацію. Завдяки цьому користувач максимально ефективно витрачає час і може зосередитись на засвоєнні конкретного матеріалу.

Інтуїтивність інтерфейсу досягається за рахунок логічної структури й зрозумілих позначень. Усі функції — додавання слова, початок тренування, редагування запису — реалізовано так, що не потребує вивчення інструкцій. Кожен новий користувач здатен повноцінно використовувати програму вже з перших хвилин, не стикаючись із бар'єрами у взаємодії. У підсумку, інтерфейс LanguageHelper є не лише зручним, а й цілеспрямовано навчальним — він не відволікає, а підтримує ефективне мовне засвоєння.

1.7.1 Загальна структура сторінок

Інтерфейс застосунку LanguageHelper логічно структурований і поділений на три основні сторінки, кожна з яких відповідає за окремий аспект навчання.

Такий поділ дозволяє чітко розмежувати функціональність, зберігаючи простоту навігації та зрозумілість користувачеві.

Перша — CardsPage, сторінка карток, реалізує класичний режим запам’ятовування лексики. Тут користувач бачить одне слово або його переклад і має визначити правильну відповідь, обравши між варіантами «Знав» або «Не знав». Цей режим призначений для швидких повторень та тренування пам’яті, і включає логіку адаптації до рівня користувача — складні слова з’являються частіше.

Друга — ListeningPage, сторінка аудіювання, дає змогу практикувати сприйняття мови на слух. Користувач прослуховує фразу іноземною мовою та обирає правильний переклад із кількох варіантів. Усі аудіофайли зберігаються локально, що дозволяє працювати з ними офлайн. Система також фіксує помилки й повторно пропонує ті фрази, які виявились складними.

Третя — DictionaryPage, це сторінка словника, де зберігаються всі додані слова та фрази. Користувач може переглядати, редагувати, додавати або видаляти записи. Доступні також пошук за ключовими словами й індикатор помилок, що виділяє слова з високим рівнем складності. Усі дії виконуються швидко та без переходів на додаткові екрани, що забезпечує ефективну взаємодію.

Завдяки такій структурі інтерфейсу, користувач завжди має під рукою основні функції, а кожна сторінка виконує свою чітко окреслену роль у загальному процесі вивчення мови.

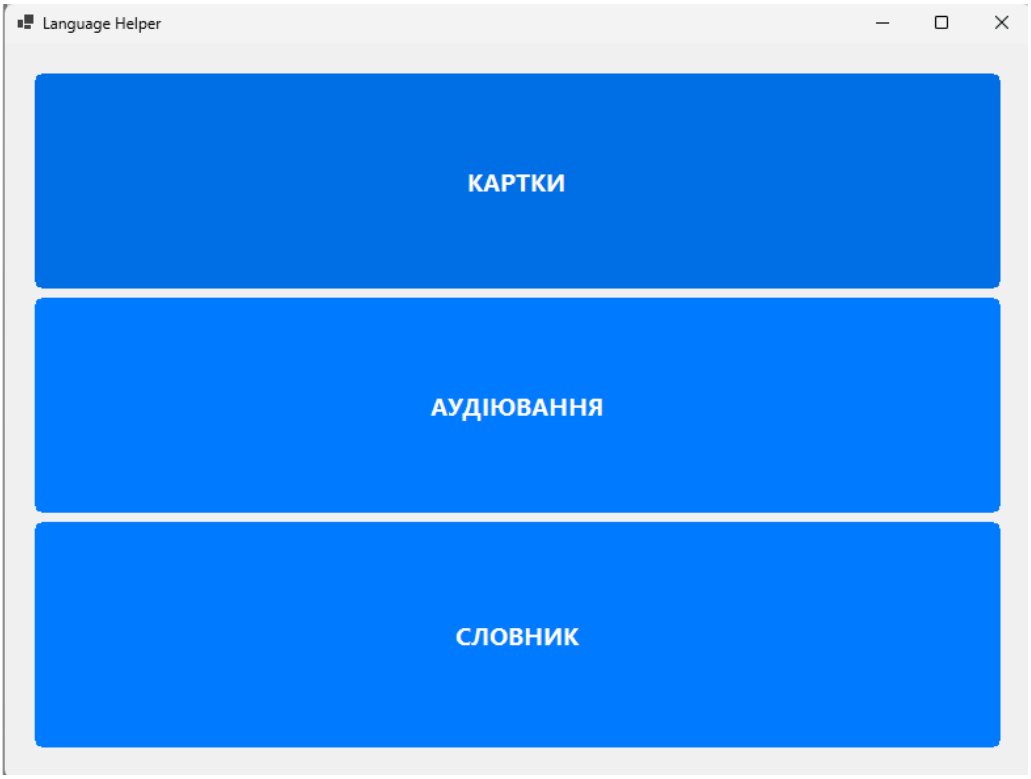


Рисунок 1.12 Інтерфейс Головного меню

Ці сторінки поєднані через нижню панель навігації (Shell Navigation), реалізовану в AppShell.xaml. Завдяки цьому перемикання між функціями здійснюється в один дотик, що особливо зручно для мобільних пристроїв.

1.7.2 Огляд Принципів UI-дизайну

Інтерфейс LanguageHelper ретельно продуманий з урахуванням ергономіки, візуальної зосередженості та відповідності вимогам доступності. Усі елементи, включно з текстами, кнопками та панелями, розміщено так, щоб не перевантажувати екран, зберігати вільний простір і сприяти повному фокусуванню уваги користувача на навчальному матеріалі.

Особливо це відчутно в модулі карток, де екран не містить жодних зайвих деталей: одночасно відображається лише одне текстове поле — або слово, або його переклад — та дві великі кнопки для взаємодії: «Знав» і «Не знав». Така організація не лише спрощує навігацію, а й допомагає користувачеві повністю зосередитися на змісті, не відволікаючись на сторонні елементи.

Колірна палітра підібрана для досягнення високої контрастності, що робить інтерфейс читабельним за будь-яких умов освітлення. Фон використовується нейтральний — білий або світло-сірий, щоб не створювати візуального

навантаження. Основні кнопки мають яскраві, інтуїтивно зрозумілі кольори: зелений означає правильну відповідь («Знав»), червоний — неправильну («Не знав»). Текст подається чорним або темно-сірим шрифтом на світлому тлі, що відповідає стандартам доступності й підходить навіть для користувачів з порушеннями зору.

Уся візуальна складова інтерфейсу оптимізована для мобільного використання. Кнопки мають достатню висоту — не менше 48 пікселів, що дозволяє комфортно працювати з ними навіть на невеликих екранах або без точного натискання. Розмір шрифтів становить щонайменше 18–20 pt, що забезпечує легкість читання без потреби масштабування. Відступи між елементами складають у середньому 12–20 pt, завдяки чому інформація сприймається структуровано та не зливається. Усе це створює приємне та ефективне середовище для навчання, яке не перевантажує й не вимагає адаптації з боку користувача.

1.7.3 UX: Зручність і передбачуваність

Інтерфейс LanguageHelper створювався з акцентом на повну прозорість і передбачуваність взаємодії. Основна мета полягала в тому, щоб користувач у будь-який момент чітко розумів, де саме він знаходиться у застосунку, яку дію виконує і що потрібно зробити далі — без необхідності читати інструкції чи вивчати логіку роботи.

Завдяки зрозумілій навігаційній структурі, кожна сторінка має власну візуальну ідентичність, що дозволяє швидко орієнтуватися у функціональних розділах. Користувач завжди бачить, у якому режимі перебуває — картки, аудіювання чи словник — завдяки відповідним заголовкам і візуальним акцентам. Усі кнопки мають інтуїтивно зрозуміле призначення та розміщені саме там, де їх очікує побачити користувач, тому навігація не потребує жодних пояснень.

Процес додавання нового слова максимально спрощений — достатньо одного-двох кліків. Вікно введення відкривається одразу з активним курсором у полі для вводу, завдяки автоматичному фокусуванню, що дозволяє миттєво почати вводити текст без зайвих рухів. У разі випадкового закриття або технічної

помилки, усі введені дані зберігаються автоматично, тож користувач не втрачає інформацію й може продовжити з того місця, де зупинився.

Запуск тренування — моментальний. Користувачеві не потрібно проходити попередні налаштування, обирати категорії чи проходити реєстрацію. Натискання на одну з головних кнопок на стартовому екрані («Картки» або «Аудіювання») негайно відкриває відповідний режим і починає роботу.

Для безпеки даних також реалізовано підтвердження при видаленні записів. Якщо користувач натискає кнопку видалення біля певного слова, з'являється діалогове вікно з уточненням дії, що дозволяє уникнути випадкових втрат важливої інформації. Усе це робить інтерфейс не лише функціональним, а й надійним, дружнім до користувача та готовим до щоденного використання без жодного бар'єру входу.

1.7.4 Розробка візуальних компонентів

У застосунку LanguageHelper використано набір візуальних компонентів, підібраних відповідно до принципів мінімалізму, зручності та ефективності взаємодії. Кожен елемент інтерфейсу має чітке функціональне призначення і відповідає завданню конкретного модуля.

Для реалізації функції аудіювання використовується MediaElement — вбудований компонент .NET MAUI для відтворення медіа. У застосунку він налаштований без зайвих елементів управління, залишено лише необхідне: можливість відтворення звуку та його повторного прослуховування. Це дозволяє сконцентруватися саме на сприйнятті мови, не відволікаючись на інші функції, і забезпечує миттєву реакцію без потреби переходів між екранами.

Для візуалізації словника використовується CollectionView — зручний елемент для відображення списку, що підтримує вертикальне прокручування навіть при великій кількості записів. Кожен елемент словника представлений окремою панеллю, яка реагує на тап користувача, відкриваючи функції редагування або детального перегляду. Поруч розміщені кнопки для швидкого редагування чи видалення запису, що дозволяє керувати словником без зайвих дій або діалогів.

Для реалізації функції пошуку по словнику використовується комбінація Entry + Button. Користувач вводить слово або його частину у текстове поле Entry, після чого натискає кнопку пошуку. В результаті відображається список слів, які відповідають запиту. Це рішення є простим, надійним і добре адаптованим під мобільний інтерфейс, дозволяючи швидко орієнтуватися у власному словнику.

1.7.5 Розробка Макетів

CardsPage побудована максимально просто й функціонально, щоб забезпечити безперервний і зосереджений процес тренування. У центрі екрана відображається поточне слово або його переклад — єдиний текстовий елемент, що дозволяє сконцентрувати увагу на змісті. Під ним розміщено дві великі кнопки — «Знав» і «Не знав», які мають контрастні кольори та зручний розмір для мобільного використання. У верхній частині сторінки знаходиться лічильник прогресу, який показує кількість пройдених карток і допомагає орієнтуватися в межах сесії.

ListeningPage призначена для аудіювання й містить усе необхідне для тренування розпізнавання фраз на слух. Угорі сторінки розташовано MediaElement, який відтворює аудіофайл. Під ним — список варіантів відповідей, зазвичай 3–4 фрази, з яких користувач має обрати правильну. Після вибору активується кнопка «Перевірити», яка ініціює перевірку відповіді. Результат негайно відображається у вигляді кольорового повідомлення: зелений індикатор — правильна відповідь, червоний — помилка, що дає чіткий і миттєвий зворотний зв’язок.

DictionaryPage реалізована як центр керування словниковим вмістом. У верхній частині розташовано поле для пошуку, що дозволяє швидко знайти потрібне слово за частиною введеного тексту. Нижче — основна частина списку слів, кожен запис містить іноземне слово, його переклад, а також кнопку видалення, розташовану збоку у вигляді іконки. У нижній частині екрана або плаваючою кнопкою в правому нижньому куті реалізовано можливість додавання нового слова — це відкриває форму введення з підтримкою текстових полів і

аудіо. Такий поділ і компонування забезпечують зручну й інтуїтивну роботу з лексичними даними.

1.8 Додавання і редагування записів у словнику

Мета реалізації словникового модуля у LanguageHelper полягає в наданні користувачеві повного контролю над лексичним вмістом. Застосунок створений таким чином, щоб кожен міг самостійно формувати словник згідно зі своїми потребами — незалежно від рівня підготовки, мови навчання чи обраної тематики. Важливою частиною стало забезпечення можливості вільного редагування будь-якого запису: користувач може не лише змінювати текст або переклад, а й оновлювати аудіо або замінювати всю фразу повністю. Усі ці дії виконуються без необхідності перезавантаження сторінки, миттєво — завдяки оновленню локального стану даних і реактивному зв’язку інтерфейсу з ViewModel.

Ключовий акцент зроблено на UX-дизайні, який не вимагає від користувача проходження інструкцій або навчання. Усе інтуїтивно зрозуміло: кнопки розміщені саме там, де їх очікують побачити, текстові поля мають підказки, і кожна дія супроводжується візуальним або логічним зворотним зв’язком.

Інтерфейс додавання нового слова реалізований у вигляді простої та лаконічної форми. Користувач бачить:

- поле для введення іноземного слова (Original), у яке автоматично встановлюється курсор при відкритті вікна;
- поле для введення перекладу (Translation);
- опціональний елемент — кнопка або поле вибору аудіофайлу (AudioPath), яке дозволяє приєднати голосову версію слова чи фрази;
- кнопку «Зберегти», яка фіксує зміни та одразу оновлює словник;
- кнопку «Скасувати» або стрілку повернення, що дозволяє вийти без збереження, не змінюючи дані.

Форма легко адаптується під мобільні екрани, підтримує валідацію введених даних та інформує про помилки (наприклад, якщо поля не заповнені). У підсумку,

ця частина інтерфейсу дозволяє швидко й комфортно керувати словниковою базою без жодних технічних складнощів.

```
ViewModel: AddEditWordViewModel.cs
public partial class AddEditWordViewModel : ObservableObject
{
    private readonly WordRepository _repo;
    private readonly INavigation _navigation;

    public bool IsEditMode { get; private set; }

    [ObservableProperty]
    private WordModel word;

    public AddEditWordViewModel(WordRepository repo, INavigation
navigation, WordModel existingWord = null)
    {
        _repo = repo;
        _navigation = navigation;

        if (existingWord != null)
        {
            Word = existingWord;
            IsEditMode = true;
        }
        else
        {
            Word = new WordModel { MistakesCount = 0, LastUpdated =
DateTime.Now };
            IsEditMode = false;
        }
    }

    [RelayCommand]
    private async Task SaveWord()
    {
        Word.LastUpdated = DateTime.Now;

        if (string.IsNullOrWhiteSpace(Word.Original) ||
string.IsNullOrWhiteSpace(Word.Translation))
            return;

        if (IsEditMode)
            await _repo.UpdateWordAsync(Word);
        else
```

```
            await _repo.AddWordAsync(Word);

            await _navigation.PopAsync(); // повернення до списку
        }
    }
```

Коли ViewModel ініціалізується без existingWord, створюється новий об’єкт WordModel, тобто режим додавання.

Якщо передано наявне слово, активується IsEditMode = true і користувач може редагувати існуючий запис.

Метод SaveWord() є ключовою частиною логіки взаємодії зі словником у LanguageHelper. Його основне призначення — перевірити, чи користувач заповнив усі необхідні поля коректно, та у разі успішної валідації — зберегти або оновити запис у базі даних через відповідний репозиторій. Після успішного збереження виконується PopAsync(), що означає повернення до сторінки словника без потреби вручну закривати форму. Таким чином, користувач одразу бачить оновлений список і продовжує роботу без додаткових дій.

Щоб зробити процес додавання або редагування максимально зручним, реалізовано низку UX-функцій, спрямованих на швидкість та інтуїтивність взаємодії. При відкритті сторінки фокус автоматично встановлюється на перше поле — введення іноземного слова, що дозволяє миттєво почати набирати текст без додаткових торкань. Кнопка «Зберегти» залишається неактивною, поки не буде заповнено обов’язкові поля — це базова валідація, яка запобігає збереженню порожніх або некоректних записів.

Натискання кнопки «Скасувати» або повернення назад здійснюється без втрати контролю: користувач повертається до попередньої сторінки без внесення змін у базу даних. Якщо йдеться про редагування існуючого запису, то відкривається та сама форма додавання, але з уже заповненими полями, що дозволяє змінити лише окремі елементи, не вводячи все з нуля. Це зменшує кількість дій і підвищує комфорт у роботі зі словником.

Усе це реалізується безпосередньо зі сторінки DictionaryPage, звідки відбувається виклик форми додавання/редагування. Зазвичай це виглядає як команда або обробник натискання кнопки «Додати слово» чи «Редагувати», яка

ініціює перехід на відповідну сторінку з передачею параметра (або порожнього слова, або вже існуючого для редагування). Це дозволяє централізовано керувати усім словниковим потоком, забезпечуючи чисту архітектуру та зрозумілий код.

```
private async void OnEditWord(WordModel selectedWord)
{
   await Shell.Current.GoToAsync(nameof(AddEditWordPage),
      new Dictionary<string, object> { { "ExistingWord", selectedWord } });
}

private async void OnAddNewWord()
{
   await Shell.Current.GoToAsync(nameof(AddEditWordPage));
}
```

З точки зору користувача, взаємодія зі словником у LanguageHelper виглядає максимально просто, логічно та безперервно. Коли користувач натискає кнопку «Додати» або вибирає вже існуюче слово зі списку, одразу відкривається окрема сторінка редагування. Інтерфейс цієї сторінки знайомий та інтуїтивний — з полями для введення іноземного слова, перекладу та, за потреби, аудіофайлу.

Користувач вводить нову інформацію або редагує наявну, не відволікаючись на сторонні дії. Після цього натискання кнопки «Зберегти» виконує збереження змін до бази даних, і користувач автоматично повертається до попередньої сторінки — словника. Жодних зайвих підтверджень чи проміжних екранів не потрібно.

Найважливіше — список оновлюється миттєво після повернення. Нове слово одразу з’являється у загальному переліку, а редагований запис замінюється актуальними даними. Це створює ефект живого словника, де будь-яка дія дає відчутний результат, а взаємодія відбувається без затримок або потреби вручну оновлювати сторінку. Така поведінка підсилює відчуття контролю та простоти, що є однією з основних цінностей інтерфейсу LanguageHelper.

1.9 Схема зберігання даних

Зберігання даних є ключовим компонентом у побудові застосунку LanguageHelper, адже саме з базою пов’язані всі основні функції: словник, модуль

карток, аудіювання, облік складності слів та персоналізація навчального досвіду. Для цього в проєкті була реалізована локальна база даних на основі SQLite, яка інтегрується безпосередньо в середовище .NET MAUI.

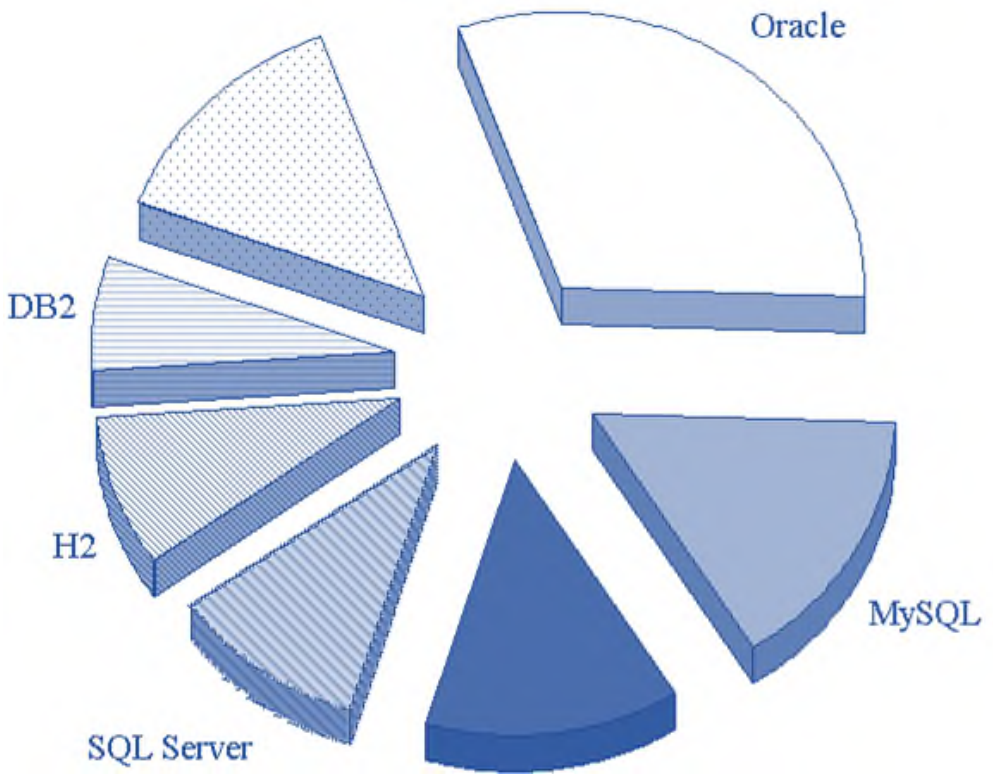


Рисунок 1.13 Інфограма порівняння використання БД

1.9.1 Обґрунтування вибору СУБД

SQLite є вбудованою реляційною системою управління базами даних, яка поєднує в собі легкість, продуктивність і повноцінну підтримку стандарту SQL. Її особливістю є відсутність необхідності в окремому сервері чи додаткових сервісах — уся база даних зберігається у вигляді звичайного локального файлу, з яким застосунок працює безпосередньо. Така архітектура забезпечує мінімальне навантаження на систему та високу швидкодію навіть за умови обробки значного обсягу даних.

Однією з ключових переваг використання SQLite є її здатність працювати в автономному режимі. Це особливо важливо у контексті мобільного навчального застосунку, для якого гарантія доступності даних незалежно від наявності інтернет-з’єднання є критичною умовою. Завдяки збереженню всіх навчальних

матеріалів, статистики користувача та інших записів безпосередньо на пристрої, застосунок залишається повністю функціональним навіть у віддалених або нестабільних мережевих середовищах. Крім того, SQLite легко інтегрується з платформою .NET MAUI, що дозволяє зручно працювати з нею в межах багатоплатформеного мобільного рішення.

1.9.2 Загальний підхід

База даних створюється автоматично при першому запуску програми. Для цього реалізовано клас WordRepository, який відповідає за ініціалізацію бази та виконання всіх операцій (CRUD): створення, читання, оновлення й видалення записів. Уся взаємодія з SQLite відбувається асинхронно за допомогою об'єкта SQLiteAsyncConnection.

Уся інформація про слова зберігається у єдиній таблиці Words, яка виступає ядром для всіх інших функцій програми.

1.9.3 Структура запису

Кожен запис у базі даних застосунку LanguageHelper представлений як об’єкт моделі WordModel, яка описує повну структуру одиниці лексики та її історію використання в навчальному процесі. Завдяки чітко визначеним полям ця модель забезпечує як збереження даних, так і їх ефективне використання в різних режимах — картки, аудіювання, перегляд словника.

Поле Id є унікальним ідентифікатором для кожного слова. Це автоматично згенероване ціле число, що використовується в базі даних для швидкого доступу, оновлення або видалення конкретного запису. Воно гарантує, що всі операції з конкретним словом відбуваються без конфліктів.

Поле Original містить текст слова або фрази іноземною мовою. Це основна навчальна одиниця, з якою працює користувач під час тренування — наприклад, слово англійською, німецькою чи будь-якою іншою мовою, яку вивчає.

Translation — переклад слова українською. Це поле виступає у парі з Original і використовується для перевірки знань або формування варіантів відповідей.

AudioPath є опціональним полем, що містить шлях до локального аудіофайлу, пов’язаного з відповідним словом. Його використання дозволяє

інтегрувати озвучення у тренування, особливо в модулі аудіювання, де користувач може прослухати фразу перед тим, як відповісти.

MistakesCount зберігає кількість неправильних відповідей, які користувач дав при взаємодії з цим словом — як у картках, так і під час аудіювання. Це поле лежить в основі реалізації адаптивного повторення: чим частіше слово викликає труднощі, тим активніше система включає його у майбутні сесії.

Поле LastTrained фіксує дату й час останнього використання цього слова під час тренування. Завдяки цьому можна контролювати частоту повторень — не перегружати користувача надто частими показами відомих слів і давати перевагу новим або забутим.

Нарешті, LastUpdated містить дату останнього редагування або створення запису. Це дозволяє не лише сортувати слова за часом додавання, а й відображати відповідні позначки в інтерфейсі — наприклад, «нове», «щойно змінено», або групувати записи за періодами активності користувача. Усі ці поля разом забезпечують повну підтримку персоналізованого навчання, статистики, гнучкого повторення та ефективної побудови інтерфейсу.

Рисунок 1.14 Структура зберігання даних

1.9.4 Логіка взаємодії з іншими модулями

Таблиця Words є центральним джерелом даних у застосунку LanguageHelper і використовується у всіх основних функціональних блоках. Її універсальність дозволяє підтримувати цілісну логіку навчання, зменшувати складність коду та забезпечувати узгодженість даних між модулями.

У модулі карток саме ця таблиця постачає слова для тренування. З неї вибираються записи відповідно до заданих критеріїв (наприклад, кількість помилок або час останнього використання), і кожне слово відображається на екрані. У випадку неправильної відповіді автоматично збільшується значення поля MistakesCount, а також оновлюється LastTrained, що дозволяє системі адаптувати подальші сесії до індивідуальних труднощів користувача.

Модуль аудіювання працює з підмножиною тієї ж таблиці — з тими словами, які мають вказаний AudioPath, тобто прикріплений звуковий файл. Після прослуховування користувач обирає правильний варіант перекладу, і залежно від відповіді також змінюється статистика: зростає MistakesCount у разі помилки та оновлюється LastTrained. Таким чином, тренування на слух не відокремлене, а є невід’ємною частиною єдиної системи запам’ятовування.

У словнику таблиця Words використовується для повного представлення всіх збережених слів. Користувач може переглядати записи, редагувати їх, сортувати за датою, фільтрувати за кількістю помилок або виконувати пошук за вмістом полів Original чи Translation. Дані також відображаються у структурованому вигляді, включаючи індикатори складності та час останнього оновлення.

Завдяки такому підходу, коли всі модулі працюють з однією таблицею, значно спрощується архітектура застосунку. Це забезпечує узгодженість даних, зменшує дублювання логіки, спрощує підтримку та дозволяє легко масштабувати функціонал у майбутньому — наприклад, додавати нові типи тренувань або експортувати статистику без потреби в додаткових джерелах.

1.9.5 Реалізація з’єднання з базою

Файл бази даних розміщується у захищеному внутрішньому каталозі застосунку (FileSystem.AppDataDirectory). Це унеможливлює доступ сторонніх додатків або користувачів до бази без рут-доступу.

```
private static readonly string dbPath =
Path.Combine(FileSystem.AppDataDirectory, "words.db");
private static SQLiteAsyncConnection _database;
public static async Task InitAsync()
{
    if (_database != null) return;
    _database = new SQLiteAsyncConnection(dbPath);
    await _database.CreateTableAsync<WordModel>();
}
```

Цей код ініціалізує базу, якщо вона ще не існує, та створює таблицю WordModel зі всіма полями.

1.9.6 CRUD-операції

Усі операції зі збереження, оновлення, пошуку та видалення даних у застосунку LanguageHelper інкапсульовані в спеціальному класі WordRepository, який виконує роль посередника між базою даних SQLite та бізнес-логікою програми. Це дозволяє суворо дотримуватись принципів чистої архітектури, із чітким розмежуванням відповідальностей: UI не працює напряму з базою, а звертається до неї виключно через ViewModel, яка, у свою чергу, використовує відповідні методи репозиторію.

Рисунок 1.15 Основні операції з базами данних

Метод AddWordAsync(WordModel word) відповідає за створення нового запису в таблиці — його викликають при додаванні слова користувачем через форму. UpdateWordAsync(WordModel word) використовується під час редагування: змінюється текст, переклад, прикріплене аудіо або статистика помилок. DeleteWordAsync(WordModel word) реалізує логіку видалення запису,

викликається при натисканні кнопки з іконкою кошика в словнику. Метод GetAllWordsAsync() повертає повний список слів і використовується при завантаженні словника або підготовці до тренування. Для зручного пошуку реалізовано SearchWordsAsync(string query), який знаходить усі слова, що містять заданий фрагмент у полі Original або Translation.

Схема зберігання даних, побудована навколо єдиної таблиці Words, є однією з найсильніших сторін реалізації LanguageHelper. Вона дозволяє не лише зберігати повноцінну мовну інформацію, а й керувати навчальним процесом на основі статистики. Такий підхід забезпечує цілісність даних — відсутність дублювань або логічних конфліктів між модулями. Крім того, єдина структура полегшує гнучку інтеграцію з різними частинами застосунку: картки, аудіювання, словник — усі використовують одні й ті самі записи з єдиними полями.

Завдяки простій структурі таблиці та легкій ORM-обгортці у вигляді SQLite-net досягається висока швидкодія, навіть на малопотужних або старих мобільних пристроях. При цьому зберігається повна автономність: усі функції програми — від пошуку до аудіювання — працюють без підключення до інтернету. Це робить LanguageHelper особливо зручним у польових умовах, у дорозі чи в місцях із нестабільним доступом до мережі. Також, завдяки полю MistakesCount та фіксації останнього використання, можливо реалізувати адаптивне навчання, орієнтоване на помилки користувача, що робить процес засвоєння лексики гнучким і персоналізованим.

Ця структура є фундаментом усієї навчальної логіки, і її можна легко розширити в майбутньому, додаючи, наприклад, теги, категорії, складність чи підтримку декількох мов без значних змін у коді.

1.10 Тестування та відлагодження застосунку

Якість програмного забезпечення відіграє вирішальну роль у його сприйнятті користувачем, адже саме від неї залежать стабільність, зручність та довготривале функціонування системи без збоїв. У контексті проєкту LanguageHelper забезпечення якості стало одним із пріоритетних завдань, що реалізувалося через комплексне тестування всіх функціональних компонентів

програми. Особлива увага приділялася тому, щоб застосунок демонстрував передбачувану поведінку в реальних умовах використання, включно з урахуванням обмежень мобільного середовища, таких як переривання сесій, зміна орієнтації екрана, обмежені ресурси пристрою та нестабільність з’єднання.

Процес тестування здійснювався у тісному зв’язку з кожним етапом розробки, що дозволило інтегрувати перевірку функціональності в саму структуру життєвого циклу програмного продукту. Завдяки такому підходу потенційні помилки виявлялися ще до їх накопичення, а відхилення від очікуваної поведінки усувалися оперативно — до того, як вони могли вплинути на загальну стабільність системи. Усі ключові модулі застосунку, зокрема словник, картки та аудіювання, проходили окреме функціональне тестування з урахуванням специфіки кожного з них.

Значну увагу було приділено перевірці базових сценаріїв, таких як створення нових записів, їхнє редагування та збереження, а також видалення з бази. У межах тренувального режиму аналізувалася поведінка застосунку у відповідь на дії користувача — як у разі правильних відповідей, так і за наявності помилок. Це дозволило переконатися в коректності логіки обробки результатів і зворотного зв’язку в інтерфейсі. Крім того, окремо тестувалися візуальні елементи — адаптація до розміру екрана, правильне відображення динамічного контенту, чутливість до дотиків, стабільне оновлення елементів після змін у базі даних та адекватне реагування на жести, що є характерним для мобільних платформ.

У результаті такого комплексного підходу було досягнуто високого рівня надійності та передбачуваності роботи застосунку, що особливо важливо в освітньому середовищі, де стабільність і зручність користування є критичними факторами успішного засвоєння матеріалу.

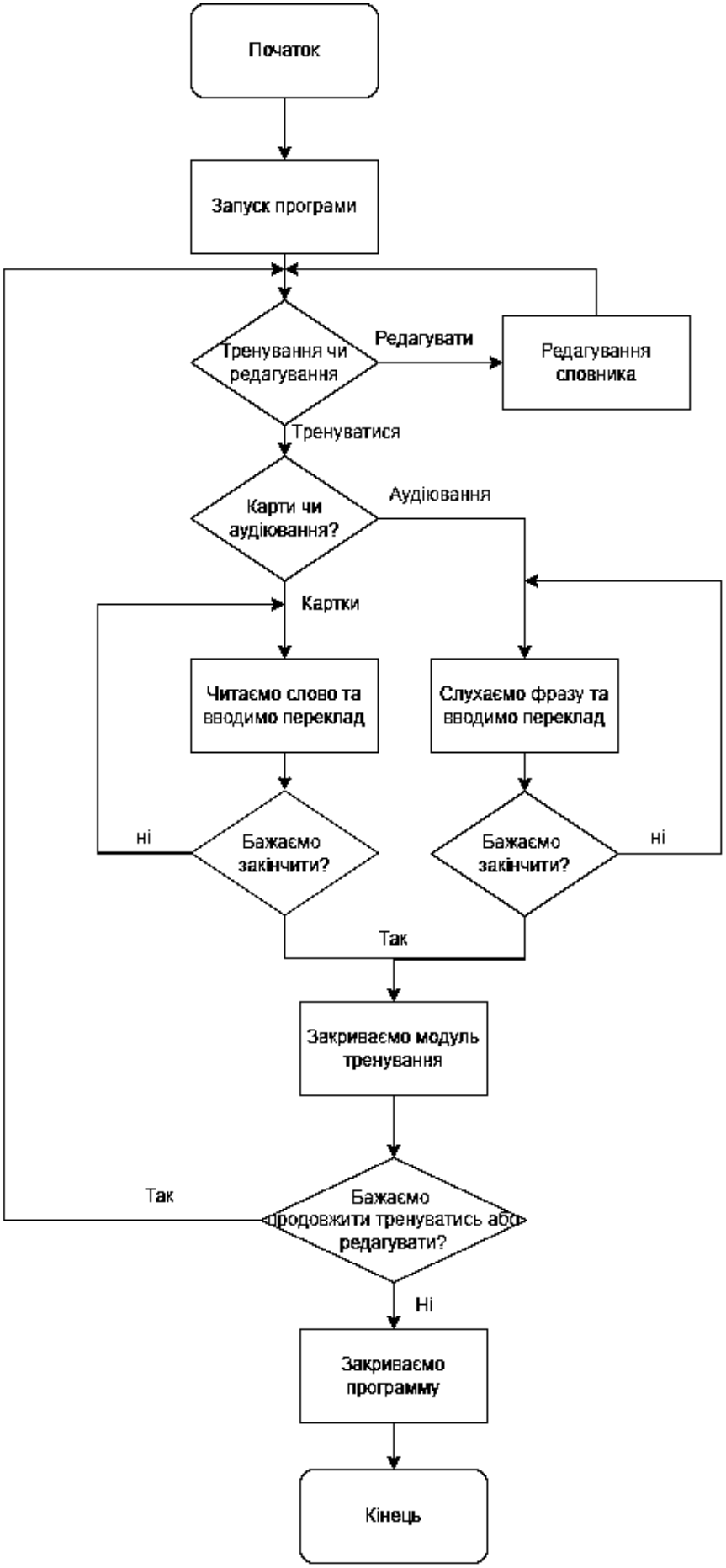


Рисунок 1.15 Загальна діаграма поведінки користувача

Окремо було проведено ручне функціональне тестування застосунку на фізичному Android-пристрої. Перевірялись усі основні сценарії використання — від першого запуску й додавання слова до роботи з аудіо, переходів між сторінками, зміни режимів тренування та роботи з локальними файлами. У результаті цього тестування вдалося виявити та виправити низку дрібних неточностей у верстці, помилки фокусування, а також оптимізувати швидкодію на різних екранах.

1.10.1. Підходи до тестування

У рамках реалізації проєкту LanguageHelper було проведено комплексне тестування, що охоплювало всі ключові рівні — від логіки даних до користувацького інтерфейсу. Такий підхід дозволив не лише перевірити працездатність окремих частин системи, а й забезпечити узгодженість усього застосунку як єдиного програмного продукту.

Перший тип — модульне тестування (unit testing) — застосовувався для перевірки окремих методів класу WordRepository. Було протестовано такі операції, як додавання нових слів до бази, оновлення існуючих записів, їх видалення, а також пошук за частиною тексту. Це дозволило переконатися, що вся базова логіка обробки даних працює коректно й не порушує цілісність бази.

Далі проводилось функціональне тестування, яке охоплювало цілісні сценарії взаємодії користувача із застосунком. Перевірялися основні функції, зокрема: проходження тренувань у режимі карток, прослуховування фраз із подальшим вибором перекладу, додавання та редагування слів у словнику, пошук і сортування. Завдяки цьому вдалося виявити логічні помилки у ланцюжках дій та удосконалити UX.

UI-тестування здійснювалось вручну на реальному пристрої. Воно включало оцінку зовнішнього вигляду кожної сторінки, перевірку відповідності розмірів шрифтів та інтерфейсних елементів, реакцію на натискання, правильність фокусування, а також плавність анімацій і переходів. Це дало змогу адаптувати інтерфейс до різних розмірів екранів та сценаріїв взаємодії.

Окремо проводилось тестування на помилки — штучне створення ситуацій з некоректними або неповними даними. Перевірялися реакції системи на порожні поля при додаванні слова, введення недопустимих символів, некоректні шляхи до аудіофайлів. Застосунок у всіх таких випадках мав або блокувати дію, або видавати повідомлення про помилку без аварійного завершення.

Завершальним етапом стало тестування навігації, яке перевіряло правильність переходів між сторінками через механізм Shell. Переконались, що всі маршрути працюють стабільно, не виникає збоїв при поверненні назад, а також що стан ViewModel зберігається між переходами. У результаті застосунок пройшов повний цикл перевірок, що дозволило досягти високої стабільності, зрозумілості й зручності у повсякденному використанні.

1.10.2 Огляд інструментів тестування

Під час розробки та тестування застосунку LanguageHelper використовувався набір інструментів, який дозволив перевірити поведінку системи як у віртуальному середовищі, так і на реальних пристроях. Це дало змогу оцінити не лише коректність логіки, а й продуктивність, адаптивність до різних екранів і стабільність у роботі.

Один із ключових засобів — емулятор Android (AVD), який входить до складу Android SDK. Він використовувався для тестування інтерфейсу на різних розмірах екранів і орієнтаціях пристрою. Завдяки цьому вдалося перевірити масштабування тексту, розташування кнопок, прокручування списків і загальну адаптивність UI до смартфонів із різною щільністю пікселів.

Паралельно проводилося тестування на фізичному смартфоні, що дозволило оцінити реальну швидкість відклику інтерфейсу, плавність анімацій, чутливість кнопок і, що особливо важливо — якість аудіовідтворення у модулі прослуховування. Таке тестування також виявило специфічні відмінності в поведінці на реальних пристроях, які не завжди помітні в емуляторі.

Для перевірки логіки програми та відстеження змін у змінних активно застосовувався Visual Studio Debugger. Використання точок зупину (breakpoints) дозволяло покроково аналізувати виконання коду: перевіряти, як формується

запит до бази, які значення передаються у ViewModel, і як саме обробляється відповідь. Це допомогло вчасно виявляти й усувати помилки, які не проявлялися візуально.

Допоміжним, але дуже важливим інструментом під час розробки була Output Console у Visual Studio. За допомогою Console.WriteLine() виводився журнал подій: завантаження сторінок, виклики методів, зміна стану змінних. Такий журнал значно полегшував виявлення логічних збоїв або неочікуваної поведінки застосунку без необхідності запускати дебаг кожного разу.

Завдяки поєднанню цих інструментів — емулятора, фізичного пристрою, відлагодження в IDE та журналу виводу — вдалося досягти глибокої перевірки застосунку та забезпечити його стабільну роботу на всіх підтримуваних платформах.

1.10.3 Типові проблеми та їх вирішення

Під час розробки LanguageHelper було виявлено низку функціональних недоліків, які могли впливати на зручність використання та точність навчального процесу. Всі вони були проаналізовані й усунуті через внесення змін до логіки ViewModel та поведінки інтерфейсу. Нижче наведено приклади ключових проблем і способів їх вирішення.

Однією з ранніх проблем була можливість збереження запису без перекладу. Це виникало через відсутність перевірки на порожні поля, у результаті чого до бази могли потрапити неповноцінні записи. Для розв’язання цієї ситуації у ViewModel було додано перевірку з умовою if (string.IsNullOrWhiteSpace(...)) return;, яка блокує виконання SaveWord() у разі відсутності обов’язкових значень.

Інша поширена ситуація — дублювання фраз під час тренування картками. Через недосконалий SQL-запит у вибірці іноді траплялися повтори одного й того самого слова. Це не лише ускладнювало навчання, а й створювало враження збою в логіці. Вирішенням стало використання Distinct() при формуванні колекції для тренування, що гарантує унікальність елементів.

Ще одна проблема стосувалась некоректного оновлення поля MistakesCount. Після правильної відповіді система не зменшувала значення

помилок, через що слово залишалось у переліку складних. У ViewModel було додано логіку: MistakesCount = Math.Max(0, MistakesCount - 1), яка знижує кількість помилок, але не дозволяє їй опуститися нижче нуля, зберігаючи цілісність даних.

Остання важлива помилка — невірне оновлення списку слів після повернення зі сторінки редагування. Раніше при поверненні на DictionaryPage зміни не відображалися без перезапуску застосунку. Це було виправлено шляхом додавання обробки події OnAppearing() у відповідних сторінках, де викликається метод LoadWords(). Таким чином, список оновлюється автоматично щоразу при повторному вході на сторінку.

Усі ці покращення значно підвищили стабільність застосунку та покращили загальний користувацький досвід, зробивши роботу з LanguageHelper більш передбачуваною, зручною та ефективною.

1.10.4 Результати тестування

Після завершення повного циклу ручного тестування на фізичному Android-пристрої (версія 11) застосунок LanguageHelper продемонстрував стабільну та надійну роботу в усіх ключових сценаріях використання. Ретельна перевірка кожного модуля в реальних умовах дозволила переконатися у високій якості реалізації та готовності застосунку до щоденного використання кінцевим користувачем.

Усі навігаційні переходи між сторінками — словник, картки, аудіювання — працювали чітко й без помилок, навіть при багаторазовому перемиканні режимів або повторному відкритті модулів. Це підтвердило правильність реалізації маршрутизації через Shell і збереження контексту між переходами.

Режими тренування, як у вигляді карток, так і в аудіюванні, функціонували без затримок чи зависань. Навіть при швидкій зміні відповідей або швидкому перемиканні між словами застосунок стабільно оновлював інтерфейс і не втрачав синхронізації з базою. Важливо, що оновлення MistakesCount і LastTrained проходило коректно навіть у ситуаціях швидкого багаторазового введення.

База даних SQLite також продемонструвала стійкість до навантаження — перевірки показали, що навіть при інтенсивному додаванні, редагуванні чи видаленні слів дані зберігаються без втрат або пошкоджень. Жодних винятків або зависань при роботі зі словником не зафіксовано.

Особливо важливим був результат тестування сценарію, коли застосунок згортається у фон, а потім відновлюється. LanguageHelper відновлює свій стан коректно — як у картках, так і в аудіюванні. Дані не губляться, поточне слово або фраза зберігається, а користувач може продовжити навчання з того місця, на якому зупинився.

Окремо було підтверджено стабільну роботу звуку на всіх сторінках, де використовується MediaElement. Аудіофайли відтворюються плавно, без затримок, повторно й незалежно від кількості переходів або перезапусків.

У результаті тестування LanguageHelper показав високий рівень стабільності, відмовостійкості та відповідності очікуваному функціоналу, що дозволяє вважати його готовим до реального використання кінцевими користувачами.

1.10.5 Перевірка на виняткові ситуації

У межах завершального етапу ручного тестування застосунок LanguageHelper було перевірено в умовах, де найчастіше можуть виникати збої або нестандартна поведінка. Основна мета цих перевірок — переконатися, що система поводиться передбачувано навіть у ситуаціях, коли користувач діє не за сценарієм або працює з неповними даними.

Було протестовано запуск програми без наявності аудіофайлів. Компонент MediaElement спрацював коректно: навіть у разі відсутності вказаного шляху до звуку жодних винятків не виникало, UI залишався стабільним, а застосунок не завершував роботу аварійно.

Ще одна ситуація — натискання кнопки «Зберегти» при порожніх полях у формі додавання або редагування слова. Завдяки вбудованій валідації, у таких випадках дія блокується, запис до бази не надсилається, а інтерфейс залишається

на місці. Це гарантує цілісність даних і виключає можливість появи порожніх або некоректних записів у словнику.

Також перевірялося швидке багаторазове натискання кнопок під час тренування. Усі дії — зокрема відповіді «Знав» / «Не знав» або натискання «Перевірити» в аудіюванні — фіксувались чітко та без дублювання. Логіка в ViewModel правильно обробляла всі події, не створюючи зайвих записів і не викликаючи повторного оновлення одного й того самого елемента.

У підсумку, LanguageHelper пройшов повноцінне ручне тестування, що включало перевірку всіх ключових функцій: додавання й обробку даних, навігацію між сторінками, стабільність взаємодії з базою, а також поведінку у нетипових ситуаціях. Завдяки використанню перевіреного архітектурного підходу MVVM і надійної технології SQLite, застосунок демонструє високу стабільність, швидкодію та стійкість до помилок, навіть у випадках інтенсивного використання чи повторюваних дій. Це робить його готовим до повсякденного використання без ризику для користувача або втрати даних.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

У результаті реалізації дипломного проєкту було створено повноцінний мобільний застосунок для вивчення іноземних слів та фраз, що поєднує в собі адаптивну систему навчання, зручний словник, функції аудіювання та інтуїтивний інтерфейс.

Рівень ефективності програмного застосунку залежить не тільки від його якісних показників, але й від того, наскільки результативним був процес його розробки. Оцінювання якості ПЗ здійснюється за низкою критеріїв, серед яких враховуються потреби кінцевих користувачів, ефективне використання ресурсів і відповідність встановленим вимогам. Із погляду користувача, важливою складовою оцінки є аналіз витрат на розробку, що охоплює обсяг трудових ресурсів і загальну вартість. У цьому розділі наведено обґрунтований розрахунок вартості створеного програмного продукту.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки залежить від ряду факторів, включаючи обсяг робіт, рівень складності, кваліфікацію команди розробників та часові обмеження, зумовлені ринковими умовами. Для оцінки масштабу програмного продукту використовується метод структурної аналогії, що передбачає звернення до спеціалізованих каталогів схожих проєктів. На їхній основі визначається кількість умовних машинних команд, необхідних для реалізації аналогічного програмного рішення (у тисячах команд).

Таблиця 2.1 Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, умовн. машинних командах.**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
1. ПП автоматизації засобів по каталогу	1060 – 5750

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2 Норма часу

Обсяг ПЗ, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 244 х 0,8 = 195,2 (люд/годин).

Трудомісткість програмного продукту визначається окремо для кожного етапу розробки, виходячи з показників трудомісткості відповідного аналога. При цьому враховують рівень складності розробки, ступінь інноваційності та частку використання стандартних модулів. Розрахунки проводяться згідно з наступними формулами:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5)

Таблиця 2.3 Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4 Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5 Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а *L_1*К_н= 195,2 *0,12*0,7= 16,39 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а *L_2*К_н =195,2 *0,11*0,7 = 15,03 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Т^а *L_3*К_н*К_т=195,2 *0,61*0,7*0,6= 50,01 \text{ (люд/годин)} \quad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на

кожен етап: технічне завдання $N_{тз}$= 3 (стр), розробка ТП $N_{тп=}$29(стр), розробка робочого проекту $N_{рп}$=6(стр), пояснювальна записка відповідно $N_{пз}$=28(стр) Розрахунок зведений у таблицю 2.6.

Таблиця 2.6 Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин**		
1.ТЗ	$T_{Ртз}$=16,39	$T_{кк}=0,7*N_{тз}=$ 0,7*3=2,1	$T_{нк}=0,15*N_{тз}=0,15*3=0,45$
2.Розробка ТП	$T_{Ртп}$=15,03	$T_{кк}=0,7*N_{тп}=0,7*29=2$ 0,3	$T_{нк}=0,15*N_{тп}=0,15*29=4,3$ 5
3.Розробка РП	$T_{Ррп}$= 50,01	$T_{кк}=0,7*N_{рп}=0,7*6=4,$ 2	$T_{нк}=0,15*N_{рп}=0,15*6=0,9$
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}=$ 1,5*28=42	$T_{кк}=0,7*N_{тз}=0,7*28=1$ 9,6	$T_{нк}=0,15*N_{пз}=0,15*28$ =4,2
Усього, в т.ч.:	179,53		
- на розробку	$\Sigma T_р$=123,43		
- контроль керівника		$\Sigma T_{кк}$=46,2	
- нормоконтроль			$\Sigma T_{нк}$=9,9

2.3 Розрахунок ціни програмного продукту

Для оцінки вартості програмного продукту розглядаються основна заробітна плата виконавців, матеріальні витрати та загальні витрати на розробку ПЗ. Детальний розрахунок основної заробітної плати наведено у таблиці 2.7. Згідно зі статтею 8 «Закону про Державний бюджет України на 2025» з 1 січня 2025 року встановлено мінімальну місячну заробітну плату у розмірі 8000 гривень, а також мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7 Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	123,43	48,00	5924,64
2.Контроль керівника	46,2	95,00	4389,00
3.Нормоконтроль	9,9	95,00	940,50
Усього	-	-	$\Sigma З_о$= 11254,14

Зробимо розрахунок матеріальних витрат на розробку ПП (табл.2.8).

Таблиця 2.8 Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	66	5.0	330,00
Разом	-	-	-	$В_{м\,і}$ =330,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$ = 0,1*285=28,5
Усього				$В_м = В_{мі} + В_{тр\_з=}$ 313,50

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9 Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	363,00	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	11254,14	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1125,41	$Зд = 0{,}1 \times Зо$ = 11254,14*0,1
4.Відрахування до єдиного фонду соціального внеску	2723,50	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (11254,14+1125,41)
5. Накладні витрати	4501,65	$Внак. = 0{,}4 \times Зо$ =0,4 * 11254,14
6. Повна собівартість	19967,7	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 363,00+11254,14+1125,41+2723,50 +4501,65

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_{пов}*Р)/100=(19967,7*10)/100=1996,77$ грн (2.7)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_{пов}+П$ = 19967,7+1996,77 = 21964,47 грн; (2.8)

Ціна реалізації розробленого програмного забезпечення становитиме:

$Ц_р=Ц_о+ПДВ$ = 21964,47 + 21964,47 *0.2 = 26357,36 грн; (2.9)

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Охорона праці спрямована на забезпечення безпечних і нешкідливих умов для робітників. На сучасному етапі технічного розвитку вона набуває дедалі більшого значення.

Забезпечення безпеки на робочому місці ґрунтується на досягненнях науки про організацію праці, ергономіки, гігієни та фізіології людини, а також психофізіологічних факторів. Крім того, рівень охорони праці залежить від швидкості впровадження новітніх технологій, рівня механізації та автоматизації виробничих процесів.

Тема мого проекту: Розробка програмного застосунку-помічника для вивчення іноземних мов

Забезпечення безпеки праці можливо лише за умови суворого дотримання вимог трудового законодавства, державних стандартів України та нормативних актів, що регламентують збереження здоров’я працівників.

3.1 Аналіз небезпечних та шкідливих чинників, що впливають на працівника

Під час роботи за комп’ютером на людину можуть впливати такі негативні фактори:

Випромінювання електромагнітного поля;

Коливання яскравості екрану;

Тривале перебування в статичному положенні.

Сукупний ефект цих факторів може зменшувати енергетичний потенціал організму, погіршувати імунітет, викликати м’язову втому та інші негативні наслідки.

3.2 Розробка заходів з охорони праці

Зменшити вплив перерахованих факторів ризику і зберегти здоров'я людині, яка постійно використовує в роботі ПК, дозволяє дотримання всіх заходів і засобів, передбачених охороною праці.

3.2.1 Мікроклімат робочої зони працівників, вентиляція

Освітлення робочого місця має бути рівномірним, а рівень шуму в межах допустимих значень, щоб запобігти перевтомі.

Рівні позитивних і негативних іонів в повітрі приміщень з ВДТ повинні задовольняти санітарно-гігієнічним нормам № 2152-80.

3.2.2 Освітлення робочого місця, шум, вібрація

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ЕОМ і ПЕОМ, має здійснюватись системою загального рівномірного освітлення. Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300 - 500 лк.

Як джерело світла при штучному освітленні застосовуються переважно люмінесцентні лампи.

Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300 - 500 лк.

Система загального освітлення має становити суцільні або переривчасті лінії світильників, розташовані збоку від робочих місць (переважно зліва), паралельно лінії зору працюючих Застосування світильників без розсіювачів та екрануючих грат заборонено.

Рівні звукового тиску в октавних смугах частот мають відповідати вимогам СН 3223-85, ГОСТ 12.1.003-83, ГР 2411-81.

3.2.3 Організація робочого місця користувача ПК

- Важливо, щоб офісний працівник сидячи за комп'ютером знаходився за добре освітленим робочим столом. Найчастіше саме погане освітлення робочого місця надає більш згубний для зору вплив, ніж сам факт перебування за комп'ютером.
- Робочі столи слід розміщувати таким чином, щоб монітори були орієнтовані бічною стороною до світлових прорізів, щоб природне світло падало переважно ліворуч.

- При розміщенні робочих місць відстань між робочими столами повинна бути не менше 2,0 м, а відстань між бічними поверхнями відеомоніторів - не менше 1,2 м.
- Конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання.
- Конструкція робочого стільця або крісла повинна забезпечувати підтримку раціональної робочої пози працівника.
- Клавіатуру слід розташовувати на поверхні столу на відстані 100..300 мм від краю, зверненого до користувача, або на спеціальній поверхні, відокремленій від основної стільниці.
- Екран відеомонітора повинен знаходитися від очей користувача на відстані 600-700 мм, але не ближче 500мм.

Рисунок 2.1. Правильне положення оператора ПК на робочому місці

Безпека праці при роботі за комп’ютером передбачає, що тривалість безперервної роботи за комп'ютером без регламентованої перерви не повинна перевищувати 2 години.

Не рекомендується працювати за комп'ютером більше 6 годин за зміну. Рекомендується робити перерви в роботі за ПК тривалістю 10 хвилин через кожні 50 хвилин роботи. Під час регламентованих перерв доцільно виконувати комплекси вправ.

При нерегламентованій роботі підвищеної інтенсивності можливі головні болі, нервові зриви та інше.

3.3 Пожежна безпека

Належна пожежна безпека – це інтегральна складова організації виробничих приміщень, що ґрунтується на дотриманні чинних законодавчих норм. Сфера пожежного захисту регулюється Правилами пожежної безпеки, затвердженими наказом Міністерства внутрішніх справ України, із періодичними доповненнями та уточненнями, що відображають сучасні вимоги.

Навіть при оснащенні будівель різноманітними системами пожежогасіння, сигналізації та внутрішніми пожежними кранами, офісні приміщення повинні бути додатково обладнані основними засобами гасіння пожеж. До них відносяться вогнегасники, кошми (спеціальні негорючі покривала), ящики з піском, бочки з водою, пожежні відра, багри, ломи, сокири та інші пристосування, причому вогнегасники вважаються найбільш зручними у використанні.

За своєчасне та повне забезпечення об’єктів заходами пожежного захисту відповідає роботодавець спільно з керівниками відповідних підрозділів. Вони зобов’язані розробити та впровадити комплекс заходів, що включає створення протипожежного режиму та інструкцій, відповідно до вимог чинних нормативних актів.

Режим пожежного захисту повинен містити детальний опис зон спеціального призначення та встановлювати правила їх експлуатації та обслуговування, зокрема:

- шляхи евакуації;
- спеціально відведені зони для куріння (так звані «курилки»);
- приміщення для зберігання продукції та сировини;
- території для стоянки автомобілів.

Ключовим елементом протипожежного режиму є розробка алгоритму дій на випадок пожежі. Обов’язковим є створення чіткого плану евакуації, в якому буде описано порядок відключення електроустановок та визначено послідовність дій співробітників у надзвичайній ситуації.

Рисунок 3.2. Засоби пожежогасіння

Основні складові алгоритму дій:

Оповіщення про пожежу – використання системи сигналізації та негайний виклик рятувальних служб.

Організація евакуації – дотримання встановлених маршрутів виходу та контроль безпечного переміщення людей.

Застосування засобів пожежогасіння – правильне використання вогнегасників та інших протипожежних систем.

Взаємодія з екстреними службами – оперативне надання інформації щодо загоряння та потенційних ризиків.

ВИСНОВКИ

У результаті реалізації дипломного проєкту було створено повноцінний мобільний застосунок для вивчення іноземних слів та фраз, що поєднує в собі адаптивну систему навчання, зручний словник, функції аудіювання та інтуїтивний інтерфейс. Проєкт не лише відповідає технічним вимогам, але й враховує когнітивні особливості користувача, що дозволяє ефективно засвоювати матеріал навіть під час коротких сесій.

Проведене тестування підтвердило стабільність, надійність і простоту використання LanguageHelper у реальних умовах. Завдяки обраному підходу до архітектури та збереження даних, програма здатна задовольнити потреби різних категорій користувачів, залишаючись водночас компактною, доступною і зрозумілою.

Окрім основного функціоналу, особливу увагу в проєкті було приділено питанням інклюзивності та персоналізації навчального процесу. Також можлива реалізація адаптивних рівнів складності, які автоматично підлаштовуються до поточних знань і темпу засвоєння користувача. Це дозволить забезпечити ще більшу гнучкість і зробити LanguageHelper універсальним інструментом для широкого кола людей, незалежно від їхнього віку, мовного рівня чи досвіду використання цифрових технологій.

Окремо варто відзначити масштаби можливого розвитку застосунку: інтеграція додаткових мовних модулів, розширення аналітики результатів користувача та додавання елементів гейміфікації можуть ще більше підвищити мотивацію та інтерес до щоденного навчання.

Таким чином, LanguageHelper демонструє, як інноваційний підхід до розробки мобільних освітніх продуктів може зробити процес вивчення мов більш гнучким та комфортним. Розроблена система здатна стати корисним помічником як для студентів, так і для самостійних учнів, сприяючи формуванню звички регулярного повторення та якісному засвоєнню матеріалу.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Козловський І. Вивчення іноземних мов за допомогою мобільних застосунків: ефективність і недоліки // Освіта і суспільство. — 2021. — №1. — С. 48–55.
2. Савчук О.І. Психолінгвістичні основи процесу засвоєння іноземної лексики. — К. : Лінгвістика, 2019. — 172 с.
3. Unity Documentation [Електронний ресурс]. — https://docs.unity3d.com (дата звернення: 01.05.2025).
4. MAUI (.NET Multi-platform App UI) Documentation [Електронний ресурс]. — https://learn.microsoft.com/en-us/dotnet/maui/ (дата звернення: 03.05.2025).
5. SQLite Documentation [Електронний ресурс]. — https://www.sqlite.org/docs.html (дата звернення: 05.05.2025).
6. Грінченко Л. Інтерфейс користувача в навчальних застосунках: принципи ергономіки // Інформаційні технології в освіті. — 2020. — №3. — С. 22–27.
7. Rosetta Stone — Language Learning Methodology [Електронний ресурс]. — https://www.rosettastone.com/why (дата звернення: 07.05.2025).
8. Duolingo: Research and Efficacy Report [Електронний ресурс]. — https://research.duolingo.com (дата звернення: 09.05.2025).
9. Busuu. Офіційний блог: методика навчання [Електронний ресурс]. — https://blog.busuu.com (дата звернення: 10.05.2025).
10. Code with C#. Building Language Learning Apps [Електронний ресурс]. — https://code-maze.com/csharp-language-learning-app/ (дата звернення: 11.05.2025).
11. Visual Studio Code — Code Editing Redefined [Електронний ресурс]. — https://code.visualstudio.com (дата звернення: 13.05.2025).
12. Литовченко М. Розробка мовних застосунків у середовищі .NET MAUI // Тези доповідей студентської конференції. — 2023. — С. 35–38.
13. Krashen, Stephen D. *Principles and Practice in Second Language Acquisition*. Oxford: Pergamon, 1982.

ДОДАТОК А. Лістинг програми

```
using LanguageHelper.Data;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.EntityFrameworkCore;
using LanguageHelper.Data;
using System.IO;

namespace LanguageHelper.App;

static class Program
{
   [STAThread]
   static void Main()
   {
      // Формируем абсолютный путь к базе в той же папке, где лежит EXE.
      var dbPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "language_helper.db");

      var services = new ServiceCollection();

      services.AddDbContext<AppDbContext>(options =>
         options.UseSqlite($"Data Source={dbPath}"));

      services.AddScoped<AppDbContext>();
      services.AddScoped<LanguageHelper.Data.Services.SpacedRepetitionService>();
      services.AddScoped<LanguageHelper.Data.Services.ListeningService>();

      using var provider = services.BuildServiceProvider();

      // Применяем все миграции – создаёт таблицы Words и Cards, если их нет.
      using (var scope = provider.CreateScope())
      {
         var db = scope.ServiceProvider.GetRequiredService<AppDbContext>();
         if (db.Database.GetPendingMigrations().Any())
         {
            db.Database.Migrate();
         }
         else
         {
            db.Database.EnsureCreated();
         }
         SeedData.Initialize(db);
      }

      ApplicationConfiguration.Initialize();
      Application.Run(new MainForm(provider));
            }
         using System;
         using System.Drawing;
         using System.Windows.Forms;
         using Microsoft.Extensions.DependencyInjection;
         using LanguageHelper.Data;
         using LanguageHelper.App.Controls;

         namespace LanguageHelper.App
         {
            public partial class MainForm : Form
            {
               private readonly IServiceProvider _services;

               public MainForm(IServiceProvider services)
               {
```

```
        _services = services;
        InitializeComponent();
    }

    private void InitializeComponent()
    {
        // Форма
        Text = "Language Helper";
        Width = 800; Height = 600;
        StartPosition = FormStartPosition.CenterScreen;
        Padding = new Padding(20);
        Font = new Font("Segoe UI", 12F, FontStyle.Regular);

        // Одноколоночная сетка 3×1
        var layout = new TableLayoutPanel
        {
            Dock = DockStyle.Fill,
            RowCount = 3,
            ColumnCount = 1
        };
        layout.RowStyles.Add(new RowStyle(SizeType.Percent, 33f));
        layout.RowStyles.Add(new RowStyle(SizeType.Percent, 33f));
        layout.RowStyles.Add(new RowStyle(SizeType.Percent, 34f));

        // 1) Кнопка «Картки»
        var btnCards = new RoundedButton
        {
            Text = "КАРТКИ",
            Dock = DockStyle.Fill,
            Font = new Font(Font.FontFamily, 14F, FontStyle.Bold)
        };
        btnCards.Click += (s, e) => new CardsForm(_services).ShowDialog();
        layout.Controls.Add(btnCards, 0, 0);

        // 2) Кнопка «Аудіювання»
        var btnListen = new RoundedButton
        {
            Text = "АУДІЮВАННЯ",
            Dock = DockStyle.Fill,
            Font = new Font(Font.FontFamily, 14F, FontStyle.Bold)
        };
        btnListen.Click += (s, e) => new ListeningForm(_services).ShowDialog();
        layout.Controls.Add(btnListen, 0, 1);

        // 3) Кнопка «Словник»
        var btnDict = new RoundedButton
        {
            Text = "СЛОВНИК",
            Dock = DockStyle.Fill,
            Font = new Font(Font.FontFamily, 14F, FontStyle.Bold)
        };
        btnDict.Click += (s, e) => new DictionaryChoiceForm(_services).ShowDialog();
        layout.Controls.Add(btnDict, 0, 2);

        Controls.Add(layout);
    }
}

/// <summary>
/// Окно выбора «Слова» или «Аудіо-фрази»
/// </summary>
public class DictionaryChoiceForm : Form
{
```

```
        private readonly IServiceProvider _services;

        public DictionaryChoiceForm(IServiceProvider services)
        {
            _services = services;
            InitializeComponent();
        }

        private void InitializeComponent()
        {
            Text = "Словник";
            Width = 300; Height = 200;
            StartPosition = FormStartPosition.CenterParent;
            Padding = new Padding(20);
            Font = new Font("Segoe UI", 12F, FontStyle.Regular);

            var flow = new FlowLayoutPanel
            {
                Dock = DockStyle.Fill,
                FlowDirection = FlowDirection.TopDown,
                WrapContents = false,
                AutoSize = true
            };

            var btnWords = new RoundedButton
            {
                Text = "Слова",
                Width = 240,
                Height = 40,
                Margin = new Padding(0, 0, 0, 10)
            };
            btnWords.Click += (_, __) =>
            {
                using var scope = _services.CreateScope();
                var db = scope.ServiceProvider.GetRequiredService<AppDbContext>();
                new WordListForm(db).ShowDialog();
                Close();
            };

            var btnPhrases = new RoundedButton
            {
                Text = "Аудіо-фрази",
                Width = 240,
                Height = 40
            };
            btnPhrases.Click += (_, __) =>
            {
                new ListeningListForm(_services).ShowDialog();
                Close();
            };

            flow.Controls.Add(btnWords);
            flow.Controls.Add(btnPhrases);
            Controls.Add(flow);
        }
    }
}
using System.Windows.Forms;
using LanguageHelper.Core.Models;
using LanguageHelper.Data.Services;
using Microsoft.Extensions.DependencyInjection;

namespace LanguageHelper.App;
```

```
public class CardsForm : Form
{
    private readonly SpacedRepetitionService _srs;
    private Card? _current;
    private bool _reverse = false; // false: EN->RU, true: RU->EN

    private Label lblQuestion = null!;
    private TextBox txtAnswer = null!;
    private Button btnCheck = null!;
    private Label lblCorrect = null!;
    private CheckBox chkReverse = null!;

    public CardsForm(IServiceProvider services)
    {
        _srs = services.GetRequiredService<SpacedRepetitionService>();
        InitializeComponent();
        LoadNext();
    }

    private void InitializeComponent()
    {
        this.Text = "Картки";
        this.Width = 400;
        this.Height = 250;
        this.StartPosition = FormStartPosition.CenterParent;

        lblQuestion = new Label { Dock = DockStyle.Top, Height = 50, TextAlign =
System.Drawing.ContentAlignment.MiddleCenter, Font = new System.Drawing.Font("Segoe UI", 16) };
        txtAnswer = new TextBox { Dock = DockStyle.Top, Height = 30 };
        btnCheck = new Button { Text = "Перевірити", Dock = DockStyle.Top, Height = 35 };
        lblCorrect = new Label { Dock = DockStyle.Top, Height = 40, TextAlign =
System.Drawing.ContentAlignment.MiddleCenter, Visible = false };
        chkReverse = new CheckBox { Text = "УКР → АНГ", Dock = DockStyle.Top, Height = 25 };

        btnCheck.Click += Check_Click;
        chkReverse.CheckedChanged += (s, e) => { _reverse = chkReverse.Checked; LoadNext(); };

        Controls.AddRange(new Control[] { lblCorrect, btnCheck, txtAnswer, lblQuestion, chkReverse });
    }

    private void LoadNext()
    {
        _current = _srs.GetNextDueCard(_reverse);
        if (_current == null)
        {
            MessageBox.Show("Карток немає!", "Інформація");
            this.Close();
            return;
        }

        lblQuestion.Text = _reverse ? _current!.Word.Translation : _current!.Word.Text;
        txtAnswer.Text = "";
        lblCorrect.Visible = false;
        txtAnswer.Focus();
    }

    private void Check_Click(object? sender, EventArgs e)
    {
        if (_current == null) return;

        var user = txtAnswer.Text.Trim().ToLower();
        var correct = (_reverse ? _current.Word.Text : _current.Word.Translation).ToLower();
```

```
        bool isCorrect = user == correct;
        lblCorrect.Text = correct;
        lblCorrect.Visible = true;
        lblCorrect.ForeColor = isCorrect ? System.Drawing.Color.Green : System.Drawing.Color.Red;

        _srs.RecordAnswer(_current, isCorrect);

        // Задержка 1 секунда для отображения, затем следующая карточк
        var timer = new System.Windows.Forms.Timer();
        timer.Interval = 1000;
        timer.Tick += (s, e) =>
        {
            timer.Stop();
            LoadNext();
        };
        timer.Start();

    }
}
```

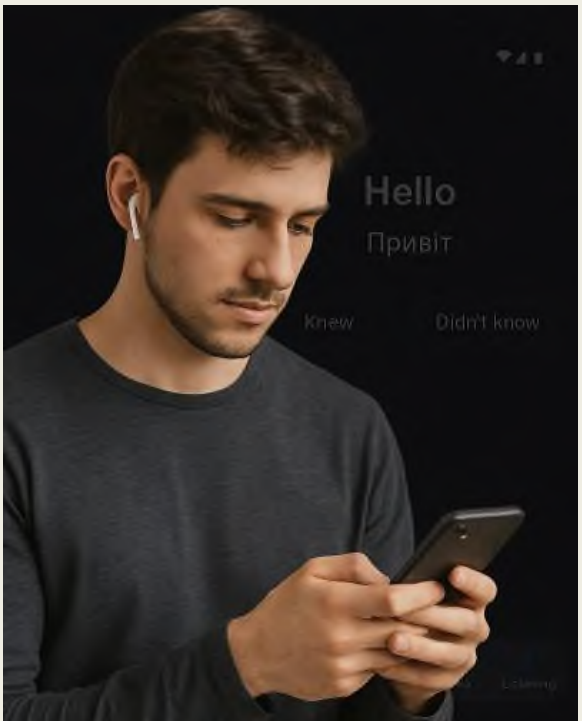
ДОДАТОК Б. Слайди мультимедійної презентації

РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ-ПОМІЧНИКА ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ

Січкар Олександр Олегович
Спеціальність: 121
Група: 4РП-08
Керівник: Шувалова І.О.
ВСП «ОТФК ОНТУ», 2025

Актуальність теми

- Сучасне суспільство потребує мобільних інструментів навчання.
- Існуючі рішення не враховують індивідуальні особливості.
- Немає автономного доступу без інтернету.
- Потрібен адаптивний застосунок-помічник.
- [Ілюстрація: людина навчається з телефона]



Мета та завдання проєкту

- Мета: створити ефективний мобільний застосунок.
- Завдання: реалізація словника, карток, аудіювання.
- Забезпечити простоту, офлайн-доступ і адаптивність.
- [Ілюстрація: структура функціоналу застосунку]

Огляд аналогів

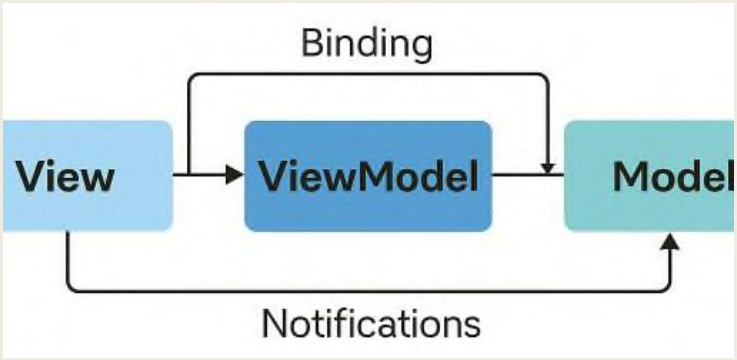
Назва	Плюси	Мінуси
Duolingo	• Яскравий дизайн • Гейміфікація	Обмежений словниковий запас
Memrise	• Велика кількість мов • Оригінальні відео	Обмежені типи вправ
Babbel	• Ретельно підібраний матеріал	Немає безкоштовної версії

Психологічні основи

- Інтервальне повторення покращує запам'ятовування.
- Мультимодальність активує кілька типів пам'яті.
- Контекст сприяє кращому засвоєнню інформації.

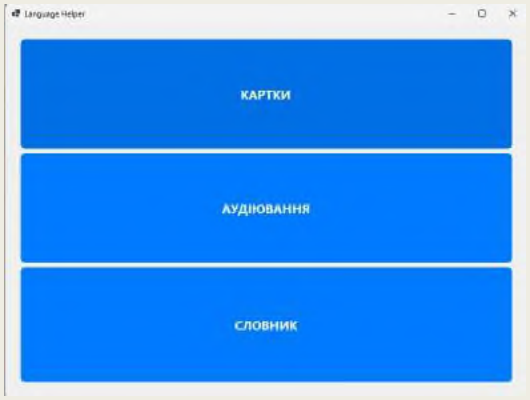
Технології реалізації

- .NET MAUI — кросплатформовий фреймворк.
- SQLite-net — легка локальна база даних.
- MVVM — структурована архітектура.
- Visual Studio 2022 — середовище розробки



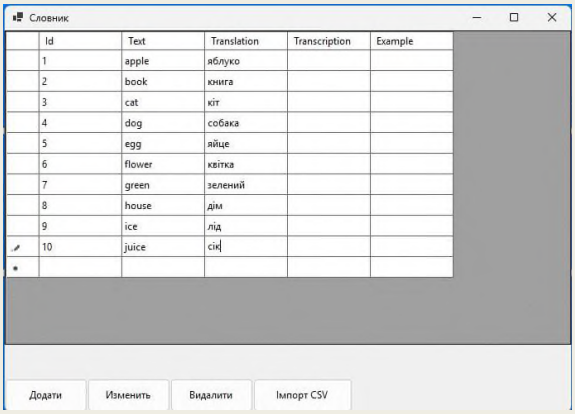
Структура застосунку

- CardsPage — модуль карток.
- ListeningPage — модуль аудіювання.
- DictionaryPage — словник зі збереженням.
- AppShell — управління навігацією.



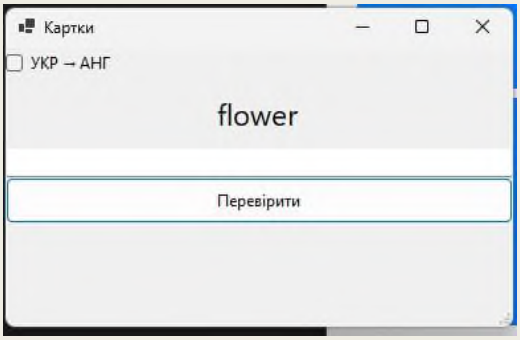
Функціонал словника

- Додавання, редагування та видалення слів.
- Пошук у реальному часі.
- Локальне зберігання в SQLite.
- Зручний інтерфейс користувача.



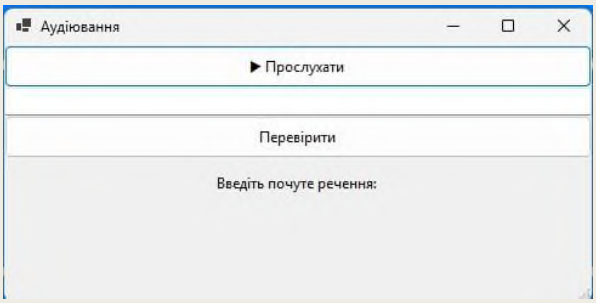
Тренування картками

- Вивід слова або перекладу.
- Оцінка відповіді користувачем.
- Пріоритет проблемних слів.
- Зміна напрямку перекладу.



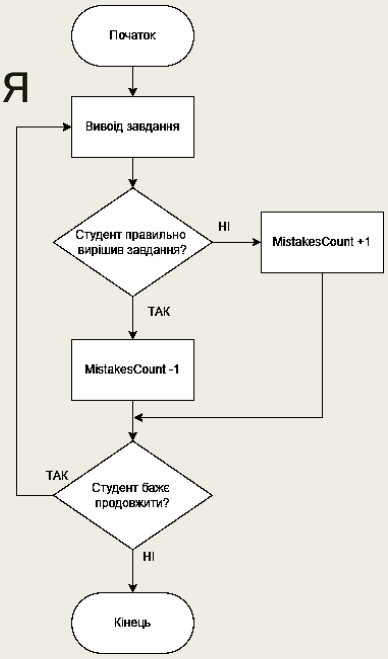
Аудіювання

- Відтворення аудіо через MediaElement.
- Вибір правильної відповіді.
- Аудіо зберігається локально.
- Облік помилок і повторення.



Адаптивна система навчання

- У застосунку реалізовано механізм адаптивного повторення слів, що ґрунтується на обліку помилок користувача. Кожен запис у базі має спеціальне поле MistakesCount, яке автоматично збільшується після кожної неправильної відповіді. Це дозволяє системі виявляти складні для користувача слова та формувати персоналізовані сесії повторення. Пріоритет надається словам, які мають більшу кількість помилок і давно не тренувались. Формування вибірки для тренування реалізується за допомогою SQL-запитів до локальної бази даних.



Висновки

Реалізовано повноцінний мовний помічник LanguageHelper, що поєднує картки, аудіювання та словник у єдину адаптивну систему навчання.
Забезпечено автономність, простоту та адаптивність: застосунок не потребує підключення до інтернету, має інтуїтивно зрозумілий інтерфейс і підлаштовується під темп користувача.
Проєкт демонструє високий рівень готовності до практичного використання та надає можливості для подальшого розширення, зокрема: інтеграцію нових мов, розширення функціоналу, або додавання системи статистики.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Січкаря Олександра Олеговича

(прізвище, ім`я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Шувалова Ірина Олегівна*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмного застосунку-помічника для вивчення іноземних мов*

Обсяг розрахунково-пояснювальної записки *85* сторінок

Обсяг графічної (презентаційної) частини *13* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений Розробка програмного застосунку-помічника для вивчення іноземних мов *електроспоживання та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації*

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 13 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки добра, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Забезпечено взаємодію таблиць та бази даних. Використання .NET MAUI, SQLite та архітектури MVVM дозволяє створити кросплатформений, автономний та масштабований застосунок. Цей підхід не тільки спрощує розробку, але й забезпечує високий рівень підтримки та тестування.*

д) основні недоліки дипломного проекту *Опис адаптивного механізму повторення та SQL-запитів мав би бути доповнений зрозумілими діаграмами або блок-схемами. Треба було детально порівняти альтернативні рішення (наприклад, відмінності між SQLite та іншими СУБД) з точки зору продуктивності та простоти розгортання, що допомогло б ще більше підкріпити прийняте рішення. Присутні деякі помилки оформлення пояснювальної записки*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис: _____

« 23 » 06 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Січкар Олександр Олегович

(прізвище, ім`я та по батькові)

Спеціальність: *121 «Інженерія програмного забезпечення»*

Освітня програма: *«Розробка програмного забезпечення»*

Тема дипломного проекту: *Розробка програмного застосунку-помічника для вивчення іноземних мов*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 68 сторінок. У пояснювальній записці описано етапи Розробка програмного застосунку-помічника для вивчення іноземних мов. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Січкар Олександр поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Січкар Олександр під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Січкар Олександр показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як Microsoft Visual Studio, Microsoft PowerPoint, Microsoft Visio та ін.*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові керівника дипломного проекту *Шувалова Ірина Олегівна*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерної техніки та програмної інженерії*

Підпис

«16» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Січкар Олександр Олегович
здобувач освіти гр. 4РП-08, та

Шувалова Ірина Олегівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка програмного застосунку-помічника для вивчення іноземних мов»
(автор роботи – Січкар О.О., керівник роботи – Шувалова І.О.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Січкар О.О. /

Керівник _____ / Шувалова І.О. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4РП-08

Січкаря Олександра Олеговича

на тему *Розробка програмного застосунку-помічника для вивчення іноземних мов*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з некритичними порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *20.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 20.06.2025 р. значення коефіцієнту подібності в роботі становить 12,77%, коефіцієнт цитування – 0,86%.*

(підпис) *20.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Січкаря О.О* (П.І.Б.)

проведена «*20*» *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/20/2025
Дата редагування 6/20/2025

Документ прийнятий

Звіт подібності

метадані

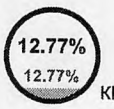
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка програмного застосунку-помічника для вивчення іноземних мов

Автор
Січкар Олександр ОлеговичШувалова Ірина Олегівна
Науковий керівник / Експерт

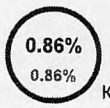
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25
Довжина фрази для коефіцієнта подібності 2

16667
Кількість слів

138719
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		21
Інтервали	A→	0
Мікропробіли		54
Білі знаки		0
Парафрази (SmartMarks)	a	110

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	138 0.83 %
2	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	72 0.43 %
3	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	70 0.42 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	57 0.34 %
5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	43 0.26 %

6	https://revolution.allbest.ru/management/00769018_0.html	39 0.23 %
7	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	37 0.22 %
8	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	37 0.22 %
9	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	36 0.22 %
10	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	35 0.21 %

з домашньої бази даних (0.20 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка мобільного застосунку-помічника майстра манікюру [redacted] 6/18/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	33 (4) 0.20 %

з програми обміну базами даних (0.21 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2017_6050903_Vitrovyi_Iurii_Iuriiovych_3405 [redacted] 10/26/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	22 (1) 0.13 %
2	Торган 44ппп [redacted] 6/19/2024 Sambir Applied College of Economics and Information Technologies (Sambir Applied College of Economics and Information Technologies)	13 (1) 0.08 %

з Інтернету (12.36 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**314 (13) 1.88 %**
2	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	306 (30) 1.84 %
3	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content**	**296 (6) 1.78 %**
4	https://card-file.ontu.edu.ua/bitstreams/12d5c0ab-e979-48f2-a8ec-d5fc31f71fd5/download	288 (28) 1.73 %
5	**https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download**	**117 (6) 0.70 %**
6	https://conferences.vntu.edu.ua/public/files/itpf/conf_itpf-2016_all.pdf	94 (7) 0.56 %
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	84 (8) 0.50 %
8	**https://revolution.allbest.ru/management/00769018_0.html**	**78 (3) 0.47 %**
9	**https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download**	**76 (4) 0.46 %**
10	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	59 (4) 0.35 %
11	**https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download**	**58 (3) 0.35 %**
12	**https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download**	**56 (3) 0.34 %**

13	https://allrefrs.ru/2-38727.html	40 (4) 0.24 %
14	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	38 (2) 0.23 %
15	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	38 (3) 0.23 %
16	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a05c07c5-bf65-4cb0-bdfa-e28694707551/content	16 (2) 0.10 %
17	https://studopedia.org/4-92436.html	16 (1) 0.10 %
18	https://card-file.ontu.edu.ua/server/api/core/bitstreams/fc8a1853-39fc-4671-8807-2fd27ddb0779/content	15 (1) 0.09 %
19	https://linguisticsgirl.com/natural-order-hypothesis-definition-criticism/	13 (1) 0.08 %
20	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	12 (1) 0.07 %
21	https://ir.nmu.org.ua/jspui/bitstream/123456789/164850/1/23_06_16_%D0%9A%D1%83%D1%82%D0%BD%D0%B8%D1%87%20%D0%A2.%D0%A2.pdf	12 (1) 0.07 %
22	https://card-file.ontu.edu.ua/bitstreams/d832942b-e631-471e-b716-8e52c042ba32/download	11 (1) 0.07 %
23	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	7 (1) 0.04 %
24	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	6 (1) 0.04 %
25	https://card-file.ontu.edu.ua/bitstreams/63ee88cb-a3d0-4005-9cf2-0cff89f28c0d/download	5 (1) 0.03 %
26	https://card-file.ontu.edu.ua/bitstreams/55e2b8f2-7d3c-4235-99fc-2be51199b96d/download	5 (1) 0.03 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма: **«Розробка програмного забезпечення»**
Група: 4РП- 08

Дипломний проєкт
здобувача освіти денної форми навчання
РП.08.18.000.ДП

СІЧКАРЯ
ОЛЕКСАНДРА ОЛЕГОВИЧА

м. Одеса
2025 **р.**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма: **«Розробка програмного забезпечення»**
Група: 4 **РП- 08**

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проєкту **на тему:**

_____

_____

Проектний матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах)