

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Система перекладу з елементами навчання»

Виконав студент групи ПЗ-21-3
ГНАТЮК Дмитро Сергійович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГНАТЮК Дмитро Сергійович**

Керівник роботи: **ЛОКТИКОВА Тамара Миколаївна**

Тема роботи: **«Система перекладу з елементами навчання»,**

затверджена Наказом закладу вищої освіти від «14» лютого 2025 р., №61/с

Вихідні дані для роботи: **об'єктом дослідження є процес створення системи перекладу з інтеграцією навчальної складової, а предметом дослідження є методи, засоби та алгоритми для реалізації системи перекладу з елементами навчання.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Керівник	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	27.02.2025р.	27.02.2025р.
2	Локтікова Т.М.	27.02.2025р.	27.02.2025р.
3	Локтікова Т.М.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з віконного застосунку і розширення для браузера системи перекладу з елементами навчання та пояснювальної записки. Пояснювальна записка до випускної роботи містить 72 сторінки, 43 ілюстрації та 12 таблиць.

Розподілена архітектура системи дозволяє інтегрувати інструменти перекладу у процес роботи з іншомовними матеріалами в Інтернеті та надає можливості для персоналізованого навчання, спрямованого на розширення словникового запасу.

Систему перекладу з елементами навчання створено з використанням сучасних технологій. Для реалізації проекту використовувався стек C#, а саме технології ASP.NET WEB API – для створення серверної частини та WPF – для реалізації віконного застосунку. При створенні розширення для браузера було використано мову TypeScript, з метою покращення типізації, бібліотеку React, для зручності створення інтерфейсу користувача, а також сучасний та швидкий компілятор Vite, який разом з додатковими бібліотеками значно спростив розробку.

Було також проведено тестування системи з метою забезпечення стабільної роботи та виявлення можливих недоліків у функціональності інтерфейсу та логіки взаємодії з користувачем.

Зручний інтерфейс системи перекладу забезпечує добре сприйняття інформації та дозволяє користувачам налаштовувати його під себе.

КЛЮЧОВІ СЛОВА: СИСТЕМА ПЕРЕКЛАДУ, РОЗПОДІЛЕНА АРХІТЕКТУРА, КЛІЄНТ-СЕРВЕР, ASP.NET WEB API, TYPESCRIPT, WPF.

					ІПЗ.КР.Б – 121 – 25 – ІЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Гнатюк Д. С.			Система перекладу з елементами навчання. Пояснювальна записка			Літ.	Арк.	Аркушів	
Перевір.		Локтікова Т.М.								3	72
Керівник		Локтікова Т.М.						ФІКТ Гр. ІПЗ-21-3			
Н. контр.		Тичина О.П.									
Зав. каф.		Вакалюк Т.А.									

ABSTRACT

The bachelor's qualification thesis consists of a desktop application and a browser extension of a translation system with learning elements, along with an explanatory note. The explanatory note for the thesis contains 72 pages, 43 illustrations, and 12 tables.

The distributed architecture of the system enables the integration of translation tools into the workflow involving foreign-language materials on the Internet and provides opportunities for personalized learning aimed at vocabulary expansion.

The translation system with learning elements was developed using modern technologies. The implementation of the project involved the C# technology stack, specifically ASP.NET Web API for the server-side component and WPF for the desktop application. The browser extension was developed using TypeScript to improve type safety, React for convenient UI creation, and the modern and fast Vite compiler, which, together with additional libraries, significantly simplified the development process.

System testing was also conducted to ensure stable performance and to identify possible shortcomings in the interface functionality and user interaction logic.

The user-friendly interface of the translation system ensures good information perception and allows users to customize it according to their preferences.

KEYWORDS: TRANSLATION SYSTEM, DISTRIBUTED ARCHITECTURE, CLIENT-SERVER, ASP.NET WEB API, TYPESCRIPT, WPF.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ ПЕРЕКЛАДУ	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури системи	16
1.4 Обґрунтування вибору інструментальних засобів	20
Висновки до першого розділу.....	22
РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ ПЕРЕКЛАДУ	23
2.1 Визначення варіантів використання	23
2.2 Розробка бази даних системи.....	25
2.3 Проектування та реалізація алгоритмів роботи системи	31
2.4 Релізація системи перекладу з елементами навчання	37
Висновки до другого розділу	44
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ	45
3.1 Порядок встановлення та налаштування параметрів системи	45
3.2 Користувацький інтерфейс розширення для браузера.....	47
3.3 Користувацький інтерфейс застосунку.....	51
3.4 Тестування системи	62
Висновки до третього розділу	67
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	69
ДОДАТКИ.....	72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс (Application Programming Interface)

REST– підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів (Representational State Transfer)

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

CRUD – основні операції, що використовуються для управління даними (Create, Read, Update, Delete)

ШІ – штучний інтелект

LLM – мовна модель (штучний інтелект) (Large Language Model)

ORM – технологія для роботи з базами даних, використовуючи об’єктно-орієнтований підхід (Object-Relational Mapping)

UI – користувацький інтерфейс (User Interface)

DI – шаблон впровадження залежностей (Dependency Injection)

DTO – легковагова модель даних для передачі (Data Transfer Object)

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми полягає у тому, що у сучасному світі володіння іноземними мовами є одним з ключових чинників професійного зростання та особистісного розвитку. Це формує потребу у інструментах, які б спрощували роботу з іншомовними матеріалами, та водночас дозволяли б вивчати мову. Існуючі рішення часто обмежуються лише базовими функціями, не сприяючи запам'ятовуванню нових слів або формуванню стійких знань. Окрім того, вони не завжди враховують потреби користувачів у зручності роботи з ними – так, недостатня інтеграція у робочі процеси відволікає увагу та уповільнює взаємодію, а недосконалий дизайн викликає втому та ускладнює сприйняття інформації.

Впровадження системи, яка об'єднує переклад і навчання безпосередньо в робочому середовищі користувача, є актуальним рішенням, яке не лише оптимізує вище зазначені процеси, але й сприятиме поступовому засвоєнню нової лексики. Це робить її особливо корисною для студентів, фахівців та всіх, хто працює з іншомовними матеріалами.

Метою випускної кваліфікаційної роботи бакалавра є розробка системи перекладу з елементами навчання.

Для досягнення поставленої мети необхідним є виконання таких **завдань**:

1. Постановка та аналіз задачі.
2. Аналіз аналогів розроблюваної системи.
3. Вибір архітектури системи.
4. Визначення технологій та інструментальних засобів реалізації проекту.
5. Визначення функціональних вимог та варіантів використання системи.
6. Опис порядку встановлення та налаштування системи та її параметрів.
7. Опис користувацького інтерфейсу.
8. Тестування системи.

Об'єктом дослідження є процес створення системи перекладу з інтеграцією навчальної складової.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Предметом дослідження є методи, засоби та алгоритми для реалізації системи перекладу з елементами навчання.

Дослідження також охоплює аналіз міждисциплінарних підходів, що можуть бути використані задля підвищення ефективності навчання.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано тези:

1) Гнатюк Д.С., Локтікова Т.М., Лисогор Ю.І. Міждисциплінарний підхід до розробки освітницького програмного забезпечення. Тези доповідей VII Всеукраїнської науково-технічної конференції «Комп’ютерні технології: інновації, проблеми, рішення», 02-03 грудня 2024 року. Житомир: «Житомирська політехніка», 2024. С. 41-42 [1];

2) Гнатюк Д.С., Локтікова Т.М. Система перекладу з елементами навчання. Тези доповідей Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених, присвяченої Дню науки, 12-17 травня 2025 року. Житомир: «Житомирська політехніка», 2025. С. 58 [2].

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ ПЕРЕКЛАДУ

1.1 Постановка задачі

У сучасному глобалізованому світі дедалі більше користувачів щодня зіштовхуються з потребою працювати з іншомовним контентом в Інтернеті. Володіння мовами стає важливою складовою як особистого розвитку, так і професійного успіху. Проте більшість доступних засобів перекладу обмежуються лише базовими функціями та не сприяють активному осмисленню й запам'ятовуванню нового матеріалу. Крім того, необхідність використання окремих інструментів для перекладу, збереження та повторного опрацювання інформації ускладнює навчальний процес і знижує мотивацію до регулярного вивчення мови. Створення системи, яка об'єднує переклад і навчання, є відповіддю на цей виклик.

Уявімо ситуацію, коли користувач читає іншомовний текст у браузері. Щойно з'являється незнайоме слово, він змушений шукати онлайн-перекладач і перемикатися між вкладками або вікнами, лише для того, щоб дізнатися значення цього слова. Така переривчастість значно відволікає від сприйняття основного змісту матеріалу. Якщо ж йдеться про художній твір, постійні зупинки та перемикання роблять читання ще більш виснажливим і дратівливим. Після повернення до потрібної сторінки користувачу знову доводиться шукати місце, на якому він зупинився, і нерідко, ще раз звертатися до перекладу чи прикладів вживання слова для глибшого розуміння контексту. Крім того, якщо користувач має намір занотувати нову лексику для подальшого вивчення, йому необхідно використовувати окреме програмне забезпечення або ж вручну записувати слова, що ще більше ускладнює і сповільнює процес навчання. Замість того, щоб постійно перемикатися між кількома програмами, застосунками чи вкладками для перекладу, збереження та повторного опрацювання нової лексики, набагато зручніше було б скористатися єдиною інтегрованою системою. Така система не лише миттєво перекладає слово або фразу, а й автоматично формує відповідні

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

навчальні матеріали, зберігає їх у персональній базі знань та пропонує інтерактивні вправи для якісного закріплення нового матеріалу. При цьому для здійснення перекладу користувачу навіть не доведеться залишати сторінку з текстом, який він читає, що значно підвищує комфорт і безперервність процесу. Саме таку ідею було покладено в основу концепції розроблюваного програмного продукту.

Основними ідеями майбутньої системи повинні стати простота та доречність використання, тобто забезпечувати максимальну інтеграцію у повсякденну роботу, в першу чергу з англomовними матеріалами, надаючи весь необхідний функціонал, при цьому не заважаючи даному процесу.

Одними із ключових аспектів постановки задачі є формування вимог до функціональності системи, серед яких виділяються основні: можливість перекладу текстів у зручному форматі; зберігання перекладених матеріалів та їх доступність на усіх синхронізованих пристроях, навіть за відсутності Інтернет-з'єднання; наявність інструментів для інтерактивного навчання; забезпечення максимально адаптивного інтерфейсу та функціоналу.

Особливу увагу слід приділити модульній структурі системи, яка дозволить додавати новий функціонал у майбутньому, та гнучкості дизайну, що сприятиме адаптації системи до змінюваних потреб користувачів.

Окрім цього, необхідно розробити алгоритми навчання, які будуть реалізовувати класичні підходи таких дисциплін як педагогіка і лінгвістика, та визначити підходи до надання функціоналу, спираючись на основи когнітивістики, психології, нейропсихології та інших наук, які допоможуть забезпечити оптимальне навантаження, ефективність запам'ятовування та засвоєння інформації.

Основними етапами розробки системи є побудова архітектури системи, її окремих компонентів та баз даних. Архітектура повинна забезпечити фундамент для масштабованої та легко підтримуваної системи. Розробка як загальних алгоритмів роботи системи так і алгоритмів таких функціональних процесів як створення навчальних матеріалів, обробка користувацьких даних тощо.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Створення дизайну розширення для браузеру та застосунку для ПК, що включає розробку як загальної концепції дизайну, так і реалізацію окремих UI компонентів. Інтеграційне тестування, в свою чергу, дозволить перевірити роботоспроможність системи у комплексі, що вкрай важливо при розробці розподіленої системи, що складається з багатьох компонентів.

1.2 Аналіз аналогів програмного продукту

Розробка системи потребує дослідження існуючих аналогів на ринку, щоб оцінити їх функціональні можливості, переваги та недоліки. Це дозволяє визначити ключові аспекти, які необхідно впровадити в розроблювану систему, а також уникнути недоліків конкурентів.

У якості першого аналогу розглянемо систему Reverso. Reverso – це багатофункціональна система, яка надає інструменти для перекладу текстів, перевірки граматики, вивчення слів та контекстів їх використання. Сервіс включає багато компонентів, серед яких: Context Reverso – дозволяє перекладати слова та фрази у контексті, Text Translation – швидкий переклад, Translate Documents – переклад тексту з файлів, Grammar Check – перевірка тексту на граматичні помилки, Dictionary – інтегрований словник із детальними визначеннями та прикладами, Vocabulary – функція для створення персоналізованого словникового запасу, Synonyms – дозволяє шукати синоніми, Conjugator – надає інформацію про відмінки слова за часом, родами та іншими параметрами [11, 22].

Reverso доступний у форматі вебплатформи, мобільного застосунку, десктопного застосунку та розширення для браузера. Завдяки широкому функціоналу, який включає не лише переклад, а й приклади вживання, граматичні підказки, функцію озвучування, є можливість збереження обраних перекладів, що значно полегшує процес навчання. Таким чином, дана система виступає одним із головних конкурентів розроблюваного програмного продукту, що вимагає ретельного аналізу її функцій, структури інтерфейсу та підходів до взаємодії з користувачем. На рис. 1.1 наведено інтерфейс розширення для

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

браузеру системи Reverso.

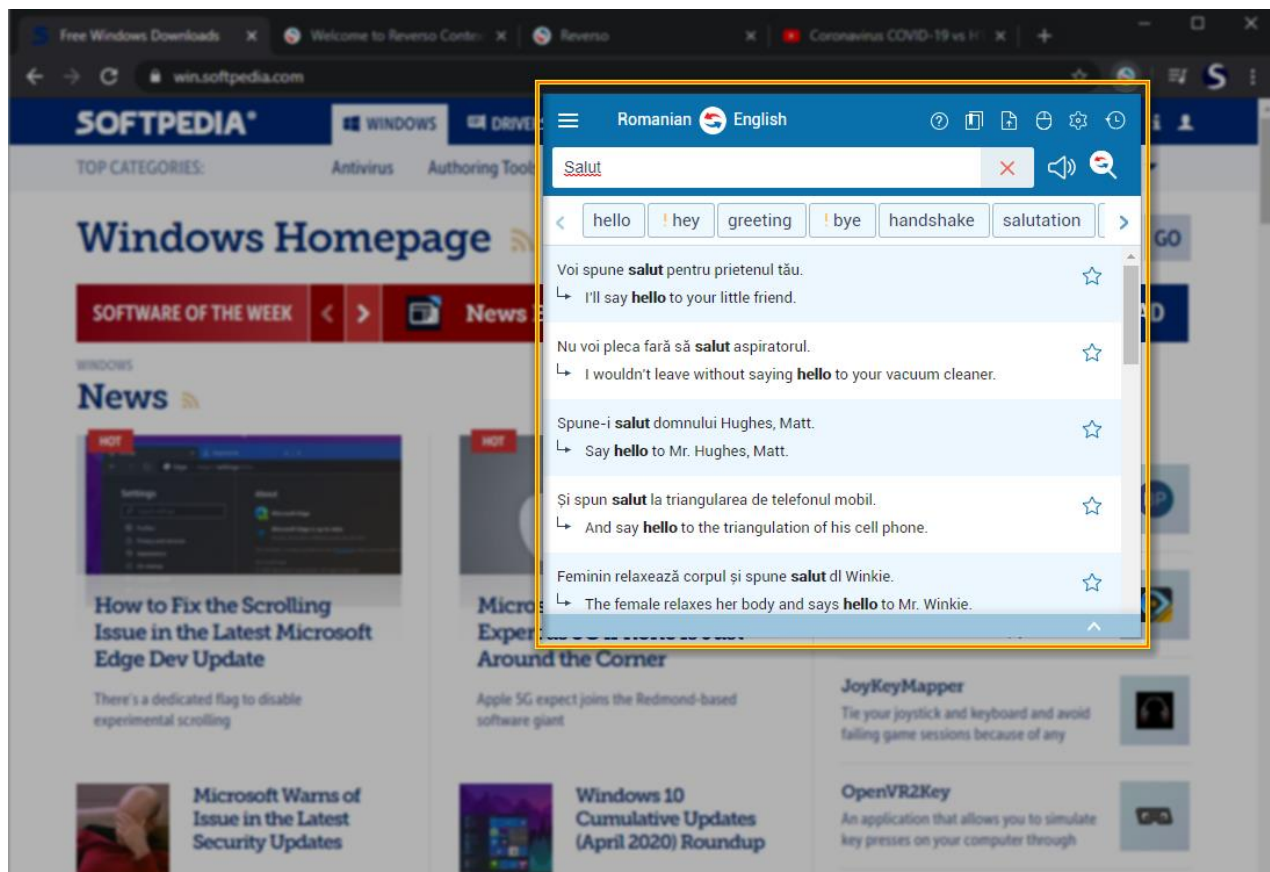


Рисунок 1.1 – Інтерфейс розширення для браузеру Context Reverso

До переваг даної системи належать широкий ряд інструментів, розширені можливості системи завдяки використанню штучного інтелекту, наявна інтеграція з третіми сервісами, велика база знань, наявна можливість тестування знань.

Серед недоліків системи можна виділити неможливість імпорту словників, недостатню персоналізацію та налаштування, неможливість коригувати зміст перекладу, недостатня інтеграція у робочий процес користувача а також те, що не увесь функціонал є інтуїтивно зрозумілим.

Наступним прикладом розглянемо веб-застосунок LingQ [17]. LingQ — це платформа для вивчення мов, яка зосереджена на навчанні через читання, слухання та розширення словникового запасу.

Основними можливостями платформи є імпорт користувацьких матеріалів (статті, книги, відео), використання вбудованих інструментів для перекладу, збереження та повторення слів а також наявність готових уроків.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток має сучасний організований дизайн (рис. 1.2), доступний через вебплатформу та мобільний застосунок.

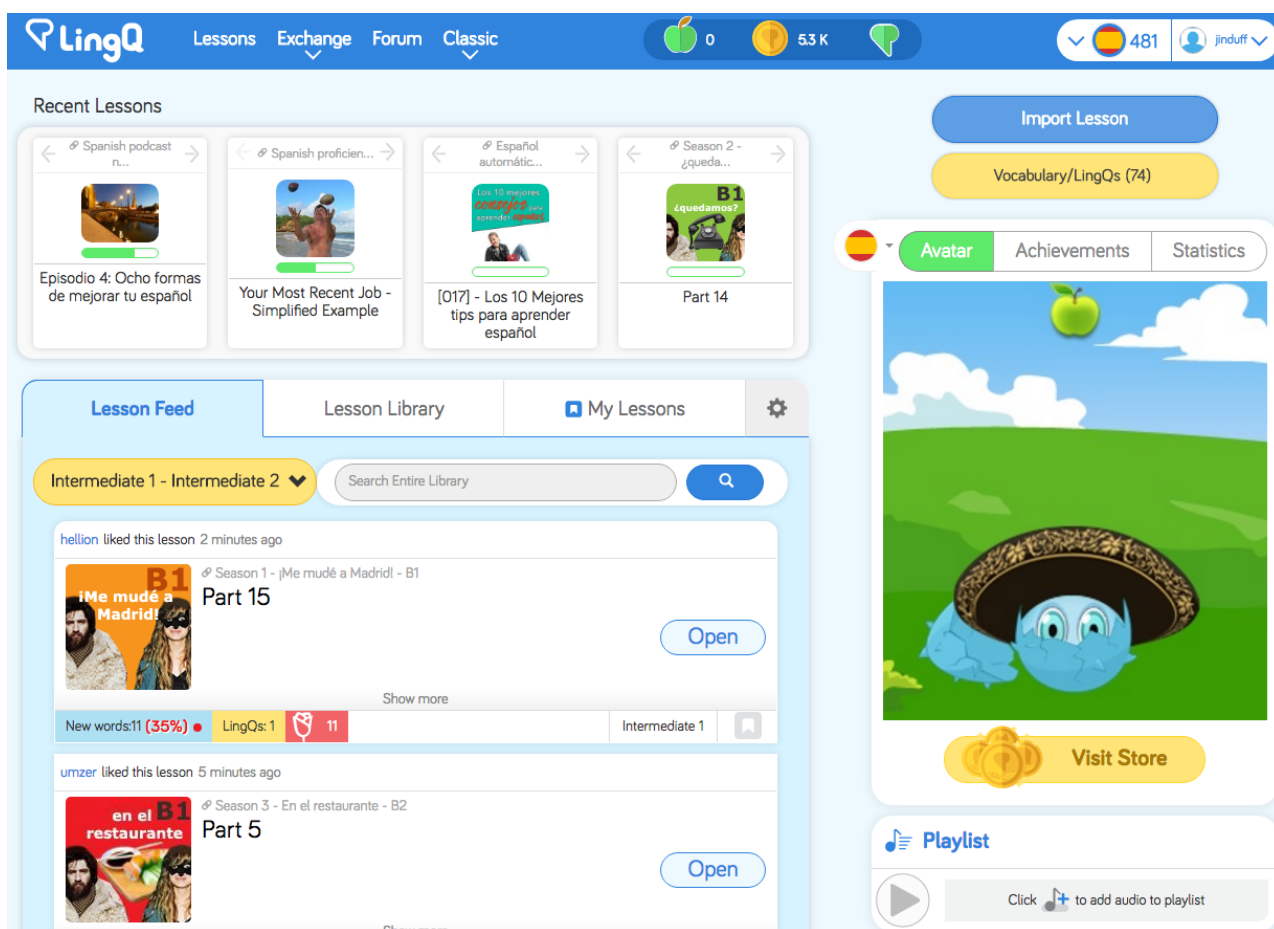


Рисунок 1.2 – Інтерфейс програми LingQ

До переваг даної системи можна віднести можливість імпорту власних матеріалів, широку бібліотеку навчальних матеріалів. Матеріали подаються у різних форматах а також присутня гейміфікація певних процесів.

Серед недоліків системи слід виокремити відсутність десктопного застосунку, перенавантажений інтерфейс, складний для новачків, відсутність офлайн функціональності та обмежена синхронізація, відсутність можливості тестування вивченого матеріала а також не реалізується головна перевага розширення для браузеру – хоч воно і розширює можливості системи, але не надає інструменти інтеграції у роботу.

В якості ще одного прикладу розглянемо Anki — систему для запам'ятовування інформації за допомогою карток із використанням

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

інтервального повторення [3]. Основними функціями є підтримка мультимедійних карток, наявність налаштованих колод та плагінів для додаткової функціональності, а також мінімалістичний інтерфейс з акцентом на перегляд карток (рис. 1.3). Програма доступна для настільних ПК і мобільних пристроїв.

Перевагами даного ПЗ є висока настроюваність (створення колод для різних цілей), ефективність навчання завдяки інтервальному повторенню, доступність на різних платформах, офлайн-функціональність із можливістю синхронізації; відкритий код і велика спільнота користувачів, велика кількість доступних розширень, висока оптимізація.

Недоліками є застарілий інтерфейс, відсутність інтеграції в робочі процеси, зокрема роботу з браузером, ручне введення даних, що потребує значних зусиль а також складність освоєння.



Рисунок 1.3 – Інтерфейс програми Anki

У результаті дослідження аналогів було сформовано цілісне бачення майбутнього програмного продукту, засноване на виявлених сильних і слабких

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

сторонах наявних рішень. Було вирішено реалізувати ряд покращень, зокрема можливість редагування вже створених записів, що значно підвищить гнучкість системи для користувачів. Також буде додано функціонал тестування знань, який дозволить користувачам самостійно перевіряти рівень засвоєння матеріалу та виявляти прогалини в знаннях, сприяючи систематичному навчанню.

Усі наведені аналоги дають змогу працювати або з будь-якою мовою, або з достатньо широким рядом найпопулярніших для вивчення мов. Таким чином було вирішено додати функціонал для роботи і з іншими мовами, наскільки це можливо, зважаючи на специфіку системи.

У табл. 1.1 наведено порівняльну характеристику розглянутих аналогів з кінцевим планом розроблюваної системи.

Таблиця 1.1

Порівняльна характеристика аналогів

Характеристика	Reverso	LingQ	Anki	Розроблювана система
Інтерфейс	Зручний	Перевантажений	Застарілий	Зручний
Робота офлайн	Частково присутня, є можливість завантажити словники	Відсутня	Весь наявний функціонал доступний офлайн	Частково присутня
Інтеграція у робочий процес	Обмежена	Відсутня	Відсутня	Висока
Простота використання	Простий у використанні	Середня складність	Ручне введення даних	Простий у використанні
Простота освоєння	Простий	Середня складність	Складний в освоєнні	Інтуїтивно зрозумілий
ПЗ	Веб-застосунок, десктопний та мобільний застосунок, розширення для браузеру	Веб-застосунок та розширення для браузеру	Десктопний застосунок та мобільний застосунок	Десктопний застосунок, розширення для браузеру

Характеристика	Reverso	LingQ	Anki	Розроблювана система
Синхронізація	Наявна	Обмежена	Наявна	Наявна
Власні дані	Власні дані імпортувати неможливо	Можливість імпорту власних матеріалів	Можливість імпорту власних матеріалів	Можливість копіювання матеріалів, редагування даних
Персоналізація	Відсутня	Середня	Висока	Розширена
Активності	Наявні	Відсутні	Є основним функціоналом	Наявні

У даній таблиці наведено саме ті характеристики, які найкраще визначають якість розроблюваної системи згідно з поставленою метою проекту, зокрема зручність використання, адаптивність інтерфейсу, можливості персоналізації, ефективність навчального процесу та технічна надійність. Обрані критерії дозволяють об'єктивно оцінити потенціал програмного забезпечення в контексті вирішення конкретних освітницьких задач. Як можна бачити, система загалом буде краще задовольняти зазначеним вимогам, ніж наявні аналоги, завдяки акценту на зручності користувача, мінімалістичному дизайну та підтримці індивідуального підходу до навчання. Водночас, реалізація додаткового функціоналу, зокрема механізмів тестування, а також взаємодії між користувачами, допоможе краще зарекомендувати розроблюване ПЗ серед аналогів на ринку, зробивши його не лише ефективним, але й привабливим для цільової аудиторії. Такий підхід сприятиме підвищенню конкурентоспроможності системи та її потенційному розширенню на нові сегменти користувачів.

1.3 Вибір архітектури системи

Для забезпечення ефективної взаємодії між різними складовими та підтримки масштабованості, обрана розподілена архітектура системи, яка поєднує можливості хмарного сервера та локальних клієнтських застосунків.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Цей підхід дозволяє оптимізувати роботу як в онлайн, так і в офлайн режимах, забезпечуючи гнучкість та надійність. Добре спроектована архітектура дозволяє чітко розмежовувати відповідальність компонентів, спрощує розробку та підтримку, а також забезпечує гнучкість системи до майбутніх змін. Для цього її було сформовано на основі концепцій чистої архітектури [10]. Ключовий підхід полягає у чіткому поділі прикладної логіки на ізольовані рівні, кожен із яких виконує свою визначену роль. Завдяки цьому мінімізується взаємозалежність частин системи, що полегшує підтримку, оновлення і подальший розвиток.

Впровадження принципів чистої архітектури забезпечує використання різноманітних технологічних рішень, кожне з яких відповідає за конкретну функціональність. Компоненти впроваджуються на декількох інфраструктурних рівнях, що дозволяє досягти оптимальної продуктивності, надійності та безперебійного доступу до сервісів.

На рис. 1.4 наведено схему логічних шарів чистої архітектури.

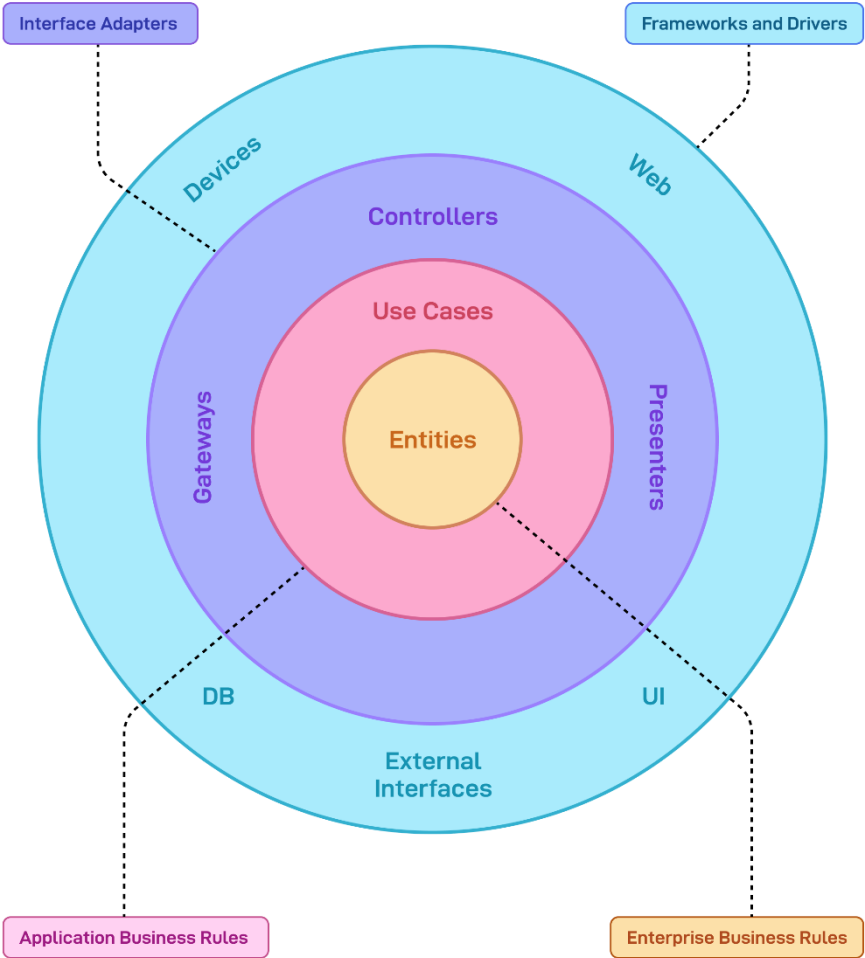


Рисунок 1.4 – Схема логічних шарів чистої архітектури

Так як дана системи поєднує у собі віддалений сервер, глобальну базу даних та клієнтське програмне забезпечення (локальний сервер, розширення для браузера, десктопний застосунок та локальна база даних), доцільно використовувати клієнт-серверну архітектуру. У цій моделі серверна частина відповідає за обробку запитів, бізнес-логіку та взаємодію з глобальною базою даних, тоді як клієнтська частина забезпечує взаємодію з користувачем та локальне зберігання даних. Такий підхід дозволяє чітко розділити функціональні обов'язки, що сприяє підтримці та розширенню системи.

Питання безпеки є надзвичайно важливим при виборі архітектури розподіленої системи, тож необхідно впроваджувати відповідні заходи безпеки на рівні мережі, застосунків та даних.

Важливою концепцією при розробці такої розподіленої системи є використання загальноприйнятих шаблонів проектування. Саме перевірені рішення для типових проблем, що виникають під час розробки програмного забезпечення, допомагають створювати структуровані, ефективні та легко змінювані системи, ґрунтуючись на досвіді та кращих практиках розробки.

Серверна частина, як віддалена, так і локальна, використовує такий шаблон проектування як REST API (Representational State Transfer). REST API – це архітектурний стиль веб-сервісів, що ґрунтується на використанні HTTP-протоколу для взаємодії між клієнтом і сервером. Він дозволяє забезпечити доступ до ресурсів через стандартні HTTP-методи (GET, POST, PUT, PATCH, DELETE), що спрощує розробку та інтеграцію веб-сервісів, роблячи їх масштабованими, а також легкими для розуміння та використання.

Крім того, для підвищення гнучкості в системі широко застосовуються принципи впровадження залежностей (Dependency Injection). Це дозволяє розробляти слабо зв'язані модулі, що спрощує розробку та покращує масштабованість.

На рис. 1.5 наведено схему шаблону впровадження залежностей.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

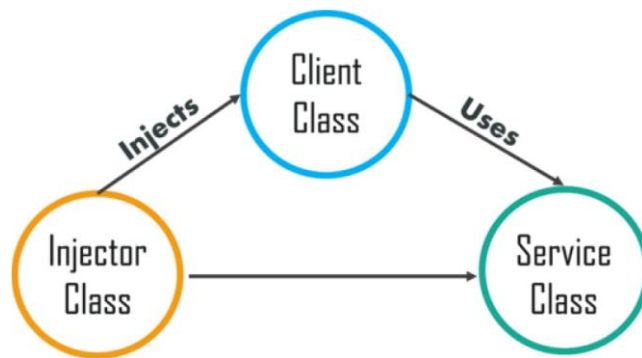


Рисунок 1.5 – Схема шаблону впровадження залежностей

Тут зображуються три основних компоненти: Injector Class, Client Class та Service Class. Client Class представляє компонент, який потребує певної залежності для виконання своєї логіки, і використовує Service Class. Service Class – це залежність, яка надає функціональність, необхідну Client Class. Injector Class відповідає за створення екземплярів Service Class та впроваджує їх у Client Class, таким чином усуваючи пряму залежність Client Class від створення Service Class.

На клієнтській частині, що відповідає за інтерфейс та взаємодію з користувачем, також застосовуються відповідні архітектурні підходи. Для десктопного застосунку застосовуються класичні підходи до архітектури, що дозволяють ефективно керувати даними та відображенням, забезпечуючи ефективну прив'язку даних (binding) до інтерфейсу користувача. Це забезпечує синхронізацію даних між моделлю та представленням, спрощуючи розробку інтерактивних інтерфейсів. Розширення для браузера, що базується на сучасних бібліотеках, також застосовує патерни управління станом, які забезпечують ефективне та передбачуване оновлення даних та компонентів.

Для забезпечення уніфікованого доступу до даних як з десктопного застосунку, так і з розширення, використовується локальний сервер. Він виступає як проміжний шар, який взаємодіє з локальною базою даних та віддаленим сервером через API. Така архітектура дозволяє централізувати логіку доступу до даних та зменшити дублювання коду.

1.4 Обґрунтування вибору інструментальних засобів

Для реалізації десктопного застосунка використано фреймворк WPF (Windows Presentation Foundation), який дозволяє створювати продуктивні та адаптивні віконні застосунки для ОС Windows [29]. WPF використовує мову C#, що забезпечує зручну інтеграцію з іншими компонентами системи та спрощує уніфікацію стека технологій. Використання WPF також знижує потребу у встановленні сторонніх інструментів на пристрої користувача.

Для реалізації віддаленого сервера обрано C#, що забезпечує уніфікацію інструментарію розробки та спрощує взаємодію між серверами і клієнтами. Віддалена база даних базується на сервісах Azure, а саме на Azure SQL Database та Blob Storage – для зберігання двійкових даних, в даному випадку аудіо файли [7]. При цьому розробка і тестування проводиться локально, з використанням SQLServer, що дозволяє не витрачати обмежені ресурси, більш швидко, зручно та безпечно працювати з БД, а у разі потреби дозволяє легко здійснити міграцію на платформу Azure [24].

Локальний сервер розробляється на основі фреймворку ASP.NET, а саме ASP.NET Web API, що ідеально доповнює використання C# для клієнтських і серверних компонентів [5, 6]. Це забезпечує зручність розробки, кращу підтримуваність та скорочення часу на інтеграцію компонентів системи. Як локальну базу даних обрано SQLServer, що вирізняється високою продуктивністю, масштабованістю та відповідністю сучасним стандартам безкоштовного ПЗ. Окрім цього однією з причин обрання саме цієї СУБД була її гарна сумісність з Azure SQL Database [8].

Для роботи з базами даних у середовищі C# використовувалась бібліотека Entity Framework Core [13]. Це сучасна, легковагова та зрозуміла ORM від компанії Microsoft. Вона дозволяє використовувати такий підхід як code-first – генерація бази даних на основі визначених класів та відносин, заданих об'єктно орієнтованим шляхом. Використання такого підходу значно спрощує та пришвидшує розробку, а можливість простої міграції змін дозволяє вносити зміни у структуру БД просто і безпечно.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Розширення для браузера (Chrome) створено з використанням бібліотеки React, яка значно спрощує розробку інтуїтивно зрозумілих інтерфейсів та підтримує швидку інтерактивність [21, 28]. У якості мови програмування обрано TypeScript, яка, у порівнянні з JavaScript надає розширений функціонал для роботи з користувацькими типами даних, що особливо корисно під час формування та обробки запитів, на яких і базується робота з даними у даній системі [16, 26]. Тестування та експлуатація розширення для браузера вимагає компіляції проекту у JavaScript та завантаження його у браузер власноруч після внесення змін. Цей процес є занадто непродуктивним, тож для швидкої компіляції використовується Vite, а для гарячого перезавантаження додаткові бібліотеки, що дозволило набагато пришвидшити розробку [12, 27].

Для отримання матеріалів, необхідних для функціонування системи, використовуються зовнішні онлайн-ресурси. Основним джерелом даних виступає платформа Reverso, яка надає доступ до якісного лінгвістичного контенту. У процесі розробки програмного забезпечення Reverso суттєво оновила свою систему захисту від автоматизованого збору даних (парсингу), що ускладнило прямий доступ до її матеріалів. З огляду на це, виникла потреба в застосуванні додаткових технічних рішень для забезпечення стабільної взаємодії з цим ресурсом. Для вирішення цієї задачі було обрано сервіс ScrapingBee, який спеціалізується на наданні інтерфейсів для роботи з веб-ресурсами, зокрема за допомогою headless-браузерів [23]. Це дозволяє ефективно обходити захисти від парсингу без необхідності самостійно налаштовувати та підтримувати складне середовище емуляції користувацької поведінки. Крім технічних переваг, важливим критерієм вибору ScrapingBee став наданий сервісом безкоштовний ознайомчий період. Завдяки цьому стало можливим провести повноцінне тестування та демонстрацію розробленої системи без додаткових фінансових витрат, що було особливо актуально в межах реалізації дипломного проекту.

Для реалізації чату зі штучним інтелектом використовувався безкоштовний сервіс-провайдер OpenRouter [19]. Це агрегатор API, що дозволяє отримувати доступ до різноманітних мовних моделей (LLM) від різних

постачальників через єдиний інтерфейс. При розробці використовувалась потужна мовна модель DeepSeek.

При реалізації функціоналу десктопного застосунку використовувались також такі бібліотеки як MailKit, для надсилання повідомлень на пошту, Xceed - бібліотека що надає додаткові елементи UI а також FontAwesome – використовувалась для отримання таких ресурсів як іконок а також для зручної взаємодії з такими елементами [14, 18, 30].

З метою забезпечення систематизованої та безпечної розробки системи, особлива увага приділялася інструментам контролю версій та базовим механізмам захисту даних. Для контролю змін у коді використовувалася система Git, яка дозволяє відстежувати зміни, зручно управляти гілками розробки та забезпечує збереження історії всіх модифікацій [15]. Це особливо важливо в умовах тривалої поетапної роботи над проєктом, де необхідна можливість відкату до попередніх версій.

Вибір саме цього стека технологій був зумовлений його відповідністю актуальним тенденціям. Основний вибір інструментів базується на необхідності використання безкоштовних або доступних ресурсів, а також відповідності сучасним вимогам ІТ-індустрії.

Висновки до першого розділу

У першому розділі було сформовано концепцію проєкту та здійснено аналіз існуючих аналогів розроблюваної системи. Розгляд наявних рішень дозволив визначити основні переваги й недоліки популярних інструментів, що поєднують переклад і навчання, а також окреслити напрямки, у яких новий продукт зможе забезпечити покращений користувацький досвід. Зібрані дані стали основою для вибору архітектури майбутньої системи а також інструментальних засобів реалізації. Таким чином, розділ заклав концептуальні та технічні основи для подальшої розробки системи, що буде конкурентноспроможною та відповідатиме сучасним вимогам.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ ПЕРЕКЛАДУ

2.1 Визначення варіантів використання

Для забезпечення зручного використання, ефективного навчання та відповідності встановленим стандартам і вимогам, наведено високорівневі вимоги, які повинні задовольняти «Система перекладу з елементами навчання».

Аналіз вимог користувачів дав підстави для визначення варіантів використання «Системи перекладу з елементами навчання».

У системі наявний лише один рівень доступу («користувач»), проте, так як система є розподіленою, розрізняються два актори типу «користувач»: користувач застосунку та користувач розширення. Таке розділення зумовлено відсутністю можливості встановити розширення для браузеру на мобільний пристрій.

Короткий опис акторів представлено в табл. 2.1.

Таблиця 2.1

Виявлення акторів

Актор	Короткий опис
Користувач застосунку	Має доступ до функціоналу десктопного застосунку.
Користувач розширення	Має доступ до функціоналу розширення.

Короткий опис варіантів використання наведено в табл. 2.2.

Таблиця 2.2

Виявлення варіантів використання

Основний актор	Найменування	Формулювання
Користувач застосунку	Авторизація	Цей варіант дозволяє вводити свої облікові дані та отримати доступ до системи.

Основний актор	Найменування	Формулювання
Користувач застосунку	Реєстрація	Цей варіант дозволяє створювати обліковий запис у системі.
	Налаштування	Цей варіант дозволяє робити налаштування різного виду, в тому числі налаштування навчальних матеріалів.
	Власні дані	Цей варіант дозволяє скопіювати готову колекцію перекладів.
	Поширення даних	Цей варіант дозволяє зробити власну колекцію загальнодоступною.
	Переклад через відповідне поле	Цей варіант дозволяє перекласти слово шляхом введення його у відповідне поле.
	Перегляд словника (Усієї бази даних)	Цей варіант дозволяє переглядати словник, в тому числі без доступу до мережі Інтернет.
	Доступ до перекладу	Цей варіант дозволяє переглядати конкретний переклад, редагувати та видалити його за потреби.
	Перегляд активностей	Цей варіант дозволяє переглядати доступні навчальні матеріали а також тести.
	Обрання активності	Цей варіант дозволяє перейти до конкретної активності, де користувач зможе її пройти.
	Перегляд користувачів	Цей варіант дозволяє переглядати список користувачів з відкритим обліковим записом.
	Перегляд користувача	Цей варіант дозволяє переглядати дані окремого користувача, його загальнодоступні колекції.
	Перегляд колекції	Цей варіант дозволяє переглядати вміст колекції, видаляти потрібні елементи, переглядати конкретні переклади.

Основний актор	Найменування	Формулювання
Користувач застосунку	ШП-чат перекладу	Цей варіант дозволяє використовувати ШП-чат у вікні окремого перекладу.
	Завершення активності	Цей варіант дозволяє переглянути результати пройденої активності.
	Перегляд облікового запису	Цей варіант дозволяє переглядати дані облікового запису поточного користувача.
	Редагування облікового запису	Цей варіант дозволяє редагувати налаштування облікового запису а також видалити його.
	Керування колекціями перекладу	Цей варіант дозволяє додавати та видаляти окремий переклад з власних колекцій.
Користувач розширення	Менеджмент останніх перекладів	Цей варіант дозволяє працювати з останніми перекладами, видаляти їх, управляти колекціями конкретного перекладу.
	Переклад з контексту за допомогою виділення	Цей варіант дозволяє перекласти слово шляхом виділення його в контексті.
	Переклад з контексту за допомогою контекстного меню	Цей варіант дозволяє перекласти слово шляхом обрання відповідної дії у контекстному меню.
	Додаткові налаштування	Цей варіант дозволяє провести додаткові налаштування, такі як зміна теми та розмір шрифту.

2.2 Розробка бази даних системи

Структура локальної і віддаленої баз даних має спільний базис. Не дивлячись на логічні відмінності у даних, що там зберігаються, таке рішення було прийнято, керуючись потребою у більш швидкій та зрозумілій розробці, простоті синхронізації, уніфікованості доступу до даних. Проте кожна БД має

додаткові унікальні таблиці, задля забезпечення специфічних задач кожної. На рис. 2.1 наведено схему глобальної бази даних.

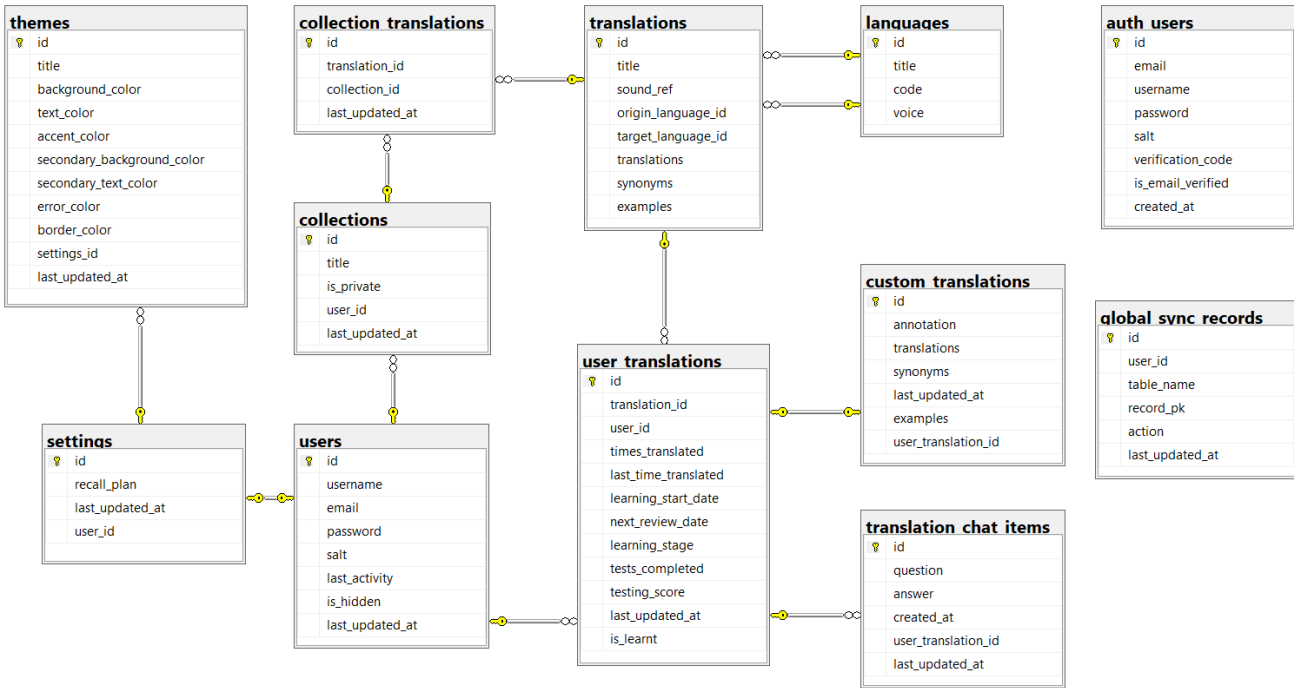


Рисунок 2.1 – Схема віддаленої бази даних

Локальна база даних містить лише дані, що відносяться до поточного користувача. Так таблиця Users буде містити лише один запис. Ключі локальної бази даних копіюються з віддаленої, задля забезпечення відповідності ключів двох БД і, в свою чергу, спрощеної взаємодії.

Загальними таблицями для обох баз даних є: translations, languages, collections, collection_translations, custom_translations, users, user_translations, settings та themes, translation_chat_items.

Короткий опис призначення загальних таблиць баз даних наведено у табл. 2.3.

Таблиця 2.3

Призначення загальних таблиць

Назва	Опис
translations	Дані про переклад: включають оригінальний текст, мови перекладу, список перекладів, приклади використання та синоніми
languages	Дані про наявні у системі мови

Назва	Опис
collections	Дані про колекції користувачів
collection_translations	З'єднувальна таблиця колекцій та перекладів
custom_translations	Таблиця, що дозволяє персоналізувати переклади без зміни оригінальної таблиці translations.
users	Дані про користувачів
user_translations	З'єднувальна таблиця користувачів та збережених перекладів
settings	Дані про налаштування користувачів
themes	Власні теми користувачів
translation_chat_items	Дані чату з III. Належить конкретному user_translation

Віддалена база даних містить дві унікальні таблиці, а саме global_sync_records та auth_users. Короткий опис призначення унікальних таблиць віддаленої бази даних наведено у табл. 2.4.

Таблиця 2.4

Призначення унікальних таблиць віддаленої бази даних

Назва	Опис
auth_users	Дані про користувачів, які знаходяться у процесі реєстрації. Таблиця необхідна для зберігання секретного коду, що надсилається користувачу на пошту. Після підтвердження пошти необхідні дані переносяться до таблиці users, а дані з поточної таблиці видаляються
global_sync_records	Дані про записи у глобальних таблицях, що вимагають синхронізації на клієнті. Реалізовані алгоритми використовують дану таблицю для зберігання видалених записів

Варто зазначити, що обидві БД зберігають інформацію про час останніх змін у більшості спільних таблиць через поле last_updated_at, що дозволяє ефективно реалізувати механізм вибіркової синхронізації. Такий підхід значно зменшує обсяг передаваних даних при оновленні.

Схему локальної бази даних наведено на рис. 2.2.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

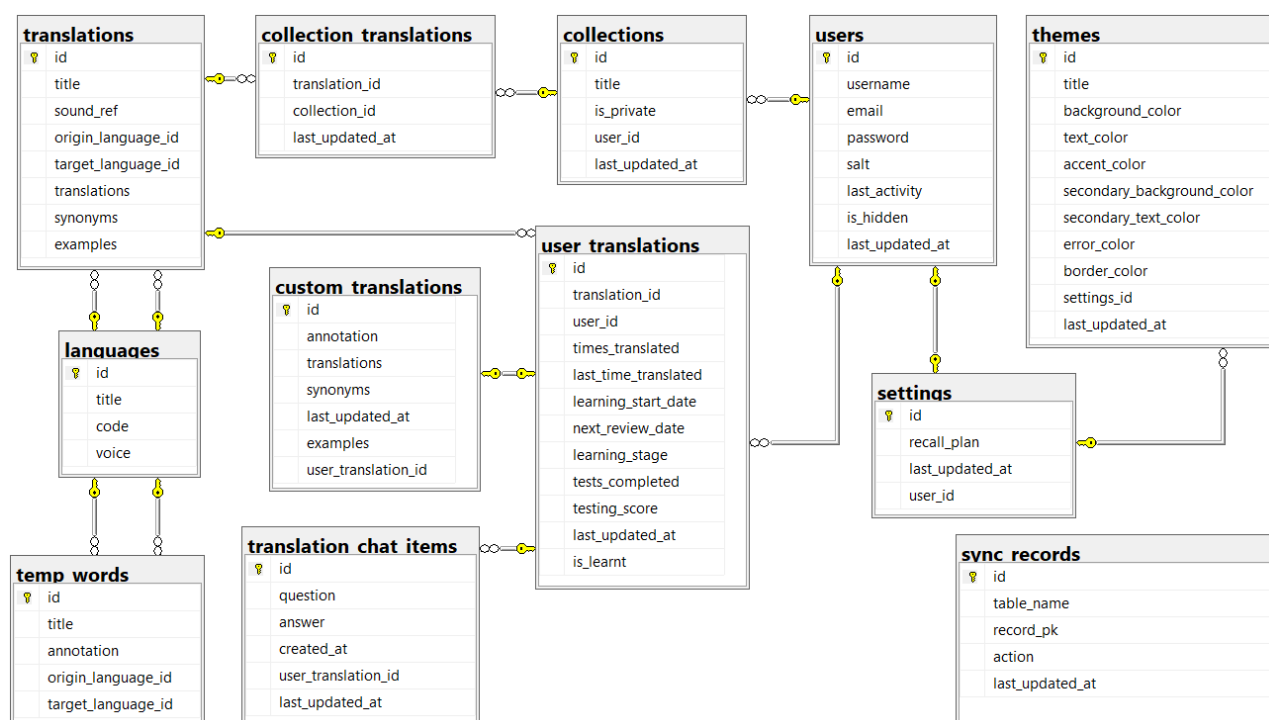


Рисунок 2.2 – Схема локальної бази даних

Локальна база даних також містить дві унікальних таблиці – temp_records та sync_records. Короткий опис призначення унікальних таблиць локальної бази даних наведено у табл. 2.5.

Таблиця 2.5

Призначення унікальних таблиць віддаленої бази даних

Назва	Опис
temp_records	Дані про текст, що було невдало спробовано перекласти без наявності Інтернет-з'єднання. У такому випадку текст потрапляє до цієї таблиці разом з необхідними параметрами
sync_records	Дані про записи у локальних таблицях, що вимагають синхронізації на сервері

У табл. 2.6-2.11 наведено більш докладну інформацію про основні таблиці.

Таблиця 2.6

Опис таблиці translations

Назва поля	Тип даних	Опис
id	uniqueidentifier	Первинний ключ
title	nvarchar	Текст що перекладається

Назва поля	Тип даних	Опис
sound_ref	uniqueidentifier	Ідентифікатор відповідного аудіо-файлу
origin_language_id	uniqueidentifier	Мова з якої перекладається
target_language_id	uniqueidentifier	Мова на яку перекладається
translations	nvarchar	Список перекладів
synonyms	nvarchar	Список синонімів
examples	nvarchar	Список прикладів використання

Тут складні дані, такі як поля translations, synonyms та examples, зберігаються у вигляді JSON, що допомагає позбавитись зайвих таблиць та зробити структуру БД більш простою і зрозумілою, а також спрощує обробку цих даних на клієнтській стороні без необхідності складних запитів.

Таблиця 2.7

Опис таблиці collections

Назва поля	Тип даних	Опис
id	uniqueidentifier	Первинний ключ
title	nvarchar	Назва колекції
is_private	bit	Визначник доступності колекції для інших користувачів системи
user_id	uniqueidentifier	Користувач, що володіє колекцією
last_updated_at	datetime2	Дата останньої зміни

Таблиця 2.8

Опис таблиці settings

Назва поля	Тип даних	Опис
id	uniqueidentifier	Первинний ключ
recall_plan	nvarchar	Масив типу int у форматі JSON, що представляє частоту, з якою користувач повторює слова згідно плану.
last_updated_at	datetime2	Дата останньої зміни
user_id	uniqueidentifier	Користувач, якому належить запис

Опис таблиці themes

Назва поля	Тип даних	Опис
id	uniqueidentifier	Первинний ключ
title	nvarchar	Назва теми
background_color	datetime2	Основний колір фону
text_color	nvarchar	Колір тексту
accent_color	nvarchar	Колір акценту
secondary_background_color	nvarchar	Другорядний колір фону
secondary_text_color	nvarchar	Другорядний колір тексту
error_color	nvarchar	Колір помилок
border_color	nvarchar	Колір границь
settings_id	uniqueidentifier	Посилання на відповідні налаштування
last_updated_at	datetime2	Дата останньої зміни

Слід зазначити, що назви кольорів несуть загальний сенс і не завжди застосовуються за прямим призначенням. Це дозволило спростити взаємодію користувача при керуванні темами, при цьому надавши достатню варіативність.

Таблиця 2.10

Опис таблиці user_translations

Назва поля	Тип даних	Опис
id	uniqueidentifier	Первинний ключ
translation_id	uniqueidentifier	Посилання на відповідний переклад
user_id	uniqueidentifier	Посилання на користувача
times_translated	int	Кількість разів, скільки текст було перекладено
last_time_translated	datetime2	Останній раз, коли текст було перекладено
learning_start_date	datetime2	Дата початку вивчення
next_review_date	datetime2	Наступна дата повторення
learning_stage	int	Етап вивчення
tests_completed	int	Кількість завершених тестів
testing_score	int	Успішність проходження тестувань

Назва поля	Тип даних	Опис
last_updated_at	datetime2	Дата останньої зміни
is_learnt	bit	Ідентифікатор, що визначає, чи навчання було завершено

Таблиця 2.11

Опис таблиці users

Назва поля	Тип даних	Опис
id	uniqueidentifier	Первинний ключ
username	nvarchar	Ім'я користувача
email	nvarchar	Електронна пошта
password	nvarchar	Хеш паролю
salt	nvarchar	Сіль, що використовувалась для хешування паролю
last_activity	datetime2	Дата останньої активності
is_hidden	bit	Наступна дата повторення
last_updated_at	datetime2	Дата останнього оновлення

Загалом, структура бази даних є масштабованою, гнучкою та орієнтованою на поступове розширення функціональності системи без необхідності внесення значних змін до загальної архітектури. Передбачена модульність, повторне використання логіки та уніфікований підхід до зберігання даних забезпечують простоту адаптації до нових вимог.

2.3 Проектування та реалізація алгоритмів роботи системи

Система використовує підхід online-first з надійним офлайн-режимом – метод роботи з даними, що першочергово спирається на взаємодію з глобальною базою даних при наявності інтернету, дозволяючи при цьому операції з локальною БД та відкладену синхронізацію у випадку його відсутності. Залежно від конкретної задачі, це може надавати такі переваги як актуальність даних та можливість безперебійної роботи офлайн.

Розглянемо один із основних процесів системи, а саме, алгоритм перекладу слова та додавання його до словника або до черги, в залежності від наявності Інтернет з'єднання. Виділимо три актори для даного опису, а саме: локальний сервер, локальну базу даних та глобальний сервер разом з віддаленою базою даних. На рис. 2.3 зображено діаграму активності, що відображає взаємодію трьох акторів у даному алгоритмі:

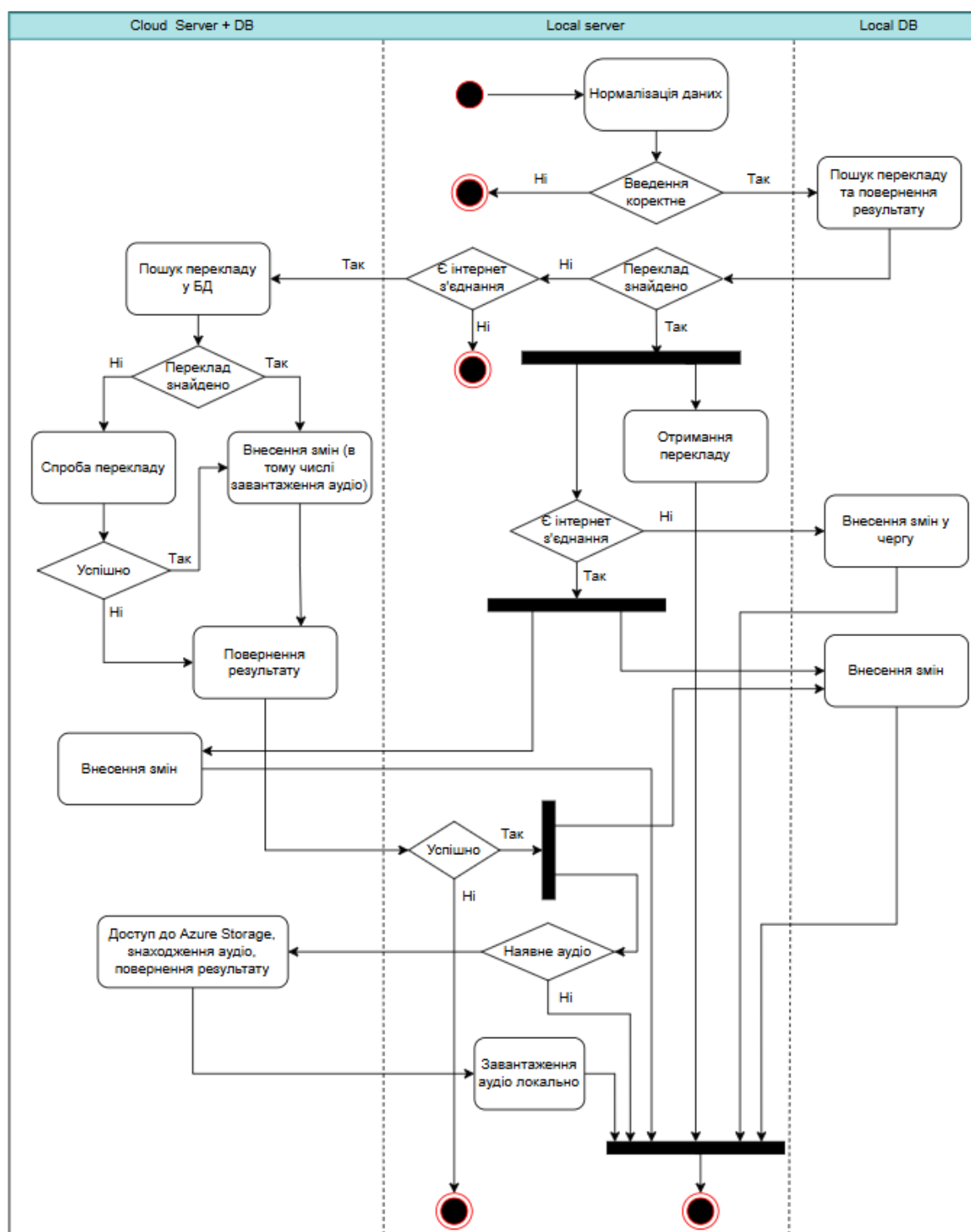


Рисунок 2.3 – Діаграма, що описує процес перекладу

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

Після отримання запиту клієнта з текстом для перекладу, цільовий текст проходить нормалізацію локально, тобто прибираються зайві символи, пробіли тощо. Якщо дані введено коректно, система виконує локальний пошук перекладу у внутрішній базі даних. Якщо переклад знайдено, він одразу повертається користувачу.

У разі, якщо переклад у локальній БД не знайдено, перевіряється наявність інтернет-з'єднання. Якщо з'єднання відсутнє, зміни ставляться в чергу на синхронізацію, яка відбудеться при відновленні мережевого доступу. Інакше, якщо інтернет доступний, локальний сервер звертається до віддаленого сервера з запитом на переклад. Глобальний сервер у свою чергу намагається знайти відповідний переклад у глобальній базі даних. Якщо такий переклад знаходиться, за необхідності він прив'язується до поточного користувача, вносяться зміни та повертається результат. При незнаходженні у БД слово перекладається та при успішному перекладі алгоритм продовжує роботу аналогічним чином. В іншому випадку – повертається результат з помилкою і алгоритм закінчує роботу.

Якщо переклад знайдено, на локальному сервері також фіксуються зеркальні зміни одразу. Якщо інтернет-з'єднання на момент повернення результату втрачене – віддалено результат вже збережено, тому при наступній синхронізації переклад буде завантажено з іншими несинхронізованими записами.

Така логіка дозволяє гарантувати стабільну роботу програми навіть в офлайн-режимі, за рахунок використання черги змін та подальшої синхронізації.

За необхідності операції супроводжуються відповідними повідомленнями про успішність чи помилки для інформування користувача.

Розглянемо ще один алгоритм, що описує процес синхронізації локальної та глобальної бази даних, який виконується при появі інтернет-з'єднання або запуску локального серверу. Враховуються обидва варіанти: коли зміни були внесені локально та коли зміни було внесено у глобальну БД (з іншого пристрою) (рис. 2.4).

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

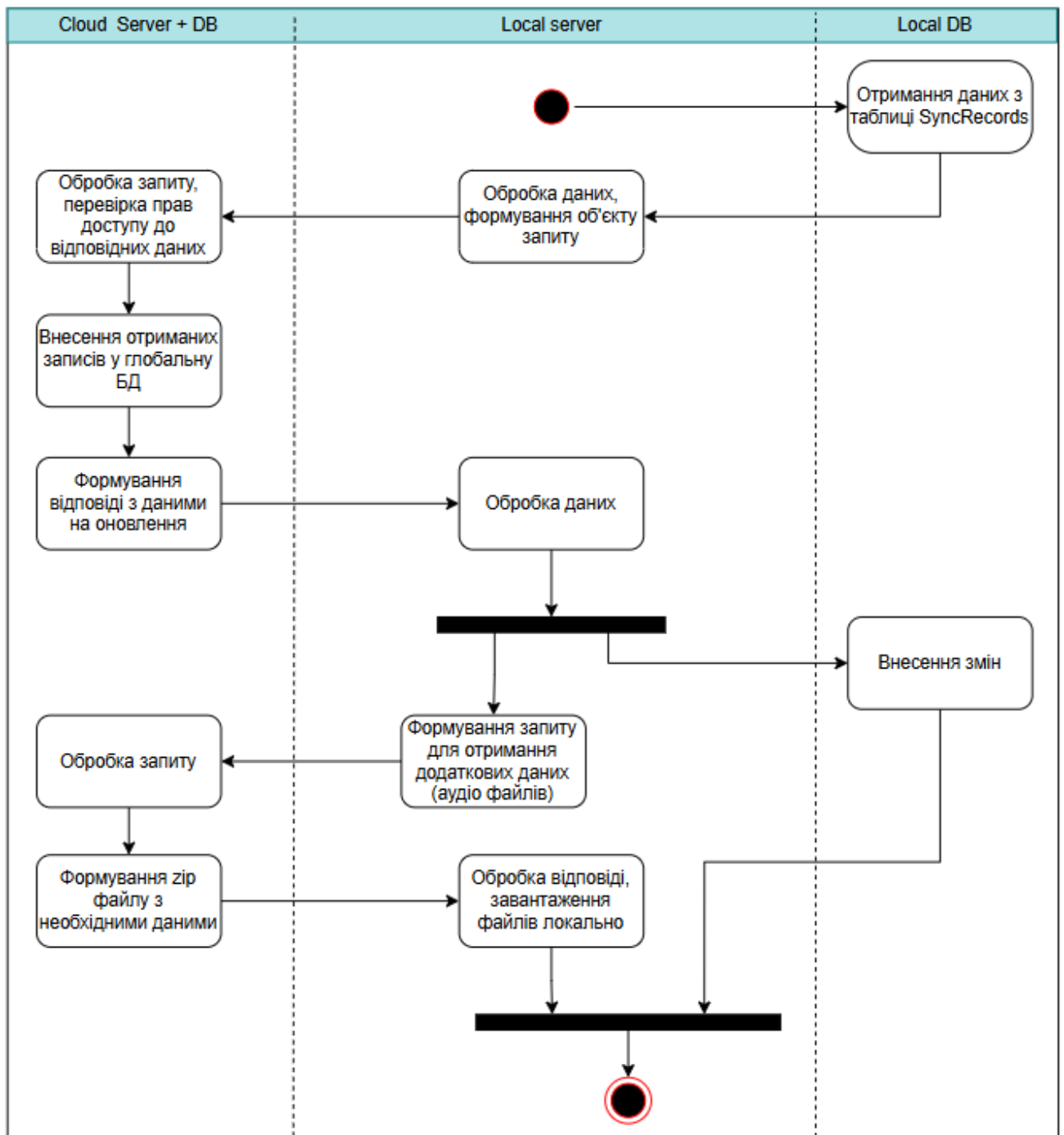


Рисунок 2.4 – Діаграма процесу синхронізації

Процес синхронізації бази даних починається з перевірки таблиці змін (SyncRecords) у локальній БД, яка зберігає інформацію про всі оновлення, зроблені у відсутності інтернет-з'єднання. Після отримання цих даних локальний сервер формує об'єкт запиту, що містить добре структуровані оновлення, які необхідно передати до глобальної бази.

Зміни можуть відбуватись як локальному сервері, так і на глобальному –

якщо користувач використовує декілька пристроїв підключених до одного облікового запису. Саме тому реалізується двостороння синхронізація.

Сервер отримує запит, перевіряє права доступу до відповідних записів та починає процес внесення змін. Отриманий об'єкт запиту представляє собою JSON об'єкт, який десеріалізується у клас C#, визначений у shared_models [9]. Цей об'єкт містить записи, які потрібно внести у БД. Об'єкти є однорівневими, тобто не містять вкладених моделей БД. При цьому кожен запис містить зовнішні ключі, які перевіряються на те, чи має користувач право здійснювати маніпуляції з тими чи іншими записами у різних таблицях. Для цього створено оптимізований алгоритм, який робить декілька швидких запитів до БД та зберігає у змінні типу HashSet ідентифікатори, до яких користувач має доступ. Після чого ключі та зовнішні ключі кожного запису перевіряються на наявність у тому чи іншому HashSet. Записи, що призначені для вставки, мають локально призначені ID. Щоб запобігти такій вразливості створюється Map-об'єкт, який об'єднує в пари локальні та новостворені глобальні ID, на які вони замінюються. У процесі валідації формуються дані для швидкої вставки у БД. Такий підхід дозволив швидко працювати з даними великих об'ємів та БД.

Після цього формується відповідь, яка включає всі нові записи, що були змінені або додані з інших пристроїв: з таблиці GlobalSyncRecords отримуються всі записи користувача, які було видалено, при цьому дані для вставки/оновлення збираються напряму з цільових таблиць, враховуючи такі дані як LastClientSyncAt (час останньої синхронізації конкретного пристрою (надсилається у запиті) та last_updated_at – поле кожної таблиці, яка може бути змінена користувачем. Таблиці без цього поля за необхідності прив'язуються до інших об'єктів відповіді. Окрім цього повертається Map-об'єкт з парами локальних та глобальних ідентифікаторів, для того, щоб уніфікувати ідентифікатори. Ці дані повертаються на локальний сервер, де обробляються та вносяться відповідні зміни до локальної БД.

Якщо існує потреба у додаткових даних (наприклад, завантаження пов'язаних аудіофайлів), формується окремий запит. У відповідь сервер створює

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

zip-архів з потрібними файлами, який завантажується на пристрій. Після завершення синхронізації, усі локальні й глобальні дані вважаються узгодженими.

Цей процес реалізується із застосуванням концепцій патерну Sync Adapter, який характеризується розділенням логіки синхронізації та підготовки даних на незалежні частини, наявністю окремих інтерфейсів для взаємодії з даними будь-яких типів, що дозволяє не змінювати, а лише розширювати наявну реалізацію, інкапсульовану обробку даних синхронізації, двонаправленість тощо.

Процес синхронізації відбувається при запуску застосунку (за наявності інтернет-з'єднання) та після відновлення з'єднання. Перевірка з'єднання відбувається кожні 10 секунд у фоновому процесі. При потребі синхронізації вона також відбувається у якості фонового процесу, що дозволяє користувачу не чекати на завантаження та продовжувати використовувати ПЗ.

Коли користувач входить до свого облікового запису – усі необхідні дані формуються у стиснутий документ та надсилаються клієнту у вигляді файлу.

Наступним кроком описано загальний алгоритм роботи активностей (рис. 2.5). Після ініціалізації активності користувачем натисканням відповідної кнопки, відбувається запит до локальної бази даних. Так як дані для активностей формуються із перекладів користувача, це означає, що вони зберігаються локально, а значить немає необхідності робити запит на віддалений сервер. Таким чином користувач може проходити активності без наявності інтернет-з'єднання.

Після отримання стандартизованої відповіді, кожна активність оброблює дані відповідно до своєї внутрішньої логіки, в тому числі для спрощеного відображення. Коли користувач потрапляє на сторінку конкретного елемента активності (наприклад картку), користувач може покинути активність або завершити її. При покиданні активності дані не будуть оновлені, в той самий час як при завершенні сформується запит на оновлення. Він представляє з себе колекцію даних про кожен переклад, а саме статистика проходження активності. Такими даними можуть бути кількість правильних та неправильних відповідей,

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

кількість пропущених карток тощо. Ці дані передаються на локальний сервер, який зразу оброблює дані та оновлює запис у БД. В той самий час в залежності від наявності інтернет-з'єднання аналогічний запит перенаправляється на віддалений сервер, або ж вносяться дані до черги синхронізації.

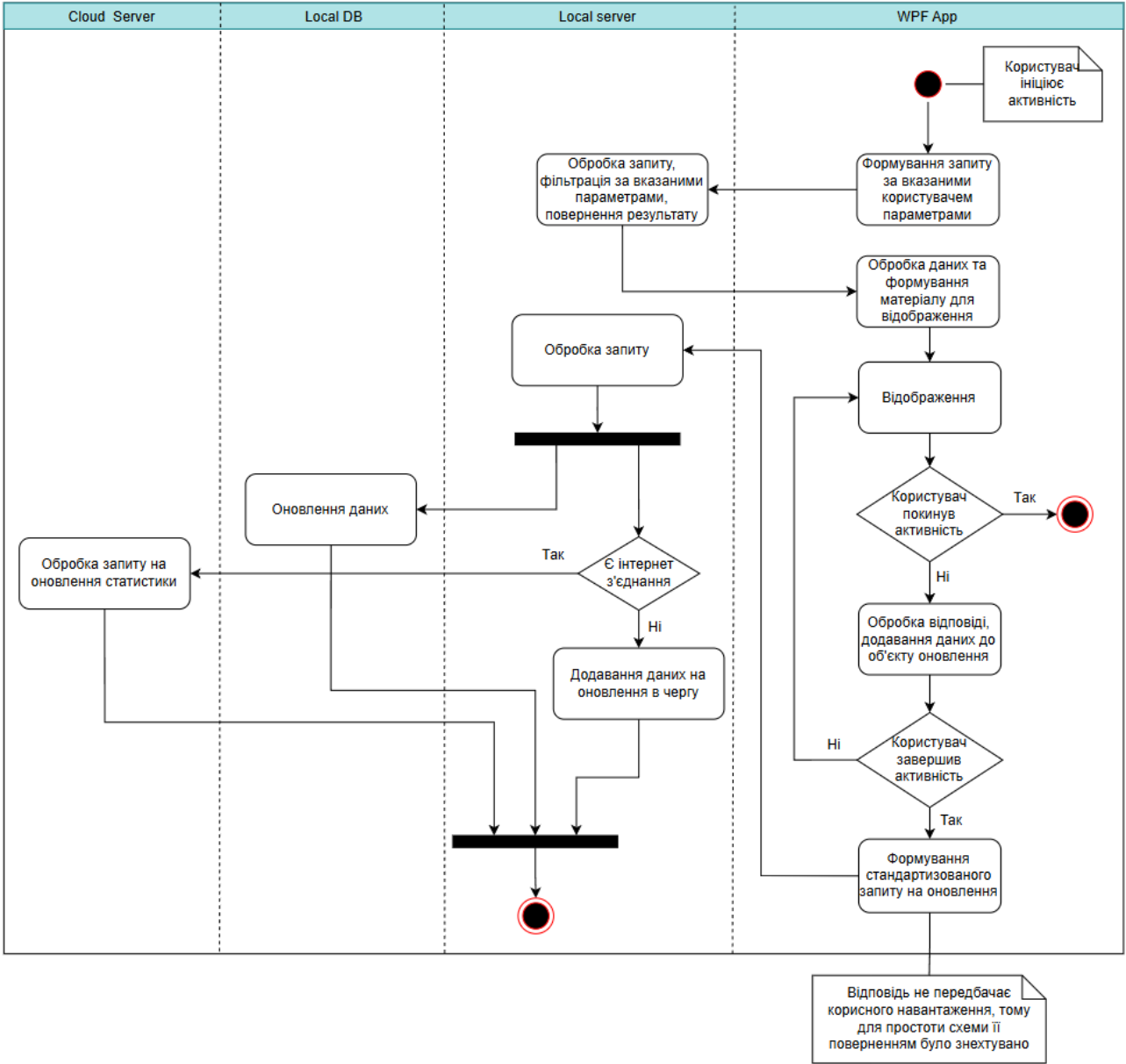


Рисунок 2.5 – Діаграма процесу проходження активності

2.4 Релізація системи перекладу з елементами навчання

На рис. 2.6 відображається багаторівнева структура локального середовища, розділена на кілька основних компонентів та їхні залежності.

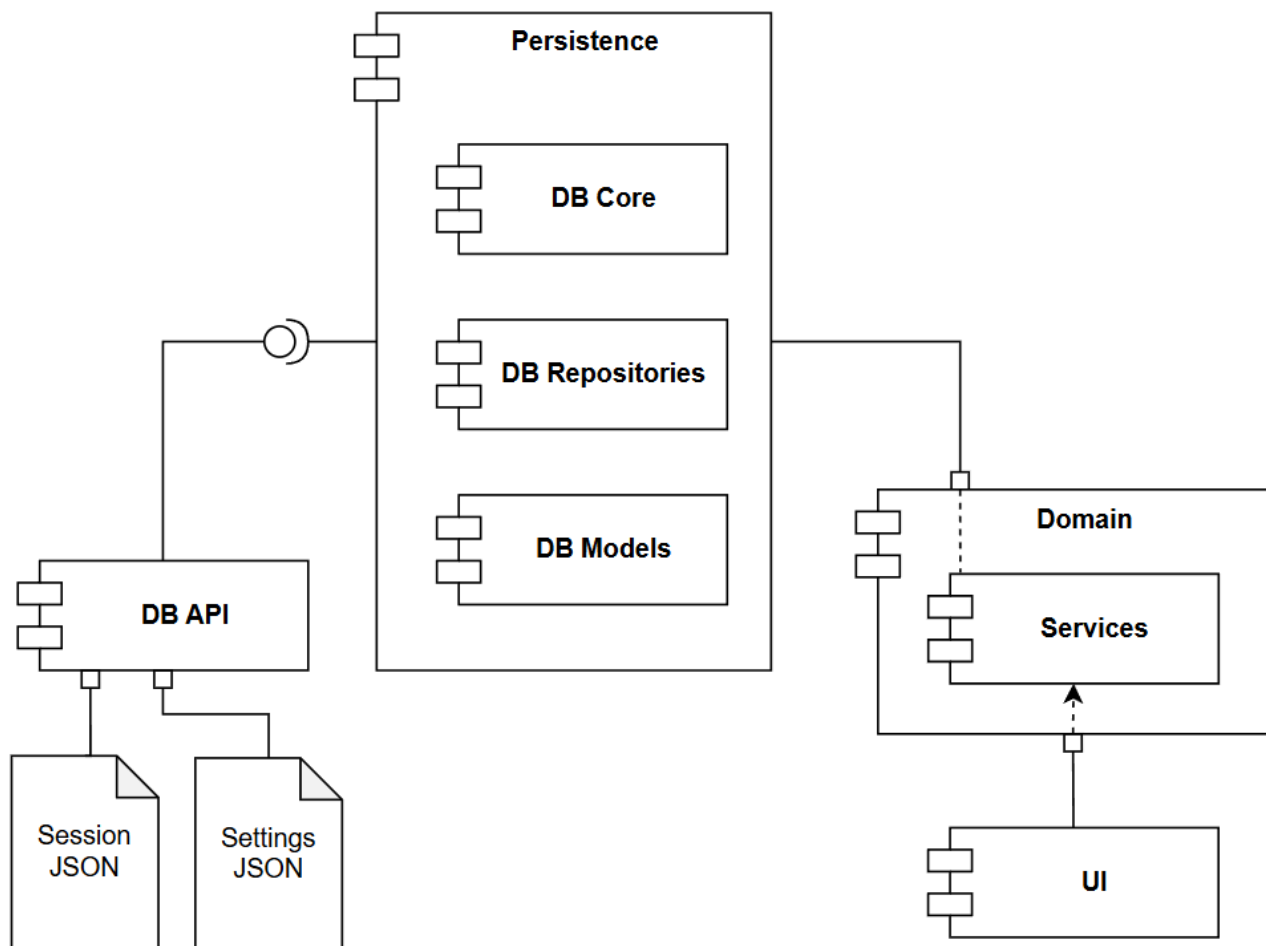


Рисунок 2.6 – Діаграма компонентів (Application)

Файли конфігурації (Session JSON та Settings JSON, та інші) показують дані, які зберігаються окремо у вигляді файлів і зберігають інформацію специфічну для даної сесії та пристрою. UI (User Interface) - це зовнішній рівень системи, з яким безпосередньо взаємодіє користувач і потрібен для обробки його дій та для відображення даних і залежить від компонента Domain. Domain представляє бізнес-логіку та дані системи. Він містить основні правила, сутності та процеси предметної області. Всередині компонента Domain знаходиться підкомпонент Services. Це набір сервісів, які реалізують бізнес-логіку, обробляють дані та координують взаємодію між різними частинами програми. Компонент Domain залежить від компонента Persistence. Для зберігання та отримання даних він область використовує функціональність, надану рівнем Persistence. Persistence – компонент, що відповідає за збереження та отримання даних із постійного сховища. Він складається з трьох підкомпонентів:

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

1. DB Core (Ядро бази даних) - містить основну логіку взаємодії з базою даних (встановлення з'єднання, виконання запитів тощо);
2. DB Repositories (Репозиторії бази даних): Надають абстракцію над доступом до даних. Кожен репозиторій може відповідати за управління певним типом сутностей домену, приховуючи деталі взаємодії з базою даних;
3. DB Models представляють структуру даних у базі даних де кожна таблиця представлена у вигляді класу (моделі).

На рис. 2.7 наведено структуру проекту у контексті розробки.

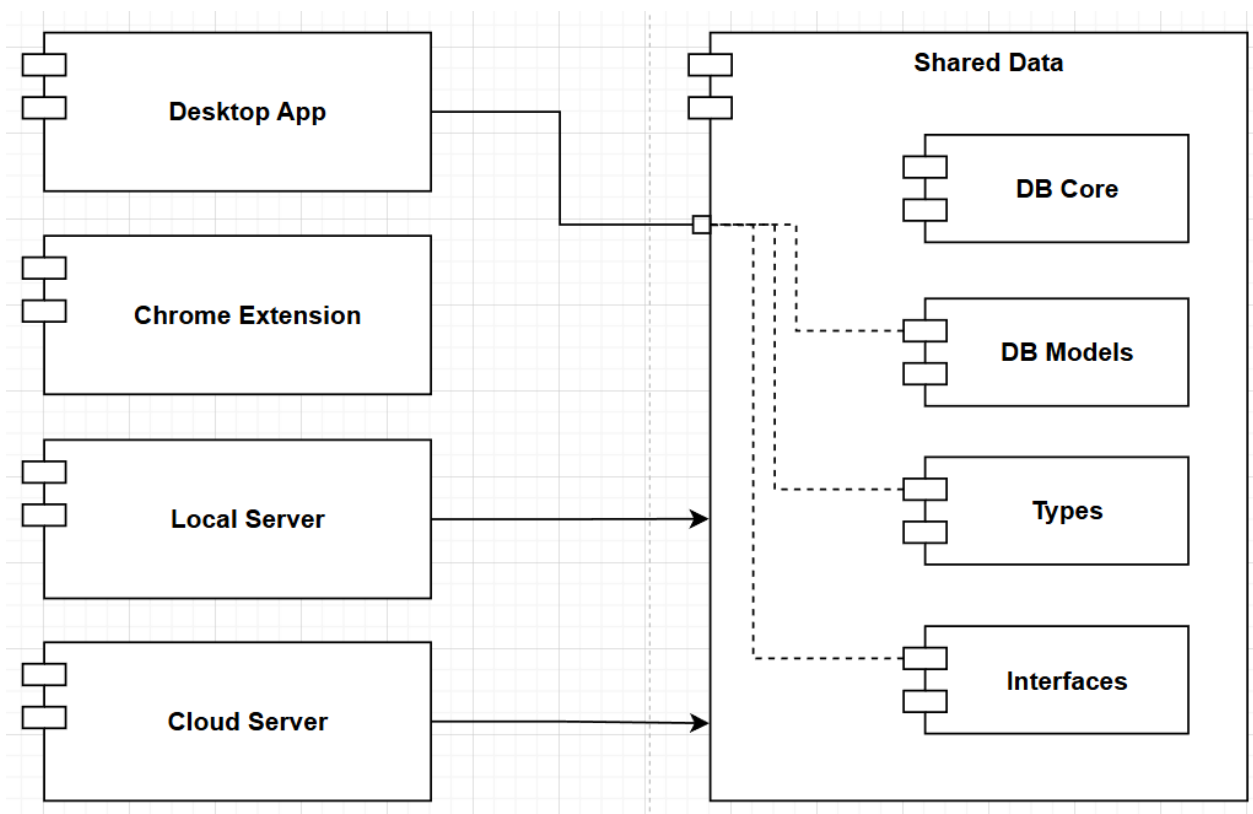


Рисунок 2.7 – Діаграма компонентів (етап розробки)

Наведена діаграма ілюструє архітектуру програмної системи в розрізі її розробки, де ключову роль відіграє централізований компонент Shared Data (спільні дані). Цей компонент визначає спільні типи даних, інтерфейси та методи для роботи з ними а також спільний контекст бази даних. Така архітектура підкреслює важливість спільної інфраструктури, що сприяє однорідності даних, повторному використанню коду, розділенню відповідальностей та гнучкості системи в процесі розробки різних клієнтських та серверних застосунків, які

працюють з єдиним ядром.

Розглянемо деякі методи на прикладі яких буде гарно видно підхід до розробки. Почнемо з реалізації механізму збереження колекцій користувачем. Цей функціонал передбачає копіювання існуючої колекції разом з перекладами у власну бібліотеку користувача. На хмарному сервері ця операція реалізована методом `SaveAsync`, який відповідає за бізнес-логіку збереження. Метод починається з асинхронного отримання оригінальної колекції за її ідентифікатором з бази даних. Важливо, що завантажуються також пов'язані переклади з їхніми відповідними об'єктами `Translation`, а дані не відстежуються `EF Core` для забезпечення оптимізації. Після цього виконується перевірка на те, чи така колекція існує та чи не є вона приватною. Якщо колекція не знайдена або є приватною, генерується виняток `KeyNotFoundException`. Далі створюється новий об'єкт `Collection` з унікальним ідентифікатором, копіюється назва оригінальної колекції, встановлюється `IsPrivate` у `false` та призначається `UserId` поточного користувача, після чого відбувається ітерація по перекладах оригінальної колекції. Для кожного перекладу створюється новий об'єкт `CollectionTranslation`, що зв'язує нову колекцію з існуючим перекладом. Потім перевіряється, чи вже належать ці переклади поточному користувачу (шляхом запиту до `UserTranslations`). Якщо переклад не належить користувачеві, він додається до списку разом з відповідним об'єктом `Translation`, щоб потім бути відправленим на клієнтську сторону для збереження в локальній базі даних користувача (Додаток А).

У більшості випадків для передачі складних даних використовуються окремі DTO-моделі (`Data Transfer Object`), що забезпечує кращу безпеку та оптимізацію.

Метод контролера на хмарному сервері `Save` обробляє `HTTP GET` запит за маршрутом «`save/{collectionId: Guid}`». Спочатку перевіряється ідентифікатор користувача. Якщо він відсутній, повертається відповідь `Unauthorized`. Далі Запускається блок `try-catch` для обробки можливих винятків. У блоці `try` викликається сервісний метод `_collectionService.SaveAsync` для виконання логіки

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

збереження колекції. Сериалізація та компресія даних: Отримані дані серіалізуються у формат JSON (з використанням JsonSerializerOptions для camelCase) і потім стискаються за допомогою GZip, щоб зменшити розмір даних для передачі по мережі. Стиснуті дані записуються в MemoryStream. Встановлюється заголовок Content-Disposition для вказівки імені файлу для завантаження (collection_{collectionId}.zip). Потім стиснутий потік даних повертається як файл із типом application/gzip. Код нижче демонструє відповідний контролер.

```
[HttpGet("save/{collectionId:Guid}")]
public async Task<ActionResult> Save([FromRoute] Guid collectionId)
{
    Guid? userId = User.GetUserId();

    if (userId == null)
        return Unauthorized("Invalid user");

    try {
        var savedData = await _collectionService.SaveAsync(userId.Value,
collectionId);

        var jsonOptions = new JsonSerializerOptions { PropertyNamingPolicy =
JsonNamingPolicy.CamelCase };

        var jsonBytes = JsonSerializer.SerializeToUtf8Bytes(savedData,
jsonOptions);

        var memoryStream = new MemoryStream();

        using (var gzipStream = new GZipStream(memoryStream,
CompressionMode.Compress, true))

            {
                await gzipStream.WriteAsync(jsonBytes, 0, jsonBytes.Length); }

        memoryStream.Position = 0;

        Response.Headers.Add("Content-Disposition", $"attachment;
filename=collection_{collectionId}.zip");

        return File(memoryStream, "application/gzip");
    }

    catch (KeyNotFoundException ex) {return NotFound(ex.Message); }

    catch (Exception ex) {return StatusCode(500, $"Internal server error:

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

```
{ex.Message}");}}
```

На локальному сервері метод `SaveCollectionAsync` у сервісі виконує завантаження колекції з хмарного сервера та її збереження в локальну базу даних із перевіркою віркою на дублювання. На початку виконується встановлення заголовка авторизації для HTTP-клієнта, що забезпечує безпечний доступ до ресурсів віддаленого серверу. Далі формується URL-запит до відповідного контролера хмарного серверу з урахуванням ідентифікатора колекції, що передається в якості аргументу. Для оптимізації читання відповіді використовується параметр `HttpCompletionOption.ResponseHeadersRead`, який дозволяє почати обробку вхідного потоку одразу після отримання заголовків, не очікуючи завершення передачі всього тіла відповіді. Якщо відповідь неуспішна – буде згенеровано виняток. У разі успішного запиту отриманий потік відповіді декомпресується за допомогою `GZipStream`, що дозволяє зменшити розмір передаваних даних, і десеріалізується в об'єкт типу `SavedCollectionDataDto`, структура якого визначена в спільній бібліотеці `shared models` і гарантує відповідність між клієнтом і сервером.

Перед додаванням нових записів до бази даних метод виконує перевірку наявності об'єктів за їх унікальними ідентифікаторами. Це дозволяє уникнути дублювання сутностей, які вже могли бути імпортовані раніше або передані у якості вкладених об'єктів отриманої відповіді від серверу. Для кожного з чотирьох типів сутностей `Collection`, `Translation`, `UserTranslation` та `CollectionTranslation` попередньо виконується запит до бази з метою формування списку ідентифікаторів уже наявних записів. Після цього нові об'єкти фільтруються та додаються лише у випадку їхньої відсутності в локальній БД. У ситуації, коли запис уже існує, його стан встановлюється як `Unchanged`, що дозволяє уникнути зайвих змін або конфліктів під час збереження даних. Після обробки пов'язаних об'єктів, колекція, отримана з сервера, додається до бази без пов'язаних перекладів, що дозволяє уникнути потенційних циклічних залежностей або помилок, пов'язаних із повторним додаванням. Усі зміни записуються до бази даних асинхронно, із використанням `await`, що дозволяє

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

зберігати реактивність інтерфейсу користувача під час виконання тривалих операцій. Метод реалізований з урахуванням потреби у безпечному оновленні локального сховища без втрати даних і конфліктів. Варто зазначити, що метод не перевіряє наявності Інтернет-з'єднання, оскільки передбачається, що його виклик ініціюється лише у випадку стабільного з'єднання. Такий підхід дозволяє уникнути зайвих перевірок, делегуючи контроль з'єднання зовнішній логіці (Додаток Б).

Розглянемо ще один метод `GetTranslationInternalAsync` та шаблон програмування «Стратегія», що в ньому застосовується, який забезпечує гнучку поведінку системи залежно від стану мережевого з'єднання [4].

Цей метод відповідає за отримання перекладу для користувача, спочатку намагаючись знайти його в локальній базі даних, а потім, у разі відсутності, звертаючись до віддаленого віддаленого сервера. Якщо переклад не знайдено локально, активується шаблон «Стратегія», реалізований через об'єкт `exes`. Цей шаблон дозволяє динамічно вибирати алгоритм виконання (онлайн або офлайн) залежно від доступності мережі. В даному випадку `onlineAction` відправляє HTTP GET запит до API хмарного сервера для отримання перекладу, після чого десеріалізує відповідь і зберігає `UserTranslation` та пов'язані сутності в локальній базі даних, завантажуючи також аудіофайли за наявності. `OfflineAction`, в свою чергу, зберігає запит на переклад у таблиці `TempWords` локальної бази даних, щоб обробити його пізніше при відновленні з'єднання. Після цього генерує виняток, що інформує про відсутність з'єднання та додавання слова (тексту) до черги (Додаток В).

Також нижче наведено код методу, що реалізує шаблон «стратегія».

```
public async Task<T> ExecuteAsync<T>(  
    Func<Task<T>> onlineAction,  
    Func<Task<T>> offlineAction,  
    bool offlineAllowed,  
    string offlineErrorMessage = "No internet connection")  
{using (var scope = _serviceProvider.CreateScope())
```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

```

        {var connectivity =
scope.ServiceProvider.GetRequiredService<ConnectivityService>();
        if (await connectivity.CheckAsync()) {return await onlineAction();}
        else if (offlineAllowed){return await offlineAction();}
        else{throw new OfflineUnavailableException(offlineErrorMessage);
        }}}

```

Як на локальному, так і на віддаленому сервері реалізовано сервіс Middleware, який оброблює всі помилки, тому не є критичним обгорнути всі методи у блоки try-catch. Це забезпечує централізовану обробку винятків, що значно спрощує підтримку коду та покращує читабельність логіки обробки запитів. У випадку виникнення непередбаченої помилки користувач отримує зрозуміле повідомлення, а технічна інформація автоматично реєструється в системі логування.

Висновки до другого розділу

У межах другого розділу було визначено варіанти використання системи. Після чого було проведено роботу над структурою бази даних, описано таблиці та поля, приведено схеми локальної та віддаленої баз даних. У розділі проектування та реалізації алгоритмів системи приведено та пояснено діаграми активностей основних з них. Також було наведено частини коду у якості прикладів реалізації конкретного функціоналу.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ

3.1 Порядок встановлення та налаштування параметрів системи

Етап встановлення та налаштування параметрів системи є критично важливим для забезпечення стабільної, передбачуваної та ефективної роботи програмного комплексу. Від правильності виконання цих процедур залежить не тільки коректне функціонування системи у повсякденній експлуатації, але й зручність її подальшої розробки, підтримки та масштабування. Ретельне планування та документування цих процесів дозволяє мінімізувати ризики виникнення помилок, оптимізувати продуктивність та забезпечити високий рівень користувацького досвіду.

На діаграмі розгортання (рис. 3.1) представлено схему компонентів системи та їх взаємодію.

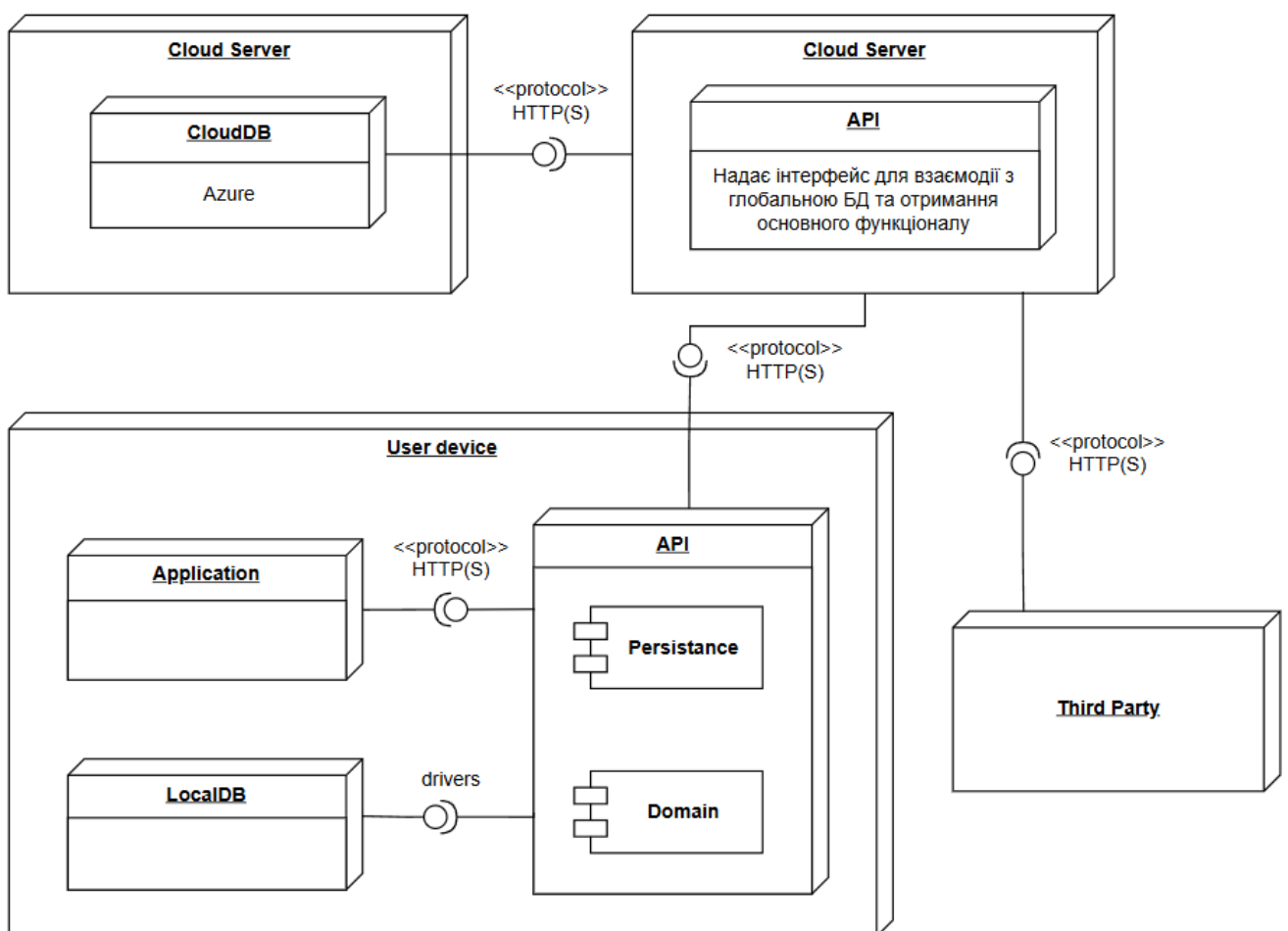


Рисунок 3.1 – Діаграма розгортання системи

Діаграма розгортання є ключовим інструментом для візуалізації фізичної

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

структури системи та ілюстрації способів взаємодії її окремих елементів. Вона надає чітке уявлення про те, як програмні компоненти, такі як клієнтські інтерфейси, API-сервери та сховища даних, розміщуються на різних апаратних пристроях, серверних платформах або в хмарних середовищах. Додатково, на діаграмі зображено мережеві з'єднання та протоколи, що використовуються для обміну інформацією між цими елементами, що є фундаментальним для забезпечення стабільної та злагодженої роботи всього програмного комплексу.

Компонент Cloud server (API) є центральним вузлом системи, відповідальним за управління глобальними даними та надання основних функціональних можливостей, і реалізований як набір сервісів, доступних через протокол HTTPS. Його основними функціями є управління доступом користувачів до системи, синхронізація даних, доступ до централізованих ресурсів, виконання складних операцій або тих, що потребують високий рівень безпеки.

CloudDb представляє собою глобальне базу даних, що є основним сховищем для всіх централізованих даних системи і надає відкритий доступ лише для API сервісу.

Далі розглянемо компонент User device, який позначає комп'ютер користувача. Тут модуль Application відповідає за функціональність десктопного застосунку або браузерного розширення, так як незалежно від варіанту використання, взаємодія з іншими складовими системи залишається однаковою. API на пристрої користувача є прошарком, що ізолює Application від прямої взаємодії з локальною базою даних та серверним API, забезпечуючи чітке розподілення відповідальності та спрощуючи розробку. Для наочності показано складові Persistence, що відповідає за роботу з локальними даними та Domain, який інкапсулює бізнес-логіку. Архітектура компоненту User device розроблена таким чином, щоб забезпечити швидку реакцію на дії користувача та ефективну роботу навіть за відсутності Інтернет-з'єднання.

Компонент Third Party акумулює в собі усі зовнішні сервіси та веб-сторінки, що використовуються системою для отримання необхідних даних

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

через надаваний інтерфейс або шляхом парсингу відповідно. Взаємодія з віддаленим сервером відбувається через протокол HTTPS, забезпечуючи стандартизований та безпечний спосіб обміну даними.

3.2 Користувацький інтерфейс розширення для браузера

Основними вимогами до розширення для браузера були його простота, зручність та налаштовуваність. Для простоти та зручності було вирішено використовувати відносно нову можливість розширень – бокову панель. Вона дозволяє зручно та нативно змінювати ширину вікна застосунку, згортати та розгортати його. Розширення складається з наступних компонентів: панель інструментів, основний блок, що представляє список усіх перекладів, відсортованих за часом останнього перекладу та динамічне вікно з повідомленнями. Кожен елемент списку може бути розгорнутий для перегляду прикладів використання, а також містить набір кнопок управління: прослухати аудіо, додати або видалити з колекції, видалити взагалі. На рис. 3.2 наведено загальний інтерфейс розширення.

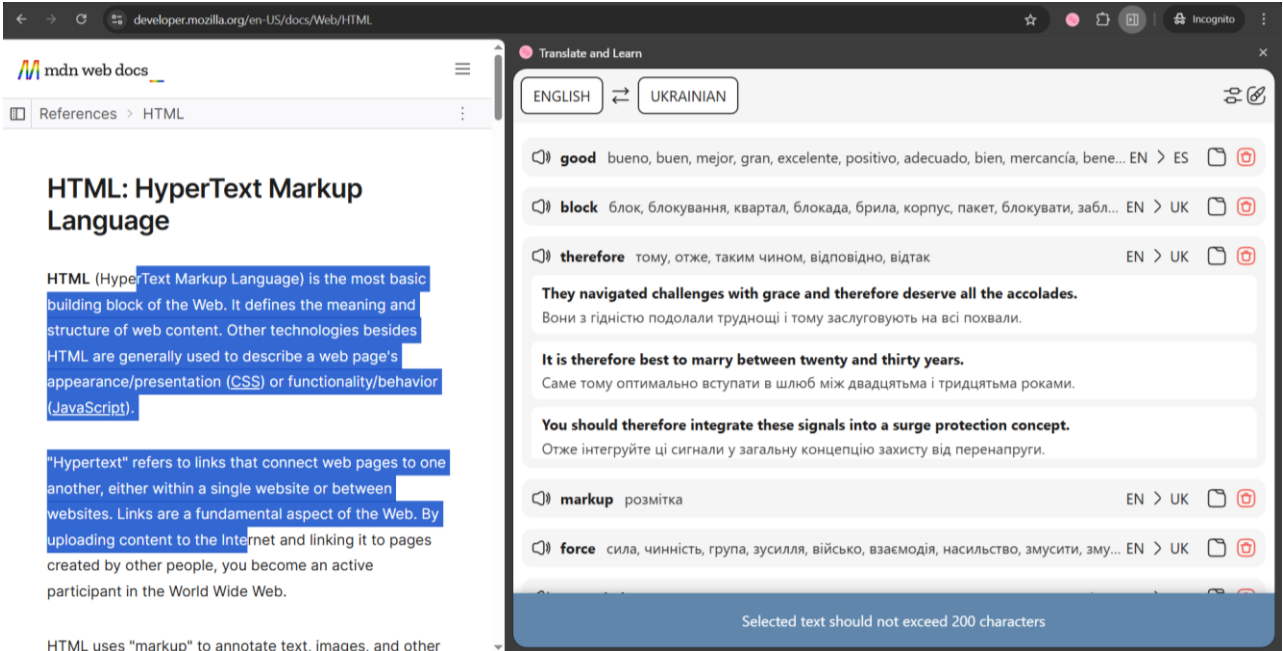


Рисунок 3.2 – Інтерфейс розширення

На рис. 3.3 наведено вікно додавання перекладу до колекцій або видалення з колекцій. Окрім цього є можливість створити нову колекцію.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

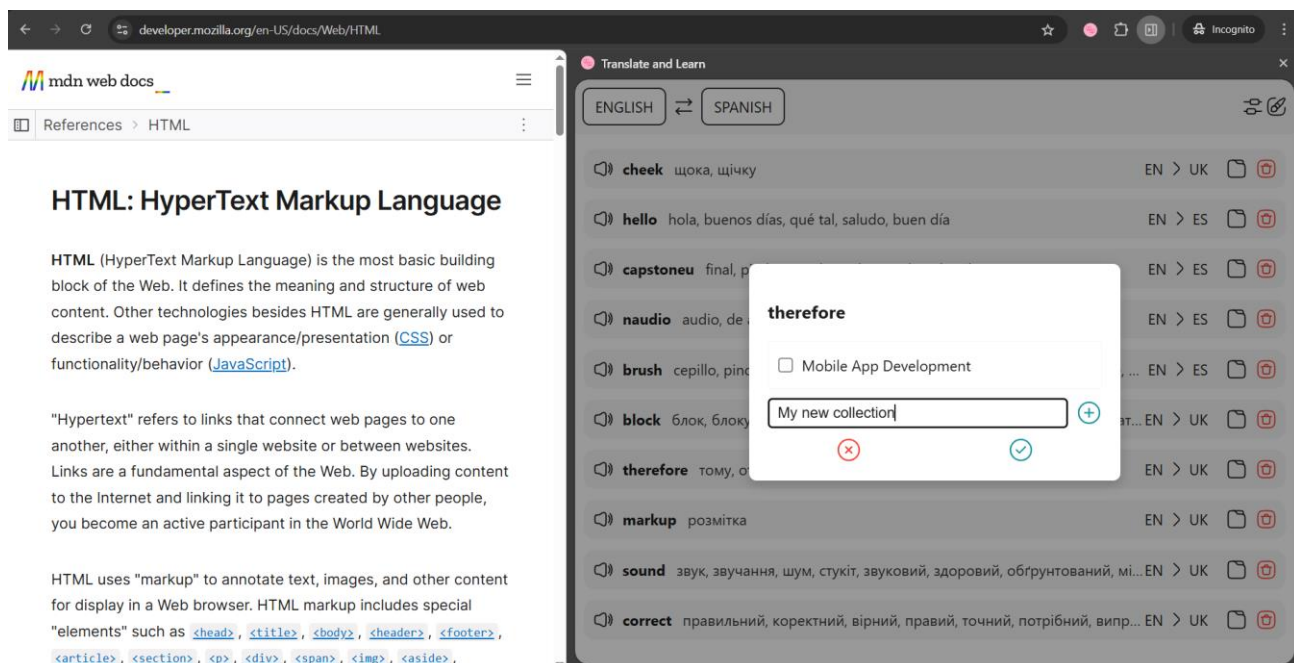


Рисунок 3.3 – Вікно керування колекціями

При цьому було необхідно враховувати такі аспекти як втомлюваність очей, якість зору тощо. Для цього було додано можливість змінювати розмір шрифту та кнопок (рис. 3.4) а також теми, для підлаштування розширення під кожну сторінку (рис. 3.5). Таким чином користувач зі слабким зором може збільшити шрифт, а тема, відповідаюча поточній сторінці за кольоровою гамою, дозволить зменшити напруженість в очах, що вкрай корисно при тривалому використанні.

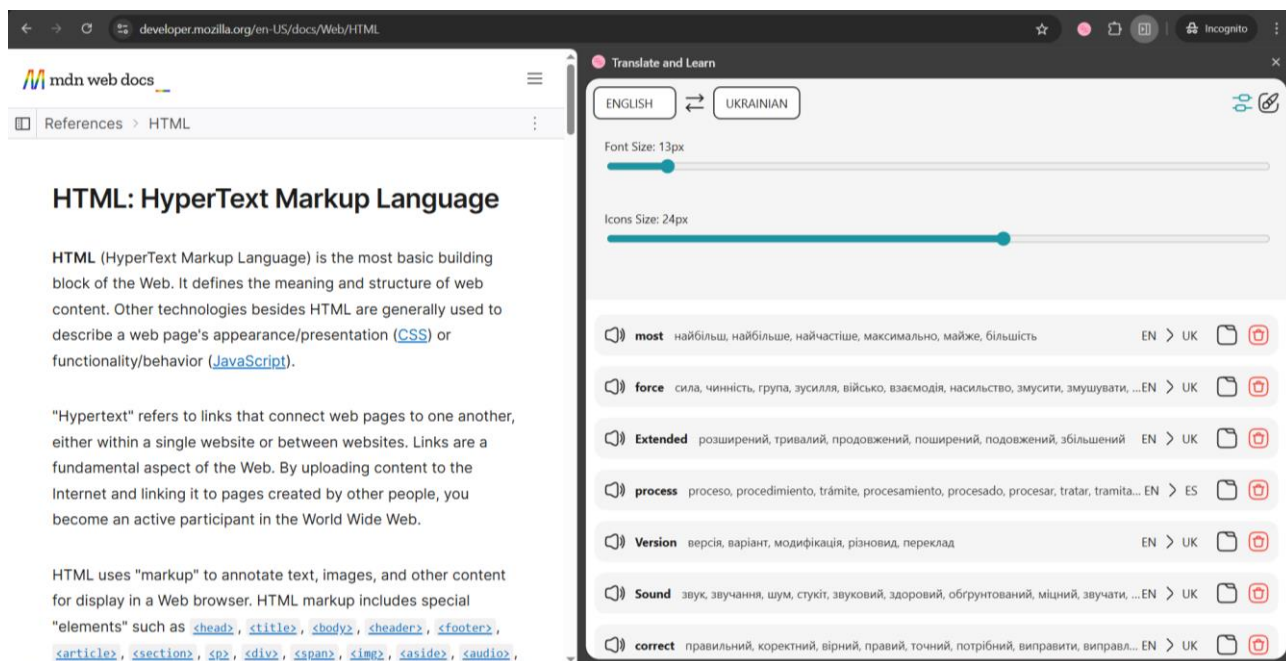


Рисунок 3.4 – Персоналізація розміру шрифту

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

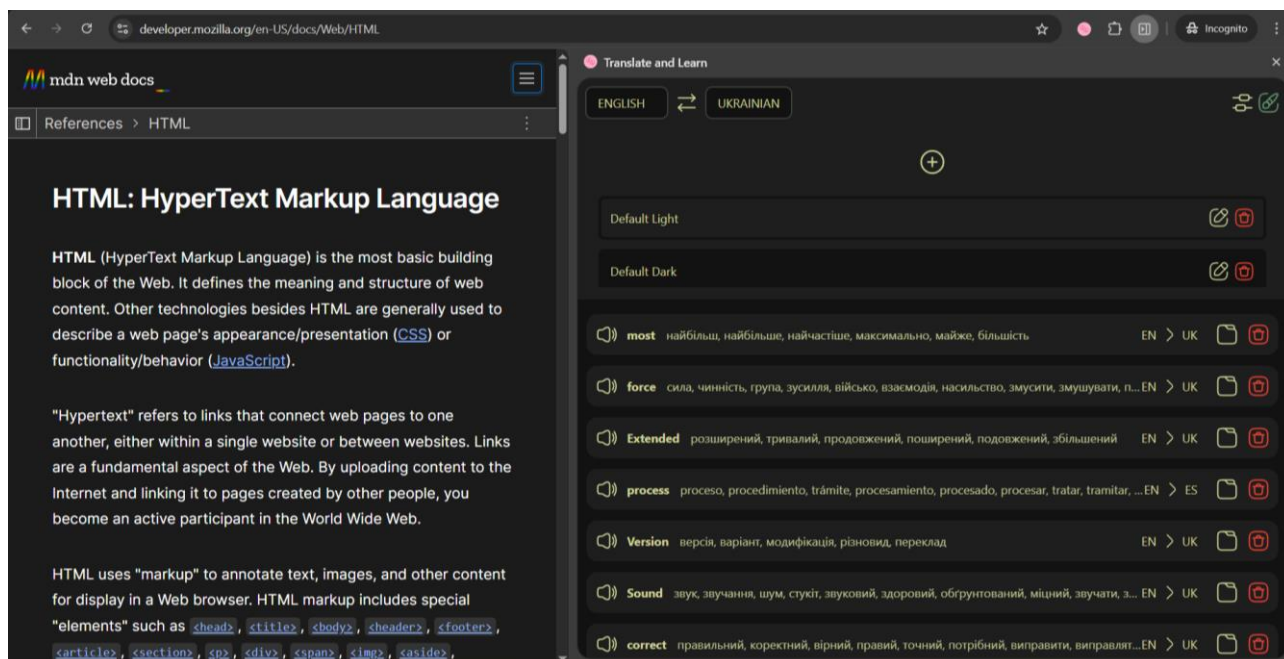


Рисунок 3.5 – Демонстрація відповідності теми розширення та сторінки

Користувач може створити власну тему, задавши потрібні кольори (рис. 3.6). Це дозволить створити набір тем і перемикатись між ними в залежності від сторінки. Для зручності додано теми за замовчуванням.

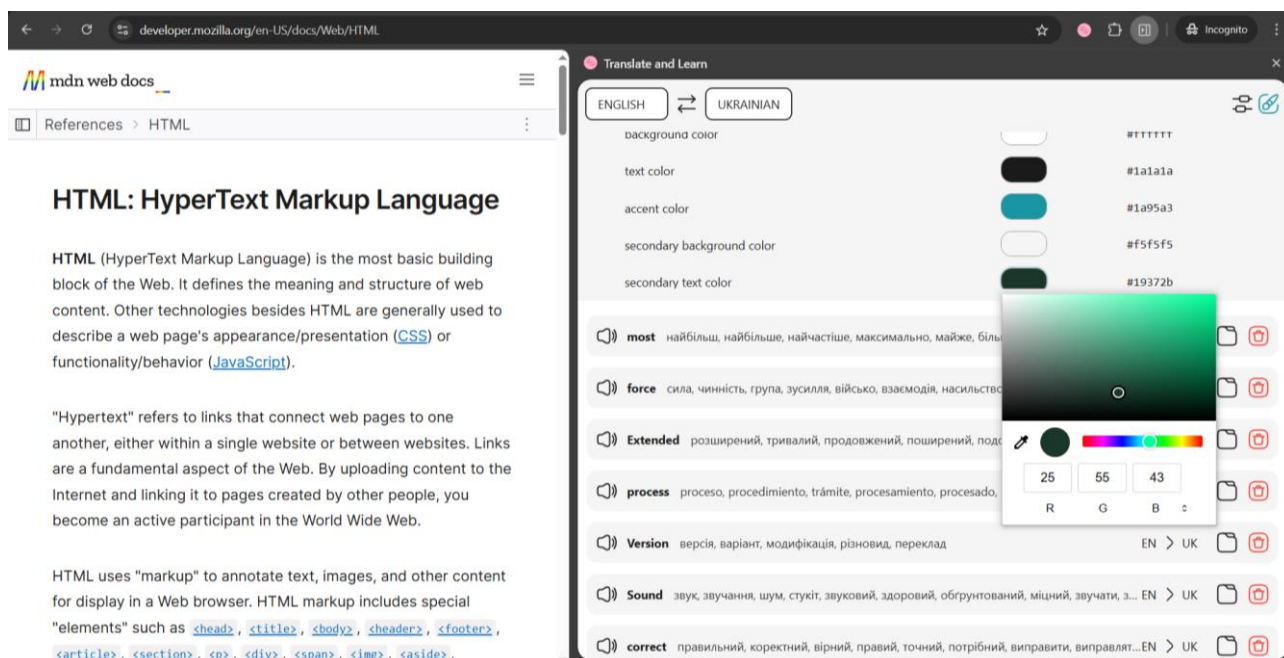


Рисунок 3.6 – Зміна кольорів теми

Переклад тексту відбувається автоматично, коли користувач виділяє текст на сторінці. При цьому виділений текст валідується, за необхідності виводиться повідомлення про помилку. Якщо переклад успішний – слово додається до

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк. 49
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		

списку. На деяких сторінках неможливо відслідкувати подію виділення або виділений текст, наприклад на сторінках типу PDF. Для таких випадків існує відповідний контролер у контекстному меню (рис. 3.7).

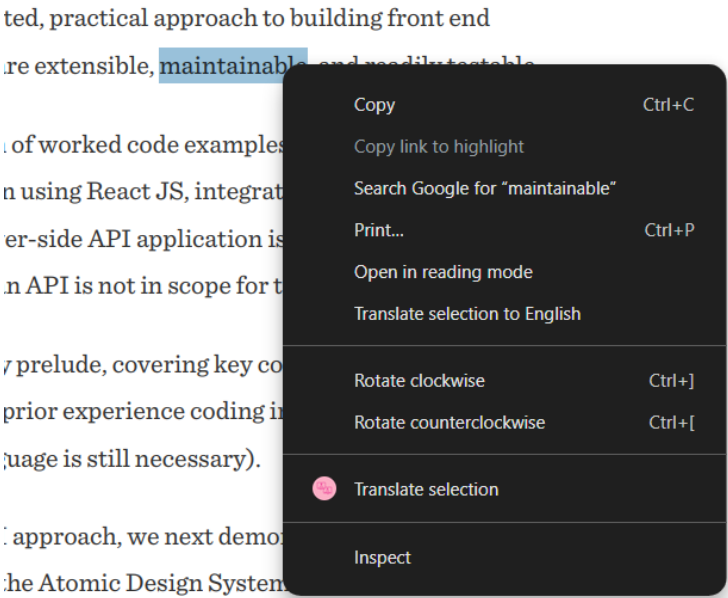


Рисунок 3.7 – Переклад через контекстне меню

При виконанні певних дій, за необхідності користувачу відображається вікно підтвердження операції (рис. 3.8). Така незначна деталь робить використання ПЗ набагато безпечніше, користувачу не доводиться остерігатись випадкових натискань кнопок, що гарно впливає на загальне сприйняття.

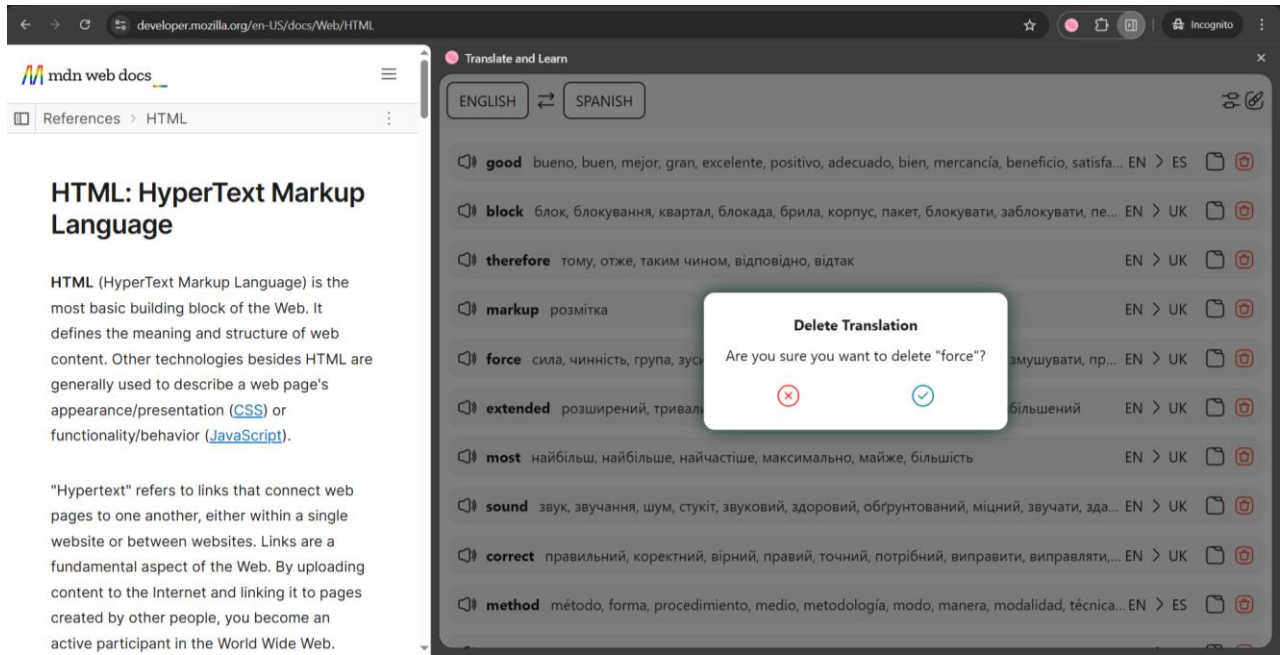


Рисунок 3.8 – Вікно підтвердження операції

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк. 50
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3 Користувацький інтерфейс застосунку

Використання системи починається з входу до облікового запису. Це можливо зробити лише через віконний застосунок. На рис. 3.9 зображено екран реєстрації де можна побачити поля, стандартні для класичного процесу реєстрації. Окрім цього можна бачити вертикальне меню, розташоване у правій частині екрану з кнопками управління, а саме стилізовані кнопки закриття та згортання застосунку, а також у низу кнопки перемикання між вкладками логіну та реєстрації.

Якщо дані реєстрації введено коректно, на пошту користувача надсилається повідомлення з секретним кодом, який необхідно ввести у відповідне поле. На рис. 3.10 – 3.11 зображено екран для введення коду та надіслане на пошту повідомлення.

Рисунок 3.9 – Сторінка реєстрації

Рисунок 3.10 – Повідомлення з секретним кодом на пошті

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

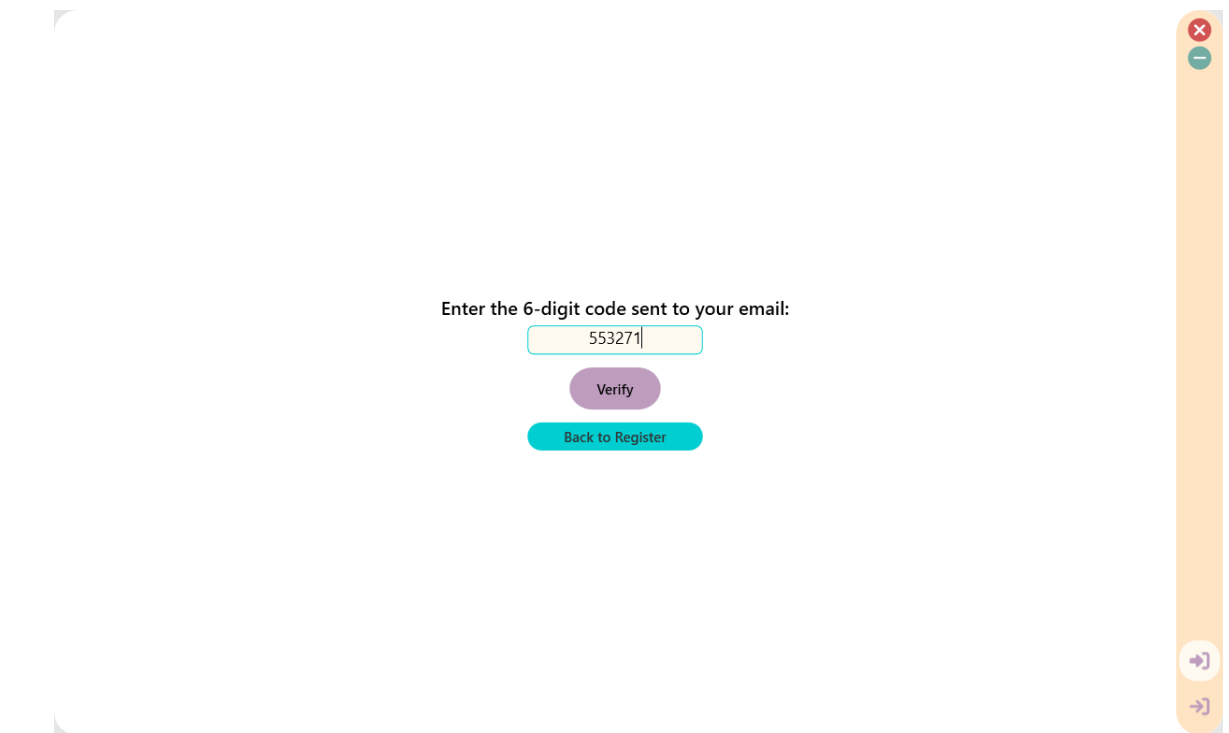


Рисунок 3.11 – Екран для введення секретного коду

Після успішного входу до облікового запису користувач потрапляє на головне вікно застосунку (Home) (рис. 3.12). На даній вкладці користувач бачить список власних перекладів, інструменти фільтрації, а саме за датою та мовами та частиною рядка, а також поле і кнопка для перекладу. Перекладене слово з'являється у списку. Відповідні поля фільтрації можливо відключити.

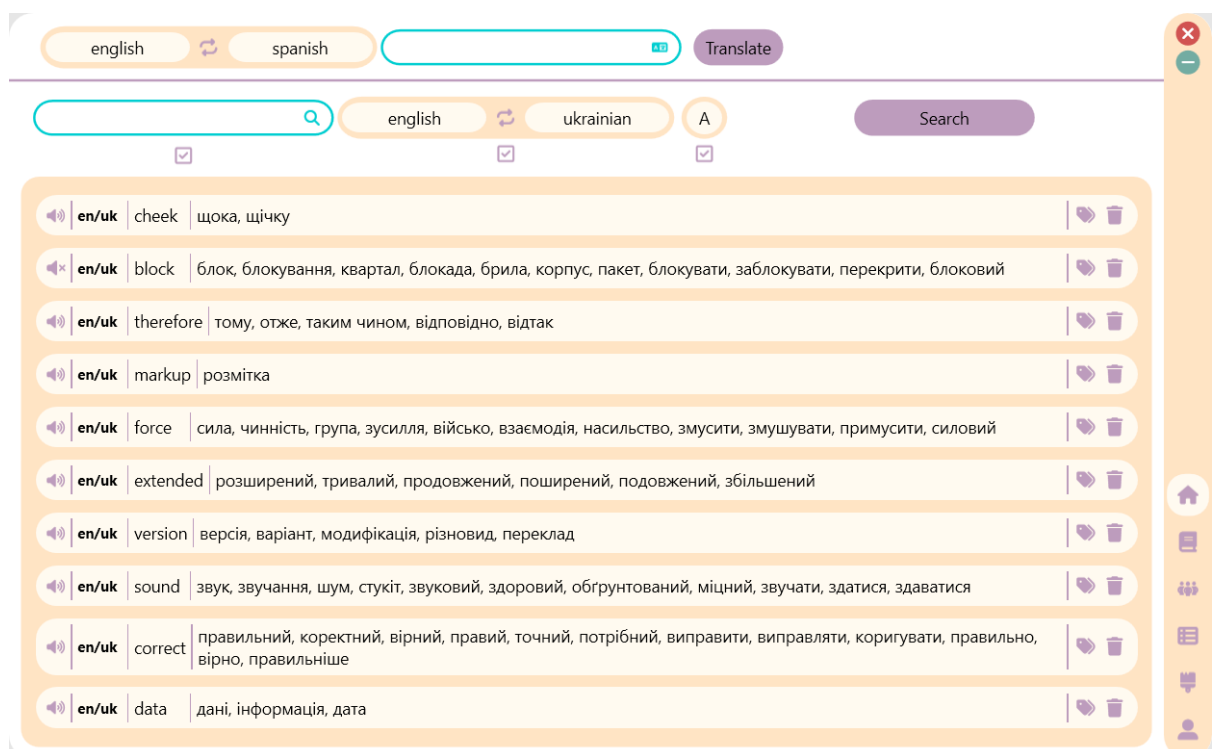


Рисунок 3.12 – Головний екран

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

На рис. 3.13 наведено вікно конкретного перекладу. На ньому відображається уся інформація про переклад і включає в себе назву, переклади, задіяні мови, приклади використання, синоніми. Користувач має змогу вносити зміни до доступних полів, редагуючи або доповнюючи інформацію відповідно до власних потреб. Така гнучкість сприяє кращій персоналізації навчального процесу. У правій частині вікна розташовано ІШ-чат, у якому можна задати питання штучному інтелекту відносно даного перекладу. Це дає змогу отримати пояснення значення, додаткові приклади вживання або поради щодо правильного вживання у контексті, що значно розширює навчальний потенціал системи.

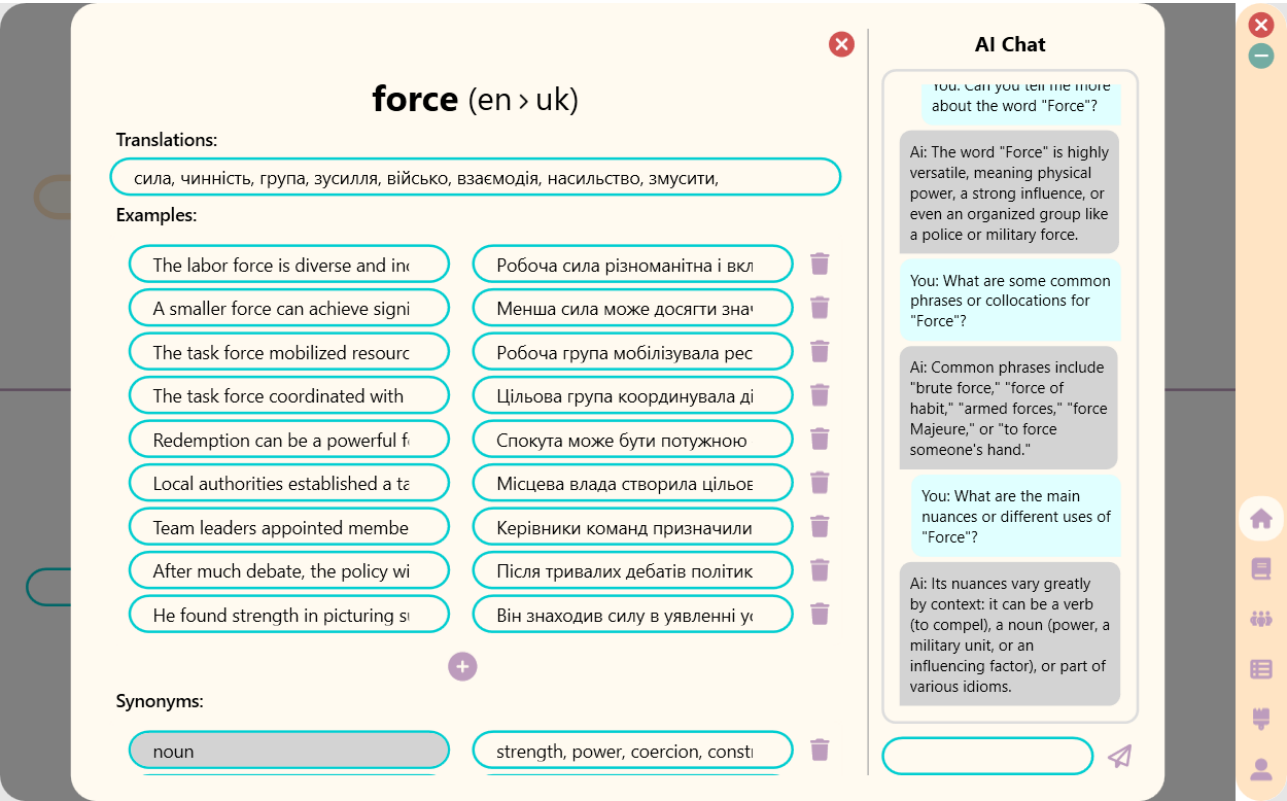


Рисунок 3.13 – Вікно перекладу

На рис. 3.14 наведено екран активностей. Даний екран поділено на дві частини. Верхня частина представляє інструменти фільтрації за мовами, бажаним об'ємом матеріалів та датою. Нижня демонструє список активностей. Після застосування фільтрів набір активностей оновлюється та користувач може перейти до цікавлячої активності натиснувши на відповідну кнопку.



Рисунок 3.14 – Екран активностей

На рис. 3.15 наведено знімок процесу проходження активності під назвою Flashcards. Головною ідеєю даної активності є надати користувачу зручний інтерфейс для повторення матеріалу шляхом демонстрації слова та перекладу на різних сторонах картки, що позитивно впливає на запам'ятовування. При цьому можливість не перегортати картку дозволяє не переглядати переклад, що може бути корисним для користувачів, що намагаються не асоціювати іншомовні слова зі словами рідної мови.

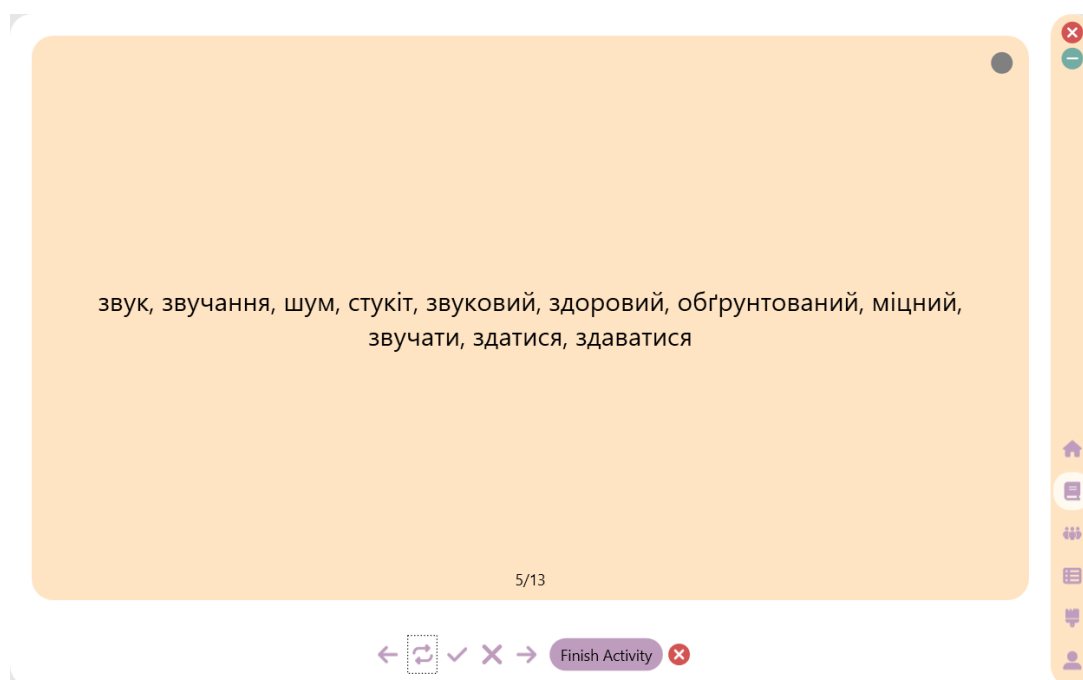


Рисунок 3.15 – Знімок процесу проходження активності під назвою Flashcards

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Після перегляду картки, та, за необхідності, перекладу, користувач має змогу оцінити рівень засвоєння (пам'ятаю/не пам'ятаю), обравши відповідну реакцію це дозволяє системі оновити статистику навчання та скоригувати графік повторення. Таким чином, процес навчання стає адаптивним і враховує індивідуальні особливості користувача, що сприяє кращому запам'ятовуванню. При цьому активність не представляє собою тестування, а, скоріше, перевірку знань у невимушеній формі, що знижує психологічний тиск та дозволяє зосередитись на матеріалі.

Активність реалізована з урахуванням принципів мінімалізму та фокусування уваги користувача, що дозволяє уникнути зайвих відволікань під час навчання. Всі візуальні елементи підібрані таким чином, щоб не перевантажувати інтерфейс, а логіка взаємодії залишалась інтуїтивно зрозумілою навіть для новачків. Завдяки цьому користувач може повністю зосередитись на змісті карток, не відволікаючись на другорядні деталі.

Екран завершеної активності (рис. 3.16) є фінальним етапом у процесі взаємодії користувача з модулем навчання.

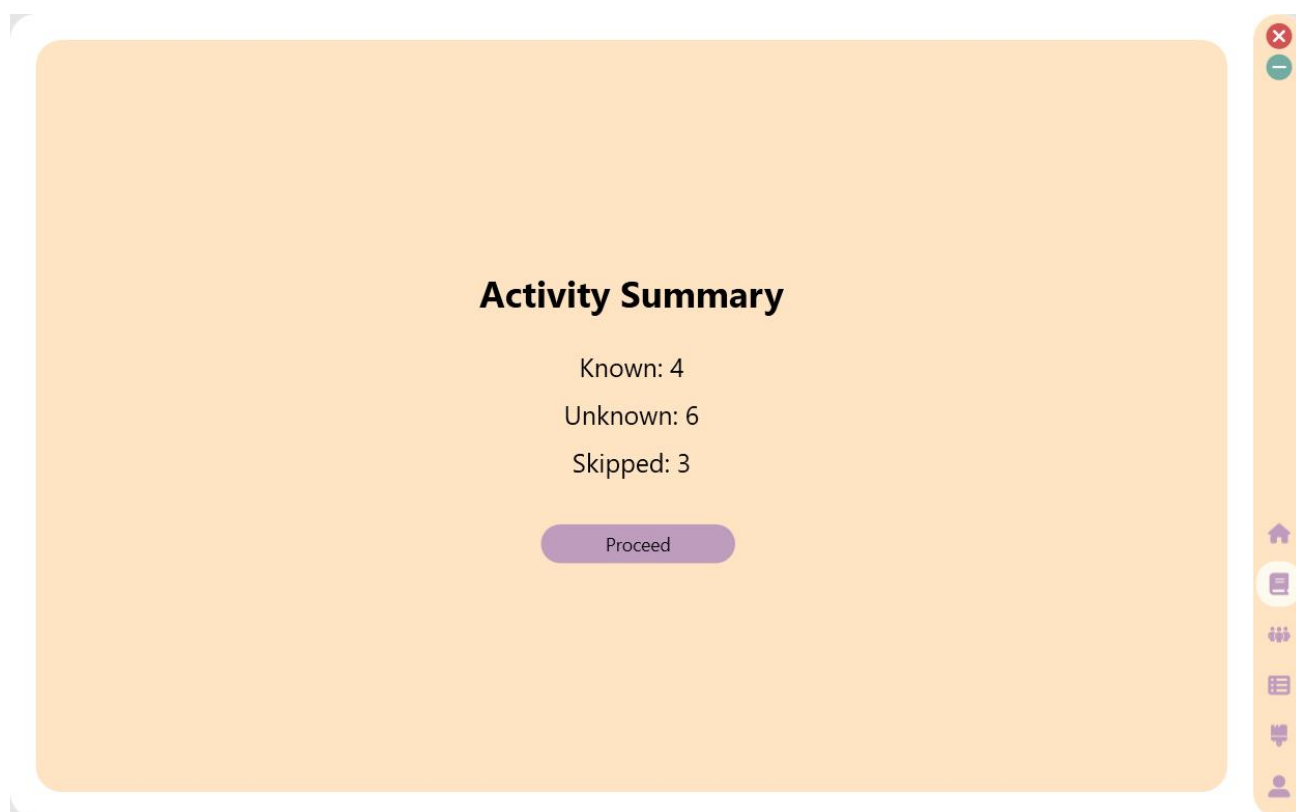


Рисунок 3.16 – Екран завершеної активності

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Його ключове призначення — відобразити короткий звіт про результати щойно завершеної сесії. Зібрані під час проходження активності дані обробляються для оновлення загальної статистики та врахуванню результатів при наступній генерації навчальних матеріалів.

Екран містить агреговану інформацію про кількість слів, які користувач позначив як відомі, невідомі або пропущені. Це дозволяє системі формувати уявлення про поточний рівень знань, а користувачу – оцінити власні досягнення. Уся взаємодія на цьому етапі є максимально мінімалістичною, що відповідає концепції безперешкодного користувацького досвіду.

На рис. 3.17. наведено знімок процесу проходження активності під назвою Synonym Match (знаходження синоніму). Після перегляду слова та варіантів відповідей, користувач має змогу обрати правильний синонім, натиснувши на відповідну.



Рисунок 3.17 – Знімок процесу проходження активності Synonym Match

На рис. 3.18 зображено екран керування темами оформлення застосунка. Цей інтерфейс дозволяє користувачеві персоналізувати зовнішній вигляд програми, створюючи та редагуючи власні теми. Основна концепція екрану полягає в забезпеченні гнучкості та простоти у налаштуванні візуальних

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

параметрів. З лівого боку екрану розташований список усіх доступних тем. Кожен елемент списку включає назву теми, кнопку редагування, видалення та застосування, що дозволяє легко перемикатись між наявними темами. Центральна частина екрану відведена під панель редагування поточної теми. Тут користувач може задати назву теми та налаштувати параметри, кожен яких впливає на різні елементи інтерфейсу. Кожне з цих полів супроводжується палітрою кольорів, що дозволяє зручно обирати бажані відтінки, та полем для введення коду кольору, що корисно для генерації гарної палітри на зовнішніх сервісах та перенесення у застосунок.

Після внесення змін, користувач має можливість зберегти ці зміни, натиснувши кнопку «Save Changes», або створити нову тему з поточними параметрами за допомогою кнопки «Create Theme». Важливою особливістю є миттєве застосування обраної або відредагованої теми до всього застосунка. Це означає, що будь-які зміни, внесені користувачем, відразу ж відображаються на інтерфейсі, дозволяючи візуально оцінити результат і швидко вносити корективи. Такий підхід значно спрощує процес налаштування та робить його більш інтерактивним.

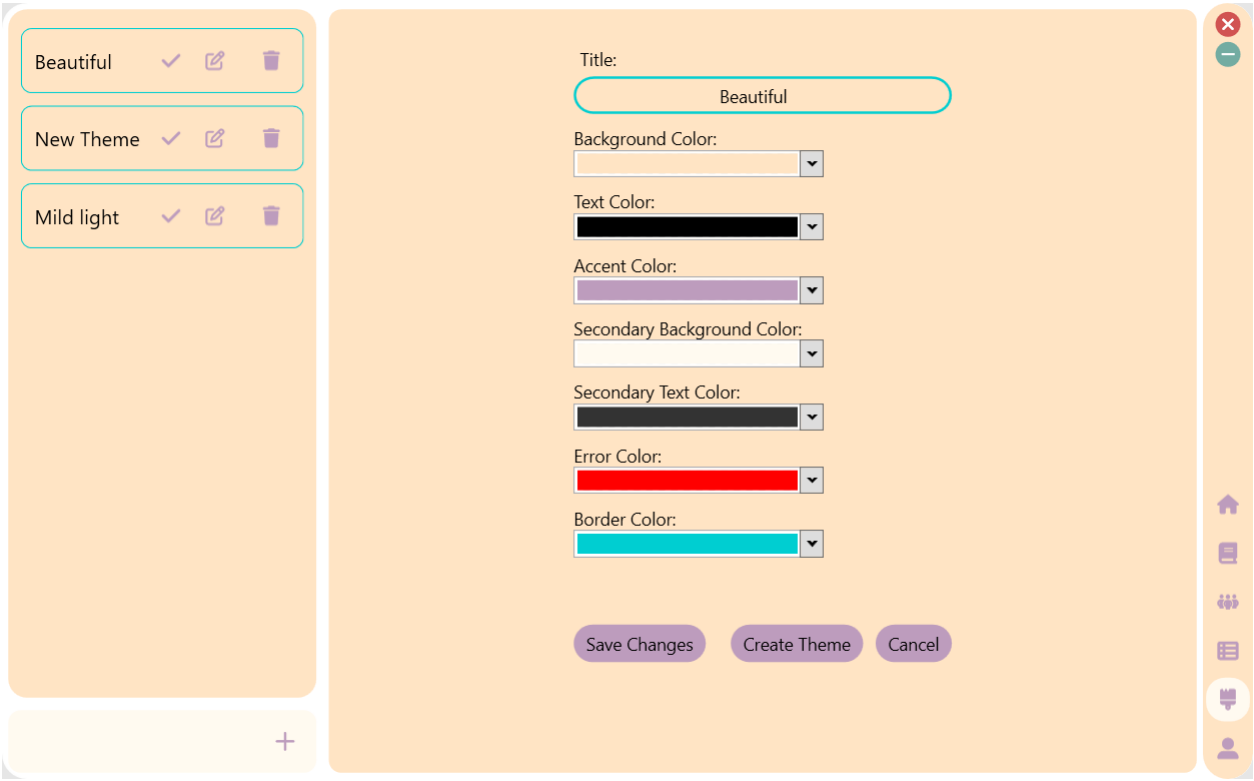


Рисунок 3.18 – Екран налаштувань тем

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

На рис. 3.19 зображено вкладку облікового запису, яка надає користувачу доступ до базових параметрів керування та простої статистики. Зокрема, можна встановити або змінити рівень видимості свого профілю для інших користувачів системи. Крім цього, тут доступна функція зміни плану запам'ятовування, що визначає частоту повторення матеріалу. Також користувач має можливість змінити своє ім'я, вийти з облікового запису або видалити його. Після виходу з облікового запису або його видалення дані з локальної бази даних будуть видалені, а користувача буде направлено на сторінку авторизації.

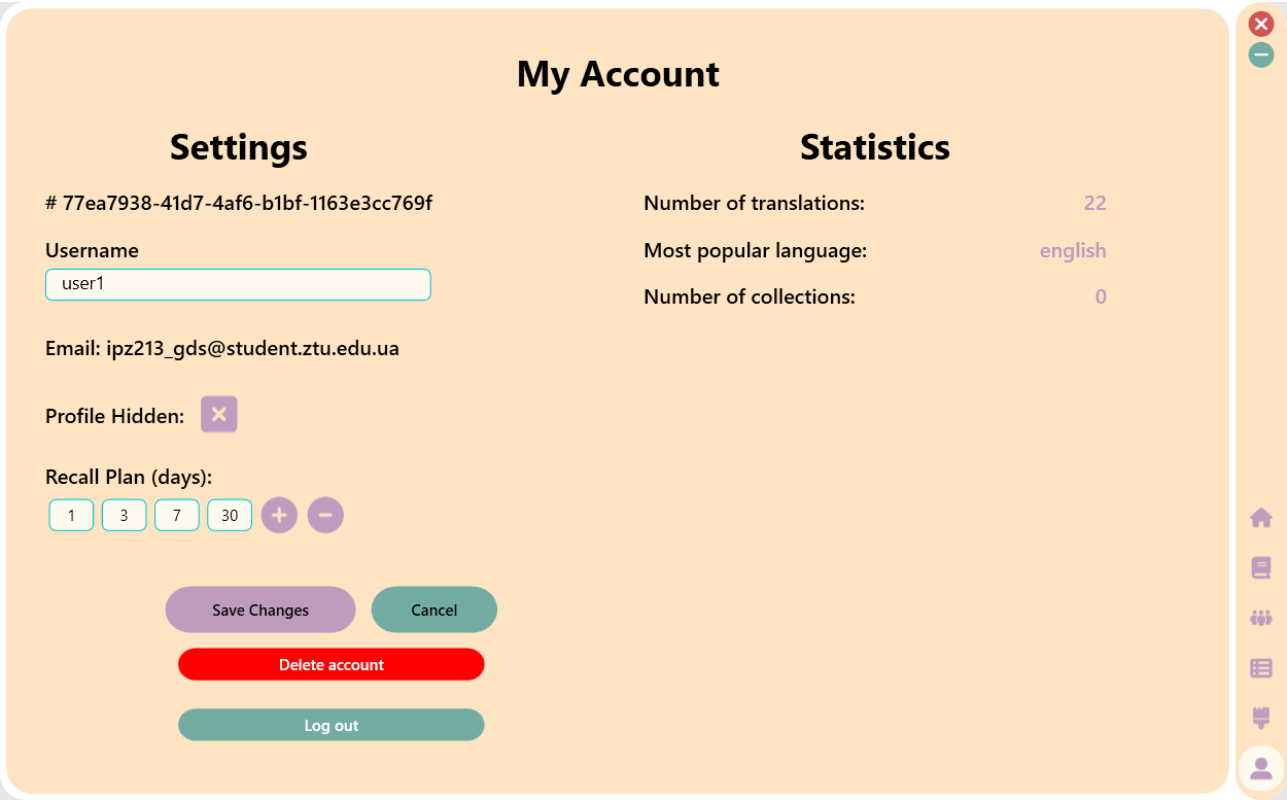


Рисунок 3.19 – Екран профілю користувача

На рис. 3.20 зображено вкладку «Користувачі», де представлено список усіх зареєстрованих користувачів, що зробили свій профіль відкритим. У списку відображається базова інформація про кожного користувача, що полегшує їх ідентифікацію. Цей функціонал виконує важливу роль у системі, оскільки дозволяє користувачам ділитися колекціями.

Username	Email	Collections
user4	user4@example.com	10
user20	user20@example.com	5
user17	user17@example.com	4
user1	ipz213_gds@student.ztu.edu.ua	3
user3	user3@example.com	3
user18	user18@example.com	3
user26	user26@example.com	2
user19	user19@example.com	2
user2	user2@example.com	1
user21	user21@example.com	1
user7	user7@example.com	1
user10	user10@example.com	0
user16	user16@example.com	0
user23	user23@example.com	0
user14	user14@example.com	0
user5	user5@example.com	0
user6	user6@example.com	0
user15	user15@example.com	0
user28	user28@example.com	0

Рисунок 3.20 – Екран користувачів

При натисканні на користувача із списку буде відображено усі його колекції, відкриті для загального доступу (рис. 3.21). Поточний користувач може скопіювати колекцію до власного облікового запису. Вона не буде зв'язана з оригінальною колекцією і може бути відредагована.

Username					Collections
user4	← user4 user4@example.com Collections Learning SQL Basics 2025-06-12 11:00 AI in Practice 2025-06-12 11:00 Database Design Principles 2025-06-12 11:00 Cybersecurity Awareness 2025-06-12 11:00 Machine Learning Fundamentals 2025-06-12 11:00 Advanced SQL Techniques 2025-06-12 11:00 Web Development Resources 2025-06-12 11:00 Data Analysis Tools 2025-06-12 11:00 Cloud Computing Essentials 2025-06-12 11:00 DevOps Practices 2025-06-12 11:00				10
user20					5
user17					4
user1					3
user3					3
user18					3
user26					2
user19					2
user2					1
user21					1
user7					1
user10					0
user16					0
user23					0
user14					0
user5					0
user6					0
user15					0
user28					0

Рисунок 3.21 – Сторінка конкретного користувача

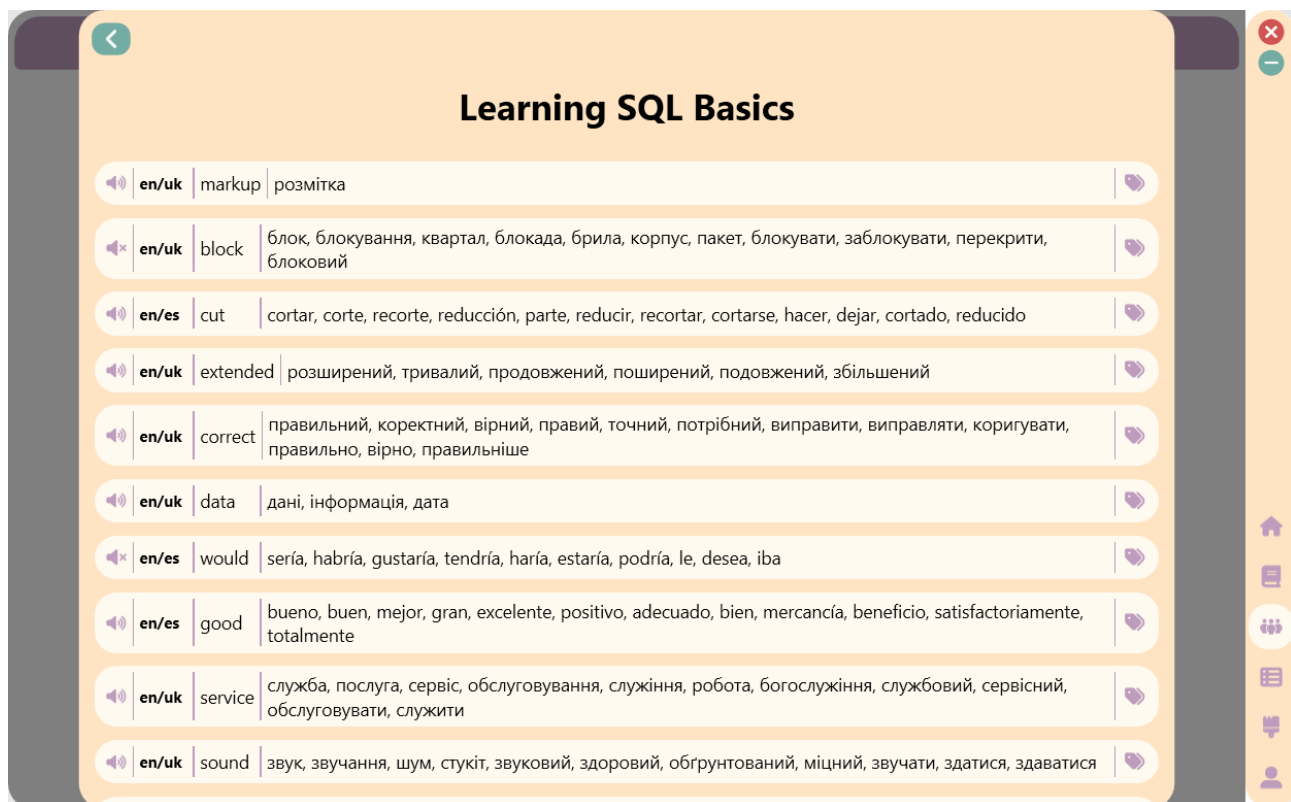


Рисунок 3.22 – Сторінка конкретної колекції

При натисканні на колекцію можна переглянути усі елементи списку (рис. 3.22). Це дозволяє попередньо оцінити зміст перед тим, як зберегти матеріали у свій профіль. Збереження можливе як усього списку цілком, так і окремих перекладів, що робить процес гнучким і адаптивним до потреб користувача.

На рис. 3.23 зображено вкладку колекцій поточного користувача. Можна побачити відмінності і в самих елементах списку. Так, на відміну від списку колекцій інших користувачів, де користувач може скопіювати колекцію, тут доступне видалення колекції. При чому видалити можна як лише саму підбірку, так і взагалі видалити усі асоційовані переклади, що може бути корисним для деяких користувачів.

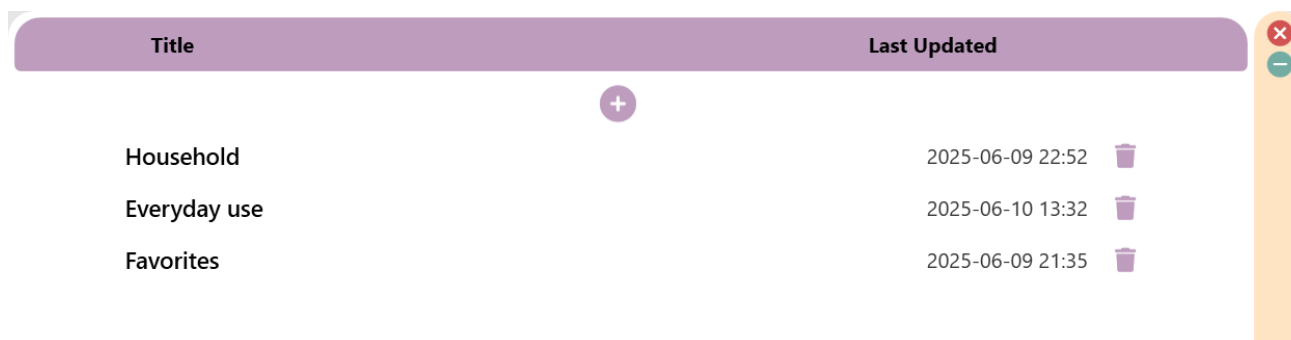


Рисунок 3.23 – Сторінка колекцій поточного користувача

При перегляді власної колекції користувач може видалити переклад взагалі, або прибрати з поточної колекції. На рис. 3.24 зображено вікно колекції, що належить поточному користувачу.



Рисунок 3.24 – Вікно власної колекції

При спробі видалити колекцію у користувача уточнюється як саме це зробити. Для цього відображається відповідне вікно (рис. 3.25).

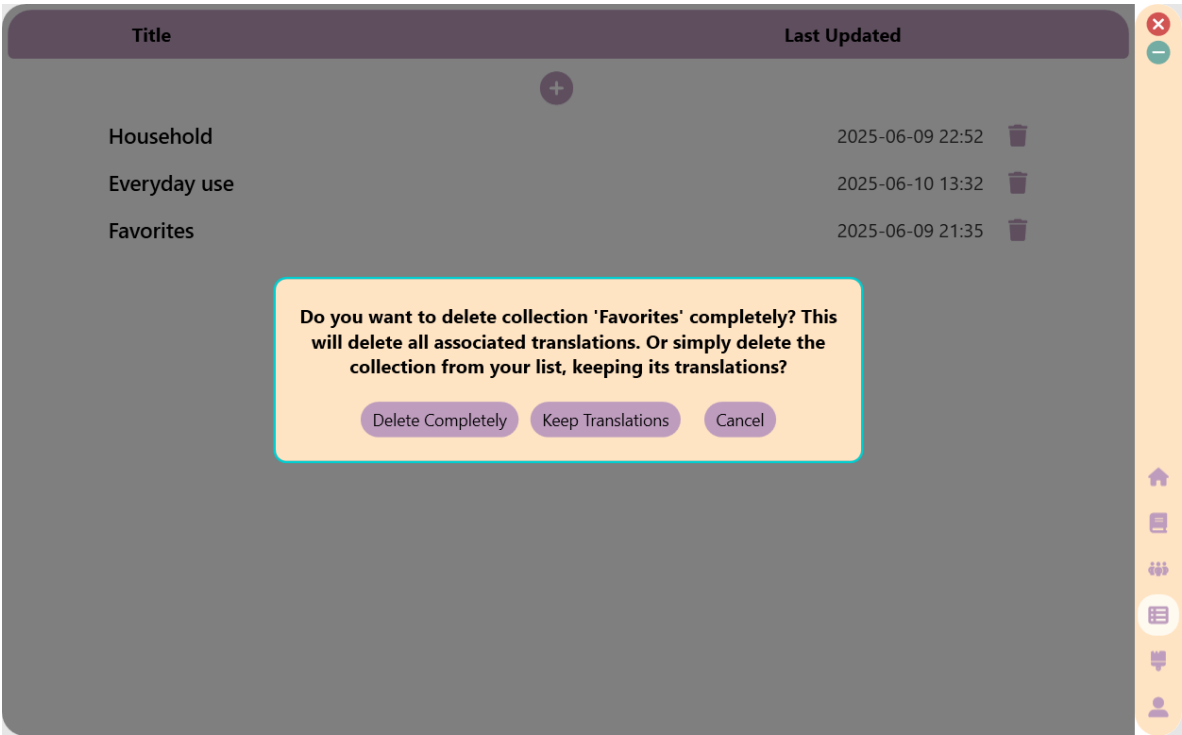


Рисунок 3.25 – Вікно обрання типу видалення та підтвердження

Як можна бачити, на вкладці наявна кнопка створення нової колекції (рис. 3.26). При натисканні на неї з'являється вікно, де можна ввести назву колекції та її видимість для інших користувачів системи.

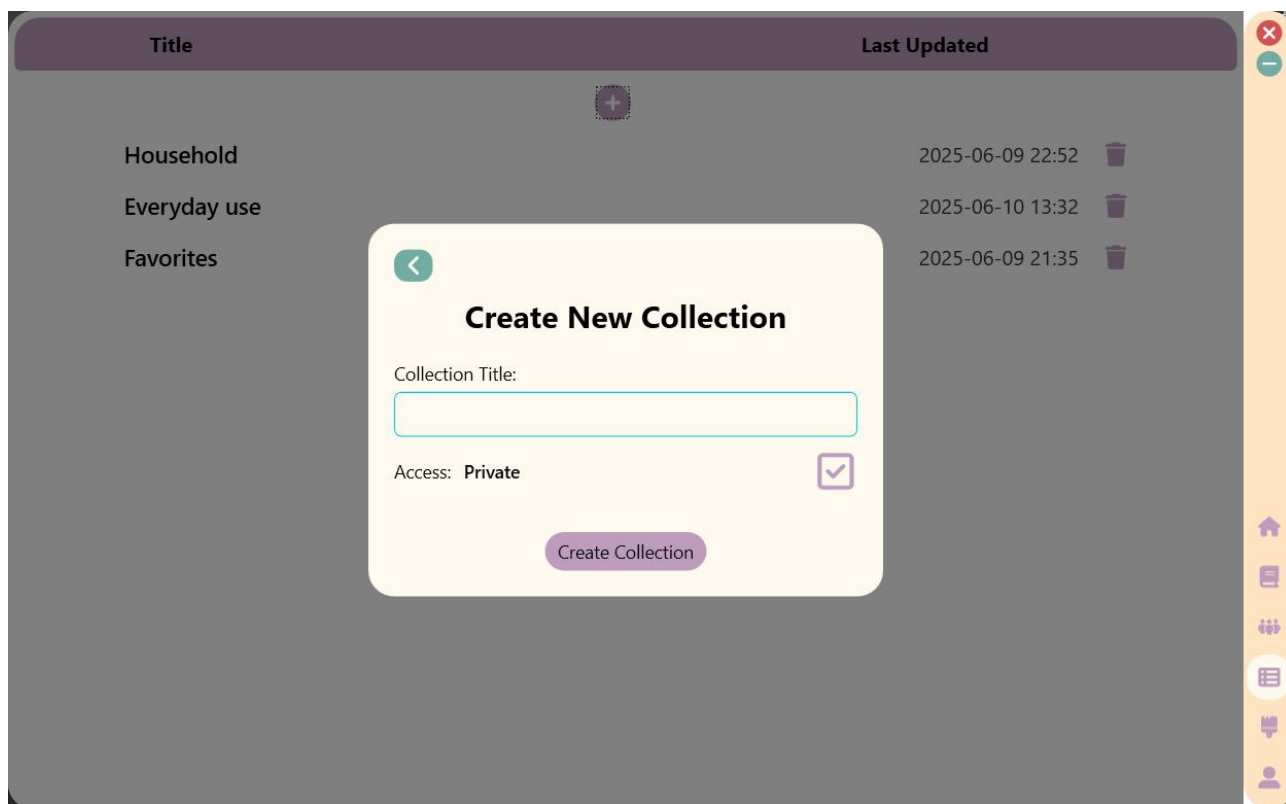


Рисунок 3.26 – Вікно створення нової колекції

Слід зазначити, що усі списки в системі реалізовані з використанням підходу *infinite scroll*, що дозволяє динамічно підвантажувати наступні порції даних у міру прокрутки сторінки. Це не лише покращує зручність користування, зменшуючи потребу у ручному перемиканні сторінок, але й оптимізує навантаження на систему, завантажуючи лише необхідні на даний момент записи, що особливо ефективно при роботі з великими наборами даних.

3.4 Тестування системи

Тестування є критично важливим етапом у життєвому циклі розробки ПЗ, особливо для розподілених систем, адже дозволяє виявляти проблеми на ранніх етапах, мінімізувати ризики та забезпечити відповідність системи заданим вимогам. Відносно велика кількість компонентів, що взаємодіють між собою, зароджують потенційні складності розробки злагодженого та надійного

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

програмного забезпечення, при цьому складність тестування зростає ще більш вагомо. Необхідно враховувати як система працює у комплексі, для чого можуть знадобитись такі ж комплексні рішення.

Проаналізувавши архітектуру та використані підходи під час розробки було виявлено найважливіші та найбільш складні частини, що потребують серйозного тестування. Так основну увагу було вирішено приділити тестуванню API, як локального, так і віддаленого, у комплексі, адже саме на передачі даних з застосуванням http протоколу і базується система, тож необхідно упевнитись у правильності роботи, надійності та ефективності роботи серверів. Для перевірки функціональності та контрактів було використано комбінацію ручного тестування та інструментів автоматизації. Такими інструментами стали Swagger та Postman – рішення, які гарно зарекомендували і показують себе у середовищі сучасного тестування та розробки значно спрощуючи роботу з API.

Так, Swagger UI виступає як незамінний інструмент для документування та інтерактивної взаємодії з RESTful API. Він надає зручний веб-інтерфейс, що дозволяє швидко переглядати доступні кінцеві точки (endpoints), їх параметри та очікувані результати.

На рис. 3.27 представлено загальний вигляд інтерфейсу Swagger UI, який відображає перелік доступних ендпоінтів для локального сервера.

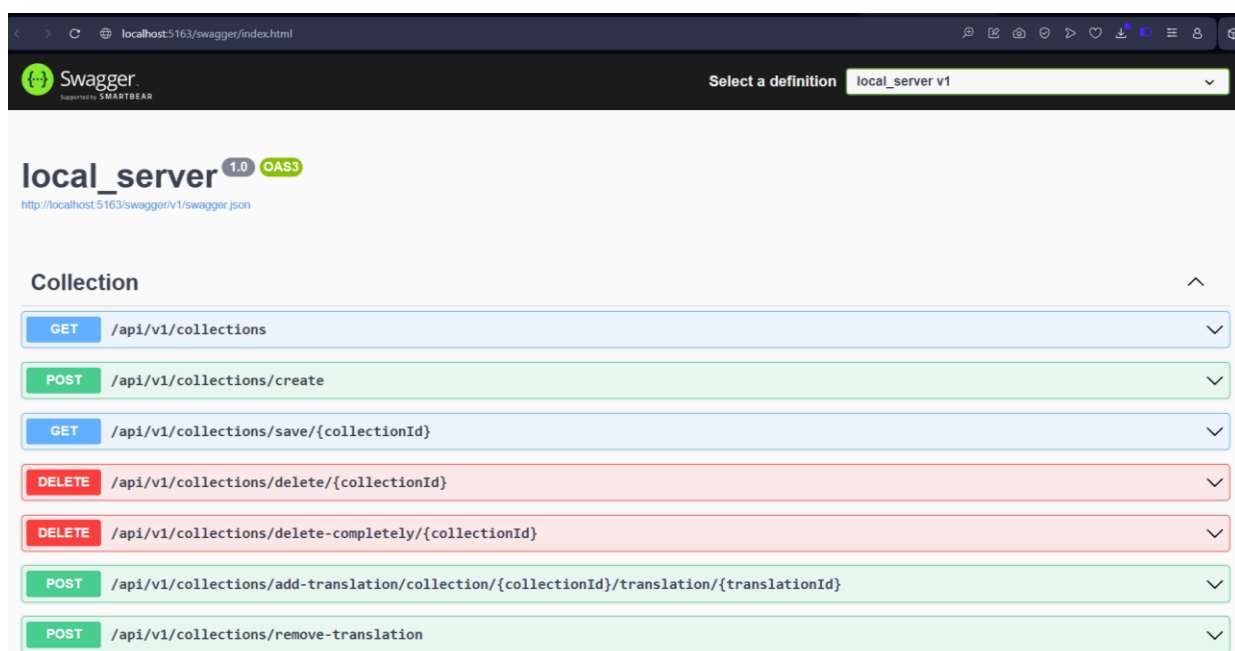


Рисунок 3.27 – Загальний інтерфейс Swagger

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

В рамках розробки даної системи Swagger використовувався на початкових етапах роботи для оперативного ручного тестування, дозволяючи відправляти запити безпосередньо з інтерфейсу, що сприяло швидкому виявленню та виправленню логічних та синтаксичних помилок. Окрім цього є можливість перегляду використовуваних типів даних, що буває корисно, коли є необхідність швидко перевірити правильність очікуваних даних.

На рис. 3.28 зображено інтерфейс, що дає розуміння про використовувані типи даних, якими оперує сервер.

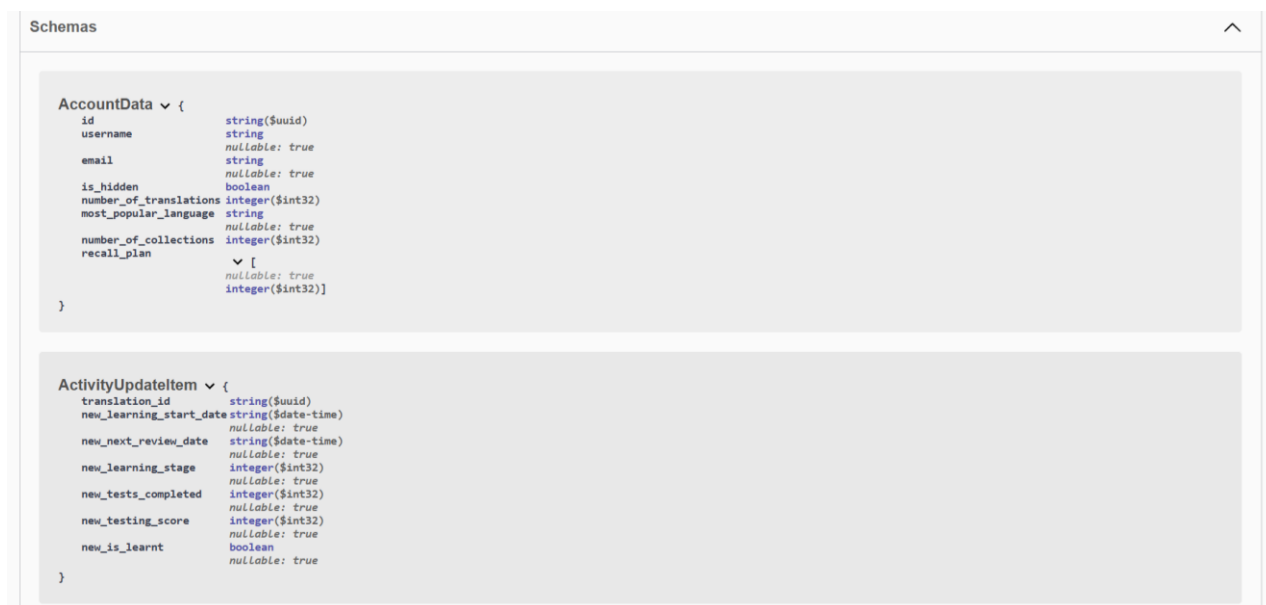


Рисунок 3.28 – Типи використовуваних даних

Окрім базового функціоналу для тестування, який надає Swagger, він також дає можливість згенерувати документацію. Це дозволило швидко створити об'ємні, комплексні набори автоматизованих тестів у Postman. Імпортувавши створену документацію Postman реалізує колекцію, яку вже можна використовувати як основу для побудови більш складних операцій.

Postman – це потужний інструмент, що використовується для систематизованого та автоматизованого підходу до тестування. Він дозволяє не лише відправляти запити, але і робити це динамічно, формуючи запити у польоті, групувати їх, писати складні обробники відповідей та керувати даними.

На рис. 3.29 зображено сторінку конкретного ендпоінту.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

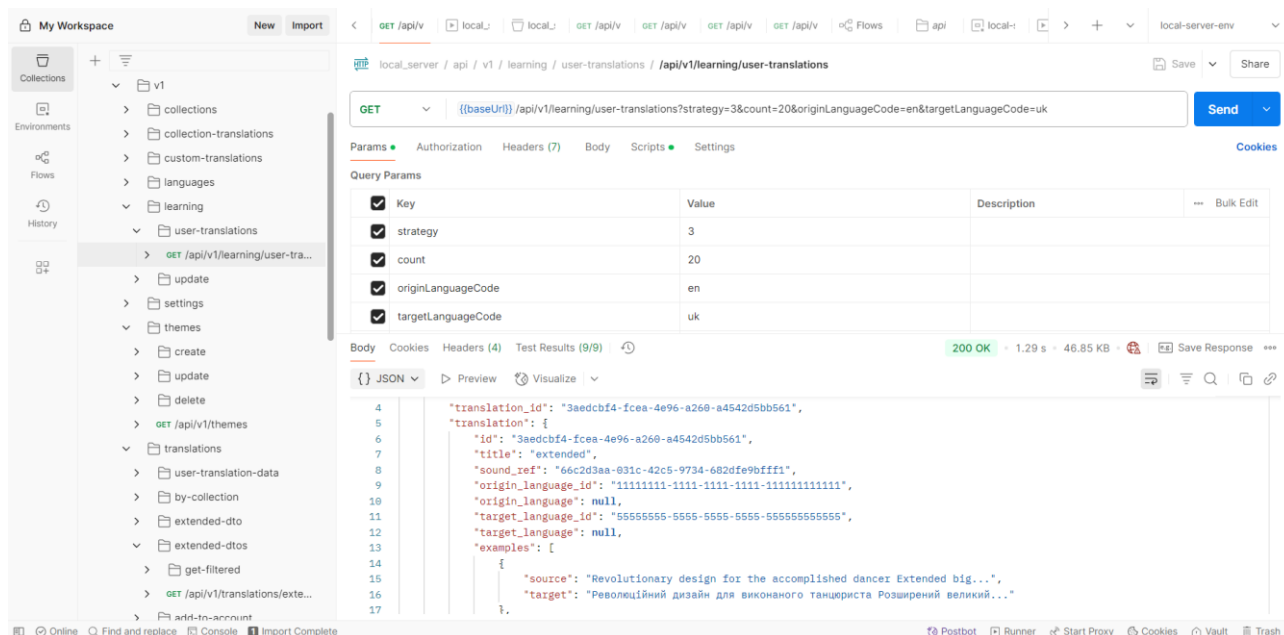


Рисунок 3.29 – Сторінка ендпоінту, вкладки Params/Body

На цьому рисунку можна побачити безпосередньо шлях до ендпоінту, що тестується, параметри запиту, які надсилаються а також сама відповідь у форматі JSON. Окрім цього інтерфейс відображає колекцію у лівому боці екрану, що представляє собою структуру папок, відповідно до згенерованої документації Swagger.

На рис. 3.30 наведено вкладки Scripts та Tests. На вкладці Scripts є можливість додати автоматизацію типів Pre-request (виконується перед відправкою запиту) та Post-response (виконується після відправки запиту).

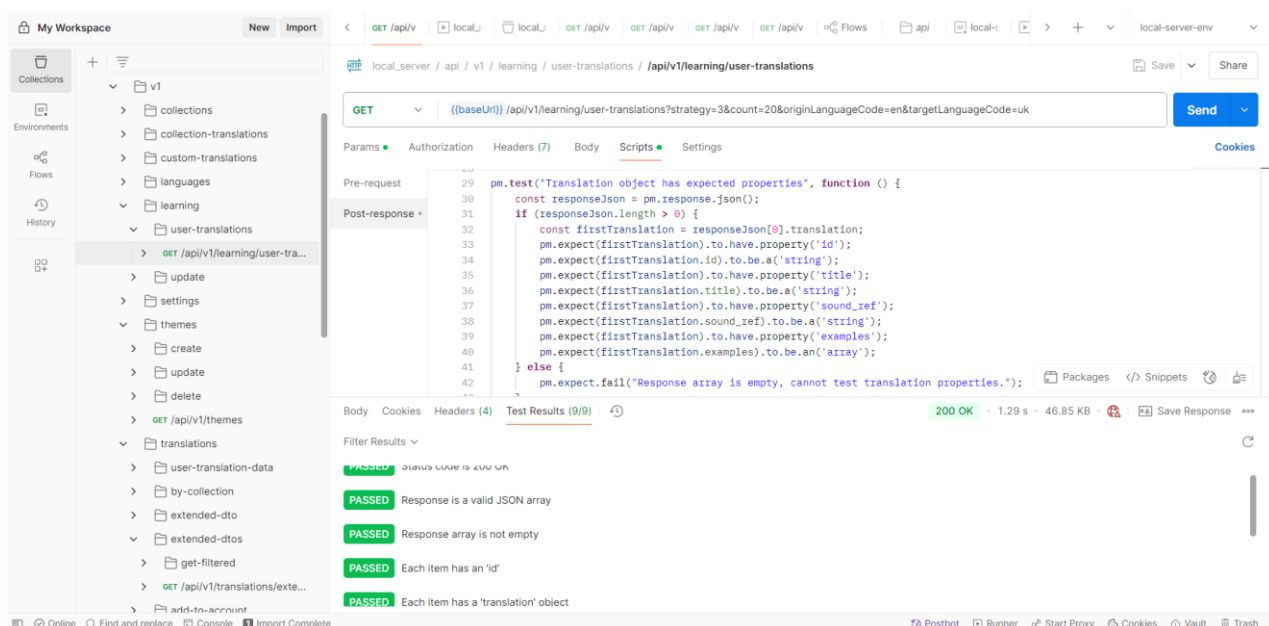


Рисунок 3.30 – Сторінка тестування ендпоінту, вкладки Scripts/Tests

		Гнатюк Д.С.				ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.					65
Змн.	Арк.	№ докум.	Підпис	Дата			

При тестуванні були написані тести, які дозволили перевірити коректність роботи API. Так на рисунку можна бачити пройдені тести для ендпоінту GET learning/user-translations.

На рис. 3.31 приведено розширені можливості Postman, що виходять за рамки простого виконання окремих HTTP-запитів.

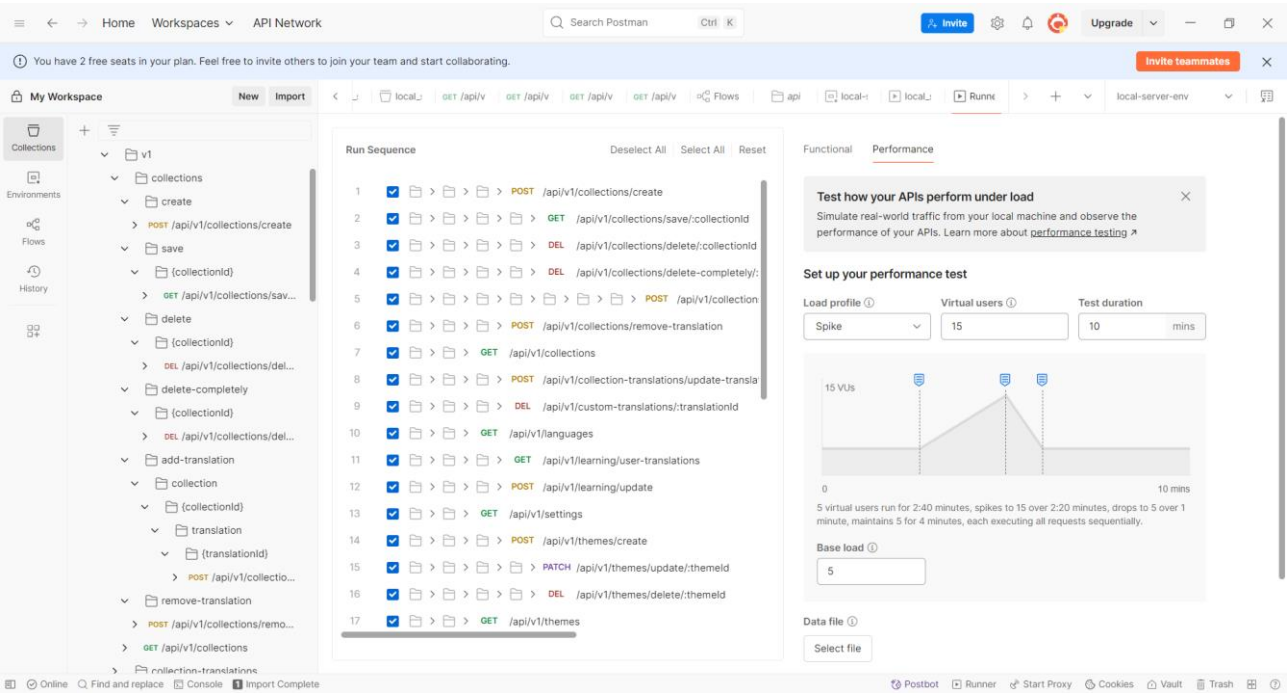


Рисунок 3.31 – Сторінка комплексного тестування

У центральній частині рисунка представлена вкладка послідовного виконання, яка є ключовим елементом для автоматизації інтеграційних та функціональних тестів. Цей функціонал Postman дозволяє створювати ланцюжки запитів, де результат одного запиту (наприклад, токен автентифікації або ідентифікатор створеного ресурсу) може бути використаний як вхідні дані для наступного запиту. Це дає змогу реалізувати складні тестові сценарії, які імітують реальні послідовності дій користувача, наприклад реєстрація нового користувача, далі підтвердження електронної пошти (на основі даних з попереднього запиту) та виконання подальших тестових запитів як автентифікований користувач, використовуючи отриманий токен. Такий підхід значно підвищує ефективність тестування, дозволяючи автоматично перевіряти складні бізнес-процеси та взаємодії між різними ендпоінтами API.

Праворуч на рисунку показано вкладку «Продуктивність», що ілюструє потужні можливості Postman для навантажувального тестування. Цей розділ дозволяє детально налаштувати симуляцію навантаження на API: надає вибір між різними профілями, такими як імпульсне навантаження, коли кількість користувачів різко зростає або постійне базове навантаження, дозволяє визначити кількість одночасних користувачів, що симулюють запити до системи.

Висновки до третього розділу

У третьому розділі було детально розглянуто порядок встановлення та налаштування системи, наведено відповідні діаграми, що ілюструють структурні рішення та спосіб розгортання компонентів. Також описано користувацький інтерфейс розроблюваної системи, зокрема інтерфейс розширення для браузера та десктопного застосунку, із зазначенням особливостей реалізації для кожної платформи. Було проаналізовано основні елементи взаємодії з користувачем, їхнє розташування, призначення та роль у забезпеченні інтуїтивної та зручної роботи з функціоналом системи. Окрему увагу було приділено уніфікованості дизайну між платформами, що сприяє цілісному користувацькому досвіду, а також реалізації механізмів синхронізації та офлайн-доступу без ускладнення інтерфейсу. Окрім цього, було описано проведені тестування, а також зроблено короткий огляд використаних для цього інструментів і бібліотек.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено програмну систему, що поєднує інструменти перекладу та вивчення мов у єдиному середовищі. На першому етапі було сформульовано основні задачі системи, сформовано перелік вимог до функціоналу, орієнтуючись на актуальні потреби користувачів, які щоденно працюють з іншомовними матеріалами та проведено аналіз аналогів. Результати дослідження лягли в основу архітектурних рішень, а також вибору інструментів реалізації.

У другому розділі було визначено акторів та варіанти використання системи. Описано та обґрунтовано структуру локальної та віддаленої баз даних, методи синхронізації та збереження даних. Значну увагу приділено формуванню алгоритмів роботи системи, побудованих з урахуванням принципів адаптивного навчання та оптимізації взаємодії з клієнтом.

У третьому розділі детально розглянуто порядок встановлення та налаштування системи, описано інтерфейс користувача, як для браузерного розширення, так і для десктопного застосунку. Було визначено основні компоненти інтерфейсу, способи взаємодії з ними та реалізовані механізми інтеграції з перекладеним матеріалом і навчальними активностями. Враховано вимоги до швидкої доступності функціоналу, мінімізації відволікань і забезпечення офлайн-роботи. Особливу увагу під час реалізації було приділено забезпеченню інтуїтивної навігації та зменшенню когнітивного навантаження на користувача.

Загалом у межах кваліфікаційної роботи було реалізовано систему, здатну адаптуватися до потреб користувача та спрощувати процес вивчення мов без відриву від звичного робочого середовища. Система об'єднує зручність, функціональність і гнучкість, що робить її потенційно корисною для широкого кола користувачів та відкриває перспективи подальшого розвитку. Усі поставлені на початку роботи задачі були успішно виконані, а отримані результати підтверджують ефективність обраного підходу.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гнатюк Д.С., Локтікова Т.М., Лисогор Ю.І. Міждисциплінарний підхід до розробки освітницького програмного забезпечення. Тези доповідей VII Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», 02-03 грудня 2024 року. Житомир: «Житомирська політехніка», 2024. С. 41-42.
2. Гнатюк Д.С., Локтікова Т.М. Система перекладу з елементами навчання. Тези доповідей Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених, присвяченої Дню науки, 12-17 травня 2025 року. Житомир: «Житомирська політехніка», 2025. С. 58.
3. AnkiWeb [Електронний ресурс] - Режим доступу до ресурсу: <https://ankiweb.net/about>.
4. Strategy [Електронний ресурс] - Режим доступу до ресурсу: <https://refactoring.guru/design-patterns/strategy>.
5. ASP.NET Core web API documentation with Swagger/OpenAPI [Електронний ресурс] - Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-8.0>.
6. ASP.NET documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0>.
7. Azure [Електронний ресурс] - Режим доступу до ресурсу: <https://portal.azure.com>.
8. Azure documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/azure>.
9. C# language documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/csharp/>.
10. Common web application architectures [Електронний ресурс] - Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application->

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

architectures.

- 11.Context.Reverso [Електронний ресурс] - Режим доступу до ресурсу:
<https://context.reverso.net/>.
- 12.CRXJS Vite Plugin [Електронний ресурс] - Режим доступу до ресурсу:
<https://crxjs.dev/vite-plugin>.
- 13.Entity Framework Core [Електронний ресурс] - Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/ef/core/>.
- 14.Font Awesome [Електронний ресурс] - Режим доступу до ресурсу:
<https://fontawesome.com/>.
- 15.Git ztu [Електронний ресурс] - Режим доступу до ресурсу:
<https://git.ztu.edu.ua>.
- 16.JavaScript [Електронний ресурс] - Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
- 17.LingQ [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.lingq.com/en/>.
- 18.MailKit [Електронний ресурс] - Режим доступу до ресурсу:
<https://mimekit.net/docs/html/Introduction.htm>.
- 19.Openrouter [Електронний ресурс] - Режим доступу до ресурсу:
<https://openrouter.ai/>.
- 20.Postman docs [Електронний ресурс] - Режим доступу до ресурсу:
<https://learning.postman.com/docs/introduction/overview/>.
- 21.React [Електронний ресурс] - Режим доступу до ресурсу: <https://react.dev/>.
- 22.Reverso Synonyms [Електронний ресурс] - Режим доступу до ресурсу:
<https://synonyms.reverso.net/synonym/>.
- 23.ScrapingBee Docs [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.scrapingbee.com/documentation/>.
- 24.SQL Server technical documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver17>.
- 25.Swagger documentation [Електронний ресурс] - Режим доступу до ресурсу:

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

<https://swagger.io/docs/>.

26.TypeScript Documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>.

27.Vite [Електронний ресурс] - Режим доступу до ресурсу: <https://vite.dev/>.

28.Welcome to Extensions! [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.chrome.com/docs/extensions?hl=en>.

29.WPF [Електронний ресурс] - Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/>.

30.Xceed Toolkit [Електронний ресурс] - Режим доступу до ресурсу: <https://xceed.com/documentation/xceed-toolkit-plus-for-wpf/index.html>.

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

ДОДАТКИ

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Файл CollectionService віддаленого сервісу, метод SaveAsync:

```

public async Task<SavedCollectionDataDto> SaveAsync(Guid userId, Guid
collectionIdToCopy)
{
    var originalCollection = await _context.Collections
        .Include(c => c.CollectionTranslations)
        .ThenInclude(ct => ct.Translation)
        .AsNoTracking()
        .FirstOrDefaultAsync(c => c.Id == collectionIdToCopy);
    if (originalCollection == null || originalCollection.IsPrivate)
    {
        throw new KeyNotFoundException($"Collection with ID
{collectionIdToCopy} not found or is private.");
    }
    var newCollection = new Collection
    {
        Id = Guid.NewGuid(),
        Title = originalCollection.Title,
        IsPrivate = false,
        UserId = userId,
    };
    var newCollectionTranslations = new List<CollectionTranslation>();
    var translationsToClient = new List<Translation>();
    var userTranslationsToClient = new List<UserTranslation>();
    var userOwnedTranslationIdsOnServer = await _context.UserTranslations
        .Where(ut => ut.UserId == userId)
        .Select(ut => ut.TranslationId)
        .ToListAsync();
    var userOwnedTranslationIdSet = new
HashSet<Guid>(userOwnedTranslationIdsOnServer);
    foreach (var originalCt in originalCollection.CollectionTranslations)
    {
        newCollectionTranslations.Add(new CollectionTranslation
        {
            CollectionId = newCollection.Id,

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        TranslationId = originalCt.TranslationId,
        LastUpdatedAt = DateTime.UtcNow});
if (!userOwnedTranslationIdSet.Contains(originalCt.TranslationId))
{translationsToClient.Add(originalCt.Translation);
    userTranslationsToClient.Add(new UserTranslation
    {UserId = userId,
        TranslationId = originalCt.TranslationId});}
}
await _context.Collections.AddAsync(newCollection);
await
_context.CollectionTranslations.AddRangeAsync(newCollectionTranslations);
await _context.SaveChangesAsync();
return new SavedCollectionDataDto
{NewCollection = newCollection,
    NewCollectionTranslations = newCollectionTranslations,
    NewTranslations = translationsToClient,
    NewUserTranslations = userTranslationsToClient};}

```

Додаток Б

Файл CollectionService локального сервісу, метод SaveCollectionAsync:

```

public async Task<Collection> SaveCollectionAsync(Guid collectionId)
{
    await HttpTools.SetAuthorizationHeader(_httpClient, _appStateService);
    string url =
PathManager.GetControllerBaseUrl($"collections/save/{collectionId}");
    using var response = await _httpClient.GetAsync(url,
HttpCompletionOption.ResponseHeadersRead);
    response.EnsureSuccessStatusCode();
    using var responseStream = await response.Content.ReadAsStreamAsync();
    using var gzipStream = new GZipStream(responseStream,

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

CompressionMode.Decompress);

    var savedData = await
JsonSerializer.DeserializeAsync<SavedCollectionDataDto>(gzipStream, new
JsonSerializerOptions { PropertyNamingPolicy = JsonNamingPolicy.CamelCase });
    if (savedData == null)
        {throw new InvalidOperationException("Failed to deserialize saved collection
data.");}

    var existingTranslationIds = await _context.Translations
        .Where(t => savedData.NewTranslations.Select(nt => nt.Id).Contains(t.Id))
        .Select(t => t.Id)
        .ToListAsync();

    var newTranslationsToAdd = new List<Translation>();
    foreach (var translation in savedData.NewTranslations)
    {if
(!existingTranslationIds.Contains(translation.Id)){newTranslationsToAdd.Add(transl
ation); }

        else{_context.Entry(translation).State = EntityState.Unchanged; }}
    await _context.Translations.AddRangeAsync(newTranslationsToAdd);

    var existingUserTranslationIds = await _context.UserTranslations
        .Where(ut => savedData.NewUserTranslations.Select(nut =>
nut.Id).Contains(ut.Id))
        .Select(ut => ut.Id)
        .ToListAsync();

    var newUserTranslationsToAdd = new List<UserTranslation>();
    foreach (var userTranslation in savedData.NewUserTranslations)
    {if (!existingUserTranslationIds.Contains(userTranslation.Id))
        {newUserTranslationsToAdd.Add(userTranslation); }
        else{_context.Entry(userTranslation).State = EntityState.Unchanged;
        }}

    await _context.UserTranslations.AddRangeAsync(newUserTranslationsToAdd);

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

var existingCollectionTranslationIds = await _context.CollectionTranslations
    .Where(ct => savedData.NewCollectionTranslations.Select(nct =>
nct.Id).Contains(ct.Id))
    .Select(ct => ct.Id)
    .ToListAsync();
var newCollectionTranslationsToAdd = new List<CollectionTranslation>();
foreach (var collectionTranslation in savedData.NewCollectionTranslations)
{
    if (!existingCollectionTranslationIds.Contains(collectionTranslation.Id))
    {newCollectionTranslationsToAdd.Add(collectionTranslation); }
    else{_context.Entry(collectionTranslation).State = EntityState.Unchanged;}}
await _context.CollectionTranslations
.AddRangeAsync(newCollectionTranslationsToAdd);
savedData.NewCollection.CollectionTranslations = null;
await _context.Collections.AddAsync(savedData.NewCollection);
await _context.SaveChangesAsync();
return savedData.NewCollection;
}

```

Додаток В

Файл TranslationService локального сервісу, метод
GetTranslationInternalAsync:

```

private async Task<UserTranslation> GetTranslationInternalAsync(
    string originName,
    string originLanguageCode,
    string targetLanguageCode,
    Func<IQueryable<UserTranslation>, IQueryable<UserTranslation>>
includeQuery)
{var query = _context.UserTranslations
    .Where(ut =>
        ut.Translation.Title == originName &&

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        ut.Translation.OriginLanguage.Code == originLanguageCode &&
        ut.Translation.TargetLanguage.Code == targetLanguageCode);
var userTranslation = await includeQuery(query).FirstOrDefaultAsync();
if (userTranslation != null) {return userTranslation; }
return await _exec.ExecuteAsync(
    onlineAction: async () =>
    {await HttpTools.SetAuthorizationHeader(_httpClient, _appStateService);
    string url = PathManager.GetControllerBaseUrl("translations") +
        $"?originName={Uri.EscapeDataString(originName)}" +
        "&originLanguageCode={originLanguageCode}" +
        "&targetLanguageCode={targetLanguageCode}";
    var response = await _httpClient.GetAsync(url);
    if (!response.IsSuccessStatusCode)
    {var error = await response.Content.ReadAsStringAsync();
        throw new Exception($"API call failed: {response.StatusCode} -
{error}");}
    var json = await response.Content.ReadAsStringAsync();
    UserTranslation utToSave =
    JsonSerializer.Deserialize<UserTranslation>(json);
    await _context.UserTranslations.AddAsync(utToSave);
    _context.Attach(utToSave.Translation.OriginLanguage).State =
    EntityState.Unchanged;
    _context.Attach(utToSave.Translation.TargetLanguage).State =
    EntityState.Unchanged;
    await _context.SaveChangesAsync();
    if (utToSave.Translation != null && utToSave.Translation.SoundRef !=
    Guid.Empty && utToSave.Translation.SoundRef != null)
    {await
    _audioFilesService.DownloadAudioByGuidAsync(utToSave.Translation.SoundRef.
    Value); }

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return utToSave; },
offlineAction: async () =>
{var ol = await _languageService.GetByCode(originLanguageCode);
    var tl = await _languageService.GetByCode(targetLanguageCode);
    TempWord tw = new TempWord();
    tw.Title = originName;
    tw.OriginLanguageId = ol.Id;
    tw.TargetLanguageId = tl.Id;
    await _context.TempWords.AddAsync(tw);
    await _context.SaveChangesAsync();
    throw new Exception("No connection! Word added to queue.");
},
offlineAllowed: true,
offlineErrorMessage: "No connection! Word added to queue.");
}

```

Додаток Г

Файл DataContext розширення для браузеру:

```

import React, { createContext, Dispatch, SetStateAction, useContext, useEffect,
useState } from 'react';
import { Collection } from '../Types/Collection.ts';
import { getUserCollections } from '../Data/collectionsApi.ts';
import { Language } from '../Types/Language.ts';
import { getLanguages } from '../Data/languagesApi.ts';
import { LanguagePreferences } from '../Types/LanguagePreferences.ts';

const COLLECTIONS_STORAGE_KEY = 'userCollections';
const LANGUAGES_STORAGE_KEY = 'userLanguages';
const LANGUAGE_PREFERENCES_STORAGE_KEY = 'languagePreferences';

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

interface DataContextProps {
  collections: Collection[];
  setCollections: Dispatch<SetStateAction<Collection[]>>;
  languages: Language[];
  setLanguages: Dispatch<SetStateAction<Language[]>>;
  languagePreferences: LanguagePreferences | null;
  setLanguagePreferences: Dispatch<SetStateAction<LanguagePreferences | null>>;
}

const DataContext = createContext<DataContextProps | undefined>(undefined);

export default function DataProvider({ children }: { children: React.ReactNode }) {
  const [collections, setCollectionsState] = useState<Collection[]>([]);
  const [languages, setLanguagesState] = useState<Language[]>([]);
  const [languagePreferences, setLanguagePreferencesState] =
  useState<LanguagePreferences | null>(null);

  useEffect(() => {
    getUserCollections()
      .then(data => {
        setCollectionsState(data);
        localStorage.setItem(COLLECTIONS_STORAGE_KEY,
        JSON.stringify(data));
      })
      .catch(console.error);
  }, []);

  useEffect(() => {
    const stored =
    localStorage.getItem(LANGUAGE_PREFERENCES_STORAGE_KEY);
    if (stored) {try {setLanguagePreferencesState(JSON.parse(stored));} catch
    {localStorage.removeItem(LANGUAGE_PREFERENCES_STORAGE_KEY);}}
    getLanguages()
  });
}

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        .then(data => {
            setLanguagesState(data);
            localStorage.setItem(LANGUAGES_STORAGE_KEY,
JSON.stringify(data));
            if (!stored && data.length >= 2) {
                const defaults = { origin: data[0], target: data[1] };
                setLanguagePreferencesState(defaults);
                localStorage.setItem(LANGUAGE_PREFERENCES_STORAGE_KEY,
JSON.stringify(defaults));
            }
        })
        .catch(console.error);
    }, []);

const setCollections: Dispatch<SetStateAction<Collection[]>> = v => {
    const upd = typeof v === 'function' ? v(collections) : v;
    setCollectionsState(upd);
    localStorage.setItem(COLLECTIONS_STORAGE_KEY, JSON.stringify(upd));
};

const setLanguages: Dispatch<SetStateAction<Language[]>> = v => {
    const upd = typeof v === 'function' ? v(languages) : v;
    setLanguagesState(upd);
    localStorage.setItem(LANGUAGES_STORAGE_KEY, JSON.stringify(upd));
};

const setLanguagePreferences: Dispatch<SetStateAction<LanguagePreferences |
null>> = v => {
    const upd = typeof v === 'function' ? v(languagePreferences) : v;
    setLanguagePreferencesState(upd);
    if (upd) {
        localStorage.setItem(LANGUAGE_PREFERENCES_STORAGE_KEY,
JSON.stringify(upd));
    } else

```

		Гнатюк Д.С.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

{localStorage.removeItem(LANGUAGE_PREFERENCES_STORAGE_KEY);}};

return (
  <DataContext.Provider
    value={{
      collections,
      setCollections,
      languages,
      setLanguages,
      languagePreferences,
      setLanguagePreferences
    }}
  >
    {children}
  </DataContext.Provider>
);
}

export function useDataContext(): DataContextProps {
  const ctx = useContext(DataContext);
  if (!ctx) throw new Error('useDataContext must be used within DataProvider');
  return ctx;
}

```

Файл EmailService віддаленого серверу:

```

using database_api.Utills;
using Microsoft.Extensions.Options;
using MailKit.Security;
using MimeKit;
using MailKit.Net.Smtp;
namespace database_api.Services
{

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

public class EmailService
{
    private readonly GoogleEnv _config;

    public EmailService(IOptions<GoogleEnv> config) {_config = config.Value; }

    public async Task<bool> SendVerificationCodeAsync(string recipientEmail,
string verificationCode)
    {
        var email = new MimeMessage();
        email.From.Add(MailboxAddress.Parse(_config.ServerEmail));
        email.To.Add(MailboxAddress.Parse(recipientEmail));
        email.Subject = "GdsBachelor email Verification Code";
        email.Body = new TextPart("plain"){Text = $"Your verification code is:
{verificationCode}"};

        try
        {
            using var smtp = new SmtpClient();
            await smtp.ConnectAsync("smtp.gmail.com", 587,
SecureSocketOptions.StartTls);

            await smtp.AuthenticateAsync(_config.ServerEmail,
_config.ServerEmailPassword);

            await smtp.SendAsync(email);
            await smtp.DisconnectAsync(true);
            return true;
        }
        catch (Exception ex){
Console.WriteLine($"Email send failed: {ex.Message}");
            return false;}}}}

```

Файл SyncService локального серверу:

local_server.Services

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

using local_server.Utills;
using Microsoft.EntityFrameworkCore;
using shared_models.Sync.Contracts;
using shared_models.Sync.Types;
using System.Text;
using System.Text.Json;
using shared_models.Models;
using shared_models.Sync.ShallowModels;
using System.IO.Compression;
namespace local_server.Services
{
    public class SyncService
    {
        private readonly HttpClient _httpClient;
        private readonly LocalDbContext _context;
        private readonly AppStateService _appStateService;
        private readonly AudioFilesService _audioFilesService;
        private readonly BatchTranslationService _batchTranslationService;
        public SyncService(LocalDbContext context, HttpClient httpClient,
AppStateService appStateService, AudioFilesService audioFilesService,
BatchTranslationService batchTranslationService)
        {
            _httpClient = httpClient;
            _context = context;
            _appStateService = appStateService;
            _audioFilesService = audioFilesService;
            _batchTranslationService = batchTranslationService;
        }
        public async Task<bool> SyncAsync()
        {

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

var request = await CreateSyncRequestFromLocalDb();

await HttpTools.SetAuthorizationHeader(_httpClient, _appStateService);
string url = PathManager.GetControllerBaseUrl("sync/synchronize");
var content = new StringContent(JsonSerializer.Serialize(request),
Encoding.UTF8, "application/json");

var response = await _httpClient.PostAsync(url, content);
if (!response.IsSuccessStatusCode)
{
    var error = await response.Content.ReadAsStringAsync();
    throw new Exception($"API call failed: {response.StatusCode} - {error}");
}

await ClearSyncRecords();
var json = await response.Content.ReadAsStringAsync();
SyncResponse data = JsonSerializer.Deserialize<SyncResponse>(json);
await ApplySyncResponseAsync(data);

await _batchTranslationService.TranslateBatchAsync();
await _appStateService.UpdateLastSync();
return true;
}

public async Task ApplySyncResponseAsync(SyncResponse syncResponse)
{
    using var transaction = await _context.Database.BeginTransactionAsync();
    try
    {
        ApplyGuidMap(syncResponse.GuidMapItems);
        await UpsertRangeAsync(_context.UserTranslations,
syncResponse.UserTranslationsToUpload);
        await UpsertRangeAsync(_context.CustomTranslations,

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

syncResponse.CustomTranslationsToUpload);
        await UpsertRangeAsync(_context.CollectionTranslations,
syncResponse.CollectionTranslationsToUpload);
        await UpsertRangeAsync(_context.Collections,
syncResponse.CollectionsToUpload);
        await UpsertRangeAsync(_context.Themes,
syncResponse.ThemesToUpload);
        await UpsertRangeAsync(_context.TranslationChatItems,
syncResponse.TranslationChatItemsToUpload);
        if(syncResponse.NewAudiosToUpload?.Count > 0)
        {await
_audioFilesService.DownloadAudiosByGuidsAsync(syncResponse.NewAudiosToU
pload); }
        if (syncResponse.LanguagesToUpload is not null)
        await UpsertRangeAsync(_context.Languages,
syncResponse.LanguagesToUpload);
        await DeleteRangeByIdsAsync(_context.Translations,
syncResponse.TranslationsToDelete);
        await DeleteRangeByIdsAsync(_context.UserTranslations,
syncResponse.UserTranslationsToDelete);
        await DeleteRangeByIdsAsync(_context.CustomTranslations,
syncResponse.CustomTranslationsToDelete);
        await DeleteRangeByIdsAsync(_context.CollectionTranslations,
syncResponse.CollectionTranslationsToDelete);
        await DeleteRangeByIdsAsync(_context.Collections,
syncResponse.CollectionsToDelete);
        await DeleteRangeByIdsAsync(_context.Themes,
syncResponse.ThemesToDelete);
        await DeleteRangeByIdsAsync(_context.TranslationChatItems,
syncResponse.TranslationChatItemsToDelete);

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

        if (syncResponse.LanguagesToDelete is not null)
            await DeleteRangeByIdsAsync(_context.Languages,
syncResponse.LanguagesToDelete);
        await _context.SaveChangesAsync();
        await transaction.CommitAsync();
    } catch { await transaction.RollbackAsync(); throw; }}
private void ApplyGuidIdMap(ICollection<GuidIdMapItem> guidMapItems)
{foreach (var item in guidMapItems)
    {switch (item.TableName)
        {case SyncUploadTable.UserTranslations:
            ReplaceEntityKey(_context.UserTranslations, item);
            break;
        case SyncUploadTable.CustomTranslations:
            ReplaceEntityKey(_context.CustomTranslations, item);
            break;
        case SyncUploadTable.CollectionTranslations:
            ReplaceEntityKey(_context.CollectionTranslations, item);
            break;
        case SyncUploadTable.Collections:
            ReplaceEntityKey(_context.Collections, item);
            break;
        case SyncUploadTable.Themes:
            ReplaceEntityKey(_context.Themes, item);
            break;
        case SyncUploadTable.TranslationChatItems:
            ReplaceEntityKey(_context.TranslationChatItems, item);
            break;}}}
private void ReplaceEntityKey<TEntity>(DbSet<TEntity> set, GuidMapItem
mapItem) where TEntity : class
    {var entity = set.Local.FirstOrDefault(e => EF.Property<Guid>(e, "Id") ==

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

mapItem.LocalGuid)

        ?? set.Find(mapItem.LocalGuid);

    if (entity is not null)
    {
        _context.Entry(entity).State = EntityState.Detached;

        var newEntity = CloneEntityWithNewId(entity, mapItem.CloudGuid);

        set.Attach(newEntity);

        _context.Entry(newEntity).State = EntityState.Modified;
    }
}

private TEntity CloneEntityWithNewId<TEntity>(TEntity entity, Guid newId)
where TEntity : class
{
    var clone = _context.Entry(entity).CurrentValue.ToObject() as TEntity;

    if (clone != null)
    {
        _context.Entry(clone).Property("Id").CurrentValue = newId;
    }

    return clone!;}

private async Task UpsertRangeAsync<TEntity>(DbSet<TEntity> dbSet,
List<TEntity> entities) where TEntity : class
{
    foreach (var entity in entities)
    {
        var id = (Guid)_context.Entry(entity).Property("Id").CurrentValue!;

        var existing = await dbSet.FindAsync(id);

        if (existing == null){dbSet.Add(entity);}

        else{_context.Entry(existing).CurrentValue.SetValues(entity);}}
    }

private async Task DeleteRangeByIdsAsync<TEntity>(DbSet<TEntity> dbSet,
List<Guid> ids) where TEntity : class
{
    if (ids.Count == 0) return;

    var items = await dbSet.Where(e => ids.Contains(EF.Property<Guid>(e,
"Id"))).ToListAsync();

    dbSet.RemoveRange(items); }

public async Task PushSyncRecord(SyncRecord newRecord)

```

```

    {if (newRecord == null)
        {throw new ArgumentNullException(nameof(newRecord), "SyncRecord
cannot be null."); }
        if (newRecord.RecordPk == Guid.Empty)
        {
            throw new ArgumentException("RecordPk cannot be an empty GUID.",
nameof(newRecord.RecordPk));
        }
        if (string.IsNullOrEmpty(newRecord.TableName))
        {
            throw new ArgumentException("TableName cannot be null or empty.",
nameof(newRecord.TableName));
        }
        var existingRecord = await _context.SyncRecords.FirstOrDefaultAsync(r =>
r.RecordPk == newRecord.RecordPk && r.TableName == newRecord.TableName);

        if (existingRecord != null)
        {switch (newRecord.Action)
            {case SyncAction.Create:
                if (existingRecord.Action == SyncAction.Create ||
existingRecord.Action == SyncAction.Update)
                    {existingRecord.LastUpdatedAt = newRecord.LastUpdatedAt; }
                else if (existingRecord.Action == SyncAction.Delete)
                {
                    existingRecord.Action = SyncAction.Create;
                    existingRecord.LastUpdatedAt = newRecord.LastUpdatedAt;
                }
                break;
            case SyncAction.Update:
                if (existingRecord.Action == SyncAction.Create)

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

        {existingRecord.LastUpdatedAt = newRecord.LastUpdatedAt; }
        else if (existingRecord.Action == SyncAction.Update)
        {existingRecord.LastUpdatedAt = newRecord.LastUpdatedAt; }
        else if (existingRecord.Action == SyncAction.Delete)
        {
            existingRecord.Action = SyncAction.Create;
            existingRecord.LastUpdatedAt = newRecord.LastUpdatedAt;
        }
        break;
    case SyncAction.Delete:
        if (existingRecord.Action == SyncAction.Create)
        {_context.SyncRecords.Remove(existingRecord);}
        else{
            existingRecord.Action = SyncAction.Delete;
            existingRecord.LastUpdatedAt = newRecord.LastUpdatedAt;
        }
        break;}}
    else
    {if (newRecord.Id == Guid.Empty) {newRecord.Id = Guid.NewGuid();}
        _context.SyncRecords.Add(newRecord);
    }
    await _context.SaveChangesAsync();
}

public async Task PushSyncRecords(ICollection<SyncRecord> records)
{
    if (records == null || !records.Any()) return;
    var groupedRecords = records
        .GroupBy(r => new { r.TableName, r.RecordPk })
        .Select(g => g.OrderByDescending(r => r.LastUpdatedAt).ThenBy(r =>
            GetSyncActionPriority(r.Action)).First())

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        .ToList();
    foreach (var record in groupedRecords){await PushSyncRecord(record);}
    await _context.SaveChangesAsync();
}

private int GetSyncActionPriority(SyncAction action)
{
    return action switch
    {
        SyncAction.Delete => 1, // highest priority
        SyncAction.Create => 2, // recreation has lowest priority (modifies an
existing state)
        _ => 99 // unhandled action
    };
}

public async Task<SyncRequest> CreateSyncRequestFromLocalDb()
{
    LocalSettings localSettings = await _appStateService.GetLocalSettings();
    var syncRequest = new SyncRequest
    {
        LastClientSyncAt = localSettings.LastSyncAt
    };
    var syncRecords = await _context.SyncRecords.ToListAsync();
    foreach (var record in syncRecords)
    {
        switch (record.TableName)
        {
            case "UserTranslations":
                var userTranslation = await
_context.UserTranslations.FindAsync(record.RecordPk);
                if (userTranslation != null)
                {

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

var sUserTranslation = new S_UserTranslation(userTranslation);
if (record.Action == SyncAction.Delete)
{
    syncRequest.UserTranslationsToDelete.Add(record.RecordPk);
}
else
{
    syncRequest.UserTranslationsToUpload.Add(new
SyncRecordDto<S_UserTranslation>(record.Action, sUserTranslation));
}
}
else if (record.Action == SyncAction.Delete)
{
    syncRequest.UserTranslationsToDelete.Add(record.RecordPk);
}
break;
case "CustomTranslations":
    var customTranslation = await
_context.CustomTranslations.FindAsync(record.RecordPk);
    if (customTranslation != null)
    {
        var sCustomTranslation = new
S_CustomTranslation(customTranslation);
        if (record.Action == SyncAction.Delete)
        {syncRequest.CustomTranslationsToDelete.Add(record.RecordPk); }
        else
        {
            syncRequest.CustomTranslationsToUpload.Add(new
SyncRecordDto<S_CustomTranslation>(record.Action, sCustomTranslation));
        }
    }

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    else if (record.Action == SyncAction.Delete)
    {
        syncRequest.CustomTranslationsToDelete.Add(record.RecordPk);
    }
    break;
case "CollectionTranslations":
    var collectionTranslation = await
_context.CollectionTranslations.FindAsync(record.RecordPk);
    if (collectionTranslation != null)
    {
        var sCollectionTranslation = new
S_CollectionTranslation(collectionTranslation);
        if (record.Action == SyncAction.Delete)
        {
syncRequest.CollectionTranslationsToDelete.Add(record.RecordPk);
        }
        else
        {
            syncRequest.CollectionTranslationsToUpload.Add(new
SyncRecordDto<S_CollectionTranslation>(record.Action, sCollectionTranslation));
        }
    }
    else if (record.Action == SyncAction.Delete)
    {
syncRequest.CollectionTranslationsToDelete.Add(record.RecordPk);
    }
    break;
case "Collections":
    var collection = await

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

```

_context.Collections.FindAsync(record.RecordPk);
    if (collection != null)
    {var sCollection = new S_Collection(collection);
        if (record.Action == SyncAction.Delete)
            {syncRequest.CollectionsToDelete.Add(record.RecordPk);}
        else{syncRequest.CollectionsToUpload.Add(new
SyncRecordDto<S_Collection>(record.Action, sCollection)); }
    }
    else if (record.Action == SyncAction.Delete)
    {syncRequest.CollectionsToDelete.Add(record.RecordPk);}
    break;
case "Themes":
    var theme = await _context.Themes.FindAsync(record.RecordPk);
    if (theme != null)
    {var sTheme = new S_Theme(theme);
        if (record.Action == SyncAction.Delete)
            {syncRequest.ThemesToDelete.Add(record.RecordPk);}
        else
            {syncRequest.ThemesToUpload.Add(new
SyncRecordDto<S_Theme>(record.Action, sTheme));
        }}
    else if (record.Action == SyncAction.Delete)
    {syncRequest.ThemesToDelete.Add(record.RecordPk);}
    break;
case "TranslationChatItems":
    var translationChatItem = await
_context.TranslationChatItems.FindAsync(record.RecordPk);
    if (translationChatItem != null)
    {
        var sTranslationChatItem = new

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

S_TranslationChatItem(translationChatItem);

        if (record.Action == SyncAction.Delete)
        {
syncRequest.TranslationChatItemsToDelete.Add(record.RecordPk); }

        else
        {syncRequest.TranslationChatItemsToUpload.Add(new
SyncRecordDto<S_TranslationChatItem>(record.Action, sTranslationChatItem));}

        }

        else if (record.Action == SyncAction.Delete)
        {syncRequest.TranslationChatItemsToDelete.Add(record.RecordPk);}

        break;}

    }

    return syncRequest;
}

private async Task ClearSyncRecords()
{await _context.SyncRecords.ExecuteDeleteAsync();}

public async Task DownloadAndInsertDataAsync()
{
    await HttpTools.SetAuthorizationHeader(_httpClient, _appStateService);
    string url = PathManager.GetControllerBaseUrl("sync/download");
    var response = await _httpClient.GetAsync(url);
    response.EnsureSuccessStatusCode();
    using (var stream = await response.Content.ReadAsStreamAsync())
    using (var archive = new ZipArchive(stream, ZipArchiveMode.Read))
    {
        var entry = archive.GetEntry("data.json");
        if (entry == null)
        {
            throw new InvalidOperationException("ZIP file does not contain
'data.json' entry.");

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

```

    }
    using (var entryStream = entry.Open())
    {
        var downloadResponse = await
System.Text.Json.JsonSerializer.DeserializeAsync<DownloadResponse>(entryStream);

        if (downloadResponse != null)
        { await
_context.Languages.AddRangeAsync(downloadResponse.Languages);

            await
_context.UserTranslations.AddRangeAsync(downloadResponse.UserTranslations);

            await
_context.CustomTranslations.AddRangeAsync(downloadResponse.CustomTranslations);

            await
_context.Collections.AddRangeAsync(downloadResponse.Collections);

            await
_context.CollectionTranslations.AddRangeAsync(downloadResponse.CollectionTranslations);

            await
_context.Themes.AddRangeAsync(downloadResponse.Themes);

            await
_context.TranslationChatItems.AddRangeAsync(downloadResponse.TranslationChatItems);

            if (downloadResponse.Settings != null)
            {
                var existingSettings = await
_context.Settings.FirstOrDefaultAsync();

                if (existingSettings != null)
                {

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

        _context.Settings.Remove(existingSettings);
    }
    await _context.Settings.AddAsync(downloadResponse.Settings);
}
await _context.SaveChangesAsync();
}}}}}}

```

Файл UserService локального серверу:

```

using shared_models.Models;
using local_server.Utills;
using shared_models.DTO.UserDto;
using Microsoft.EntityFrameworkCore;
namespace local_server.Services
{public class UserService
    {
        private readonly HttpClient _httpClient;
        private readonly AppStateService _appStateService;
        private readonly LocalDbContext _context;
        private readonly AutosyncService _autosyncService;
        public UserService(HttpClient httpClient, LocalDbContext context,
AppStateService appStateService, AutosyncService autosyncService)
        {
            _httpClient = httpClient;
            _context = context;
            _appStateService = appStateService;
            _autosyncService = autosyncService;
        }
        public async Task<UserNoSensitive?> Get()
        {
            await HttpTools.SetAuthorizationHeader(_httpClient, _appStateService);

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

        using var response = await
_httpClient.GetAsync(PathManager.GetControllerBaseUrl("users"));

        UserNoSensitive? uns = await
response.Content.ReadFromJsonAsync<UserNoSensitive>();

        return uns;}

public async Task<Guid?> GetCurrentIdAsync()
{return _context.Users.FirstOrDefault()?.Id; }

public async Task<bool> PreRegister(PreRegisterRequest authDto)
{
    string path = PathManager.GetControllerBaseUrl("users/pre-register");
    using var response = await _httpClient.PostAsJsonAsync(path, authDto);
    return response.IsSuccessStatusCode;
}

public async Task<RegisterResponse>
CompleteRegister(CompleteRegisterRequest authDto)
{
    string path = PathManager.GetControllerBaseUrl("users/complete-register");
    using var response = await _httpClient.PostAsJsonAsync(path, authDto);
    RegisterResponse? rr = await
response.Content.ReadFromJsonAsync<RegisterResponse>();
    if (response.IsSuccessStatusCode)
    {
        await _context.Languages.AddRangeAsync(rr.Languages);
        await _appStateService.UpdateSession(rr.Token);

        await _context.Users.AddAsync(new User(rr.User));
        await _context.SaveChangesAsync();
    }
    else{throw new Exception("Register failed");}
    return rr;}

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public async Task<LoginResponse?> Login(LoginRequest authDto)
{
    using var response = await
_httpClient.PostAsJsonAsync(PathManager.GetControllerBaseUrl("users/login"),
authDto);
    LoginResponse? rr = await
response.Content.ReadFromJsonAsync<LoginResponse>();
    if (response.IsSuccessStatusCode)
    {
        await _context.Languages.AddRangeAsync(rr.Languages);
        await _appStateService.UpdateSession(rr.Token);
        await _context.Users.AddAsync(new User(rr.User));
        await _context.SaveChangesAsync();
        _ = Task.Run(async () =>
        {
            await _autosyncService.TriggerInitialDownloadAsync();
        });
    }
    else{throw new Exception("Login failed");}
    return rr;
}

public async Task Logout()
{
    await HttpTools.SetAuthorizationHeader(_httpClient, _appStateService);
    await _appStateService.UpdateSession(null);

    await _context.ClearDatabaseAsync();
    await _appStateService.UpdateSession(null);
}

public async Task<bool> IsAuthenticated()
{

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return await _appStateService.GetToken() != null;
    }

    public async Task<bool> DeleteUser()
    {
        await HttpTools.SetAuthorizationHeader(_httpClient, _appStateService);
        using var response = await
            _httpClient.DeleteAsync(PathManager.GetControllerBaseUrl("users"));
        if (response.IsSuccessStatusCode)
        {
            await _context.ClearDatabaseAsync();
        }
        return response.IsSuccessStatusCode;
    }

    public async Task<List<UserListItemDto>> GetUsersAsync(int page, int
itemsPerPage)
    {
        await HttpTools.SetAuthorizationHeader(_httpClient, _appStateService);
        using var response = await
            _httpClient.GetAsync(PathManager.GetControllerBaseUrl("users/list-items") +
                $"?page={page}" +
                $"&itemsPerPage={itemsPerPage}");
        if (response.IsSuccessStatusCode)
        {
            List<UserListItemDto> users = await
response.Content.ReadFromJsonAsync<List<UserListItemDto>>();
            return users;
        }
        throw new Exception("Something went wrong fetching users!");
    }

    internal async Task<AccountData> GetAccountDataAsync()
    {
        var response = new AccountData();
        var user = await _context.Users.Include(u => u.Settings).FirstAsync();
    }

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

response.Id = user.Id;
response.IsHidden = user.IsHidden;
response.Username = user.Username;
response.Email = user.Email;
response.RecallPlan = user.Settings.RecallPlan;
response.NumberOfCollections = await _context.Collections
    .CountAsync(c => c.UserId == user.Id);
var userTranslationData = await _context.UserTranslations
    .Where(ut => ut.UserId == user.Id)
    .GroupBy(ut => ut.Translation.OriginLanguage)
    .Select(group => new
    {
        Language = group.Key,
        Count = group.Count()
    })
    .OrderByDescending(x => x.Count)
    .ToListAsync();
response.NumberOfTranslations = userTranslationData.Sum(x => x.Count);
var mostPopularLanguage = userTranslationData.FirstOrDefault()?.Language;
if (mostPopularLanguage != null) response.MostPopularLanguage =
mostPopularLanguage.Title;
else response.MostPopularLanguage = "N/A";
return response;}
public async Task<bool> Update(UpdateUserDto update)
{
    using var response = await
_httpClient.PostAsJsonAsync(PathManager.GetControllerBaseUrl("users/update"),
update);
    return response.IsSuccessStatusCode;}}}}

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл UserService віддаленого серверу:

namespace database_api.Services

```
{  
    public class UserService  
    {  
        private readonly CloudDbContext _context;  
        private readonly string _jwtIssuer;  
        private readonly string _jwtAudience;  
        private readonly string _jwtSecret;  
        private readonly EmailService _emailService;  
  
        public UserService(CloudDbContext context, IOOptions<JwtSecrets> jwtSecrets,  
EmailService emailService)  
        {  
            _context = context;  
            _jwtIssuer = jwtSecrets.Value.Issuer;  
            _jwtAudience = jwtSecrets.Value.Audience;  
            _jwtSecret = jwtSecrets.Value.SecretKey;  
            _emailService = emailService;  
        }  
        public async Task<bool> PreRegister(PreRegisterRequest authDto)  
        {  
            var existingUser = await _context.Users  
                .FirstOrDefaultAsync(u => u.Username == authDto.Username || u.Email  
== authDto.Email);  
            if (existingUser != null)  
                throw new Exception("User already exists.");  
            var existingAuthUser = await _context.AuthUsers  
                .FirstOrDefaultAsync(u => u.Email == authDto.Email);  
            if (existingAuthUser != null)
```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        _context.AuthUsers.Remove(existingAuthUser);
var (hash, salt) = HashPassword(authDto.Password);
var verificationCode = GenerateVerificationCode().ToString();
var authUser = new AuthUser
{
    Email = authDto.Email,
    Username = authDto.Username,
    Password = hash,
    Salt = salt,
    VerificationCode = verificationCode,
    IsEmailVerified = false,
    CreatedAt = DateTime.UtcNow
};
_context.AuthUsers.Add(authUser);
await _context.SaveChangesAsync();
return await _emailService.SendVerificationCodeAsync(authDto.Email,
verificationCode);
}

public async Task<User> CompleteRegister(CompleteRegisterRequest authDto)
{
    var authUser = await _context.AuthUsers
        .FirstOrDefaultAsync(u => u.Email == authDto.Email);
    if (authUser == null)
        throw new Exception("Verification request not found.");
    if (authUser.VerificationCode != authDto.EmailVerificationCode.ToString())
        throw new Exception("Invalid verification code.");
    var user = new User()
    {
        Username = authUser.Username,
        Email = authUser.Email,

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Password = authUser.Password,
        Salt = authUser.Salt,
        LastActivity = DateTime.UtcNow,
        Settings = new Settings(),
        LastUpdatedAt = DateTime.UtcNow,
    };

    _context.Users.Add(user);
    _context.AuthUsers.Remove(authUser);
    await _context.SaveChangesAsync();
    return user;
}

public async Task<User> Login(LoginRequest authDto)
{
    var existingUser = await _context.Users.FirstOrDefaultAsync(u =>
u.Username == authDto.UsernameOrEmail || u.Email ==
authDto.UsernameOrEmail);

    if (existingUser == null)
    {throw new Exception("User does not exist.");}

    else if (!VerifyPassword(authDto.Password, existingUser.Password,
existingUser.Salt))
    {
        throw new Exception("Wrong password.");
    }

    existingUser.LastActivity = DateTime.UtcNow;
    _context.Users.Update(existingUser);
    await _context.SaveChangesAsync();
    return existingUser;
}

public string GenerateJwtToken(User user)
{

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

var claims = new[]
{
    new Claim(JwtRegisteredClaimNames.Sub, user.Id.ToString()),
    new Claim(JwtRegisteredClaimNames.Email, user.Email),
};
var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_jwtSecret));
var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
var token = new JwtSecurityToken(
    _jwtIssuer,
    _jwtAudience,
    claims,
    expires: DateTime.UtcNow.AddDays(30),
    signingCredentials: creds);
return new JwtSecurityTokenHandler().WriteToken(token);
}

public (string Hash, string Salt) HashPassword(string password)
{
    byte[] salt = new byte[16];
    using (var rng = RandomNumberGenerator.Create())
    {
        rng.GetBytes(salt);
    }
    string saltBase64 = Convert.ToBase64String(salt);
    string hash = Convert.ToBase64String(KeyDerivation.Pbkdf2(
        password: password,
        salt: salt,
        prf: KeyDerivationPrf.HMACSHA256,
        iterationCount: 10000,
        numBytesRequested: 32));
}

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Локтікова Т.М.				104
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return (hash, saltBase64);
    }

    public bool VerifyPassword(string enteredPassword, string storedHash, string
storedSalt)
    {
        byte[] salt = Convert.FromBase64String(storedSalt);
        string hash = Convert.ToBase64String(KeyDerivation.Pbkdf2(
            password: enteredPassword,
            salt: salt,
            prf: KeyDerivationPrf.HMACSHA256,
            iterationCount: 10000,
            numBytesRequested: 32));
        return hash == storedHash;
    }

    private static int GenerateVerificationCode()
    {
        Random random = new Random();
        return random.Next(100000, 999999);
    }

    public async Task<List<UserListItemDto>> GetUsersAsync(int page = 1, int
itemsPerPage = 20)
    {
        if (page < 1)
            page = 1;
        if (itemsPerPage < 1)
            itemsPerPage = 20;
        var usersQuery = _context.Users
            .Where(u => !u.IsHidden);
        var paginatedUsers = await usersQuery
            .OrderByDescending(u => u.Collections.Count)

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        .Skip((page - 1) * itemsPerPage)
        .Take(itemsPerPage)
        .Include(u => u.Collections)
        .ToListAsync();
    return UserListItemDto.List(paginatedUsers);
}

public async Task<bool> UpdateUserAsync(Guid userId, UpdateUserDto
update)
{
    var user = await _context.Users.Include(u => u.Settings).FirstAsync(u => u.Id
== userId);

    if(update.Username.Length < 3) throw new InvalidDataException("Username
length must be not less then 3.");

    if (user == null) throw new InvalidDataException("User does not exist.");
    if (await _context.Users.AnyAsync(u => u.Username == update.Username))
throw new InvalidDataException("The username is already exists.");

    user.Username = update.Username;
    user.IsHidden = update.IsHidden;
    user.Settings.RecallPlan = update.RecallPlan;
    _context.SaveChanges();
    return true;
}

public async Task<bool> DeleteUserAsync(Guid userId)
{
    var user = await _context.Users.FirstOrDefaultAsync(u => u.Id == userId);
    if (user == null)
    { throw new InvalidOperationException("User not found.");}
    _context.Users.Remove(user);
    await _context.SaveChangesAsync();
    return true;
}
}
}

```

		Гнатюк Д.С.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Локтікова Т.М.				106
Змн.	Арк.	№ докум.	Підпис	Дата		