Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромережо»

Виконав: студент II курсу групи 1ПІ-23м
спеціальності 121 – Інженерія програмного забезпечення
Ольхов Максим Русланович
Керівник: к.т.н., доц. каф. ПЗ Ткаченко. О. М.
«09» грудня 2024 р.
Опонент: к.т.н., доц. каф. ОТ Тарновський М.Г.,
«09» грудня 2024 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«09» грудня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор О.Н. Романюк
_____
«18» вересня 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ольхову Максиму Руслановичу

1. Тема роботи – Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж.

Керівник роботи: Ткаченко Олександр Миколайович, к.т.н., доц. каф. ПЗ, затверджені наказом вищого навчального закладу від
« _17_ » _вересня 2024 р. № 310.

2. Строк подання студентом роботи
_____3 грудня 2024 р._____

3. Вхідні дані до роботи: модель розробки – вдосконалення; моделі нейронних мереж для автоматизованого перекладу – трансформери; моделі Generative Pre-trained Transformer; методи обробки контексту – механізм уваги (Attention Mechanism); інтерфейс для виклику перекладу – OpenAI ChatGPT API; кешування перекладів – база даних для зберігання результатів.

4. Зміст текстової частини: вступ; аналіз питання та постановка задач; розрозбка методів підвищення якості процесу перекладу та алгоритмів роботи

програмного продукту; розробка програмного забезпечення для автоматизова
перекладу; економічна частина; висновки; список використаних джерел; додат

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність те об'єкт та предмет дослідження; теоретичні основи; наукова новизна; порівняль аналіз аналогів; метод і блок-схема алгоритму робути; структура графічн інтерфейсу головного вікна розширення; тестування програмного забезпече економічна частина; апробація та публікації роботи; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ткаченко О. М., к.т.н., доц. каф. ПЗ	19.09.2024	20.11.2024
5	Причепа І.В., к.е.н., доцент кафедри ЕПВМ	21.11.2024	26.11.2024

7. Дата видачі завдання 19 вересня 20 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів автоматизованого перекладу	20.09.2024-30.09.2024	Вик.
2	Розробка методів підвищення якості автоматизованого перекладу	01.10.2024-08.10.2024	Вик.
3	Розробка алгоритмів роботи програмного забезпечення	09.10.2024-09.11.2024	Вик.
4	Розробка програмного забезпечення	10.11.2024-24.11.2024	Вик.
5.	Економічна частина	24.11.2024-30.11.2024	Вик.

Студент _____ **Ольхов М. Р.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ткаченко О. М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.8

Ольхов М. Р. Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж. Магістерська кваліфікаційна робота зі спеціальності 121 - Інженерія програмного забезпечення.
Вінниця: ВНТУ, 2024. 107с.
На укр. мов. Бібліогр.: 30 назв; рис.: 31; табл.: 21.

Ця магістерська робота присвячена дослідженню можливостей та застосувань інформаційної технології автоматизованого перекладу текстів у веб-додатках з використанням нейромережевих моделей. Основний акцент зроблено на дослідженні сучасних методів автоматизованого перекладу, таких як моделі на основі трансформерів і механізмів уваги.

Дослідження включає розробку та тестування алгоритмів автоматизованого перекладу, спрямованих на підвищення точності та релевантності перекладів у багатомовних веб-додатках. Результати роботи допоможуть краще зрозуміти, як нейромережеві методи та алгоритми впливають на оптимізацію процесів перекладу текстів, сприяючи зручності користування для різних мовних груп.

Ключові слова: автоматизований переклад, нейромережі, веб-додатки, ChatGPT-4 API, оптимізація перекладу, контекстуальний переклад.

ANNOTATION

Olkhov M. R. Development of a method and software tool for automated text translation in web applications based on neural networks. Master's qualification work in the specialty 121 - Software Engineering.

Vinnytsia: VNTU, 2024. 107c.

In Ukrainian. Bibliogr.: 30 titles; figs. 31; tables: 21.

This master's thesis is devoted to the study of the possibilities and applications of information technology for automated text translation in web applications using neural network models. The main focus is on the study of modern methods of automated translation, such as models based on transformers and attention mechanisms.

The research includes the development and testing of automated translation algorithms aimed at improving the accuracy and relevance of translations in multilingual web applications. The results of the work will help to better understand how neural network methods and algorithms affect the optimization of text translation processes, contributing to the usability of different language groups.

Keywords: automated translation, neural networks, web applications, ChatGPT-4 API, translation optimization, contextual translation.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Актуальність дослідження зумовлена необхідністю вдосконалення методів автоматизованого перекладу текстів у веб-додатках з використанням нейромережевих технологій. Ураховуючи глобалізацію та зростаючі вимоги до багатомовних платформ, дослідження в цій сфері є надзвичайно актуальним, оскільки забезпечення точного та контекстуально залежного перекладу є основним викликом для сучасних веб-додатків.

Сучасні методи автоматизованого перекладу, зокрема на основі традиційних статистичних моделей та методів машинного перекладу, демонструють обмежену здатність адекватно відображати контекст і культурні відмінності між мовами. Використання більш потужних технологій, таких як нейромережі, дозволяє значно покращити якість перекладу, але це також ставить нові вимоги до продуктивності та ефективності систем.

Аналіз існуючих рішень у сфері автоматизованого перекладу показує, що трансформерні моделі (наприклад, BERT, GPT) забезпечують високу точність, їх застосування в реальному часі на багатомовних платформах потребує значних обчислювальних ресурсів. Більш того, недоліком є потреба в великих навчальних вибірках, що може обмежити їх адаптивність до специфічних доменів і контекстів, характерних для конкретних веб-додатків. Водночас методи на основі моделі послідовність-послідовність (Seq2Seq) також мають свої обмеження, особливо при перекладі складних ідіоматичних виразів чи спеціалізованих термінів.

На сьогодні існуючі методи перекладу на основі нейромереж дозволяють досягти значних результатів, вони все ще стикаються з викликами, зокрема щодо інтеграції в реальний час, масштабованості та адаптації до специфічних контекстів.

Таким чином, актуальними є задача розробки методів та програмних засобів для підвищення якості перекладу та адаптивності систем автоматизованого

перекладу, а також розробка нових підходів до інтеграції нейромережевих моделей у веб-додатки.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення якості автоматизованого перекладу текстів у веб-додатках за допомогою нейромережевих моделей, зокрема шляхом вибору оптимальних архітектур для точного перекладу з урахуванням контексту та стилістики тексту.

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів нейромережевого перекладу для автоматизованої обробки текстів у веб-додатках;

- визначити оптимальну для подальшого розвитку архітектуру нейромереж для забезпечення точного перекладу з урахуванням контексту і стилістики тексту;

- вдосконалити метод автоматизованого перекладу текстів у веб-додатках на основі нейромереж,

- розробити програмні засоби для інтеграції автоматизованого перекладу в веб-додатки з використанням OpenAI ChatGPT API;

- створити механізм кешування перекладів для підвищення продуктивності та швидкості обробки повторюваних текстів;;

- провести експериментальні дослідження розроблених засобів для оцінки точності та ефективності перекладу.

Об'єкт дослідження – процес перекладу текстів у веб-додатках за допомогою нейромереж.

Предмет дослідження – методи та засоби автоматизованого перекладу текстів у веб-додатках за допомогою нейромереж.

Методи дослідження: методи математичної статистики для обробки й аналізу навчальних даних; методи машинного навчання для створення моделей

автоматизованого перекладу; комп’ютерне моделювання для аналізу та перевірки теоретичних положень.

Наукова новизна отриманих результатів.

1. Вдосконалено метод автоматизованого перекладу текстів у веб-додатках на основі нейромереж, який, на відміну від існуючих, поєднує переваги моделей трансформерів і механізмів уваги, що дозволяє підвищити точність перекладу складних фраз та контекстуальних виразів, особливо в багатомовних середовищах.

2. Отримала подальшого розвитку архітектуру програмної системи для автоматизованого перекладу, яка, на відміну від існуючих, включає інтеграцію OpenAI ChatGPT API разом з механізмом кешування в базі даних, що дозволило знизити навантаження на сервери та підняти швидкість перекладу.

3.**Практичне значення отриманих результатів.** Розроблено інформаційну технологію, алгоритми та програмні засоби, які дозволяють підвищити точність автоматизованого перекладу текстів у веб-додатках на основі нейромережевих моделей, що включають обробку великих обсягів текстових даних та адаптацію перекладів до контексту.

Особистий внесок здобувача.

Всі наведені у магістерській кваліфікаційній роботі наукові результати отримані автором особисто. Наукову працю опубліковано одноосібно.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи, представлені та обговорені на Міжнародній науково-практичній інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Публікації. Основні результати досліджень були опубліковані в одній науковій праці, яка входила у матеріали Міжнародної науково-практичної інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [1].

1 АНАЛІЗ МЕТОДІВ АВТОМАТИЗОВАНОГО ПЕРЕКЛАДУ ТЕКСТІВ У ВЕБ-ДОДАТКАХ

1.1 Аналіз сучасних підходів до автоматизованого перекладу текстів у веб-додатках

Автоматизований переклад текстів у веб-додатках стає невід'ємною частиною багатьох онлайн-сервісів, забезпечуючи користувачам доступ до інформації на різних мовах без необхідності залучення професійних перекладачів. Це особливо актуально для платформ із глобальним охопленням, де аудиторія складається з користувачів різних культур і мов. Таким чином, автоматизований переклад покращує користувацький досвід та сприяє доступності контенту для ширшої аудиторії.

Однією з провідних технологій у цій галузі є моделі на основі нейромереж, зокрема архітектура трансформерів, яка дозволяє отримувати більш точні й контекстуально правильні переклади. Завдяки механізмам уваги такі моделі здатні "розуміти" контекст усього речення або навіть більшого обсягу тексту, що особливо для перекладу складних фраз та ідіом. Це відкриває можливість значного покращення якості перекладу у порівнянні з традиційними підходами, що зосереджуються на послідовному перекладі слів чи фраз[1].

Сучасні веб-додатки також вимагають швидких рішень, тому важливим є баланс між якістю перекладу та швидкістю його виконання. Технології автоматизованого перекладу оптимізуються для забезпечення високої продуктивності, зокрема через кешування результатів перекладів та адаптивне налаштування нейронних мереж для конкретних контентів. Це знижує затримки в роботі додатку та покращує ефективність обробки великих обсягів текстової інформації.

Для ефективної інтеграції перекладу в роботу веб-додатків розробляються спеціальні API, такі як ChatGPT API, які дозволяють легко підключати

нейромережеві моделі до існуючих систем. Це забезпечує не лише високу якість перекладу, але й простоту в налаштуванні та експлуатації, що є важливим для розробників, які прагнуть розширити функціонал своїх додатків без великих витрат на інфраструктуру.

Таким чином, сучасні підходи до автоматизованого перекладу текстів у веб-додатках орієнтовані на поєднання нейромережевих моделей, оптимізації продуктивності та забезпечення багатомовної підтримки, що дозволяє покращити якість перекладу та зробити його більш зручним для користувачів у глобальному масштабі.

1.2 Роль нейромереж та штучного інтелекту в автоматизованому перекладі

Штучний інтелект (ШІ) та нейромережі відіграють ключову роль в автоматизованому перекладі текстів, дозволяючи суттєво підвищити точність, швидкість та якість перекладу. Застосування нейромереж, зокрема моделей на основі трансформерів, забезпечує можливість глибшого аналізу тексту, врахування контексту і створення перекладів, що наближені до людських. Це особливо важливо для складних текстів, де важливе розуміння не лише окремих слів, а й загального змісту та стилю.

Одним із важливих досягнень у цій галузі є використання глибокого навчання для розвитку систем перекладу, таких як ChatGPT та інші мовні моделі. Ці моделі здатні аналізувати великі обсяги текстів, визначати закономірності та створювати точні переклади з урахуванням культурних та мовних нюансів. Це значно полегшує роботу у сферах, де точність перекладу є важливою, наприклад, в медичній або юридичній документації.

Крім того, роль ШІ полягає в оптимізації та автоматизації рутинних процесів перекладу, що дозволяє звільнити людські ресурси для більш творчих та стратегічних завдань. Веб-додатки, що використовують ШІ для перекладу, можуть виконувати складні операції майже миттєво, що особливо цінно для великих

платформ із глобальною аудиторією. Це дозволяє значно скоротити час на обробку текстів та збільшити продуктивність сервісів.

Іншою важливою функцією ШІ в автоматизованому перекладі є здатність адаптуватися до змін у мовних структурах та нових виразів. З часом моделі можуть навчатися нових слів і фраз, розширюючи свій словниковий запас та покращуючи точність перекладу. Це дозволяє їм швидко реагувати на зміни у мовних трендах та підтримувати актуальність перекладів.

Також нейромережі надають можливість забезпечувати багатомовну підтримку та розуміння специфіки різних мовних груп. Це важливо для глобальних веб-додатків, де є необхідність у доступі до контенту на різних мовах.

У результаті, ШІ та нейромережі стають невід'ємними інструментами для створення високоякісних перекладацьких систем, які можуть задовольняти потреби сучасного користувача та забезпечувати масштабованість у процесі автоматизованого перекладу текстів у веб-додатках.

Нейромережі стали основним інструментом в автоматизованому перекладі текстів, завдяки їх здатності розпізнавати складні мовні структури та адаптуватись до контексту. Ранні підходи до машинного перекладу базувались на статистичних методах і правилах, однак із розвитком нейромереж з’явилися інноваційні рішення, які забезпечили суттєве підвищення якості перекладу. Сьогодні такі методи, як Seq2Seq[5], трансформери[2] та механізм уваги, широко застосовуються в більшості передових моделей для перекладу.

Кожен метод має свої унікальні характеристики та підходи до обробки текстів, що дозволяють досягти різних рівнів ефективності та точності. Наприклад, Seq2Seq[5] підходить для перекладу коротких фраз, Transformer[2], пропонуючи високу продуктивність. Модель BERT[3] враховує двонаправлений контекст слів, а T5[4] є універсальним інструментом, а також є модель GPT-4 на основі трансформерів, розроблена OpenAI для генерації текстів. Основні методи автоматизованого перекладу з використанням нейромереж наведені у таблиці 1.1.

Таблиця 1.1 – Основні методи автоматизованого перекладу

Метод	Опис	Приклади використання
Seq2Seq (послідовність-послідовність)	Використовується для обробки послідовностей тексту та перекладу фраз. Складається з двох частин: енкодера (захоплює контекст вхідного речення) і декодера (створює вихідне речення мовою перекладу).	Підходить для коротких фраз та речень. Використовується в системах, де важливе дотримання порядку слів.
Трансформер (Transformer)	Модель з механізмом уваги, що дозволяє одночасно аналізувати слова в реченні, незалежно від їхньої позиції. Застосовується для високоякісного перекладу з урахуванням контексту.	Google Translate, Microsoft Translator, OpenAI API.
ChatGPT (OpenAI)	Модель, основана на архітектурі трансформерів і здатна обробляти контекст, що дозволяє перекладати тексти з високою точністю та гнучкістю.	Застосовується для багатомовного перекладу, надаючи можливість генерувати точні переклади та контекстуалізувати їх.
BERT (Bidirectional Encoder Representations from Transformers)	Двонаправлена модель, що використовує контекст навколо слів для точного перекладу. Не підходить для створення повного тексту, але ефективна в задачах вибору правильного значення слів.	Використовується для доповнення перекладу, особливо у фахових текстах.

BERT (Bidirectional Encoder Representations from Transformers) – ця модель використовує двонаправлену увагу, що дозволяє враховувати контекст як зліва, так і справа від кожного слова. Це особливо корисно для задач перекладу, де точне розуміння контексту для отримання правильного результату. Завдяки такій архітектурі, BERT досягає високих результатів у перекладі складних фраз і виразів, де врахування контексту є важливим.

GPT (Generative Pre-trained Transformer) – ця модель є генеративною, що означає здатність не лише до перекладу, а й до створення нових текстів на основі наданих даних. GPT показує відмінні результати в задачах, де необхідно генерувати природні та зрозумілі переклади. Модель також легко адаптується до різних специфічних доменів, таких як технічний чи медичний переклад, завдяки своїй гнучкості.

T5 (Text-To-Text Transfer Transformer) – T5 є універсальною багатозадачною моделлю, що вирішує широкий спектр мовних завдань, включаючи переклад. Однією з ключових особливостей цієї архітектури є її здатність працювати з різними задачами за допомогою єдиного підходу, що дозволяє зменшити складність та покращити узгодженість результатів. Завдяки цій універсальності T5 показує ефективні результати у багатьох типах перекладу.

Детальний порівняльний аналіз нейромережевих методів за точністю представлено в таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз нейромережевих методів

Метод	Точність	Ефективність (Швидкість, Продуктивність)	Використання
Трансформер (Transformer)	Висока точність завдяки механізму уваги, добре працює з контекстом	Швидкий і ефективний при обробці великих обсягів тексту, висока продуктивність	Великі текстові обсяги, високоякісний переклад з урахуванням контексту
GPT	Висока точність для генеративних завдань, створення природних перекладів	Висока швидкість в генерації тексту, добре адаптується до доменів	Генерація текстів, специфічні домени (медичний, технічний переклад)

Продовженн таблиці 1.2

Метод	Точність	Ефективність (Швидкість, Продуктивність)	Використання
Seq2Seq	Помірна точність, залежить від довжини фраз	Повільніший, особливо при обробці довгих текстів	Переклад коротких фраз і речень, де важливий порядок слів
BERT	Дуже висока точність для контекстуальних завдань	Середня швидкість, залежить від конкретної задачі	Переклад фахових текстів, завдання, що вимагають контекстного розуміння

Нейромережі значно покращили якість автоматизованого перекладу завдяки здатності враховувати контекст і структуру тексту. Серед різних методів, GPT є найкращим вибором для багатьох завдань перекладу завдяки своїй гнучкості та здатності генерувати природні переклади, що відповідають специфічним вимогам доменів, таких як технічний або медичний переклад. Модель GPT має високу точність у генеративних завданнях, може адаптуватися до різних стилів тексту і забезпечує високу швидкість перекладу, що робить її ідеальною для інтеграції в сучасні системи автоматизованого перекладу.

1.3 Інтеграція моделей трансформерів і механізмів уваги в системи перекладу

Інтеграція моделей трансформерів і механізмів уваги в системи автоматизованого перекладу стала важливим етапом у розвитку технологій машинного перекладу. Завдяки своїм архітектурним особливостям, трансформери забезпечують високу продуктивність та точність при обробці текстових даних, дозволяючи досягати значного прогресу у вирішенні проблем, пов’язаних із перекладом.

Моделі трансформерів, зокрема Transformer та його варіанти, такі як GPT та T5, зосереджені на обробці тексту за допомогою механізму уваги, що дозволяє

аналізувати всі слова в реченні паралельно, а не по черзі, як це було в традиційних моделях Seq2Seq. Це значно зменшує час на обробку тексту і підвищує точність перекладу. Завдяки здатності трансформерів ефективно опрацьовувати довгі послідовності слів і враховувати глобальний контекст, ці моделі здобули популярність у системах перекладу, зокрема, у таких відомих продуктах, як Google Translate та Microsoft Translator.

Механізм уваги є важливою складовою трансформерів, оскільки дозволяє моделі фокусуватися на найбільш релевантних частинах тексту під час перекладу. Завдяки цьому забезпечується точніше відображення контексту та значень слів у різних мовах. Наприклад, у складних реченнях, де важливе значення кожного слова в контексті іншого, увага дозволяє моделі правильно інтерпретувати значення, навіть якщо слова знаходяться далеко одне від одного в реченні.

Інтеграція в системи перекладу цих моделей у дозволяє забезпечити високий рівень точності та швидкості обробки текстів. Для більш ефективного перекладу системи часто поєднують кілька підходів, наприклад, трансформери з механізмом уваги для загального перекладу і додаткові моделі, такі як BERT, для контекстуалізації термінів і спеціалізованих завдань, наприклад, перекладу технічних або юридичних текстів. Це дозволяє підвищити якість перекладу в специфічних сферах, де важлива точність і відповідність термінів.

Використання таких моделей у поєднанні з розподіленими обчислювальними ресурсами також дозволяє масштабувати системи перекладу для роботи з великими обсягами текстів у реальному часі, що є важливою вимогою для сучасних веб-додатків та багатомовних платформ.

З розвитком обчислювальних потужностей та вдосконаленням алгоритмів, таких як GPT-4, інтеграція трансформерів і механізмів уваги продовжить еволюціонувати, забезпечуючи ще більшу точність, швидкість і адаптацію до конкретних завдань. Очікується, що в майбутньому моделі зможуть ще краще

справлятися з нюансами різних мов і специфікою текстів, забезпечуючи більш точний та природний переклад у більш широкому спектрі застосувань.

1.4 Аналіз основних платформ і програмних засобів для автоматизованого перекладу

Автоматизоване перекладання текстів стало важливим аспектом роботи з багатомовними веб-додатками, програмним забезпеченням та іншими системами, що потребують швидкої обробки великих обсягів текстових даних. У цьому розділі буде розглянуто основні платформи, які використовують такі технології, а також їхні переваги та обмеження.

Основні платформи для автоматизованого перекладу наприклад кілька популярних платформ для автоматизованого перекладу, серед яких варто виділити Google Translate[6], Microsoft Translator[7], Amazon Translate[8] та DeepL[9]. Кожна з цих платформ використовує різні технології для обробки та перекладу тексту, механізмів уваги та інші передові нейромережеві підходи.

В таблиці 1.3, наведено порівняльний опис популярних платформ для автоматизованого перекладу.

Таблиця 1.3 – Порівняння моделей машинного навчання

Платформа	Опис	Особливості	Використання
Google Translate	Одна з найпопулярніших платформ для перекладу текстів та веб-сторінок.	Підтримує понад 100 мов, інтеграція з різними продуктами Google, автоматичне розпізнавання мови.	Переклад текстів, веб-сайтів, мобільних додатків.
Amazon Translate	Сервіс перекладу від Amazon, спеціально орієнтований на обробку великих обсягів тексту.	Підтримує більше 50 мов, інтеграція з іншими сервісами AWS.	Електронна комерція, підтримка багатомовності для веб-сайтів.

Продовження таблиці 1.3

Платформа	Опис	Особливості	Використання
DeepL	Платформа, відома своєю високою точністю та природністю перекладу.	Використовує передові нейромережеві моделі, зокрема трансформери, для досягнення високої якості.	Переклад технічних текстів, професійний переклад.
Microsoft Translator	Потужний сервіс для перекладу, заснований на трансформерах і механізмах уваги, підтримує велику кількість мов.	Інтеграція з продуктами Microsoft, включаючи Office, Skype і Azure.	Переклад для корпоративних та приватних користувачів.

Сучасні платформи для автоматизованого перекладу активно інтегруються в різноманітні веб-додатки, що дозволяє забезпечити багатомовну підтримку користувачів по всьому світу. Зокрема, Google Translate та Microsoft Translator часто використовуються для перекладу веб-сайтів, мобільних додатків та електронних документів, що потребують оперативного перекладу. Amazon Translate особливо підходить для інтернаціоналізації електронної комерції, де необхідно швидко перекладати величезні обсяги даних і підтримувати актуальність перекладів на великих платформах. DeepL, з його високою точністю, здобув популярність серед професіоналів, які працюють з технічними, юридичними та іншими спеціалізованими текстами.

Для кращого розуміння процесу роботи платформ, можна звернути увагу на рисунок 1.1 , що показує основні етапи перекладу з використанням трансформерів та механізмів уваги.

Рисунок 1.1 – Етапи перекладу з використанням трансформерів

У розділі розглянуто основні платформи для автоматизованого перекладу, такі як Google Translate, Microsoft Translator, Amazon Translate та DeepL, які використовують сучасні нейромережеві моделі, зокрема трансформери та механізми уваги, для досягнення високої точності перекладу. Кожна з платформ має свої особливості, зокрема Google Translate підтримує понад 100 мов і інтегрується з продуктами Google, Microsoft Translator є потужним інструментом для корпоративних користувачів, Amazon Translate орієнтований на обробку великих обсягів тексту, а DeepL славиться високою точністю перекладу, особливо для технічних текстів. Порівняльний аналіз точності та швидкості показує, що хоча всі платформи використовують подібні технології, відмінності в точності перекладу можуть бути значними для різних мовних пар, зокрема DeepL показує найкращі результати для технічних матеріалів, тоді як Google Translate є швидким і ефективним для загальних перекладів.

1.5 Оцінка ефективності існуючих рішень у галузі перекладу текстів у веб-додатках

Автоматизоване перекладання текстів є ключовим елементом сучасних веб-додатків, що працюють з багатомовними інтерфейсами та великими обсягами текстових даних. Вибір ефективного рішення для перекладу залежить від точності, швидкості, можливості інтеграції та специфічних вимог до типів текстів. У цьому розділі оцінюються основні платформи для автоматизованого перекладу, зокрема Google Translate, Microsoft Translator, Amazon Translate та DeepL, а також їх переваги та обмеження.

Точність перекладу є основним фактором при виборі платформи для автоматизованого перекладу текстів. Платформи, що використовують нейромережеві моделі на основі трансформерів, як-от Google Translate, Microsoft Translator, Amazon Translate та DeepL, демонструють високу точність для основних мовних пар. Однак точність може варіюватися залежно від специфічних типів текстів.

DeepL має найвищу точність при перекладі технічних текстів і спеціалізованих термінів, завдяки використанню вдосконалених нейромережевих архітектур. Google Translate, в свою чергу, показує високі результати при перекладі загальних мов, але може помилятись у спеціалізованих термінах.

Швидкість обробки тексту є важливим аспектом, особливо для веб-додатків, де потрібно швидко обробляти великі обсяги даних. Amazon Translate виділяється швидкістю, оскільки спеціалізується на обробці великих обсягів тексту в реальному часі. Відповідно, цей сервіс є ідеальним для інтеграцій у веб-платформи електронної комерції та інші реальні часи.

Microsoft Translator також показує добру швидкість, хоча її продуктивність іноді може бути середньою порівняно з Amazon Translate. Google Translate демонструє високу швидкість, але для великих обсягів тексту може бути менш ефективним.

Інтеграція рішень для перекладу в існуючі веб-додатки є важливим критерієм, оскільки дозволяє забезпечити масштабованість і легкість у використанні. Google Translate і Microsoft Translator надають зручні API, що дозволяє інтегрувати ці платформи в різні програмні середовища. Вони також підтримують великий набір мов і функціональність для перекладу текстів на різних платформах.

Amazon Translate добре інтегрується з іншими сервісами AWS, що забезпечує масштабованість і ефективність на великих веб-платформах. DeepL, хоча й має обмежену інтеграцію, забезпечує високу точність, що робить його ідеальним для професійного перекладу, зокрема технічних текстів.

Спеціалізація на конкретних типах текстів специфічних типів текстів, таких як технічні або юридичні матеріали, точність і ефективність перекладу мають вирішальне значення. DeepL виділяється як лідер у перекладі технічних текстів, завдяки високій точності та точному використанню спеціалізованих термінів. Microsoft Translator також справляється з технічними текстами, хоча його точність може бути меншою порівняно з DeepL.

Google Translate і Amazon Translate підходять для загальних текстів, таких як веб-контент або електронна пошта, але їх точність для спеціалізованих матеріалів може бути нижчою. Порівняння платформ за характеристиками такі як точність, швидкість та інтеграцією наведені в таблиці в таблиці 1.4, а також на рисунку 1.2 наведено порівняння точності та швидкості платформ.

Таблиця 1.4 – Порівняння платформ за характеристиками

Платформа	Точність	Швидкість обробки	Інтеграція та масштабованість	Спеціалізація
Google Translate	Висока для основних мовних пар, але можуть бути помилки для спеціалізованих термінів	Швидка обробка текстів	Інтеграція через API в різні платформи	Підходить для загальних текстів
Microsoft Translator	Висока точність для стандартних умов і технічних текстів	Швидкість середня	Інтеграція з продуктами Microsoft	Добре працює з технічними текстами
Amazon Translate	Висока точність, особливо для великих обсягів тексту	Дуже швидка обробка	Добра інтеграція з AWS	Спеціалізується на великих обсягах і реальному часі
DeepL	Найвища точність для технічних текстів	Швидкість середня	Обмежена інтеграція	Спеціалізується на професійному та технічному перекладі

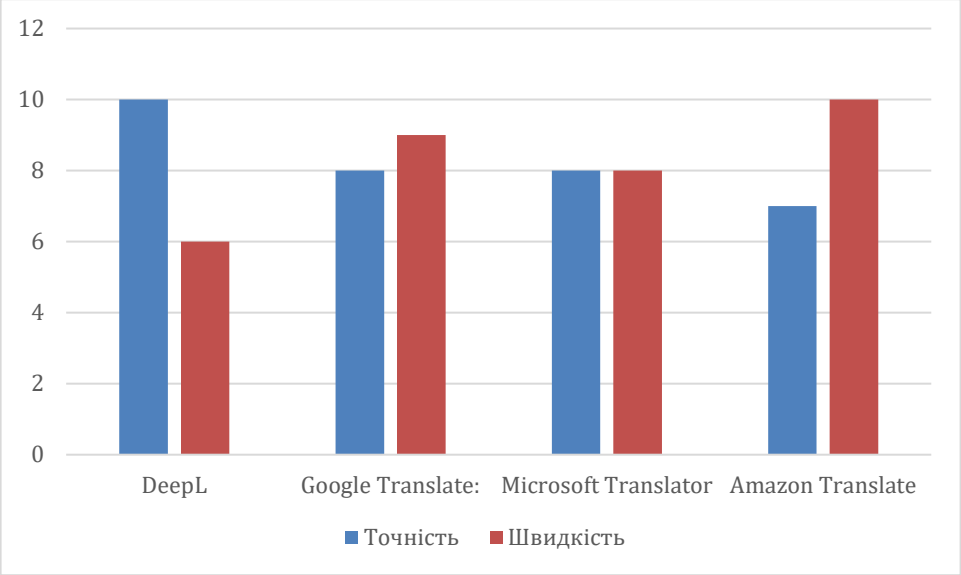


Рисунок 1.2 – Порівняння точності та швидкості платформ

Загальний аналіз показує, що всі основні платформи для автоматизованого перекладу текстів мають свої сильні та слабкі сторони. Google Translate і Microsoft Translator є ідеальними для загальних перекладів завдяки високій швидкості та хорошій інтеграції з іншими платформами. Amazon Translate демонструє відмінну швидкість і добре підходить для обробки великих обсягів тексту, а DeepL є лідером у точності перекладу технічних текстів і спеціалізованих матеріалів. Вибір платформи залежить від конкретних вимог до точності, швидкості, а також типу текстів, що перекладаються.

1.6 Формування вимог до програмного забезпечення для автоматизованого перекладу текстів у веб-додатках на основі нейромереж

Беручи до уваги актуальність та необхідність ефективного автоматизованого перекладу текстів у веб-додатках, а також особливості існуючих методів машинного перекладу, переваги та недоліки існуючих рішень, можна сформулювати вимоги до програмного забезпечення, яке буде розроблятися для автоматизованого перекладу текстів.

Конкурентоспроможне програмне забезпечення для автоматизованого перекладу текстів повинно відповідати наступним характеристикам:

- Простота у використанні: інтерфейс повинен бути інтуїтивно зрозумілим, без необхідності додаткового навчання користувачів.
- Автоматизація процесів: система повинна забезпечити автоматичний переклад без необхідності втручання користувача, а також автоматичне збереження результатів перекладу в базі даних.
- Прогнозування та адаптація до контексту: система повинна бути здатною адаптувати переклади залежно від контексту і специфіки використаного тексту (наприклад, технічні терміни, ідіоматичні вирази, культурні особливості).
- Мультиплатформність: програмне забезпечення повинно бути доступним на різних платформах (веб, мобільні додатки, десктопи).
- Наявність аналітичних інструментів: для аналізу якості перекладу повинні бути надані інструменти для моніторингу точності і швидкості перекладу, а також можливість вдосконалення моделей на основі отриманих даних.
- Доступність та інтеграція: система повинна мати API для інтеграції з іншими веб-додатками та системами, забезпечуючи безперебійне функціонування в реальному часі.

З метою підвищення точності перекладу, система буде використовувати методи машинного навчання, зокрема трансформери та механізми уваги, в поєднанні з оптимальним набором даних для навчання моделей. Основним джерелом даних будуть тексти, що включають різноманітні теми та стилі мови для забезпечення максимальної універсальності і точності перекладу.

Необхідною є автоматизація збору та обробки текстових даних для навчання моделей, що включає:

- тексти різних типів (технічні, наукові, бізнес-матеріали);
- мовні особливості, включаючи діалекти, неформальні вирази та специфічну лексику;
- культурні та контекстуальні аспекти, що впливають на точність перекладу.

Отже, сформовані вимоги до програмного забезпечення для автоматизованого перекладу текстів передбачають створення системи, яка поєднує простоту використання, автоматизацію перекладу, адаптацію до контексту та забезпечення високої точності, що дозволить ефективно інтегрувати переклад у веб-додатки.

1.7 Висновки

Машинне навчання та нейромережеві моделі, зокрема трансформери та механізми уваги, стали важливими інструментами для автоматизованого перекладу текстів у веб-додатках. Вони дозволяють забезпечити високу точність перекладу, ефективно обробляти великі обсяги текстів та швидко реагувати на зміни в контексті користувацьких запитів. Однією з головних переваг таких підходів є здатність нейромереж швидко адаптуватися до нових мовних пар і контекстів, забезпечуючи точний переклад навіть для складних або спеціалізованих текстів.

Після аналізу існуючих платформ для автоматизованого перекладу, таких як Google Translate, Microsoft Translator, Amazon Translate і DeepL, можна зробити висновок, що найбільш підходящим для реалізації автоматизованого перекладу в веб-додатках є вибір трансформерних моделей, зокрема таких, як GPT. Вони забезпечують високу точність, зокрема для загальних та технічних текстів, і можуть ефективно працювати з великими обсягами даних. GPT моделі мають здатність швидко адаптуватися до нових контекстів та забезпечують високу швидкість обробки текстів, що робить їх ідеальним рішенням для інтеграції в багатомовні веб-додатки.

2 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗОВАНОГО ПЕРЕКЛАДУ ТЕКСТІВ У ВЕБ-ДОДАТКАХ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

2.1 Задачі підвищення точності перекладу в багатомовних веб-середовищах

У сучасному цифровому світі автоматизований переклад у веб-середовищах став необхідним для спілкування між користувачами з різних культурних та мовних груп. Попри значний прогрес у розвитку нейронних мереж, багато викликів залишаються невирішеними. Зокрема, проблеми точності перекладу можуть виникати через синтаксичні, семантичні та культурні розбіжності між мовами. Розглянемо основні труднощі та підходи до покращення точності перекладу в багатомовних середовищах.

Кожна мова має свої граматичні та синтаксичні правила, які визначають порядок слів у реченні, способи вираження часових та модальних форм, а також узгодження між словами. Наприклад, у багатьох мовах порядок слів SVO (суб'єкт-дієслово-об'єкт) є звичним (наприклад, англійська), тоді як інші мови можуть використовувати порядок SOV (наприклад, японська). Через ці розбіжності автоматизовані системи перекладу часто стикаються із завданням правильного перестановки слів, що впливає на точність перекладу.

Формула для оцінки розбіжностей у порядку слів

Для оцінки точності перекладу враховуючи порядок слів, можна використовувати спеціальні метрики, які враховують розташування n-грам. Однією з таких є метрика Kendall’s, яка вимірює узгодженість порядку слів між еталонним та автоматичним перекладом:

$$T = \frac{\mathrm{P} - \mathrm{Q}}{\frac{n(n-1)}{2}} \tag{2.1}$$

де:

- P — кількість пар, що зберігають порядок слів,
- Q — кількість пар із зворотним порядком,
- n — загальна кількість слів.

Ця формула дозволяє оцінити, наскільки добре система перекладу зберігає логіку оригінального речення.

На рисунку 2.1 показано, як змінюється точність перекладу залежно від синтаксичної структури.

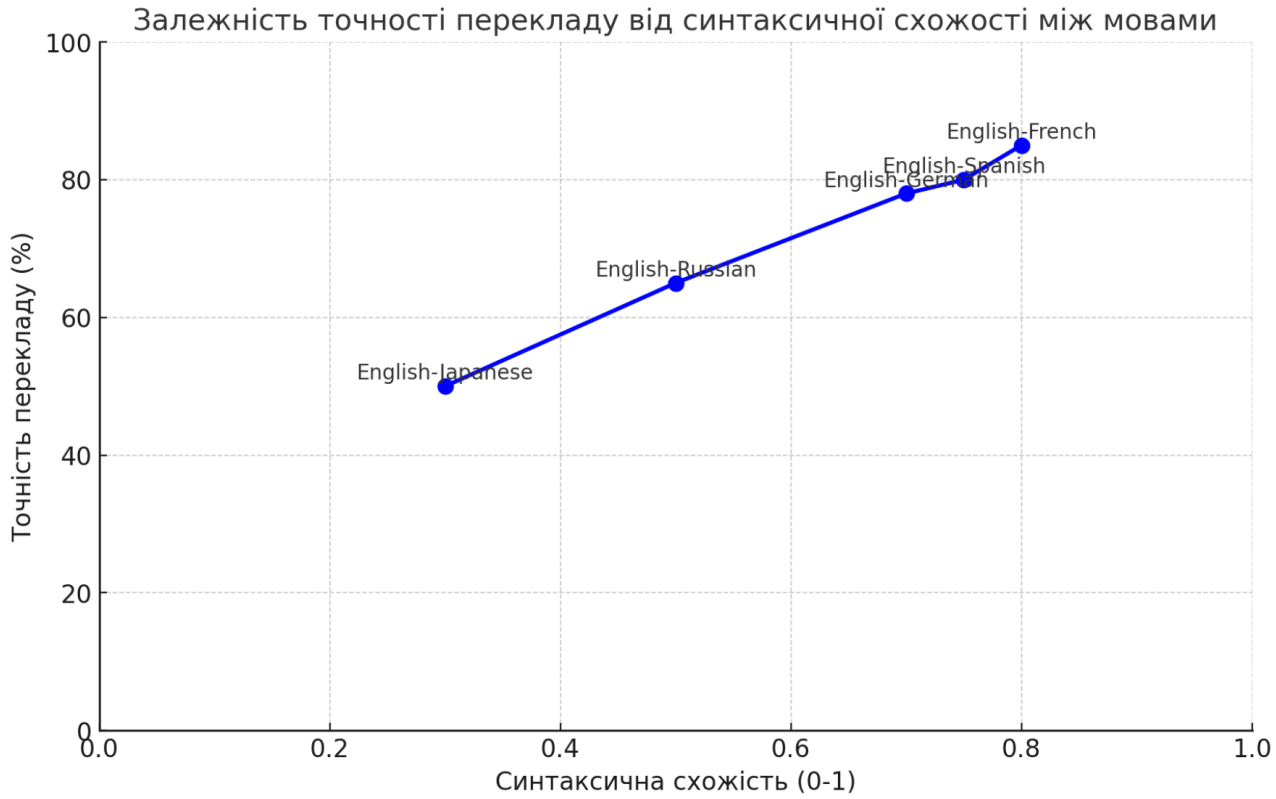


Рисунок 2.1 – Точність перекладу

Багато слів мають різні значення залежно від контексту. Наприклад, слово "spring" в англійській мові може означати "джерело", "весна" або "стрибок". Для точного перекладу необхідно правильно визначити значення слова в конкретному контексті. Нейронні мережі, зокрема, моделі трансформерів, здатні розглядати весь

контекст, що оточує слово, але навіть такі моделі можуть помилятися в складних випадках.

Одним із підходів до вирішення проблеми контексту є використання косинусної подібності для обчислення семантичної близькості між контекстами:

$$Similarity = \cos(\theta) = \frac{\text{A} * \text{B}}{\left||A|\right| * ||B||} \quad (2.2)$$

де:

- cos(θ) - косинус кута θ між векторами A і B;
- A * B — скалярний добуток (або внутрішній добуток) векторів A і B;
- $\left||A|\right| * ||B||$ — нормування векторів A і B, тобто їх довжини або величини.

Цей метод застосовується в перекладі для знаходження значень слів або фраз, які найбільше відповідають оригіналу.

Ідіоми та культурні особливості часто створюють труднощі для автоматизованого перекладу. Наприклад, англійське "break the ice" дослівно означає "розбити лід", однак насправді це ідіома, що означає "почати спілкування". Автоматичні системи перекладу не завжди здатні вловити такі нюанси, що призводить до дослівного, але неточного перекладу.

Для вирішення проблеми культурних відмінностей можуть застосовуватись вагові коефіцієнти для конкретних культурно забарвлених слів чи фраз. Наприклад, при тренуванні моделі можна налаштувати ваги так, щоб ідіоми отримували вищий коефіцієнт при перекладі з однієї мови на іншу.

Для деяких мов, особливо тих, що мають меншу кількість носіїв, даних для тренування моделей часто недостатньо. Наприклад, європейські мови, такі як англійська, німецька та французька, мають значні корпуси текстів для навчання моделей, у той час як менш розповсюджені мови не мають достатньої кількості якісних текстів. Це знижує точність перекладу для таких мов і ускладнює розробку універсальних багатомовних моделей.

На рис. 2.2 показано вплив обсягу навчального корпусу на точність перекладу для різних мов.

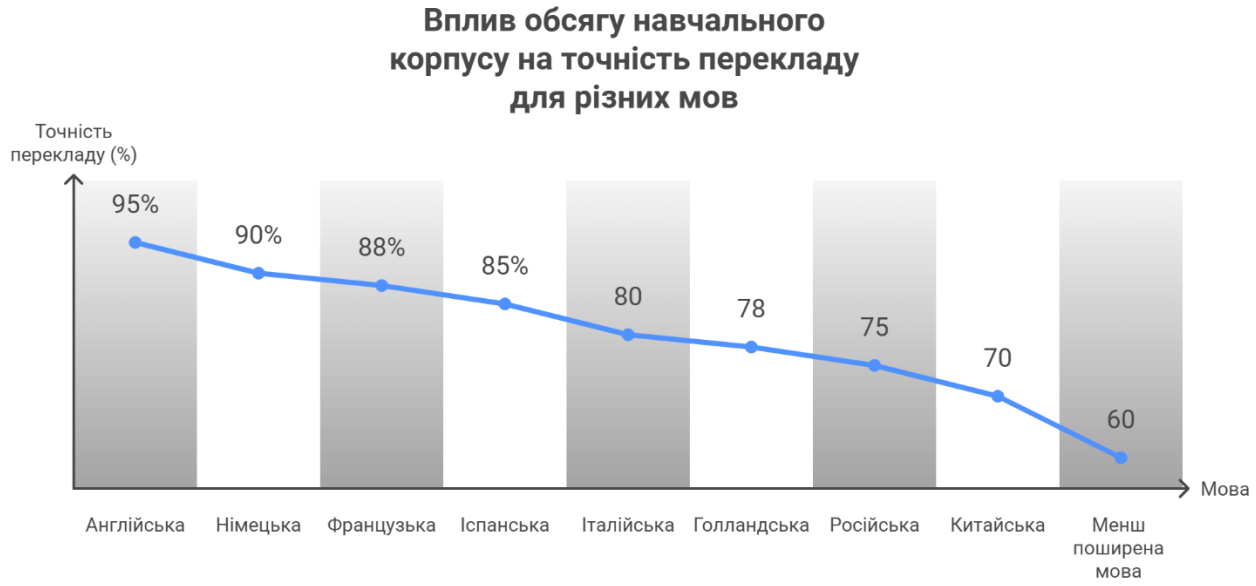


Рисунок 2.2 – Вплив обсягу навчального корпусу на точність

Стиль і тон є важливими компонентами будь-якого тексту. Наприклад, офіційні документи часто вимагають формальної мови, тоді як у неформальному спілкуванні допускається використання сленгу або скорочень. Машинні перекладачі можуть мати труднощі з адаптацією стилю, особливо при перекладі з мов, що мають різні рівні ввічливості (наприклад, японська мова).

При налаштуванні тональності перекладу можуть використовуватись спеціальні коефіцієнти. Наприклад, формула для збереження регістра та формальності може виглядати так:

$$Tone_{Score} = w_f * Formal + w_i * Informal \quad (2.3)$$

де:

- w_f — вага для формального стилю,
- w_i— вага для неформального стилю.

Архітектура трансформерів, особливо за рахунок використання механізму уваги, забезпечує можливість обробляти значні контекстуальні відомості та вибирати релевантні слова для перекладу. Механізм уваги дозволяє моделі звертати увагу на важливі слова в реченні, що є особливо корисним для складних або довгих речень.

Формула механізму уваги у трансформерах

Механізм уваги описується наступною формулою:

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (2.4)$$

де Q, K, V — матриці запитів, ключів і значень, а d_k— розмірність простору ключів. Завдяки цій формулі трансформери можуть обирати найбільш значущі частини тексту для точного перекладу.

Автоматизований переклад у багатомовних веб-середовищах стикається з численними викликами, зокрема з лексичними неоднозначностями, збереженням культурного контексту та адаптацією до стилю тексту. Використання трансформерів із механізмом уваги дозволяє значно покращити точність перекладу, проте повне вирішення проблем перекладу вимагає глибшого налаштування моделей для врахування мовних та культурних особливостей. Подальший розвиток технологій та збільшення корпусів даних для рідкісних мов сприятиме подоланню бар'єрів у багатомовних веб-додатках.

2.2 Адаптація до контексту та культурних особливостей

Однією з найбільших проблем при автоматизованому перекладі є необхідність адаптації перекладу до специфічних контекстів і культурних відмінностей. Навіть за умови високої точності алгоритмів машинного перекладу, адаптація до місцевих традицій, культурних нюансів та специфічного контексту

залишається складним завданням. Пояснимо основні труднощі, які виникають у цьому процесі.

Культурні відмінності впливають на вибір слів і фраз у процесі перекладу, що може призвести до втрати або спотворення сенсу. Наприклад, англійська фраза "break the ice" (буквально "розламати лід") у культурі англомовних країн має значення "порушити мовчання в складній ситуації". У мовах, де цей вираз не має прямого еквівалента, переклад може бути досить далеким від першочергового сенсу.

Адаптація контексту для культурно-забарвлених висловів у нейронних мережах вимагає врахування культурних особливостей мови. Для цього використовуються технології навчання на великих корпусах текстів з культурними специфіками, що дозволяє точніше вгадувати ідентичність виразів у перекладі [1].

Слова часто мають різні значення залежно від контексту. Наприклад, слово "bat" в англійській мові може означати як "кажан", так і "бита для бейсболу". Для точного перекладу необхідно визначити правильне значення слова в контексті речення. Використання моделей на основі семантичної подібності, таких як трансформери, допомагає вирішити ці проблеми.

Один із способів вирішення цих проблем — це використання адаптивних моделей, що включають культурні контексти в процес перекладу. Моделі на основі трансформерів, такі як BERT [3] або T5 [4], дозволяють враховувати контекст, що сприяє більш точному перекладу в межах специфічних культур.

Для кращого розуміння процесу роботи, можна звернути увагу на рисунок 2.3, що показує використання адаптивних моделей, що включають культурні контексти в процес перекладу.

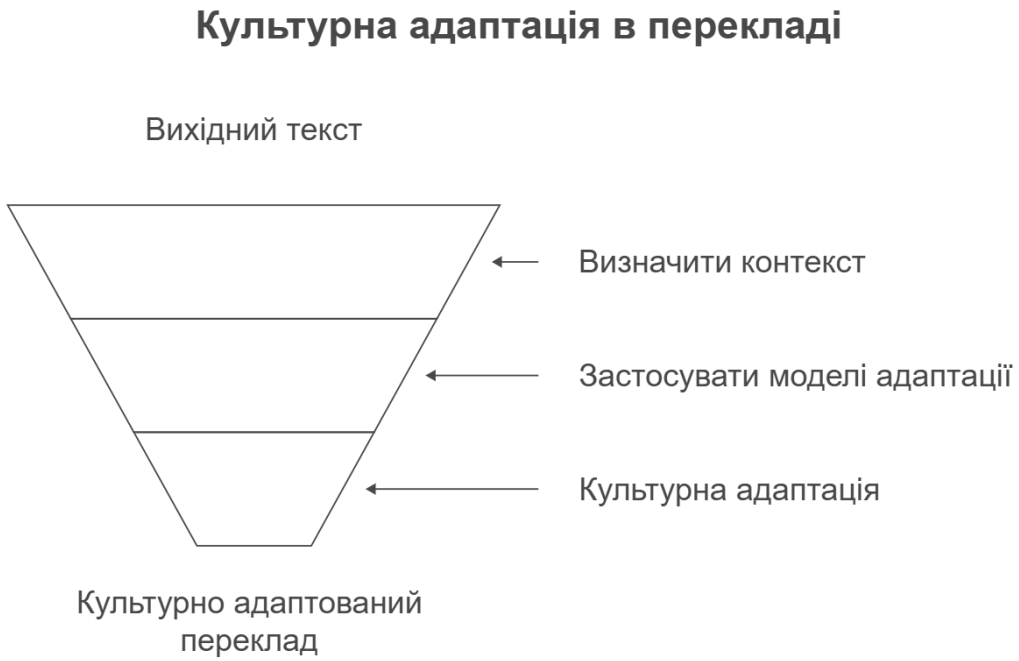


Рисунок 2.3 – Використання адаптивних моделей;

Трансформер є архітектурою, яка застосовується для багатьох завдань у машинному навчанні, включаючи переклад. Вона використовує механізм уваги, який дозволяє моделі фокусуватися на важливих частинах тексту.

Механізм уваги є ключовим у трансформерах і допомагає моделі вибирати важливі частини вхідного тексту. Він працює за принципом побудови ваг для кожного слова або фрагмента тексту, що дозволяє точно передавати значення в перекладі. На рисунку 2.3 можна спостерігати механізм уваги трансформера.

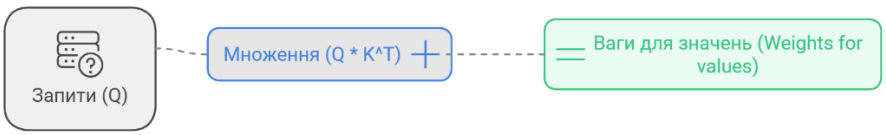


Рисунок 2.4 – Механізм уваги трансформера;

Формула механізму уваги в трансформері виглядає так:

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.5)$$

де:

- Q — матриця запитів,
- K — матриця ключів,
- V — матриця значень,
- d_k — розмірність простору ключів.

Цей механізм дозволяє краще інтерпретувати віддалені залежності в текстах, що є важливим для точності перекладу.

Не всі мови мають однакові конструкції, що ускладнює переклад. Деякі мови можуть мати відмінності в порядку слів, у вираженні часу, тощо. Це створює труднощі для побудови універсальних моделей перекладу. Одним із підходів є використання гібридних моделей, що поєднують традиційні методи з нейронними мережами. Наприклад, модель Seq2Seq [5] показала хороші результати у завданнях, що вимагають гнучкості при перекладі.

Таблиця 2.1 показує приклади культурно-забарвлених висловів для різних мов та їх адаптовані переклади:

Таблиця 2.1 – Приклади культурно-забарвлених висловів

Мова	Вираз	Переклад	Примітка
Англійська	Break the ice	Почати розмову	Використовується в соціальних контекстах для розв'язання напруги
Японська	頑張って (Ganbatte)	Бажаю удачі	Мотиваційний вираз з емоційним забарвленням
Іспанська	Ser pan comido	Легко як хліб	Мотиваційний вираз з емоційним забарвленням

Адаптація до контексту та культурних особливостей є важливим аспектом у машинному перекладі, який вимагає врахування лексичних, граматичних та культурних відмінностей між мовами для досягнення точності і збереження

значення оригінального тексту. Використання трансформерів та адаптивних моделей значно покращує ці процеси в багатомовних веб-додатках.

2.3 Удосконалення методу автоматизованого перекладу текстів у веб-додатках на основі нейромереж

Для покращення точності автоматизованого перекладу в багатомовних веб-додатках важливе значення має вибір відповідних методів нейромереж. Сучасні методи, зокрема трансформери та моделі з механізмами уваги, значно підвищують точність і адаптивність перекладу. Особливу роль у цьому контексті відіграє використання великих передтренованих мовних моделей, таких як GPT (Generative Pretrained Transformer), які можуть забезпечити високу точність навіть без додаткового навчання на конкретних корпусах текстів.

Щоб визначити найкращу моделей перекладу було порівняно аналоги між собою такі як DeepL, Google Translate, GPT-3.5, GPT-4. Було обранообрав 12 популярних мов: іспанську, китайську (мандарин), російську, французьку, німецьку, японську, португальську, корейську, нідерландську, гінді, індонезійську та арабську. Для кожної з них протестовано переклади, використовуючи кожного з кандидатів. Як метрику якості перекладу Було використано стандарт — метрику BLEU-4[30].

BLEU-4 (Bilingual Evaluation Understudy) — це метрика автоматичної оцінки якості машинного перекладу. Вона вимірює схожість між перекладом, створеним моделлю, і еталонними перекладами (людськими). BLEU-4 враховує до чотирислівних послідовностей (4-грам), що дозволяє краще оцінювати точність і плавність перекладу, зокрема збереження структури фраз. Результат виражається в діапазоні від 0 до 1 (або 0–100%), де вищі значення означають більшу якість.

Було виконано візуалізацію розподілу метрики BLEU для кожної мовної пари у вигляді графіків типу «ящик із вусами». Жирна лінія, розташована всередині

ящика, відображає медіану, межі ящика відповідають 25-му та 75-му процентилю. Трикутний виріз у центрі («notch») вказує на довірчий інтервал для медіани.

Переклад з іспанської на англійську є відносно нескладним завданням, і всі моделі демонструють високу ефективність. Аналогічна ситуація спостерігається й для інших романських мов. Результат було візуалізован на рисунку 2.5.

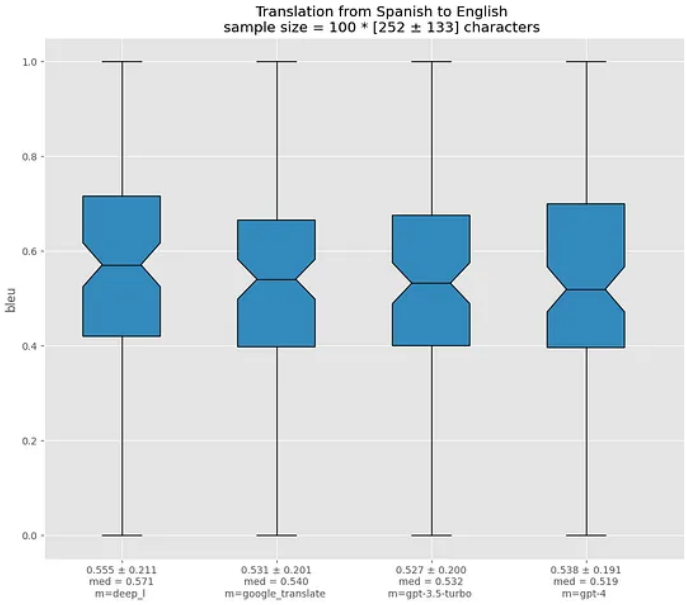


Рисунок 2.5 – Візуалізація порівння моделей англійська-іспанська;

Для російської мови моделі GPT-3.5 та GPT-4 показали найгірші результати за метрикою BLEU порівняно з іншими системами. Це, ймовірно, пояснюється труднощами з токенізацією текстів, написаних кирилицею. ЩО можна спостерігати на рисунку 2.6.

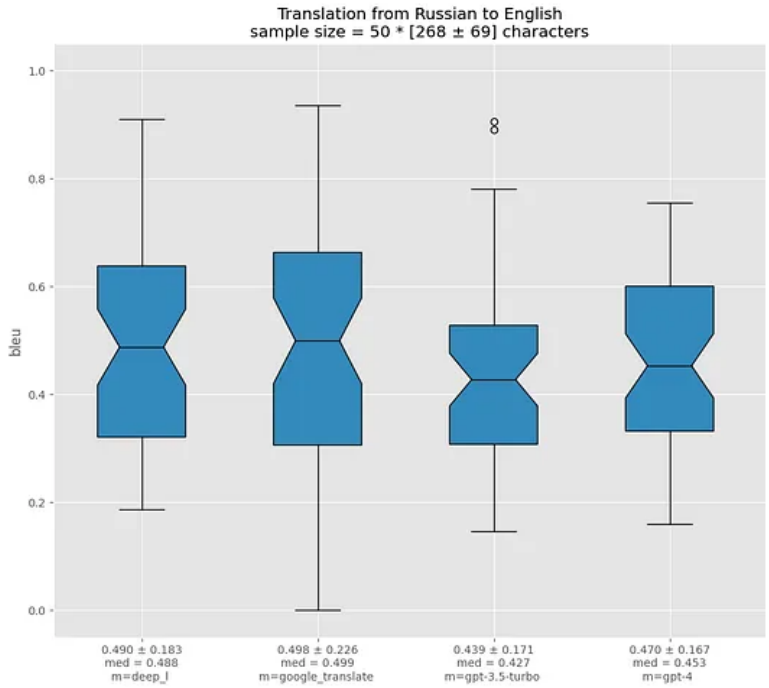


Рисунок 2.6 – Візуалізація порівння моделей російська-англійська;

Для моделі DeepL переклад із японської виявився не важким. Натомість GPT-3.5 і GPT-4 демонструють високі результати для цієї мовної пари, забезпечуючи якісний та точний переклад. Що можна спостерігати на рисунку 2.7.

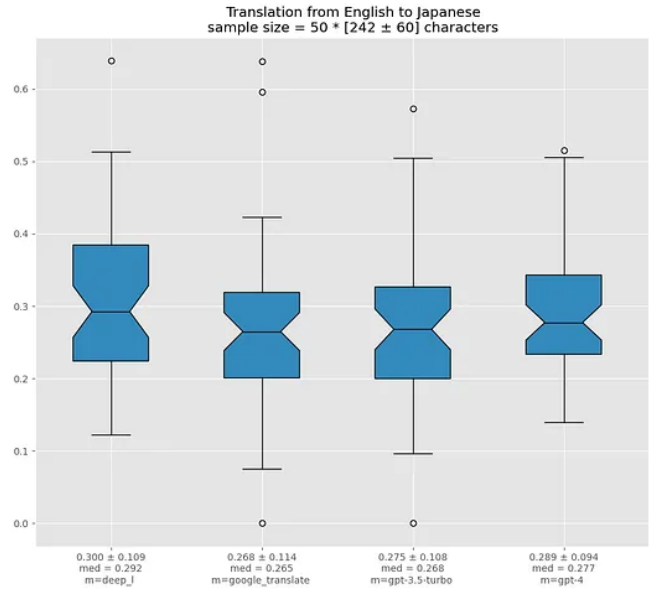


Рисунок 2.7 – Візуалізація порівння моделей Англійська-Японська;

Google Translate та DeepL значно переважають OpenAI за швидкістю обробки запитів. Крім того, їхні API в даний час є набагато стабільнішими: сервери OpenAI перебувають під великою навантаженістю, що іноді призводить до помилок під час обробки запитів, і ці запити доводиться повторно надсилати. Очікується, що ці технічні недоліки будуть усунені в майбутньому.

Проблеми зі швидкістю в OpenAI частково компенсуються можливістю потокового отримання відповіді, коли модель надсилає результати поступово, токен за токеном, що зменшує час очікування для користувачів.

Однією з переваг використання GPT-3.5/GPT-4 для перекладу є можливість подальшої кастомізації під конкретні потреби:

- Можна змінювати промт, що дозволяє виконувати переклади в певному стилі.
- Є можливість генерувати різні варіанти перекладу за допомогою різних промтів та/або встановлення температури > 0. Потім можна вибирати найкращий переклад за певними критеріями, наприклад, за семантичним збігом з оригіналом, який може бути обчислений іншою нейронною мережею.

Переваги використання ChatGPT для перекладу:

- Універсальність: Модель підтримує численні мови, що робить її підходящою для багатомовних веб-додатків.
- Адаптивність: ChatGPT здатний враховувати культурні та контекстуальні особливості, що підвищує точність перекладу.
- Простота інтеграції: Завдяки API ChatGPT можна легко інтегрувати цю модель у веб-додатки для автоматизованого перекладу без необхідності в додатковому навчанні.

Трансформери, зокрема ті, що використовуються в моделях на зразок ChatGPT, дозволяють ефективно обробляти текст завдяки використанню механізму уваги. Це дозволяє моделі фокусуватися на важливих для контексту частинах

речення, забезпечуючи високий рівень точності навіть при складних синтаксичних конструкціях.

Механізм уваги є основою архітектури трансформерів і дозволяє моделі виділяти значущі елементи в реченні або абзаці, що забезпечує кращу обробку контексту та точність перекладу. Для ефективного перекладу важливо, щоб модель могла зберігати залежності між словами, навіть якщо вони знаходяться далеко одне від одного в тексті.

Для покращення якості перекладу було використано модель OpenAI. Використовуючи підрахунки BLEU-4 можна порахувати наступне :

Формула BLEU-4:

$$\text{BLEU-4} = BP \times \exp\left(\sum_{n=1}^{4} w_n \cdot \log P_n\right)$$

де:

- BP — штраф за довжину перекладу (Brevity Penalty), що забезпечує покарання для коротших перекладів, ніж оригінал.
- P_n— точність n-грам для кожного n (1, 2, 3, 4).
- W_n — ваги для кожного значення n (зазвичай 1/4 для кожного).

Для мовної пари (наприклад, з англійської на українську), результати BLEU-3 для трьох систем можна побачити у таблиці 2.2.

Таблиця 2.2 – Blue-4 англійської на українську

Модель	BLEU-4
DeepL	0.45
OpenAI (GPT-4)	0.48
Google Translate	0.42

Результату показав на 4-5% кращого за метрикою BLEU-4, OpenAI GPT-4.

Вибір моделей, таких як ChatGPT, які базуються на архітектурі трансформерів та механізмах уваги, є оптимальним для досягнення високої точності перекладу в багатомовних веб-додатках. Модель ChatGPT вже має вбудовані механізми для роботи з контекстом і культурними особливостями, що робить її ідеальним рішенням для автоматизованого перекладу.

2.4 Використання моделей трансформерів і механізмів уваги для перекладу складних текстів

Моделі трансформерів із механізмами уваги стали основою для сучасних методів машинного перекладу, зокрема таких, що використовуються в ChatGPT. Ці моделі дозволяють здійснювати переклад складних текстів, зберігаючи контекстуальну точність та коректність, що є важливим для багатьох сфер, таких як наука, техніка, право та культура. Завдяки здатності трансформерів до обробки великих обсягів тексту в одному проході і використанню механізмів уваги, переклад складних текстів стає точнішим, ніж за допомогою традиційних моделей машинного перекладу.

Основна перевага трансформерів полягає в їх здатності обробляти всю послідовність тексту за один раз, без необхідності послідовної обробки слів. Це дозволяє моделі значно ефективніше враховувати контекст, що особливо важливо для перекладу складних і багатозначних текстів. Одним із ключових компонентів цієї архітектури є механізм уваги, який дозволяє моделі зосереджуватися на різних частинах тексту в залежності від важливості кожного слова або фрази.

У трансформерних моделях, таких як GPT, увага реалізована через так звану механізм багаторазової уваги (Multi-head Attention), який дозволяє моделі одночасно звертати увагу на різні аспекти тексту.

Механізм багаторазової уваги (Multi-head Attention) забезпечує одночасну обробку кількох запитів уваги, що дає змогу моделі "дивитися" на різні частини

тексту одночасно. Це дозволяє трансформеру враховувати різні контексти і синтаксичні структури в одному вході.

Формула багаторазової уваги в трансформерах:

$$\text{Multi} - \text{head Attention}(\text{Q}, \text{K}, \text{V}) = Concat(head_1, head_2 \ldots\ldots, head_h)W^O \quad (2.6)$$

де:

- Q,K,V — матриці запитів (queries), ключів (keys) та значень (values),
- $head_i = \text{Attention}(QW^Q{}_i, KW^k{}_i, VW^V{}_i,$ — кожна «голова» механізму уваги обчислює свою увагу за різними параметрами,
- W^O— матриця перетворення, що об'єднує результати різних голов у єдиний вихід.

Ця формула дозволяє моделі враховувати різні аспекти контексту для кожного слова або фрази, забезпечуючи більш точний переклад при складних структурних елементах.

При перекладі складних текстів, таких як технічні інструкції чи юридичні документи, точність перекладу є важливою. Трансформери з механізмом багаторазової уваги дозволяють зберігати значення термінів і фраз, які можуть змінюватися в залежності від контексту. Наприклад, термін "bank" може означати або фінансову установу, або берег річки, і трансформер здатен вибрати правильне значення на основі контексту.

Приклад:

- "The bank of the river" (берег річки) — в цьому контексті трансформер визначає, що «bank» означає саме берег, а не фінансову установу.

Завдяки використанню багаторазової уваги, модель здатна ефективно орієнтуватися на слова, що забезпечують ключові змісти фрази, і правильно обирати значення термінів залежно від їх розташування в контексті.

Моделі трансформерів із механізмами багаторазової уваги є потужним інструментом для перекладу складних текстів. Вони здатні забезпечити високий рівень точності, зберігаючи контекст і визначаючи правильне значення термінів залежно від їхнього місцезнаходження в тексті. Однак для ефективного застосування цих моделей необхідне достатнє забезпечення даними та врахування специфіки тексту.

2.5 Висновки

Автоматизований переклад у веб-додатках, використовуючи нейромережеві технології, стикається з низкою викликів, зокрема з точністю перекладу, що залежить від синтаксичних, семантичних та культурних розбіжностей між мовами. Використання трансформерних моделей, таких як

GPT та T5, дозволяє покращити точність перекладу завдяки здатності моделей розуміти контекст та обробляти складні синтаксичні конструкції. Однак навіть найсучасніші методи не завжди можуть подолати проблеми, пов'язані з лексичними неоднозначностями або збереженням культурних відмінностей у текстах.

Адаптація перекладу до культурних контекстів і специфіки мови є одним із ключових аспектів автоматизованого перекладу. Проблеми виникають через відсутність точних відповідників для деяких виразів або культурно забарвлених фраз. Наприклад, ідіоми та фразеологізми часто мають значення, яке важко передати дослівно. Для вирішення цієї проблеми використовуються адаптивні моделі, що дозволяють зберігати культурний контекст і тональність мови, враховуючи мовні особливості та контекстуальні нюанси.

Інтеграція передтренованих трансформерних моделей, таких як ChatGPT, у веб-додатки для автоматизованого перекладу дозволяє досягти високої точності, забезпечуючи універсальність та адаптивність перекладу. Завдяки використанню механізму уваги, трансформери можуть виділяти важливі частини тексту, що

дозволяє зберігати точність навіть при складних мовних конструкціях. Однак для покращення якості перекладу необхідно продовжувати вдосконалювати ці технології, зокрема шляхом розширення бази даних для менш поширених мов та адаптації моделей до специфічних культурних контекстів.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ АВТОМАТИЗОВАНОГО ПЕРЕКЛАДУ ТЕКСТІВ У ВЕБ-ДОДАТКАХ

3.1 Проектування архітектури веб-додатку для перекладу текстів

Веб-додатки для автоматизованого перекладу текстів є важливими інструментами для швидкої локалізації контенту, особливо для інтернаціональних веб-платформ. У цьому розділі буде описано проектування архітектури такого додатку, з використанням сучасних підходів до машинного перекладу, а також розглянуто взаємодію компонентів системи та способи оптимізації її роботи.

Автоматизований переклад текстів стає все більш популярним завдяки розвитку технологій машинного навчання та нейромереж. Застосування таких технологій дозволяє забезпечити точний переклад, що зберігає контекст і значення оригінального тексту. Моделі, побудовані на архітектурі **Transformer**, зокрема GPT-4 від OpenAI, стали основою для багатьох сучасних систем машинного перекладу, що використовуються в реальних веб-додатках.

Для реалізації веб-додатку для перекладу текстів важливо правильно спроектувати архітектуру системи. Вона повинна включати компоненти для обробки запитів, інтеграції з API, кешування результатів і підтримки користувацького інтерфейсу.

Існують кілька основних підходів до автоматизованого перекладу текстів у веб-додатках:

- Моделі послідовність-послідовність (Seq2Seq): Ранні моделі машинного перекладу, які використовували дві рекурентні нейронні мережі для кодування і декодування тексту. Ці моделі мають обмеження у роботі з довгими залежностями.
- Механізм уваги (Attention Mechanism): Механізм, який дозволяє моделі "зосереджуватись" на різних частинах тексту під час перекладу, що значно покращує якість перекладу, зокрема для складних речень і фраз.

- Трансформер (Transformer): Базова архітектура для сучасних моделей машинного перекладу. Вона застосовує механізм самостійної уваги (self-attention), що дозволяє ефективно працювати з довгими текстами та зберігати контекст на всіх етапах перекладу.
- Генеративні моделі (Generative Models): Моделі, такі як GPT, генерують текст на основі заданого контексту. Вони забезпечують високу точність перекладу, оскільки здатні адаптуватися до різних стилів мови та специфічних запитів.

Архітектура веб-додатку для перекладу текстів

Основні компоненти архітектури веб-додатку для автоматизованого перекладу текстів, що використовує API OpenAI:

- Інтерфейс користувача (UI). Простий і зручний для введення тексту та вибору мови перекладу. Для інтеграції цього компонента використовується веб-фреймворк (наприклад, React або Vue.js).
- Модуль обробки запитів (API Handler). Це компонент, що відповідає за взаємодію з API OpenAI, формує запити та отримує результати перекладу.
- Система кешування. Для оптимізації роботи та зменшення витрат на API запити, результат перекладу зберігається в базі даних. Це дозволяє прискорити подальші запити з однаковим текстом.
- База даних. Використовується для збереження історії перекладів, даних користувачів та кешованих результатів. Для цього можна застосувати реляційну базу даних (наприклад, MS SQL).

MS SQL Server[16] є реляційною системою управління базами даних (RDBMS), яка забезпечує зберігання даних у таблицях з чітко визначеними зв'язками. Це робить її відмінним вибором для нашої моделі бази даних, що включає численні взаємозв'язки між таблицями, такими як користувачі, групи, пости та повідомлення. Продуктивність MS SQL Server вражає завдяки підтримці

паралельної обробки запитів, що дозволяє швидко отримувати доступ до даних навіть у умовах високого навантаження. Крім того, система пропонує різноманітні типи індексів, включаючи повнотекстові індекси, що значно прискорює виконання запитів на пошук і фільтрацію. Вбудований оптимізатор запитів автоматично вибирає найбільш ефективний план виконання запиту, що додатково підвищує загальну продуктивність системи.

Одним із ключових аспектів MS SQL Server є безпека, яка забезпечується завдяки підтримці різноманітних механізмів автентифікації, таких як Windows Authentication і SQL Server Authentication, які дозволяють гнучко керувати правами доступу до бази даних. Для захисту чутливих даних система пропонує такі методи шифрування, як Transparent Data Encryption (TDE) та Always Encrypted, що гарантує високий рівень безпеки навіть під час збереження та передачі інформації. Крім того, детальне налаштування ролей та прав доступу для різних категорій користувачів забезпечує надійний контроль за доступом до даних, що є важливим для відповідності сучасним стандартам безпеки та конфіденційності[16].

MS SQL Server також забезпечує високу масштабованість, що є важливим для систем автоматизованого перекладу текстів у веб-додатках на основі нейромереж. Система підтримує як вертикальне, так і горизонтальне масштабування, що дозволяє їй адаптуватися до зростаючих обсягів даних і навантаження в процесі обробки запитів. Функції реплікації сприяють ефективному розподілу навантаження та оптимізації доступу до даних, що є важливими для забезпечення швидкості та якості перекладу. Також варто зазначити надійність підтримки та документації для MS SQL Server. Широка спільнота користувачів, безкоштовні ресурси та професійна підтримка від Microsoft гарантують стабільність системи. Докладна документація на офіційному сайті Microsoft охоплює всі аспекти роботи з MS SQL Server, що полегшує навчання та вирішення проблем, необхідних для розробки ефективного рішення автоматизованого перекладу.

Останнім часом MS SQL Server пропонує функції, які можуть бути корисними для системи автоматизованого перекладу текстів. Інструменти, такі як SQL Server Reporting Services (SSRS), дозволяють створювати звіти для аналізу ефективності перекладів, що може бути важливим для оптимізації роботи системи. SQL Server Integration Services (SSIS) забезпечують можливості інтеграції даних, що може бути корисним при обробці та кешуванні перекладів з різних джерел. На основі проведеного аналізу, можна стверджувати, що MS SQL Server є оптимальним вибором для розробки невеликої бази даних для кешування перекладів, завдяки своїй реляційній архітектурі, високій продуктивності, безпеці та можливостям інтеграції. Ця платформа забезпечить ефективне управління даними, що сприятиме зручності та швидкості доступу до кешованих перекладів.

База даних для системи кешування перекладів текстів організована навколо кількох основних таблиць, які взаємодіють одна з одною для забезпечення ефективного управління оригінальними текстами, перекладами та мовами.

Таблиця SourceTexts містить ключову інформацію про оригінальні тексти, які підлягають перекладу. Вона включає такі поля, як унікальний ідентифікатор, сам текст, код мови оригіналу та дату створення запису. Це дозволяє зберігати оригінальні тексти в централізованому місці, спрощуючи їх оновлення та повторне використання.

Таблиця Translations служить для зберігання перекладів оригінальних текстів. Вона містить поля, такі як унікальний ідентифікатор перекладу, ідентифікатор оригінального тексту (посилання на таблицю SourceTexts), перекладений текст, код мови перекладу та дату створення запису. Це дозволяє уникнути дублювання оригінальних текстів і полегшує їх оновлення, адже зміна оригінального тексту автоматично оновлює всі відповідні переклади.

Враховуючи наведені складові архітектури системи, що розробляється, було розроблено UML-діаграму розгортання програмного забезпечення. Розроблену UML-діаграму розгортання наведено на рис. 3.1.

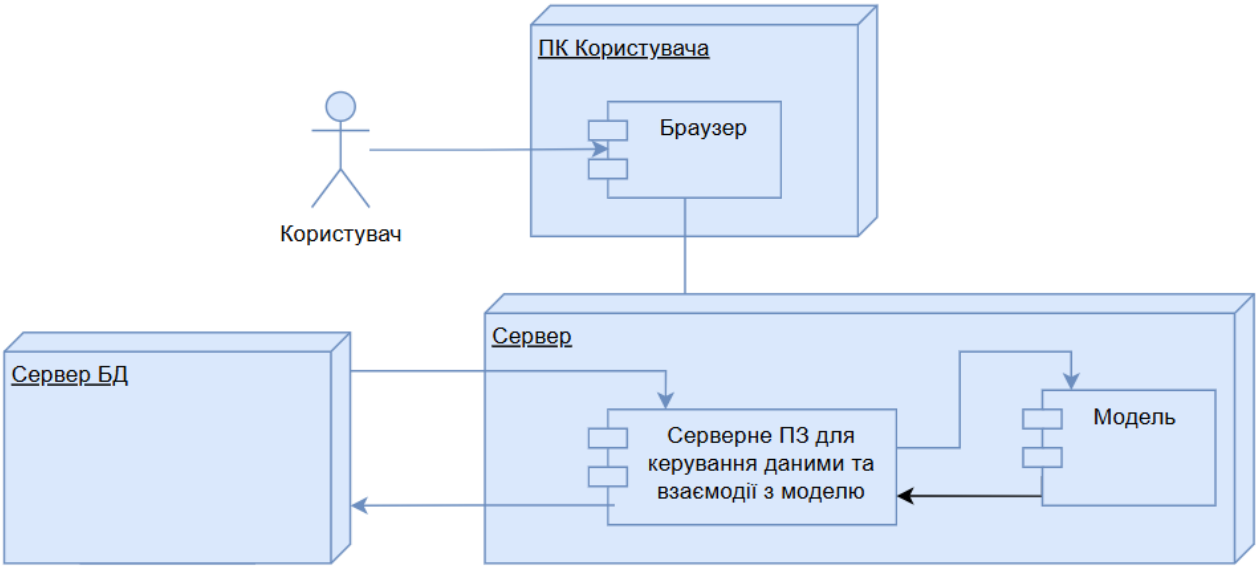


Рисунок 3.1 – UML-діаграма розгортання системи

Усі ці таблиці пов'язані між собою через встановлені зв'язки, що забезпечує цілісність даних і дозволяє виконувати запити для аналізу та звітності. Зв'язок між таблицями дозволяє, наприклад, швидко отримати всі переклади для конкретного оригінального тексту або визначити, які мови підтримуються для конкретного перекладу. Цей підхід не лише спрощує доступ до інформації, але й підвищує ефективність роботи бази даних, оскільки користувачі можуть здійснювати складні запити без значних витрат часу.

Крім того, налагоджені зв'язки між таблицями сприяють швидшій обробці запитів, що є важливим для систем, які працюють з великими обсягами даних. Це дозволяє зменшити навантаження на сервери і поліпшити загальну продуктивність системи. Таким чином, структурованість бази даних не тільки підвищує її зручність для користувачів, але й дозволяє зберігати та аналізувати дані більш ефективно, що в умовах зростаючих вимог до швидкості та точності інформації є важливим аспектом для забезпечення конкурентоспроможності системи.

Для створення ефективної системи кешування перекладів у веб-додатках,

побудованих на нейромережах, важливо враховувати як архітектурні, так і технічні аспекти, щоб забезпечити надійність, масштабованість та простоту використання. Однією з моделей, яка допоможе досягти цих цілей, є архітектура MVVM (Model-View-ViewModel), яка розділяє бізнес-логіку, інтерфейс користувача і модель даних. Це дозволяє зробити систему більш гнучкою, спрощує процес тестування і підвищує підтримуваність, що є надзвичайно важливим для динамічних веб-додатків.

Детальний робота архітектури MVVM зображена на рисунку 3.2.

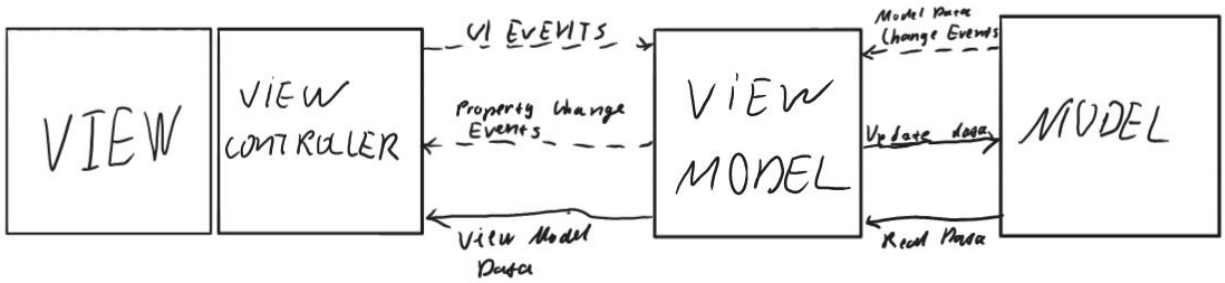


Рисунок 3.2 – Архітектура MVVM

Архітектура MVVM включає три ключові компоненти: Model, View та ViewModel. Model представляє дані та бізнес-логіку системи. В рамках системи для поширення інформації в соціальних мережах, модель може містити інформацію про користувачів, пости, аналітичні дані та інші важливі елементи. Цей компонент взаємодіє з базою даних та зовнішніми джерелами для отримання і обробки даних, забезпечуючи основу для бізнес-логіки та процесів системи.

View відповідає за відображення інформації користувачам, а ViewModel містить логіку відображення і взаємодії з даними, що дає змогу з'єднувати модель з інтерфейсом користувача, не порушуючи цілісності архітектури [17]. Таким чином, MVVM дозволяє спростити процес розробки, тестування та підтримки програмного забезпечення, зокрема в контексті динамічних та інтерактивних систем соціальних

мереж.

Архітектура MVVM є важливим аспектом, оскільки вона забезпечує чітке розділення логіки та інтерфейсу, що сприяє кращій структурі коду та його підтримці. Це дозволяє розробникам легко масштабувати проєкти, додавати нові функції та компоненти, не порушуючи наявний код [18].

Детальна діаграма пакетів зображена на рисунку 3.3.

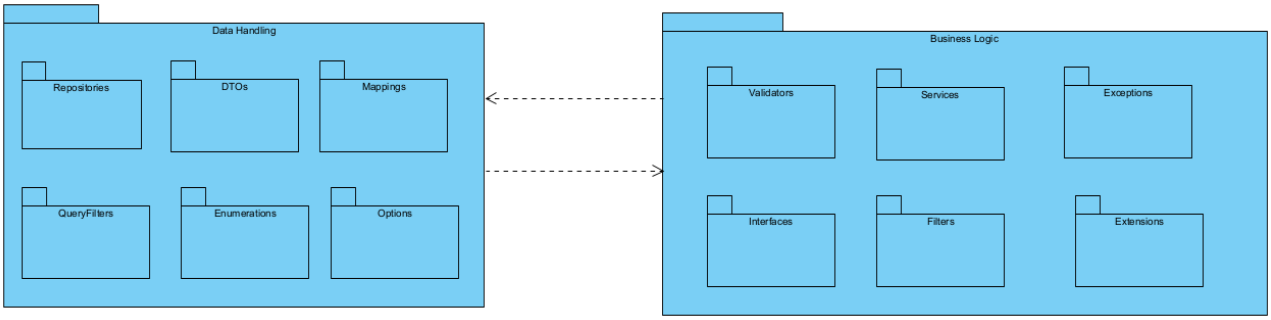


Рисунок 3.3 – Детальна діаграма пакетів

Для створення архітектури системи автоматизованого перекладу текстів у веб-додатках можна розділити компоненти на два основні пакети: "Data Handling" (Обробка даних) і "Business Logic" (Бізнес-логіка). Це дозволить відокремити компоненти, які займаються обробкою даних, від тих, що виконують бізнес-логіку. Такий підхід сприяє кращому управлінню системою та спрощує процес розробки, тестування і підтримки.

Пакет 1: Data Handling

Цей пакет охоплює всі компоненти, пов’язані з управлінням, зберіганням і передачею даних у системі:

- Repositories: відповідають за взаємодію з базою даних, зберігання і отримання сутностей.
    - DBContext: клас для управління транзакціями і доступом до бази даних.
    - Entities: сутності бази даних, що представляють таблиці з перекладами,

текстами та іншими даними.

- DTOs (Data Transfer Objects): використовуються для передачі даних між бізнес-логікою та інтерфейсом користувача.
- Mappings: забезпечує перетворення даних між бізнес-логікою та DTO, підтримуючи узгодженість і відповідність даних.
- QueryFilters: застосовуються для фільтрації запитів до бази даних, обмежуючи або модифікуючи результати відповідно до встановлених критеріїв.
- Enumerations: містять фіксовані константи для використання в різних частинах системи, наприклад, коди мов.
- Options: зберігають конфігураційні налаштування, які можуть бути змінені під час роботи програми або на рівні налаштувань середовища.

Пакет 2: Business Logic

Цей пакет відповідає за управління бізнес-процесами, обробку запитів та їхню валідацію:

- Validators: класи для перевірки вхідних даних перед їх передачею до інших компонентів системи.
- Services: логіка управління операціями і процесами, що пов'язані з обробкою запитів на переклад.
- Interfaces: визначають інтерфейси для різних компонентів, забезпечуючи можливість змінювати реалізації відповідно до принципу інверсії залежностей (Dependency Inversion).
- Filters: фільтри для попередньої обробки даних, наприклад, авторизації або логування.
- Exceptions: класи для обробки помилок, що виникають у процесі роботи різних компонентів.
- Extensions: розширення, які додають нову функціональність або модифікують поведінку наявних компонентів.

Взаємодія між пакетами

Пакет "Data Handling" відповідає за всі операції з даними — їх зберігання, доступ і передачу. Компоненти цього пакета абстрагують доступ до бази даних через репозиторії та фільтри запитів. Пакет "Business Logic" опрацьовує ці дані, керує потоками інформації, валідує їх і обробляє винятки. Обидва пакети взаємодіють через DTO, забезпечуючи чіткий розподіл завдань між управлінням даними та бізнес-логікою [19][20].

3.2 Вибір мови програмування та інструментів для реалізації веб-додатку

3.2.1Вибір мови програмування

Для розробки системи поширення інформації в соціальних мережах було обрано мову програмування C#/.NET. Такий вибір обумовлений багатими можливостями платформи .NET, включаючи її кросплатформеність і підтримку різних операційних систем. Додатково, .NET пропонує потужні бібліотеки для автоматизації, як-от Selenium, що робить його ідеальним для розробки програмного забезпечення з високими вимогами до масштабованості та продуктивності [10].

Порівнюючи .NET з іншими мовами програмування, такими як Python, можна виділити кілька суттєвих переваг. По-перше, .NET забезпечує високу продуктивність завдяки компіляції коду у байт-код, який виконується на платформі .NET. На відміну від цього, Python є інтерпретованою мовою, що може призводити до зниження швидкості виконання у великих проєктах. По-друге, .NET має розвинену екосистему інструментів і бібліотек, що значно спрощує інтеграцію з різними технологіями, такими як бази даних, веб-служби та інші системи [11].

Крім того, використання Selenium разом з .NET надає можливість розробки надійних і масштабованих рішень для автоматизації процесів у соціальних мережах. Це є особливо актуальним для нашого проекту, де необхідно

виконувати авторизацію, навігацію та взаємодію з різними елементами веб-сторінок. У порівнянні з Python, який також підтримує Selenium, .NET пропонує кращу інтеграцію з Windows-середовищем, що може мати вирішальне значення для компаній, що працюють на цій платформі. Також, завдяки потужній екосистемі .NET, розробники отримують доступ до додаткових інструментів і бібліотек, що спрощують автоматизацію та розширюють функціональність рішень. Це дозволяє швидше реагувати на зміни вимог та забезпечувати високу продуктивність у процесах автоматизації.

В таблиці 3.1 наведено більш розгорнуту порівняльну характеристику C#/.NET та Python.

Таблиця 3.1 − Основні відмінності між C# і Python

C#	Python
C# має деструктори для автоматичного очищення ресурсів при знищенні об'єктів.	Python використовує методи del для деструкції об'єктів, але очищення ресурсів зазвичай реалізується за допомогою контекстних менеджерів (через with).
C# використовує статичну типізацію, що забезпечує високий рівень безпеки типів. Всі змінні мають визначений тип на етапі компіляції.	Python, навпаки, є мовою з динамічною типізацією, що дозволяє змінювати типи змінних під час виконання. Хоча це надає гнучкість, воно може призводити до помилок, які виявляються лише під час виконання.
C# підтримує інтерфейси, що дозволяє реалізувати множинне наслідування через інтерфейси.	Python підтримує множинне наслідування класів, дозволяючи класу успадковувати властивості та методи від кількох батьківських класів.
C# має сувору систему управління пам'яттю з автоматичним збором сміття, що підвищує безпеку та продуктивність.	Python також має автоматичний збір сміття, проте це може призводити до затримок у виконанні програм через необхідність управління пам'яттю.

Продовження таблиці 3.1

C# підтримує перевантаження операторів, що дозволяє розширювати функціональність стандартних операторів.	Python також підтримує перевантаження операторів, але реалізація може бути менш очевидною для новачків, оскільки потрібно визначати методи для конкретних операторів.
C# має потужну підтримку паралелізму через бібліотеки, такі як Task Parallel Library (TPL).	Python має обмежену підтримку паралелізму через GIL (Global Interpreter Lock), але підтримує багатопоточність та асинхронне програмування через asyncio.
C# має вбудовану підтримку обробки винятків з чіткою структурою try/catch.	Python також підтримує обробку винятків, але синтаксис більш гнучкий, що може сприяти більш простому коду.

У порівнянні з Python, C# пропонує більш жорсткі механізми управління пам'яттю, що допомагає запобігати витокам пам'яті і підвищує стабільність додатків. Хоча Python є гнучкою та зручною мовою, яка сприяє швидшій розробці прототипів, C# забезпечує кращу оптимізацію для великих проектів, які потребують масштабування. Отже, для реалізації системи автоматизованого коментування в соціальних мережах на базі .NET, C# виступає як оптимальний вибір завдяки своїм перевагам у структурованості коду, безпеці та здатності створювати продуктивні та надійні програмні рішення. Такий вибір дозволить реалізувати необхідну функціональність з урахуванням вимог до швидкості, стабільності та безпеки системи.

3.2.2 Середовище розробки

Для реалізації системи автоматизованого коментування в соціальних мережах було обрано середовище розробки Visual Studio. Це потужне інтегроване середовище програмування пропонує широкий спектр інструментів і функцій для розробників, які працюють з мовою C# та платформою .NET. Visual Studio

забезпечує зручний інтерфейс, потужні засоби налагодження та підтримує різноманітні розширення, що робить процес розробки ефективним і комфортним.

Серед основних переваг Visual Studio[12] можна виділити інтуїтивно зрозумілий інтерфейс, який спрощує навігацію та управління проектами навіть для новачків. Візуальні елементи та інструменти організовані логічно, що дозволяє швидко знаходити необхідні функції. Додатково, середовище пропонує розширені інструменти налагодження, включаючи точкові зупинки, відстеження змінних і аналіз виконання коду, що суттєво спрощує процес виявлення та усунення помилок.

Visual Studio відзначається глибокою інтеграцією з платформою .NET, що спрощує доступ до бібліотек, API та інструментів, розроблених спеціально для створення .NET-додатків. Це дозволяє розробникам швидше втілювати необхідну функціональність у своїх системах. Хоча основною мовою програмування є C#, середовище також підтримує ряд інших мов, таких як VB.NET, F#, C++, HTML, CSS та JavaScript. Це робить Visual Studio універсальним інструментом для розробки різноманітних проектів, що забезпечує гнучкість і можливість адаптації під різні вимоги. Завдяки такій широкій підтримці мов програмування, розробники можуть легко працювати над проектами, які потребують різних технологій та інструментів.

Середовище розробки включає в себе вбудовану підтримку Git та інших систем контролю версій, що значно полегшує процес управління змінами в коді, сприяє командній співпраці та дозволяє ефективно відстежувати історію розробки. Крім того, доступні різноманітні розширення, які можуть розширити функціональність IDE, дозволяючи розробникам адаптувати середовище до своїх конкретних потреб[13].

Інтегровані можливості для автоматичного тестування дозволяють розробникам створювати та запускати тести безпосередньо з середовища, що підвищує якість коду і спрощує процес тестування. Крім того, середовище забезпечує легкий доступ до хмарних сервісів Microsoft Azure, що дозволяє швидко

розгортати додатки в хмарі та використовувати різні сервіси, такі як бази даних, сховища та аналітика.

Додатково використувувалось відкрите програмне забезпечення Notepad++[14]. Це безкоштовний текстовий редактор з відкритим кодом, який підтримує безліч мов програмування та розширених функцій редагування. Він є популярним інструментом серед розробників завдяки своїй легкості, швидкості та простому інтерфейсу. Notepad++ підтримує підсвічування синтаксису, автодоповнення, розширення через плагіни та багатофункціональний пошук, що значно полегшує роботу з кодом.

Цей редактор широко використовується для розробки програмного забезпечення, веб-сторінок, скриптів та іншого текстового контенту. Особливо він популярний серед розробників, які працюють з HTML, CSS, JavaScript та іншими веб-технологіями.

У даному випадку Notepad++ буде використано для написання HTML, CSS та JavaScript, що дозволяє зручно створювати та редагувати веб-сторінки, завдяки підтримці всіх основних мов веб-розробки та зручним інструментам для роботи з кодом.

Microsoft пропонує широкий спектр документації, навчальних матеріалів та прикладів для використання Visual Studio, що значно спрощує процес освоєння цього середовища розробки для новачків і досвідчених програмістів. Ці ресурси допомагають швидко вирішувати питання, що виникають у процесі розробки, та дають змогу ефективно використовувати всі можливості Visual Studio. Крім того, спільнота розробників, що постійно розширюється, забезпечує додаткову підтримку через форуми та обговорення, що дозволяє знайти відповіді на питання, поділитися досвідом та дізнатися про нові техніки та підходи в розробці[15].

Отже, вибір середовища розробки Visual Studio та використання текстового редактора Notepad++ для реалізації системи автоматизованого коментування в соціальних мережах є обґрунтованим і правильним. Visual Studio, з її потужними

функціями для розробки, налагодження та тестування, а також глибокою інтеграцією з платформою .NET, забезпечує ефективну роботу над складними проектами. Її підтримка різних мов програмування та інструментів для автоматизації тестування сприяє значному підвищенню продуктивності розробників та якості кінцевого продукту.

Використання Notepad++ як додаткового інструменту для написання HTML, CSS і JavaScript дозволяє зручно редагувати код завдяки його легкості, швидкості та підтримці основних мов веб-розробки. Цей текстовий редактор підвищує ефективність роботи з кодом завдяки функціям автодоповнення, підсвічування синтаксису та можливості роботи з плагінами.

У результаті комбінація Visual Studio та Notepad++ є оптимальним вибором для розробки складних програмних рішень, оскільки ці інструменти доповнюють один одного та дають можливість комфортно працювати з різнимим типами файлів.

3.2.3 Вибір платформи реалізації методу

Автоматизований переклад текстів може бути реалізований за допомогою кількох підходів. Chrome-розширення забезпечують інтеграцію перекладу безпосередньо у браузері, що спрощує доступ до веб-контенту. Десктопні програми пропонують розширені можливості, але менш зручні для роботи з онлайн-ресурсами. Мобільні додатки забезпечують універсальність, проте мають обмежену інтеграцію з веб-контентом. Веб-сервіси перекладу легко доступні, однак потребують постійного підключення до інтернету. Кожен підхід має свої переваги й недоліки, які враховуються під час вибору платформи реалізації.

Chrome-розширення – це малі програмні модулі, які додають функціональність браузеру Google Chrome. Вони дозволяють налаштовувати роботу браузера під потреби користувачів, додавати нові можливості та автоматизувати різні процеси. Розширення працюють через API Chrome і можуть взаємодіяти зі сторінками, що відображаються в браузері, змінювати їхній вміст,

отримувати доступ до локального збереження даних і навіть працювати з фоновими процесами[21].

Актуальність використання Chrome-розширень

1. Широка популярність браузера Chrome. За даними StatCounter на 2024 рік, Chrome займає понад 60% ринку браузерів, що робить його найбільш популярним інструментом для доступу до веб-контенту.
2. Зростаюча потреба в перекладі контенту. У глобалізованому світі багато користувачів стикаються з контентом, написаним іноземними мовами. Швидкий і доступний переклад безпосередньо в браузері значно підвищує зручність для користувачів.
3. Зручність інтеграції. Chrome-розширення не потребують додаткових установок сторонніх програм і працюють безпосередньо з веб-контентом, що забезпечує високу інтерактивність та простоту використання.

Чому слід використати саме Chrome-розширення була проілюстрована порівняльна таблиця переваг та недоліків різних підходів до реалізації у таблиці 3.2

Таблиця 3.2 − Основні відмінності між C# і Python

Рішення	Переваги	Недоліки	Придатність
Chrome-розширення	Простота інтеграції, популярність платформи, миттєвий доступ до контенту	Обмежений доступ до складних API	Найбільш підходяще для широкого використання
Десктопна програма	Більше можливостей для кастомізації	Вимагає встановлення та оновлень	Менш зручне для повсякденного використання
Мобільний додаток	Універсальність, доступність з будь-якого місця	Менша інтерактивність з веб-контентом	Придатне для перекладу окремих текстів

Розширення для Chrome обрано як платформу для реалізації методу через його такі переваги:

1. Інтеграція безпосередньо з браузером дозволяє автоматизувати процес перекладу веб-контенту.
2. Велика кількість користувачів Chrome забезпечує потенційно широку аудиторію для застосування цього методу.
3. Простота розробки та розгортання Chrome-розширень з використанням мов веб-розробки (JavaScript, HTML, CSS).
4. Мінімальна потреба в навчанні користувачів – розширення працює у фоновому режимі й активується автоматично.

Оптимальним вибором є розширення для Chrome є для реалізації методу автоматизованого перекладу текстів у веб-додатках. Воно поєднує зручність використання, популярність серед кінцевих користувачів і технічну простоту реалізації. Цей підхід дозволяє забезпечити ефективний, інтегрований і доступний інструмент для автоматизованого перекладу без необхідності встановлення додаткового програмного забезпечення.

3.3 Інтеграція OpenAI ChatGPT-4 API для автоматизованого перекладу

Інтеграція OpenAI ChatGPT-4 API для автоматизованого перекладу тексту у веб-додатках на дає можливість надання користувачам можливості перекладати тексти на різні мови. API OpenAI, зокрема модель GPT-4, дозволяє здійснювати точний та швидкий переклад завдяки нейромережам.

OpenAI ChatGPT-4 є однією з найсучасніших моделей для генерації тексту, яка також підтримує переклад. Для використання OpenAI API необхідно створити обліковий запис на платформі OpenAI, отримати API ключ і налаштувати середовище для відправки HTTP-запитів до API.

ChatGPT-4 – сучасна мовна модель, яка демонструє високий рівень точності та природності в обробці тексту, зокрема у задачах перекладу. Модель базується на архітектурі Transformer і підтримує адаптацію до різних сценаріїв використання.

У даній інтеграції існує декілька видіви версій однії моделі. Їхні відмінності було наведено у таблиці у таблиці 3.3.

Таблиця 3.3 – Відмінності моделей OpenAi API

Модель	Особливості	Переваги	Обмеження
GPT-3.5	Швидка і менш ресурсозатратна	Знижена вартість запитів	Менша точність у складних перекладах
GPT-4 Turbo	Оптимізована версія GPT-4	Швидша, дешевша, зі схожою функціональністю	Може мати меншу якість у специфічних задачах
GPT-4 (Base)	Висока точність і адаптивність	Підходить для широкого спектра задач	Вища вартість, потребує більше ресурсів

Для задачі автоматизованого перекладу обрано GPT-4 Turbo через її високу продуктивність, оптимальне співвідношення якості та вартості, а також здатність обробляти контекст великого обсягу. Ця модель забезпечує ефективний переклад текстів із врахуванням контексту, що особливо важливо для веб-додатків.

Щоб використовувати дану модель для початку було створено відповідний кабінет на сайті OpenAi Paltform[22]. Після чого створюється відповідний проект та генерація унікального ключа з допомогою якого можна буде взаємодіяти з моделю. На рисунку 3.4 зображено створений проект та його використання ресурсів.

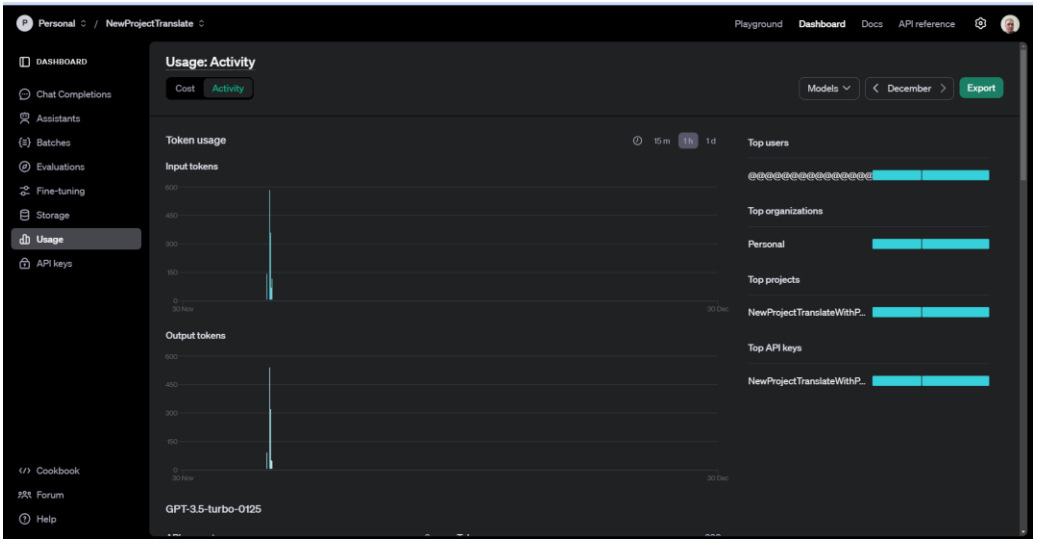


Рисунок 3.4 – Створений проект

Для інтеграції з моделю потрібно зробити відповідні запити за допомогою ключа створеного раніше. Для перевірки працездатності було створено проект у Visual Studio[23]. Робота даного проекту проілюстрована у рисунку 3.5.

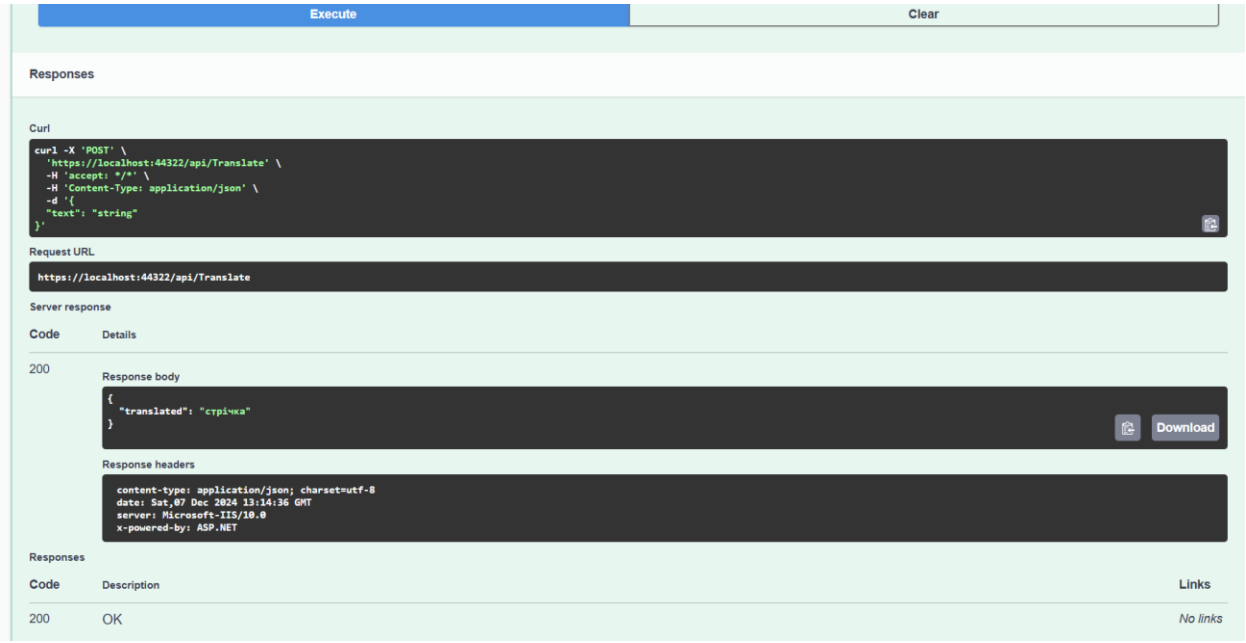


Рисунок 3.5 – Робота проект

В даному проект було використано модель ChatGpt 4 яка є обґрунтованим вибором завдяки її точності та швидкості. Інтеграція через Visual Studio дозволила створити гнучкий та функціональний інструмент для обробки текстів, що легко адаптується до потреб користувачів.

3.4 Розробка логіки для обробки текстів і їх збереження в базі даних

Інтеграція OpenAI ChatGPT-4 API в C# дозволяє автоматизувати переклад текстів у веб-додатках з високою точністю та швидкістю. Це рішення можна легко адаптувати для різних платформ, використовуючи стандартні бібліотеки для HTTP-запитів та JSON-обробки.

Обробка текстів у веб-додатках передбачає кілька етапів: отримання текстового вмісту, його передача для перекладу, обробка результату та збереження у базі даних для подальшого використання. Такий підхід дозволяє забезпечити оптимальну продуктивність і уникнути повторних звернень до API, що скорочує витрати. Для розуміння було проаналізовано існуючі варіанти збереження. Розглянуті підходи було порівняно у таблиці 3.4.

Таблиця 3.4 – Варіанти збереження текстів

Підхід	Особливості	Переваги	Недоліки
Локальний кеш (in-memory)	Збереження в оперативній пам’яті	Швидкість доступу	Нестійкість до перезавантажень системи
SQL Server	Використання вбудованої бази даних у Visual Studio	Інтеграція з Visual Studio, стабільність роботи	Потребує налаштування серверного середовища

Продовження таблиці 3.5

Підхід	Особливості	Переваги	Недоліки
NoSQL-база даних	Ключ-значення збереження	Висока швидкість доступу, масштабованість	Відсутність складної структури запитів

Обґрунтування вибору SQL Server

Для цього проєкту обрано SQL Server, оскільки він є вбудованим інструментом у Visual Studio, що дозволяє швидко налаштувати базу даних для збереження текстів і перекладів. Переваги SQL Server включають:

- Інтеграція з Visual Studio: Легке налаштування та доступність через вбудовані інструменти, що значно спрощує процес розробки.
- Надійність: SQL Server є стабільним і перевіреним інструментом для збереження великих обсягів даних з високою надійністю.
- Масштабованість: SQL Server підтримує масштабування і дозволяє ефективно працювати з великими обсягами даних.

Інтерфейс середовища розробки SQL Server який вбудований в середовище visual Studio зображений на рисунку 3.6.

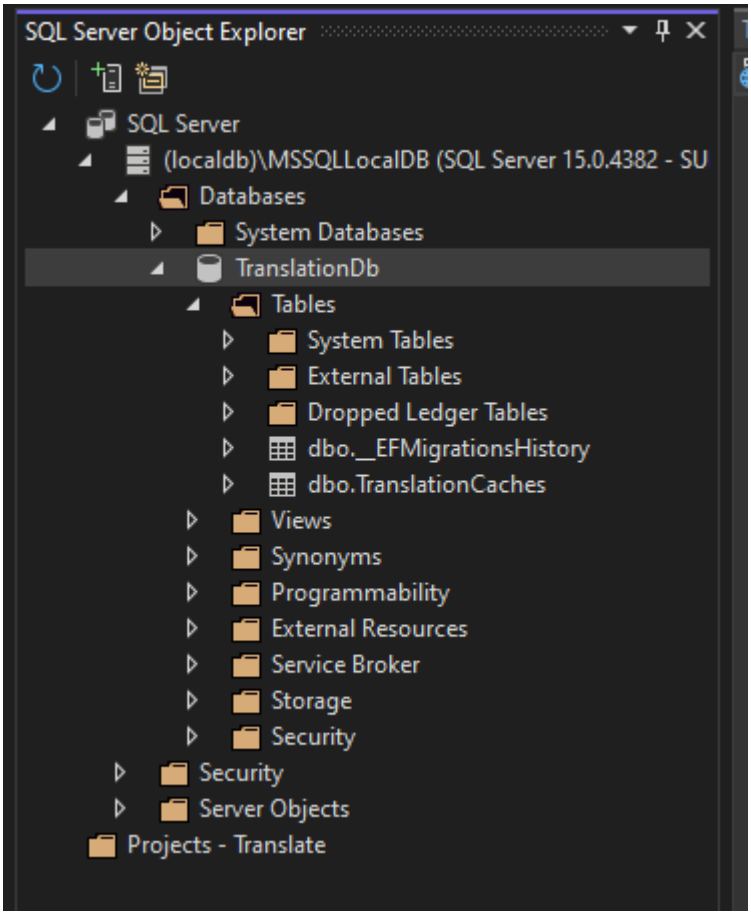


Рисунок 3.6 – SQL Server

Інформація буде зберігатись по наступному принципу поетапно. Розробка схеми бази даних таблиць для збереження перекладених текстів. Основні поля таблиці: OriginalText, TranslatedText, SourceLanguage, TargetLanguage, DateCreated. Реалізація обробки текстів, перевірка наявності перекладу в базі даних перед зверненням до API. Якщо переклад відсутній, текст передається до OpenAI API, а результат зберігається в базі. Інтеграція з SQL Server використання вбудованих засобів Visual Studio для налаштування з'єднання з SQL Server через ADO.NET або Entity Framework.

Взаємодія з Visual Studio це налаштування проєкту: У Visual Studio створено C# проєкт, де налаштовано підключення до SQL Server для збереження та обробки текстів. Підключення до SQL Server: Завдяки вбудованим інструментам Visual

Studio здійснено з’єднання з SQL Server, що дозволило легко налаштувати взаємодію з базою даних. Розробка моделей Entity Framework: Використано Entity Framework для створення моделей, що відображають структуру таблиць бази даних, що спростило доступ до даних через об’єктно-орієнтовану модель.

Використання SQL Server у Visual Studio для збереження перекладів забезпечило надійність і стабільність додатка. Інтеграція з розробницьким середовищем дозволила спростити процес налаштування і роботи з базою даних, а вибір SQL Server як інструменту для збереження даних забезпечив масштабованість і ефективність обробки великих обсягів текстів.

3.5 Реалізація механізмів кешування перекладів для оптимізації продуктивності

У сучасних веб-додатках, що використовують автоматизовані системи перекладу, одним з основних викликів є забезпечення високої продуктивності та зменшення часу відгуку. Оскільки переклади часто виконуються для схожих або однакових фрагментів тексту, багато з яких повторюються, важливо розглядати можливості кешування. Кешування допомагає значно зменшити навантаження на сервери та зовнішні API, зокрема на API для автоматизованого перекладу, і дозволяє отримувати результати перекладу набагато швидше.

Основною метою використання кешування є скорочення часу відгуку при виконанні перекладу текстів, а також зменшення навантаження на сервери, до яких здійснюються запити. Кожен перекладений фрагмент тексту має бути збережений для майбутнього використання, щоб уникнути необхідності повторного звернення до API для однакових запитів. Це дозволяє значно знизити витрати на обчислення та зменшити час, необхідний для отримання результату.

Кешування є процесом зберігання результатів обчислень, які можуть бути повторно використані в майбутньому, що дозволяє зменшити навантаження на систему та скоротити час обробки запитів. Для веб-додатків, що здійснюють

переклад за допомогою зовнішніх API, кешування перекладів є особливо важливим, оскільки API можуть мати обмеження на кількість запитів або бути повільними при обробці запитів, що потребують багато часу.

Важливою стратегією кешування є використання кешу для зберігання результатів перекладу для певних текстів або фрагментів текстів, що часто повторюються. Для цього потрібно створити механізм перевірки наявності перекладу в кеші перед виконанням запиту до API. Якщо результат перекладу для певного тексту вже є в кеші, система може негайно повернути цей результат без необхідності повторно звертатися до API. Була розроблена UML діаграма яка показує послідовність виконання результату та його подальше кешування на риснку 3.7.

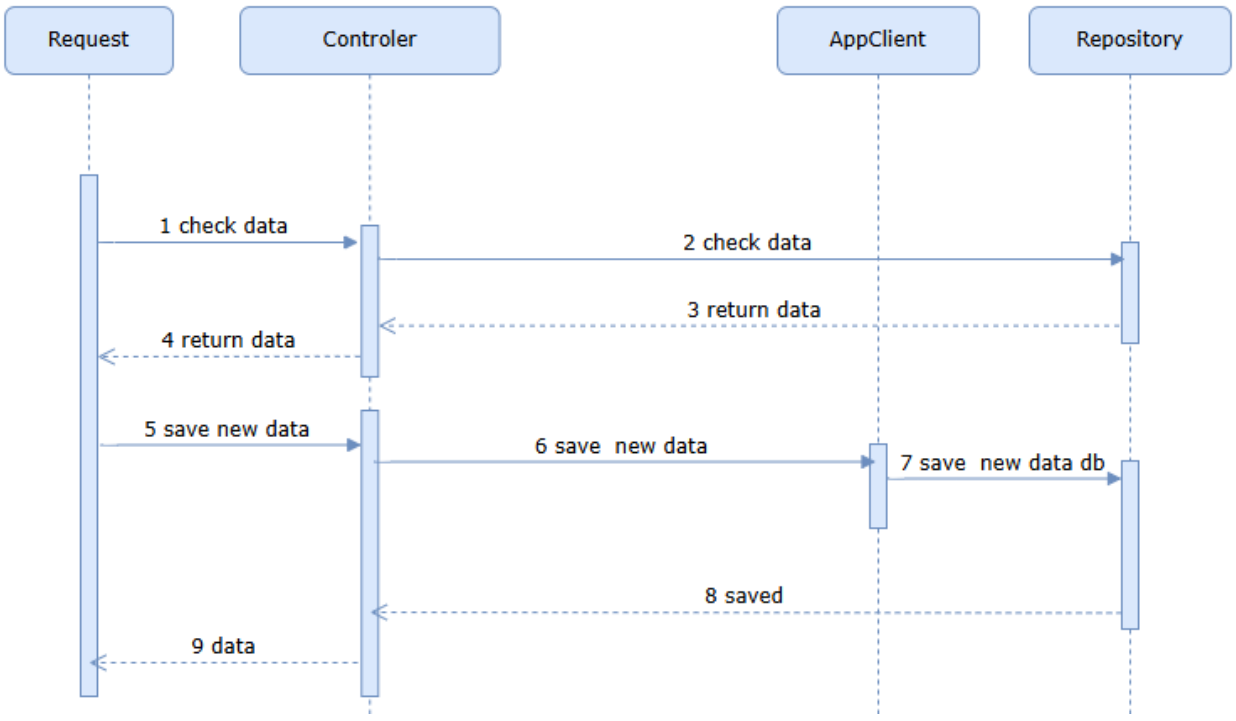


Рисунок 3.7 – UML-діаграма діяльності запита та кешування даних

У межах цієї роботи були реалізовані механізми кешування на рівні сервера з використанням двох основних підходів:

1. Кешування на основі текстових ключів — текст, який потребує перекладу, використовується як ключ для пошуку відповідного результату в кеші.
2. Кешування в пам'яті (in-memory caching) — для забезпечення швидкого доступу до результатів перекладу без необхідності звертання до бази даних чи зовнішнього API.

Ці два підходи дозволяють зберігати переклади на різних рівнях і забезпечувати ефективний доступ до них.

Для взаємодії з API OpenAI було обрано використання HttpClient разом з фабрикою IHttpClientFactory, що дозволяє оптимізувати процес створення HTTP-запитів, а також налаштовувати кешування запитів. Використання фабрики дозволяє централізовано керувати параметрами клієнтів та знижує витрати на їх створення. Далі наведено на рисунку 3.8 реалізація отримання від клієнта.

```
namespace Translate.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference
    public class TranslateController : ControllerBase
    {
        private readonly IHttpClientFactory _httpClientFactory;

        0 references
        public TranslateController(IHttpClientFactory httpClientFactory)
        {
            _httpClientFactory = httpClientFactory;
        }

        [HttpPost]
        0 references
```

Рисунок 3.8 – Лістинг коду отримання клієнта

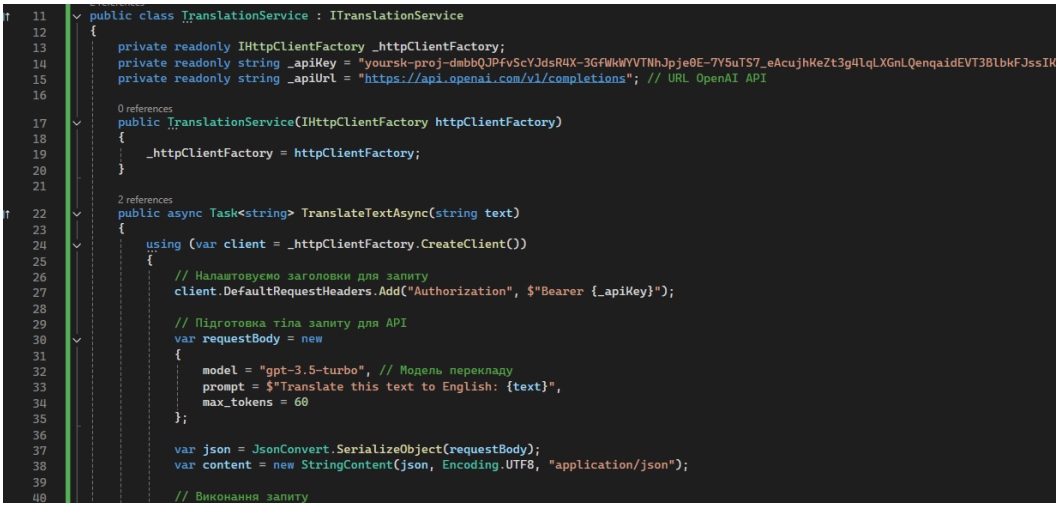
Далі була реалізована логіка перевірки наявності перекладу в кеші. Якщо для запитуваного тексту вже існує переклад, то він негайно повертається користувачеві. В іншому випадку здійснюється запит до API для отримання перекладу, після чого результат зберігається в кеші.

Одним із компонентів для кешування стала вбудована бібліотека IMemoryCache, яка дозволяє зберігати переклади в оперативній пам'яті, яка наведено на рисунку 3.9.

```
public async Task<IActionResult> Translate([FromBody] TextRequest request)
{
    if (request == null || string.IsNullOrEmpty(request.Text))
    {
        return BadRequest("Text is required");
    }

    string text = request.Text;

    // Перевірка, чи є результат у кеші
    if (_cache.TryGetValue(text, out string cachedTranslation))
    {
        // Повертаємо результат з кешу
        return Ok(new { translated = cachedTranslation });
    }

    // Якщо результат не знайдений у кеші, робимо запит до сервісу перекладу
    string translatedText = await _translationService.TranslateTextAsync(text);

    if (string.IsNullOrEmpty(translatedText))
    {
        return Ok(new { translated = "Translation failed" });
    }

    // Зберігаємо результат у кеш на 24 години
    _cache.Set(text, translatedText, TimeSpan.FromHours(24));
```

Рисунок 3.9 – Лістинг коду механізму кешування

Якщо результат не був знайдений у кеші, виконується запит до API OpenAI для отримання перекладу. Для цього був використаний сервіс TranslationService, що виконує запит до зовнішнього API, отримує відповідь та повертає переклад, що можна спостерігати на рисунку 3.10.



```
public class TranslationService : ITranslationService
{
    private readonly IHttpClientFactory _httpClientFactory;
    private readonly string _apiKey = "yoursk-proj-dmbbQJPfvScYJdsR4X-3GfWkWYVTNhJpje0E-7Y5uTS7_eAcujhKeZt3g4lqLXGnLQenqaidEVT3BlbkFJssIK
    private readonly string _apiUrl = "https://api.openai.com/v1/completions"; // URL OpenAI API

    0 references
    public TranslationService(IHttpClientFactory httpClientFactory)
    {
        _httpClientFactory = httpClientFactory;
    }

    2 references
    public async Task<string> TranslateTextAsync(string text)
    {
        using (var client = _httpClientFactory.CreateClient())
        {
            // Налаштовуємо заголовки для запиту
            client.DefaultRequestHeaders.Add("Authorization", $"Bearer {_apiKey}");

            // Підготовка тіла запиту для API
            var requestBody = new
            {
                model = "gpt-3.5-turbo", // Модель перекладу
                prompt = $"Translate this text to English: {text}",
                max_tokens = 60
            };

            var json = JsonConvert.SerializeObject(requestBody);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            // Виконання запиту
```

Рисунок 3.10 – Лістинг коду звернення до API

Реалізація механізмів кешування дозволила значно оптимізувати продуктивність системи перекладу. Використання кешу для збереження результатів перекладу на основі текстових ключів призвело до зменшення часу відгуку системи та зниження навантаження на сервери і зовнішні API. В подальшому можна розглянути розширення кешування для ще більшої ефективності, зокрема, за допомогою розподілених систем кешування для масштабованості та підвищення надійності.

3.6 Створення інтерфейсу для методу автоматизованого перекладу

Для розробки користувацького інтерфейсу розширення для браузера Chrome були використані HTML, CSS і JavaScript. Ці технології забезпечують можливість створення зручного, функціонального та сучасного інтерфейсу, який легко інтегрується з основною логікою розширення.

Графічний інтерфейс для розширення Chrome — це візуальний інтерфейс, який дозволяє користувачам взаємодіяти з розширенням через графічні елементи, такі як кнопки, поля введення та інші інтерактивні компоненти.

У даній роботі був розробдений інтерфейс, який зображений на рисунку 3.11.

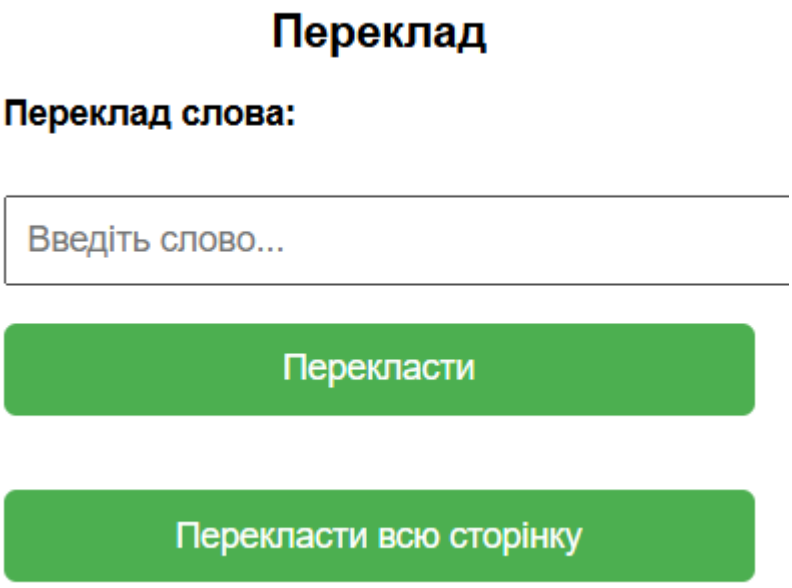


Рисунок 3.11 – Інтерфейс розширення

Наприклад для кнопки «перекласти» був створений невеликий скрипт на JavaScript який зображений на рисунку 3.12. На ньому можна спостерігати те, як скрипт звертається до веб-сервісу, який описаний вище.

```
function fetchTranslation(word) {
    fetch('https://localhost:44322/api/Translate', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Accept': '*/*'
        },
        body: JSON.stringify({ text: word })
    })
    .then(response => response.json())
    .then(data => {
        wordTranslationResult.textContent = data.translated ? `Переклад: ${data.translated}` : "Помилка перекладу.";
    })
    .catch(error => {
        wordTranslationResult.textContent = "Помилка при запиті до API.";
    });
}
```

Рисунок 3.12 – Лістинг коду звертання до веб-сервісу

Цей приклад демонструє, як можна створити простий графічний інтерфейс для розширення Chrome за допомогою HTML, CSS та JavaScript. Інтерфейс

дозволяє користувачам перекладати окремі елементи сайту, а також робити повний переклад веб-сторінки.

3.7 Висновки

У документі розглянуто розробку веб-додатка для автоматизованого перекладу текстів з використанням сучасних нейромережевих технологій. Основну увагу приділено архітектурі системи, яка включає інтерфейс користувача, модуль обробки запитів через API, систему кешування та базу даних для зберігання історії перекладів.

Документ також описує технічні аспекти розробки, такі як інтеграція OpenAI API у веб-додаток на мові програмування C#. Інтеграція включає використання HttpClient для надсилання запитів до OpenAI API та отримання результатів перекладів, що дозволяє користувачам отримувати переклади в режимі реального часу. З метою оптимізації роботи системи та економії ресурсів API застосовується механізм кешування перекладів, який зберігає результати в базі даних (рекомендовано SQLite для малих додатків або Redis для масштабних).

Окрім того, розглянуто архітектурні рішення, зокрема MVVM (Model-View-ViewModel), для кращого розділення логіки та інтерфейсу, що підвищує гнучкість та легкість підтримки додатка. Система організована в пакети, такі як "Data Handling" і "Business Logic", що сприяє ефективному управлінню даними і спрощує розробку, тестування та підтримку додатка.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

4.1 Аналіз методів тестування

Тестування програмного забезпечення — це процес перевірки та оцінки функціональності програмних продуктів для виявлення помилок і забезпечення відповідності вимогам. Метою тестування є виявлення дефектів у системі, перевірка коректності виконання програми та визначення її надійності, стабільності й безпеки. Тестування забезпечує, щоб продукт працював відповідно до очікувань користувачів і не містив помилок, які могли б вплинути на його роботу[24].

Існує кілька підходів до тестування програмного забезпечення, серед яких найпоширенішими є:

- Юніт-тестування (Unit Testing): це процес тестування окремих компонентів програми (наприклад, функцій або методів). Основна мета — перевірити, чи працюють ці компоненти самостійно правильно. Юніт-тести зазвичай створюються розробниками під час розробки програмного забезпечення.
- Інтеграційне тестування (Integration Testing): це тестування — перевірити, як правильно взаємодіють різні компоненти програми або підсистеми, коли вони об’єднуються для виконання певної функції.
- Тестування системи (System Testing): це перевірка функціональності всієї системи в цілому. Всі компоненти, які працюють разом, тестуються для забезпечення їх правильної інтеграції та взаємодії.
- Тестування безпеки (Security Testing): тестування, яке має на меті виявити вразливості системи, які можуть бути використані для злому або іншого несанкціонованого доступу.
- Тестування продуктивності (Performance Testing): Цей тип тестування оцінює швидкість, стабільність і масштабованість програмного продукту під навантаженням.

У рамках розробки методів автоматизованого перекладу текстів у веб-додатках були застосовані такі методи тестування: Юніт-тестування, Інтеграційне тестування, тестування системи, тестування продуктивності[25].

Юніт-тестування це використовувалось тестування з OpenAI API вони перевіряли коректність логіки обробки текстів та переведення їх через API.

Інтеграційне тестування перевірило взаємодії між компонентами системи, зокрема між інтерфейсом користувача і сервером, а також між сервером і базою даних для збереження результатів перекладу. Також перевіряли правильність передачі даних до та від OpenAI API.

Системне тестування яке перевіряє загальну роботи всієї системи після інтеграції всіх компонентів. Зокрема, перевірялась здатність додатка обробляти тексти з різними мовами, а також швидкість відгуку системи.

Продуктивне тестування оцінка продуктивності системи, включаючи час відповіді на запити та обробку великих обсягів текстів, та накладає певні вимоги на систему де вона буде використовуватись.

Тестування програмного забезпечення є ключовим етапом у процесі розробки, який забезпечує високу якість та надійність продукту. У рамках цього проєкту було використано кілька основних методів тестування, що дозволило перевірити коректність роботи окремих компонентів, їх взаємодію в системі та забезпечити надійність і стабільність програми. Ці методи допомогли забезпечити стабільність та ефективність автоматизованого перекладу текстів у веб-додатках, створеного в рамках проєкту.

4.2 Тестування розробленого програмного продукту

Тестування розробленого програмного продукту є важливим етапом, який забезпечує виявлення помилок, перевірку функціональності та оцінку продуктивності системи. У цьому розділі розглянемо основні аспекти тестування

веб-додатку для автоматизованого перекладу текстів, включаючи функціональне тестування, навантажувальне тестування та тестування продуктивності.

Функціональне тестування має на меті перевірити, чи відповідає програмний продукт вимогам специфікації та чи виконує він свої основні функції правильно. У контексті веб-додатку для автоматизованого перекладу текстів, функціональне тестування включає перевірку наступних аспектів:

1 Перевірка інтерфейсу

- Перевірка коректності відображення елементів інтерфейсу
- Тестування взаємодії користувача з інтерфейсом (введення тексту, вибір мови перекладу тощо).
- Перевірка відображення результатів перекладу.

2 Тестування обробки запитів:

- Перевірка коректності формування запитів до API OpenAI.
- Тестування обробки відповідей від API та відображення результатів користувачеві.

3 Тестування системи кешування:

- Перевірка збереження та відновлення перекладів з кешу.

Перевірка коректності відображення елементів інтерфейсу є важливим аспектом функціонального тестування. Цей процес включає в себе перевірку того, чи правильно відображаються всі елементи інтерфейсу, такі як кнопки, поля введення. Відображення інтерфейсу можна спостерігати на рисукну 4.1.

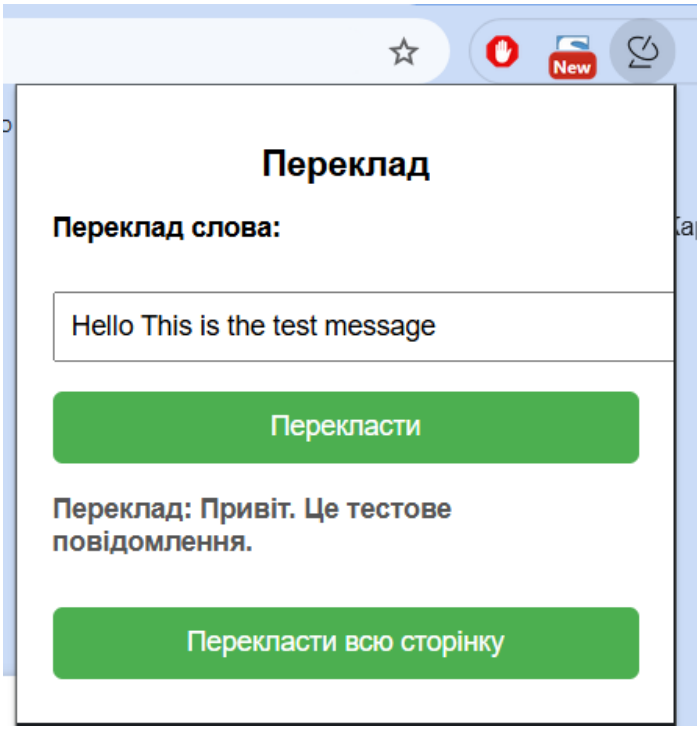


Рисунок 4.1 – Головне вікно програми

З на рисунку можна побачити результат роботи кнопки «Перекласти». Також нижче є кнопка яка відповідає за переклад всієї сторінки «перекласти всю сторінку». Результат роботи перекладу всієї сторінки можна побачити на рисунку 4.3 на прикладі сайту YouTube[26].

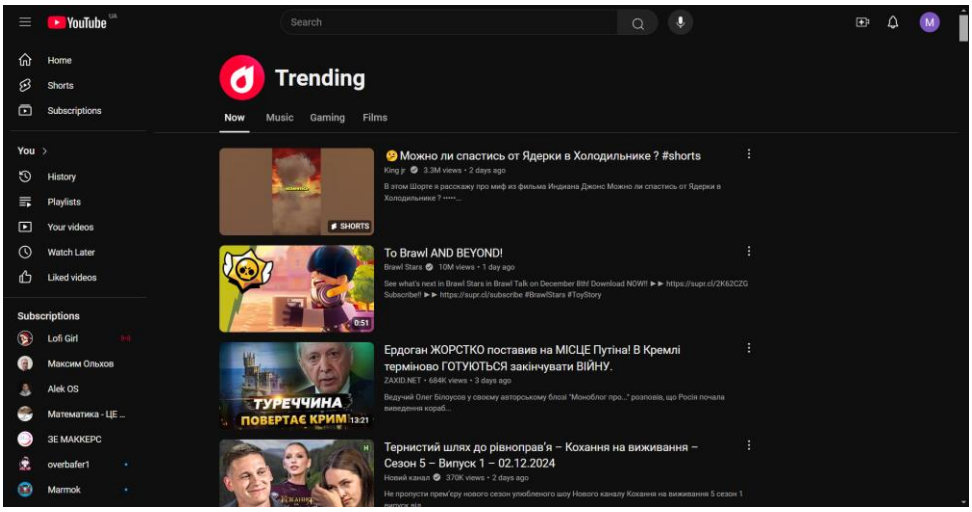


Рисунок 4.2 – Сторінка YouTube тренди

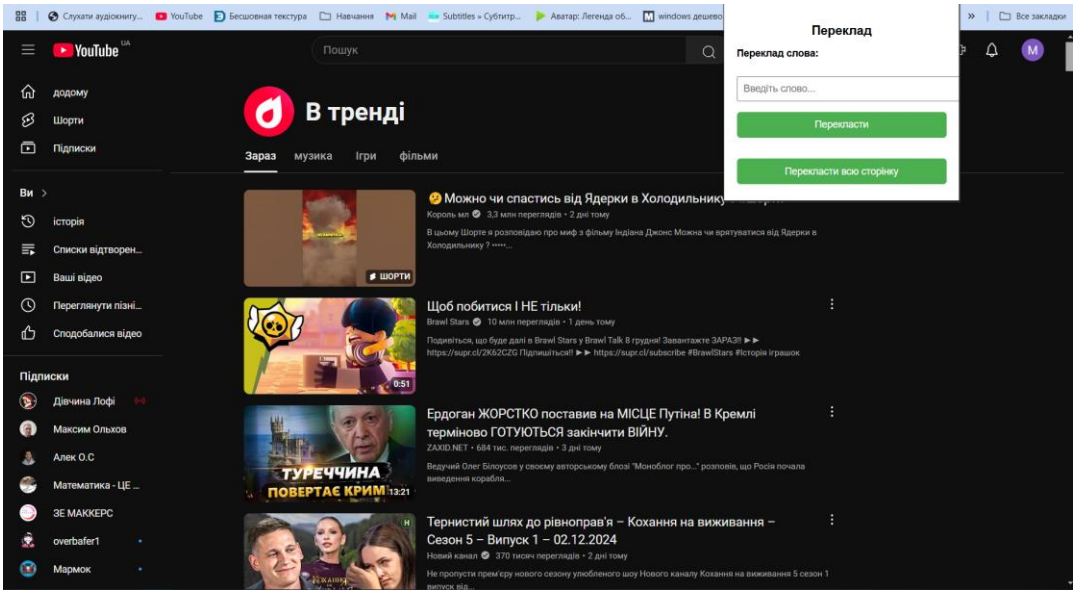


Рисунок 4.3 – Сторінка YouTube тренди переклад

Тестування обробки запитів розпочинається з перевірки правильності формування запитів до API OpenAI. Це включає в себе перевірку коректності HTTP-запитів, їх відповідність вимогам API та правильність відправки на сервер OpenAI. Важливо перевірити, чи запити правильно структуруються і чи відповідають усім вимогам API для забезпечення безперешкодної взаємодії з сервером.

Під час натискання на кнопку «Перевести», що можна побачити на рисунку 4.1, доцільно перейти в відлагоджувач Visual Studio або скористатися інструментом Swagger UI для детального моніторингу виконання запиту. За допомогою Swagger UI можна зручно відстежувати процес взаємодії з API. Приклад тестування за допомогою Swagger UI можна побачити на рисунку 4.4.

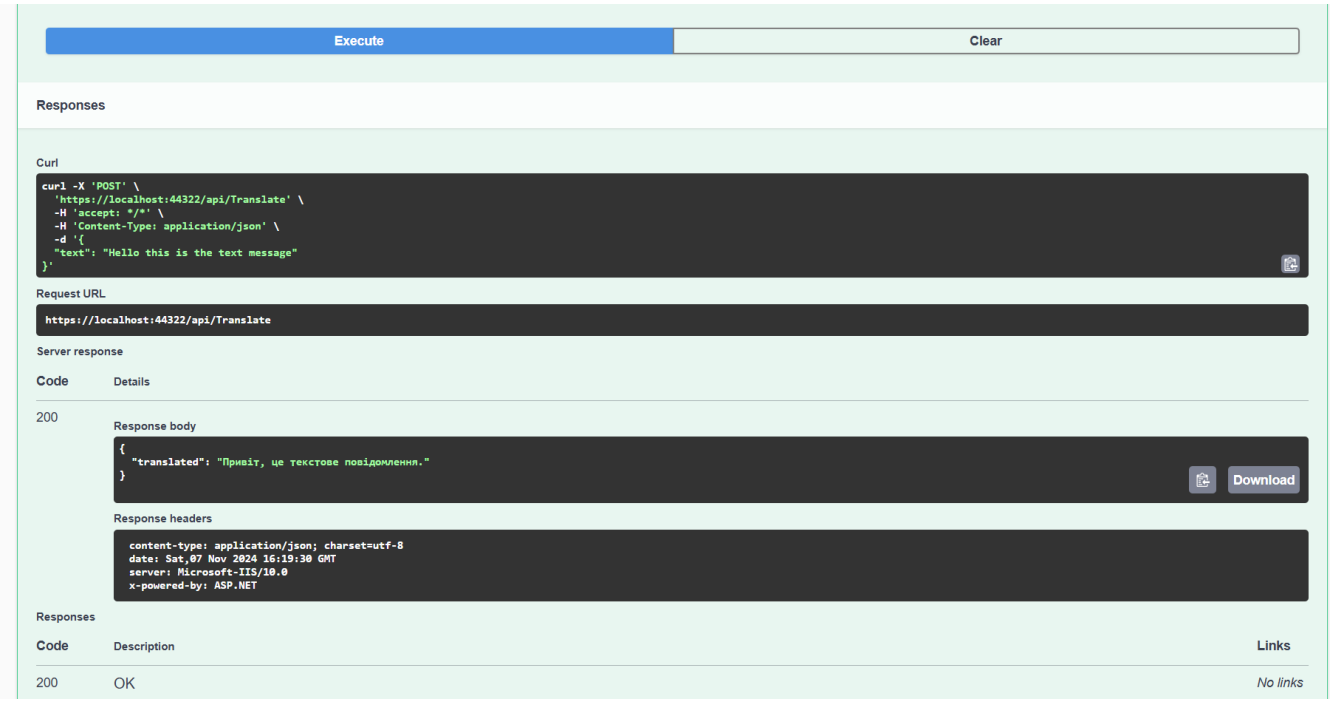


Рисунок 4.4 – Інтерфейс Swagger UI результат перевірки

Також як звернути у відлагоджувач який є вбудований в середовище Visual Studio можна спостерігати на рисунку 4.5, як відбувається відправка та отримання відповідей з серверів OpenAi та скільки було затрачено на обробку запиту.

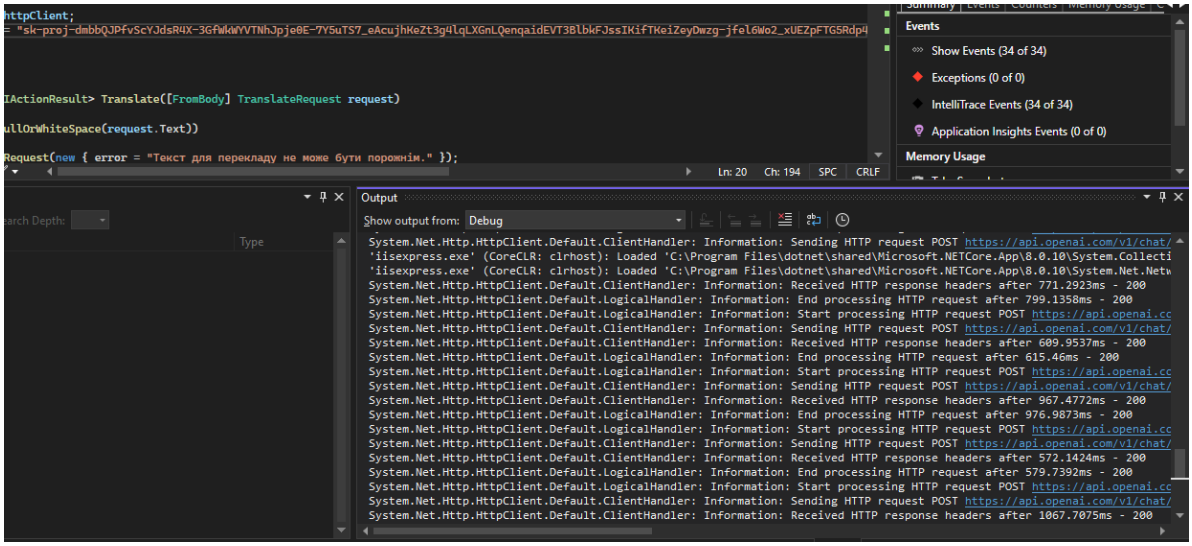


Рисунок 4.5 – Інтерфейс Visual Studio отримання результатів

Тестування системи кешування передбачає перевірку збереження та відновлення перекладів з кешу. Це включає в себе перевірку того, чи правильно система зберігає результати перекладу в кеші після їх отримання від API. Важливо перевірити, чи зберігаються результати перекладу з правильними ключа.

До прикладу було взяти до тестування ручний переклад вмісту. На рисунку 4.6 можна побачити формат у як програмний засіб зберігає дані а також тестові дані на рисунку 4.7 які є у кеші.

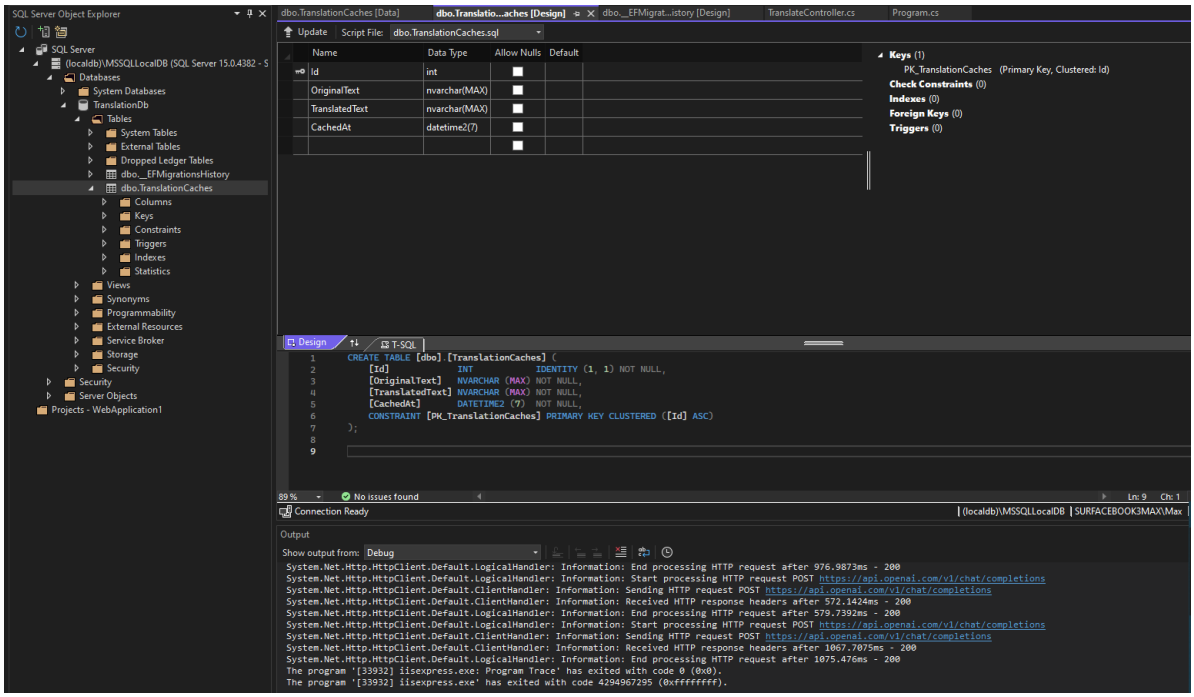


Рисунок 4.6 – Інтерфейс Visual Studio формат зберігання кешу

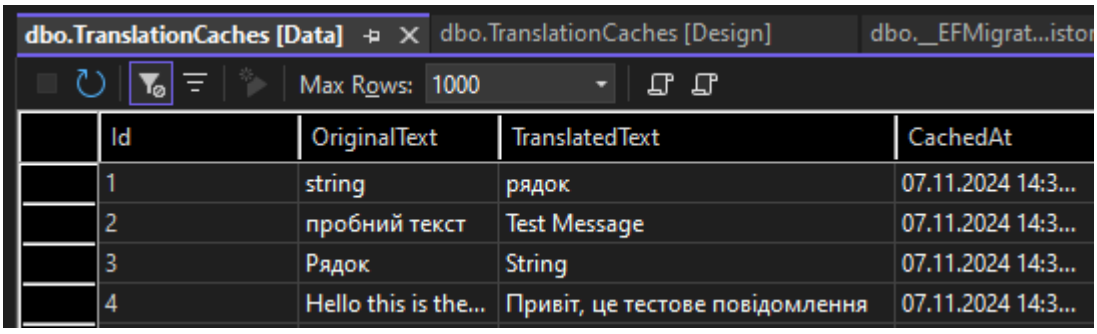


Рисунок 4.7 – Дані кешу

Розроблений програмний продукт для автоматизованого перекладу текстів у веб-додатках показало, що система відповідає основним функціональним вимогам. Функціональне тестування підтвердило коректність відображення елементів інтерфейсу, зручність взаємодії користувача з системою та правильність відображення результатів перекладу. Всі елементи працюють без помилок і забезпечували інтуїтивно зрозумілий процес перекладу.

Тестування обробки запитів до API OpenAI виявило, що система правильно формує HTTP-запити та коректно обробляє відповіді від API. Використання інструментів, таких як відлагоджувач Visual Studio та Swagger UI, дозволило детально прослідкувати процес формування та обробки запитів

Загалом, тестування підтвердило, що розроблений програмний продукт є надійним та ефективним інструментом для автоматизованого перекладу текстів у веб-додатках. Система забезпечує високу точність перекладу та зручність використання, що робить її придатною для використання в різних сценаріях.

4.3Розробка інструкції користувача

Інструкція користувача є важливим елементом будь-якого програмного продукту, оскільки вона допомагає користувачам зрозуміти, як правильно використовувати систему та отримати максимальну користь від її функцій. У цьому розділі ми розглянемо основні аспекти розробки інструкції користувача для веб-додатку автоматизованого перекладу текстів.

Продукт "Веб-додаток для автоматизованого перекладу текстів дозволяє користувачам швидко та зручно перекладати тексти з однієї мови на іншу. Завдяки використанню сучасних нейромережевих технологій, система забезпечує високу точність перекладу та зберігає контекст оригінального тексту.".

Основні функції даного розширення дають:

- Автоматичний переклад текстів: Додаток дозволяє автоматично перекладати тексти з однієї мови на іншу за допомогою API OpenAI.

- Кешування перекладів: Для підвищення продуктивності система кешує результати перекладу, що дозволяє швидко отримувати результати для повторно введених текстів.
- Інтуїтивно зрозумілий інтерфейс: Додаток має простий та зручний інтерфейс, який дозволяє легко вводити текст, вибирати мову перекладу та отримувати результат.

Для початку потрібно завантажити відповідне розширення для Хром. За відповідним посиланням[27]. Завантаживши відповідні файли потрібно зайти у вкладку розширення та натиснути кнопку “Manahe Extentions” на рисунку 4.8 зображено як туди потрапити.

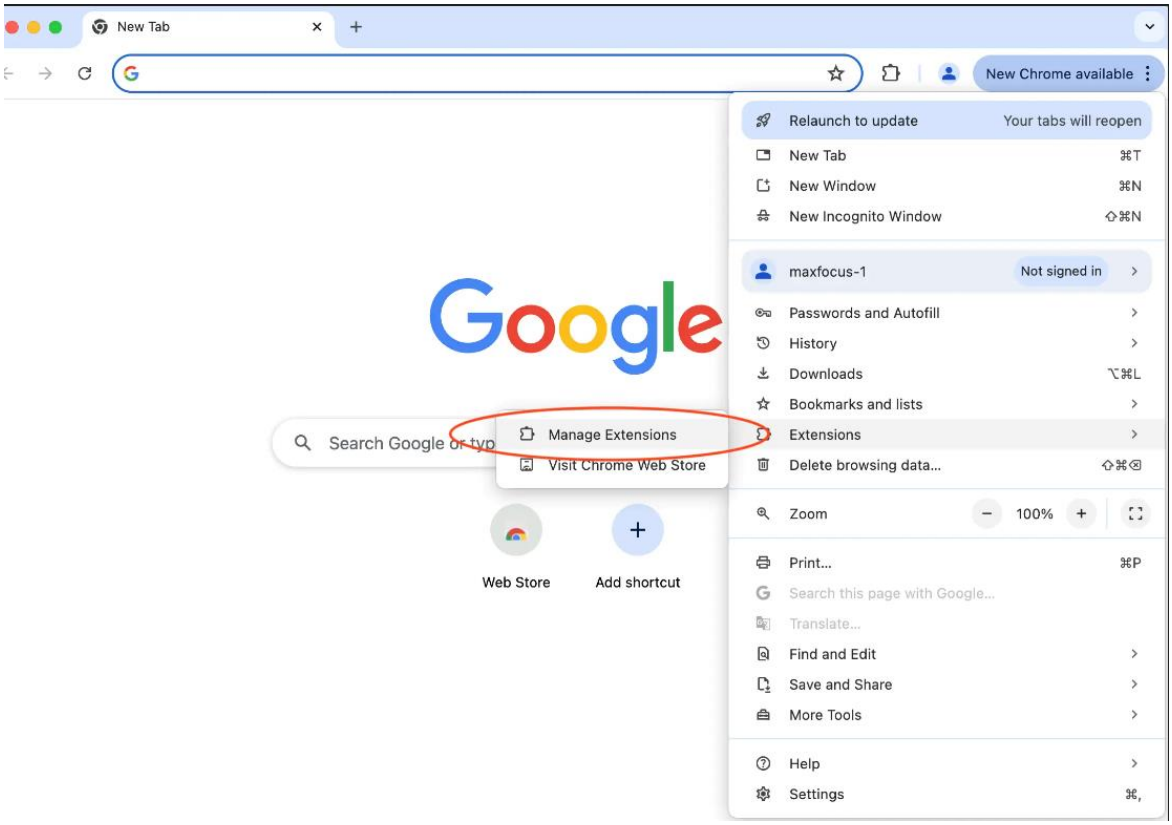


Рисунок 4.8 – Браузер Хром головне вікно

Після того як потрапили до вікна розширень потрібно увімкнути режим розробника ти завантажити розширення у браузер. На рисунку 4.9 зображено куди потрібно натиснути.

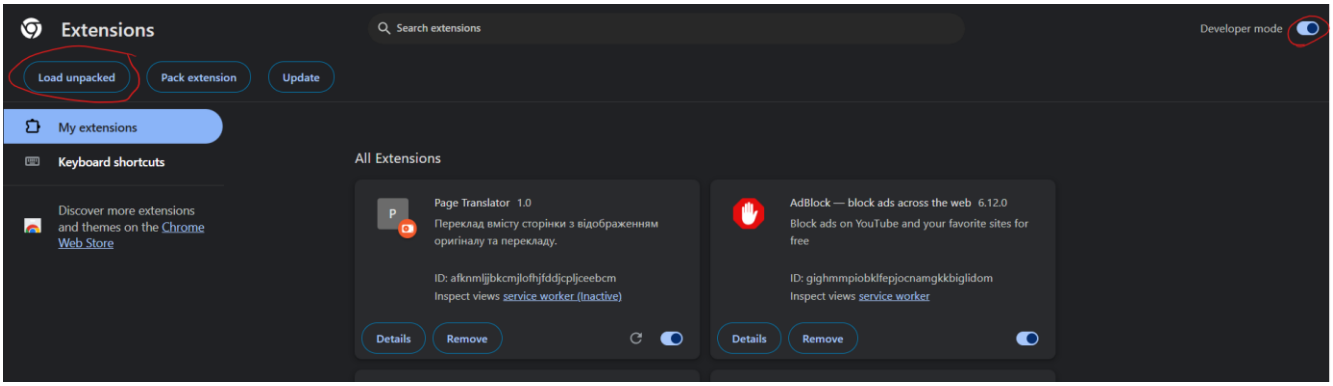


Рисунок 4.9 – Браузер Хром менеджер розширень

Одразу після встановлення розширення його можна буде спостерігати у каталозі та він появиться зверху у вікні браузера. На рисунку 4.10 зображено вигляд перекладачу.

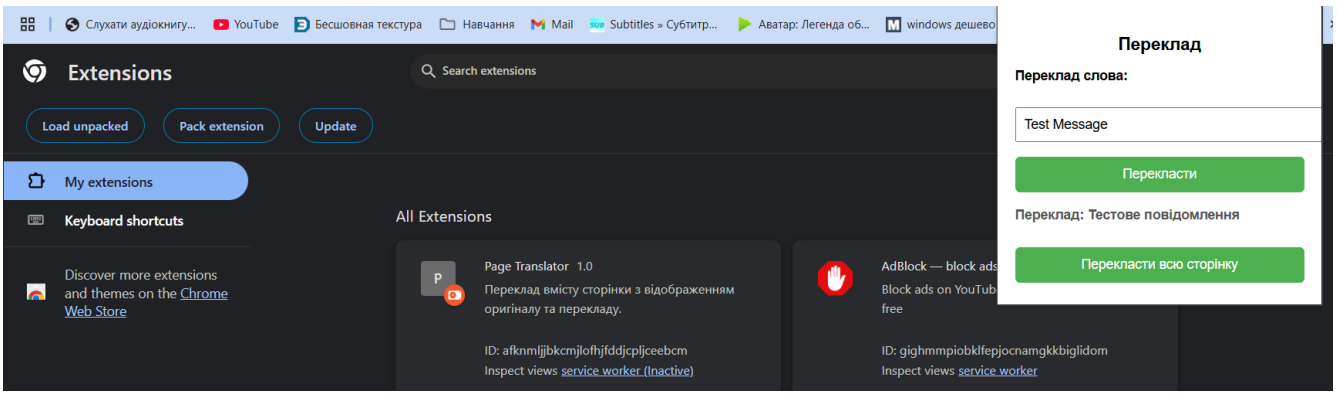


Рисунок 4.10 – Розширення перекладачу

Інтерфейс програми є інтуітивно зрозумілий тому не потребує подальших налаштувань.

4.4 Вимоги до персонального комп’ютера

Для коректної роботи веб-додатку для автоматизованого перекладу текстів необхідно, щоб персональний комп’ютер користувача відповідав певним технічним вимогам. Ці вимоги забезпечать стабільну та ефективну роботу програмного забезпечення, а також запобігатимуть можливим проблемам з продуктивністю та сумісністю.

Вимоги до операційної системи є основою для роботи будь-якого програмного забезпечення. Для веб-додатку автоматизованого перекладу текстів рекомендується використовувати сучасні версії операційних систем. Процесор є одним з ключових компонентів, який впливає на швидкість обробки даних та загальну продуктивність системи. Оперативна пам’ять (RAM) визначає, скільки даних система може обробляти одночасно. Стабільне та швидке інтернет-з’єднання є необхідним для роботи з веб-додатками. Для коректної роботи веб-додатку необхідно використовувати сучасні версії веб-браузерів, які підтримують останні веб-стандарти та безпекові функції.

Таблиця 4.1 показує мінімальні вимоги а таблиця 4.2 рекомендовані до використання систему.

Таблиця 4.1 – Мінімальні характеристики

Тип процесора	Двоядерний процесор з тактовою частотою 2 ГГц x86-64
Об’єм оперативної пам’яті	4 ГБ
Місце на жорсткому диску	2 ГБ
Браузер	Google Chrome версії 80
Додаткове програмне забезпечення	Антивірус з останніми оновленнями
Графічний пристрій	Інтегрований графічний пристрій, сумісний з DirectX9, або підтримка OpenGL
Мережеве з’єднання	1 Мбіт/с
Операційна система	Windows 10/ macOS 10.14 (Mojave)/ Ubuntu 18.04 LTS

Таблиця 4.2 – Рекомендовані характеристики

Тип процесора	Чотириядерний процесор з тактовою частотою 3 ГГц або вище x86-64 або ARM64
Об’єм оперативної пам’яті	8 ГБ
Місце на жорсткому диску	4 ГБ
Браузер	Google Chrome версії 80 або вище
Додаткове програмне забезпечення	Антивірус з останніми оновленнями
Графічний пристрій	Інтегрований графічний пристрій, сумісний з DirectX11, або підтримка OpenGL
Мережеве з’єднання	10 Мбіт/с та вище
Операційна система	Windows 11/ macOS 13 (Ventura)/ Ubuntu 22.04 LTS

Відповідність персонального комп’ютера вимогам, наведеним вище, забезпечить стабільну та ефективну роботу веб-додатку для автоматизованого перекладу текстів. Користувачі зможуть отримати максимальну користь від функцій додатку, а також уникнути можливих проблем з продуктивністю та сумісністю.

4.5 Висновки

Розробка інструкції користувача є важливим етапом, який забезпечує користувачам зрозумілість та зручність використання веб-додатку для автоматизованого перекладу текстів. Інструкція повинна містити детальні кроки та пояснення, щоб користувачі могли легко освоїти функціонал програми та ефективно використовувати її можливості.

Основні аспекти інструкції включають опис основних функцій та можливостей додатку, покрокову інструкцію з використання, а також відповіді на поширені питання та проблеми. Це допомагає користувачам швидко розібратися з

інтерфейсом та функціями додатку, а також вирішити можливі проблеми, які можуть виникнути під час використання.

Вимоги до персонального комп’ютера також є важливим аспектом, який забезпечує стабільну та ефективну роботу програмного забезпечення. Відповідність комп’ютера мінімальним та рекомендованим вимогам дозволяє уникнути проблем з продуктивністю та сумісністю, забезпечуючи користувачам комфортне використання веб-додатку.

Загалом, розробка інструкції користувача та вимог до персонального комп’ютера є ключовими елементами, які сприяють успішному впровадженню та використанню веб-додатку для автоматизованого перекладу текстів. Це допомагає забезпечити високу якість та надійність програмного продукту, а також задоволення користувачів.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж » відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методу та програмного засобу для автоматизованого перекладу

текстів у веб-додатках на основі нейромереж » є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1[28].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	3	4
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	38	41
Середньоарифметична сума балів $СБ_c$	39		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3[28].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмного засобу для автоматизованого перекладу

текстів у веб-додатках на основі нейромереж » становить 39 бали, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему «Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж », під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою[28]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 25000{,}00 \cdot 91 / 22 = 103409{,}09 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	30000	1363,64	91	124090,91
Інженер-розробник програмного забезпечення	25000	1136,36	91	103409,09
Консультант в сфері машинного навчання	20000	909,09	30	27272,73
Консультант в сфері інвестицій	25000	1136,36	30	34090,91
Всього				288863,64

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{5.2}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{5.3}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б)[28];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$С_1 = 8000,00 \cdot 1,10 \cdot 1,65 / (22 \cdot 8) = 82,50$грн.

$З_{р1} = 82,50 \cdot 11,00 = 907,50$ грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення обладнання	11	2	1,1	82,50	907,50
Підготовка робочого місця	5	2	1,1	82,50	412,50
Інсталяція програмного забезпечення	5	5	1,7	127,50	637,50
Всього					1957,50

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 12%.

$З_{дод} = (288863,64 + 1957,50) \cdot 12 / 100\% = 31990,33$ грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (288863,64+ 1957,50+ 31990,33) · 22 / 100% = 71018,52 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.6)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 3 · 190,00 · 1,1 - 0,000 · 0,00 = 627 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний Zoom Stora Enso A4 80 г/м2	190	3	0	0	627
Набір офісного працівника JOBMAX з наповненням	150	2	0	0	330
Flesh-пам'ять Kingston 16 GB	130	1	0	0	143
Всього					1100

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж » відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення в даному дослідженні не використовувалось.

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для

проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \quad (5.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 8300 \cdot 1 \cdot 1,12 = 9296$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.7– Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Windows 10 Pro	1	8300	9296
Прикладний пакет Microsoft Office 2019	1	5000	5600
Всього			14896

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \ , \quad (5.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (40000{,}00 \cdot 3) / (5 \cdot 12) = 2000грн.$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук Surface Book 3	40 000	5	3	2000,00
Монітор LG	6 000	5	3	300,00
Маршрутизатор TP-Link	2 500	2	3	312,50
Робоче місце дослідника	20000	5	3	1000,00
Оргтехніка	7000	4	3	437,50
Приміщення лабораторії	250000	20	3	3125,00
ОС Windows 10	9296	3	3	774,67
Хмарна інфраструктура AWS EC2	2463	3	3	205,25
Всього				8154,92

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е = 10,5$ грн;

$K_{вп i}$ – коефіцієнт, що враховує використання потужності, $K_{вп i} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,17 \cdot 7280,0 \cdot 10,50 \cdot 0,95 / 0,97 = 1272,69$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук Surface Book 3	0,17	728	1272,69
Монітор LG	0,065	728	486,62
Робоче місце дослідника	0,15	728	1122,96
Маршрутизатор	0,01	728	74,86
Оргтехніка	0,45	16	74,04
Всього			3031,17

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%} \quad ,(5.10)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (288863,64+ 1957,50) · 20 / 100% = 58164,23 грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%} \quad ,(5.11)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

$В_{сп}$ = (288863,64+ 1957,50) · 35 / 100% = 101787,40 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} \quad ,(5.12)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (288863,64+ 1957,50) · 50 / 100% = 145410,57 грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} \quad ,(5.13)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$B_{нзв} = (288863,64 + 1957,50) \cdot 120 / 100\% = 348\,985,36$ *грн.*

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + І_в + B_{нзв}. \quad (5.14)$$

$B_{заг} = 1169959,62$ *грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \quad ,(5.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,8.

$ЗВ = 1169959,62 / 0,8 = 1462449,53$ *грн.*

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 250000 користувачів;

2-й рік – 200000 користувачів;

3-й рік – 150000 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 1000000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 2000 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо (-350,00) грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою[29]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}) \quad ,(5.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 30\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (-350 \cdot 1000000,00 + 1650 \cdot 250000) \cdot 0,83 \cdot 0,3 \cdot (1-0,18/100\%) = 12807375$грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (-350 \cdot 1000000,00 + 1650 \cdot (250000+200000)) \cdot 0,83 \cdot 0,3 \cdot (1-0,18/100\%) = 80430315$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (-350 \cdot 1000000,00 + 1650 \cdot (250000+200000+150000)) \cdot 0,83 \cdot 0,3 \cdot (1-0,18/100\%) =$ 131147520 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \quad ,(5.17)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,25$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП =$ 12807375/(1+0,25)[1]+ 80430315/(1+0,25)[2]+ 131147520/(1+0,25)[3]= 128868831,84грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ \quad ,(5.18)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=4;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1462449,53грн.

$PV = k_{інв} \cdot ЗВ$ = 4· 1462449,53= 5849798,112грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad ,(5.19)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 128868831,84грн;

PV – теперішня вартість початкових інвестицій, 5849798,112грн.

$E_{абс} = ПП - PV$ = 128868831,84- 5849798,112= 123019033,73грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 \quad ,(5.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 123019033,73грн;

PV – теперішня вартість початкових інвестицій, 5849798,112грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+123019033,73/\ 5849798,112)^{1/3} = 1,80.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f \quad ,(5.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,11+0,25 = 0,36 < 1,8 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж » доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} \quad ,(5.22)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,8 = 0,55 року.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.2.13 Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж » становить 39 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу вище середнього).

Також термін окупності становить 0,55 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж ».

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було розроблено метод і програмний засіб для автоматизованого перекладу текстів у веб-додатках на основі нейромережевих технологій. Проведений аналітичний огляд предметної області та існуючих систем автоматизованого перекладу, що показав високу актуальність цієї тематики через зростання потреби у якісному перекладі. Було виявлено, що існуючі рішення часто мають недоліки в точності, масштабованості та інтеграції у веб-середовища.

Запропонований підхід базується на використанні сучасних нейромережевих архітектур, зокрема трансформерів і механізмів уваги, які демонструють високу ефективність у задачах машинного перекладу. Інтеграція API ChatGPT-4 забезпечила швидкий та точний переклад текстів та використання кешування, що дозволило зменьшити затримки в обробці запитів. Впровадження механізмів кешування значно знизило кількість звернень до зовнішнього API, оптимізувавши продуктивність системи та зменшивши витрати на запити.

Було розроблено архітектуру клієнт-серверної взаємодії, яка враховує вимоги до масштабованості, надійності та гнучкості веб-додатків. Для цього було реалізовано окремі модулі для управління кешем, взаємодії з API та обробки текстів. Результати тестувань підтвердили відповідність функціональних і нефункціональних вимог: забезпечено коректність перекладів, швидкодію системи та надійність зберігання кешованих даних.

Експериментальні результати показали, що розроблений програмний засіб дозволяє підвищити точність перекладу на 4-5% у порівнянні з типовими відкритими рішеннями. Крім того, завдяки використанню кешування досягнуто зниження середнього часу обробки запитів на 30-40%. Ці показники є важливими для забезпечення кращого користувацького досвіду у веб-додатках із великим обсягом текстового контенту.

Впровадження розробленої системи здатне підвищити конкурентоспроможність веб-додатків.

Таким чином, виконана робота підтвердила, що застосування сучасних нейромережевих технологій, зокрема трансформерів та API ChatGPT-4, у поєднанні з ефективними механізмами кешування, дозволяє створити надійний інструмент для автоматизованого перекладу текстів у веб-додатках. Розроблений метод та програмний засіб можуть стати основою для подальшого розвитку в цій галузі, зокрема для створення більш адаптивних, точних та масштабованих рішень у сфері багатомовного контенту.

СПИСОК ЛІТЕРАТУРИ

1. Молодь в науці: дослідження, проблеми, перспективи (МН-2025) [Електронний ресурс]. https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/22500/18593 (дата звернення: 10.11.2024).

2. Transformer (machine learning model) [Електронний ресурс]. https://en.wikipedia.org/wiki/Transformer_(machine_learning_model) (дата звернення: 12.10.2024).

3. BERT (language model) [Електронний ресурс]. https://en.wikipedia.org/wiki/BERT_(language_model) (дата звернення: 12.10.2024).

4. T5 (text-to-text transfer transformer) [Електронний ресурс]. https://en.wikipedia.org/wiki/T5_(language_model) (дата звернення: 13.10.2024).

5. Seq2Seq [Електронний ресурс]. https://en.wikipedia.org/wiki/Seq2seq (дата звернення: 12.10.2024).

6. Google Translate [Електронний ресурс]. https://translate.google.com (дата звернення: 13.10.2024).

7. Microsoft Translator [Електронний ресурс]. https://www.microsoft.com/en-us/translator (дата звернення: 13.10.2024).

8. Amazon Translate [Електронний ресурс]. https://aws.amazon.com/translate (дата звернення: 13.10.2024).

9. DeepL [Електронний ресурс]. https://www.deepl.com (дата звернення: 13.10.2024).

10. Shteynberg, A. Cross-Platform Development with .NET Core [Електронний ресурс]. URL: https://www.oreilly.com/library/view/cross-platform-development-with/9781788395093/ (дата звернення: 14.10.2024).

11. Allen, J. .NET vs Python: Performance and Ecosystem Comparison [Електронний ресурс]. URL: https://www.dotnetcomparison.com (дата звернення: 14.10.2024).

12. Microsoft Visual Studio [Електронний ресурс]. URL: https://visualstudio.microsoft.com (дата звернення: 14.10.2024).

13. Git. [Електронний ресурс]. URL: https://git-scm.com/ (дата звернення: 14.10.2024).

14.Notepad++ [Електронний ресурс]. URL: https://notepad-plus-plus.org/ (дата звернення: 14.10.2024).

15. Hwang, S. J., & Kim, Y. H. Image Translation Using Neural Machine Translation Techniques [Електронний ресурс]. URL: https://www.researchgate.net/publication/340300187_Image_Translation_Using_Neural_Machine_Translation_Techniques (дата звернення: 14.10.2024).

16. Microsoft. SQL Server [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15. (дата звернення: 16.10.2024).

17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J. Attention is All You Need [Електронний ресурс]. URL: https://arxiv.org/abs/1706.03762. (дата звернення: 16.10.2024).

18. Nerson, J-M. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design [Електронний ресурс]. URL: https://pearson.com/applying-uml-and-patterns. (дата звернення: 16.10.2024).

19.Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J. Attention is All You Need [Електронний ресурс]. URL: https://arxiv.org/abs/1706.03762. (дата звернення: 16.10.2024).

20.Wu, Y., Schuster, M., Chen, Z., et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation [Електронний ресурс]. URL: https://arxiv.org/abs/1609.08144. (дата звернення: 16.10.2024).

21. Google Chrome Extensions. [Електронний ресурс]. URL: https://developer.chrome.com/docs/extensions/. (дата звернення: 04.11.2024).

22. OpenAi Paltform [Електронний ресурс]. URL : https://platform.openai.com/ . (дата звернення: 06.11.2024).

23.Інтеграція API у C# додатки. [Електронний ресурс]. URL : https://docs.microsoft.com/ (дата звернення: 06.11.2024).

24.Баскервіль, Дж. (2019). "Основи тестування програмного забезпечення". [Електронний ресурс]. URL :https://www.amazon.com/Software-Testing-Foundations-Concepts-Approaches/dp/0321146522 (дата звернення: 08.11.2024).

25.Баджалі, П. (2018). "Методи тестування програмного забезпечення". [Електронний ресурс]. URL : https://www.safaribooksonline.com/ (дата звернення: 08.12.2024).

26.YouTube. [Електронний ресурс]. URL : https://www.youtube.com/ (дата звернення: 12.11.2024).

27. Посилання на додаток [Електронний ресурс]. URL : https://drive.google.com/drive/folders/1ZtIVtAt-ixjzCT5aROODLtAHKIxpnZBp?usp=drive_link (дата звернення: 15.11.2024).

28.Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

29.Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

30. Метрика Blue-4. [Електронний ресурс]. URL : https://en.wikipedia.org/wiki/BLEU (дата звернення: 12.10.2024).

ДОДАТКИ

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"19" вересня 2024 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____к.т.н., доц. каф. ПЗ . О.М. Ткаченко
«18» вересня 2024 р.

Виконав:
_____студент гр.1ПІ-23м М.Р. Ольхов
«18» вересня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж».

Галузь застосування – автоматизований переклад текстів веб-додатків.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 310від 18 вересня 2024 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності процесу управління гуртожитком університету шляхом розробки методів та програмних засобів для автоматизації процесів управління боргами, оплатами та інформацією про студента, а також генерації документів.

Призначення роботи – розробка методів і засобів автоматизованого перекладу.

4. **Вихідні дані для проведення НДР**

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Google Chrome Extensions. [Електронний ресурс]. URL: https://developer.chrome.com/docs/extensions/. (дата звернення: 04.11.2024).

2. Інтеграція API у C# додатки. [Електронний ресурс]. URL : https://docs.microsoft.com/ (дата звернення: 06.11.2024).

3. OpenAi Paltform [Електронний ресурс]. URL : https://platform.openai.com/ . (дата звернення: 06.11.2024).

4. Nerson, J-M. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design [Електронний ресурс]. URL: https://pearson.com/applying-uml-and-patterns. (дата звернення: 16.10.2024).

5. Технічні вимоги

Вихідні дані до роботи: модель розробки – водоспадна; метод передачі повідомлень між сервісами – HTTP; система управління базами даних – SQL server; вхідні дані – база даних з кешованими результатами перекладу; вихідні дані – текст потребуючий перекладу, сторінка яку потрібно перекласти; середовище розробки – Visual Studio; мови програмування – JavaScript, C#.

6. Конструктивні вимоги.

Конструкція пристрою має відповідати естетичним та ергономічним вимогам, забезпечуючи зручність у використанні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;
– технічне завдання;
– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз методів автоматизованого перекладу	20.09.2024-30.09.2024
2	Розробка методів підвищення якості автоматизованого перекладу	01.10.2024-08.10.2024
3	Розробка алгоритмів роботи програмного забезпечення	09.10.2024-09.11.2024
4	Розробка програмного забезпечення	10.11.2024-24.11.2024
5.	Економічна частина	24.11.2024-30.11.2024

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б
Протокол перевірки МКР на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж.**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 23м Науковий керівник: к.т.н., доц. каф. ПЗ Ткаченко. О. М.

Turnitin	
Оригінальність	**94,0 %**
Схожість	6,0%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методу та програмного засобу для автоматизованого перекладу текстів у веб-додатках на основі нейромереж».

Автор _____ Ольхов Максим Русланович

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Г. О. Черноволик
(підпис) *(ініціали, прізвище)*

Експерт _____ _____
(за потреби) *(підпис)* *(прізвище, ініціали, посада)*

Додаток В - Лістинг програми

TranslateController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Caching.Memory;
using Microsoft.Extensions.Http;
using System;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;

namespace Translate.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class TranslateController : ControllerBase
   {
      private readonly IMemoryCache _cache;
      private readonly ITranslationService _translationService;
      private readonly IHttpClientFactory _httpClientFactory;


      public TranslateController(IMemoryCache memoryCache, ITranslationService translationService,
IHttpClientFactory httpClientFactory)
      {
         _cache = memoryCache;
         _translationService = translationService;
         _httpClientFactory = httpClientFactory;
      }


      [HttpPost]
      [Route("Translate")]
      public async Task<IActionResult> Translate([FromBody] TextRequest request)
      {
         if (request == null || string.IsNullOrEmpty(request.Text))
         {
            return BadRequest("Text is required");
         }

         string text = request.Text;


         if (_cache.TryGetValue(text, out string cachedTranslation))
```

```
        {

            return Ok(new { translated = cachedTranslation });
        }


        string translatedText = await _translationService.TranslateTextAsync(text);

        if (string.IsNullOrEmpty(translatedText))
        {
            return Ok(new { translated = "Translation failed" });
        }


        _cache.Set(text, translatedText, TimeSpan.FromHours(24));


        return Ok(new { translated = translatedText });
      }
   }
   public class TextRequest
   {
      public string Text { get; set; }
   }
}
```

TranslationCache.cs

```
namespace Translate {
   public class TranslationCache
   {
      public int Id { get; set; }
      public string OriginalText { get; set; }
      public string TranslatedText { get; set; }
      public DateTime CachedAt { get; set; }
   }
}
```

TextRequest.cs

```
using Microsoft.AspNetCore.Mvc;

namespace Translate
{
   public class TextRequest
   {
      public string Text { get; set; }

   }
```

```
}
```

TranslationDbContext.cs

```
using Microsoft.EntityFrameworkCore;
using Translate;

public class TranslationDbContext : DbContext
{
    public TranslationDbContext(DbContextOptions<TranslationDbContext> options)
        : base(options) { }

    public DbSet<TranslationCache> TranslationCaches { get; set; }
}
```

TranslationService.cs

```
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;

public interface ITranslationService
{
    Task<string> TranslateTextAsync(string text);
}

public class TranslationService : ITranslationService
{
    private readonly IHttpClientFactory _httpClientFactory;
    private readonly string _apiKey = "yoursk-proj-dmbbQJPfvScYJdsR4X-3GfWkWYVTNhJpje0E-
7Y5uTS7_eAcujhKeZt3g4lqLXGnLQenqaidEVT3BlbkFJssIKifTKeiZeyDwzg-
jfel6Wo2_xUEZpFTG5Rdp4DC0aOjWl_o7pobHSVpplxnGUaveSeeQx4Aapi-key";
    private readonly string _apiUrl = "https://api.openai.com/v1/completions";

    public TranslationService(IHttpClientFactory httpClientFactory)
    {
        _httpClientFactory = httpClientFactory;
    }

    public async Task<string> TranslateTextAsync(string text)
    {
        using (var client = _httpClientFactory.CreateClient())
        {

            client.DefaultRequestHeaders.Add("Authorization", $"Bearer {_apiKey}");
```

```
        var requestBody = new
        {
            model = "gpt-3.5-turbo",
            prompt = $"Translate this text to English: {text}",
            max_tokens = 60
        };

        var json = JsonConvert.SerializeObject(requestBody);
        var content = new StringContent(json, Encoding.UTF8, "application/json");


        var response = await client.PostAsync(_apiUrl, content);

        if (response.IsSuccessStatusCode)
        {

            var responseContent = await response.Content.ReadAsStringAsync();
            dynamic result = JsonConvert.DeserializeObject(responseContent);


            return result.choices[0].text.ToString().Trim();
        }
        else
        {
            return "Translation failed";
        }
    }
  }
}
```

WeatherForecast.cs

```
namespace Translate
{
  public class WeatherForecast
  {
    public DateOnly Date { get; set; }

    public int TemperatureC { get; set; }

    public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

    public string? Summary { get; set; }
  }
```

```
}
```

Program.cs

```
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddHttpClient();


builder.Services.AddDbContext<TranslationDbContext>(options =>
   options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddControllers();
static IHostBuilder CreateHostBuilder(string[] args) =>
      Host.CreateDefaultBuilder(args)
         .ConfigureWebHostDefaults(webBuilder =>
         {
            webBuilder.ConfigureServices(services =>
            {

               services.AddMemoryCache();


               services.AddHttpClient();


               services.AddScoped<ITranslationService, TranslationService>();

               services.AddControllers();
            });
            webBuilder.Configure(app =>
            {
               app.UseRouting();
               app.UseEndpoints(endpoints =>
               {
                  endpoints.MapControllers();
               });
            });
         });

var app = builder.Build();
```

```
if (app.Environment.IsDevelopment())
{
   app.UseSwagger();
   app.UseSwaggerUI();
}
app.UseHttpsRedirection();

app.MapControllers();

app.Run();
```

manifest.json

```
{
   "manifest_version": 3,
  "name": "Web Page Translator",
  "description": "Translate web pages and text using AI",
  "version": "1.0",
  "permissions": [
    "activeTab",
    "storage",
    "scripting"
  ],
  "background": {
      "service_worker": "background.js"
  },
  "action": {
      "default_popup": "popup.html"
  },
  "host_permissions": [
    "http://*/*",
    "https://*/*"
  ]
}
```

popup.html

```
using System.Runtime.InteropServices;

< !DOCTYPE html >
< html lang = "en" >
< head >
   < meta charset = "UTF-8" >
   < meta name = "viewport" content = "width=device-width, initial-scale=1.0" >
   < title > Translate Extension </ title >
   < link rel = "stylesheet" href = "styles.css" >
```

```
</ head >
< body >
  < div class= "container" >
    < h1 > Перекладач </ h1 >

    < !--Переклад слова-- >
    < div class= "translate-section" >
      < input id = "wordInput" type = "text" placeholder = "Введіть слово для перекладу" />
      < button id = "translateWordButton" > Перекласти слово </ button >
      < p id = "wordTranslationResult" ></ p >
    </ div >

    < !--Переклад сторінки-- >
    < div class= "translate-page" >
      < button id = "translatePageButton" > Перекласти всю сторінку</button>
    </div>
  </div>

  <script src = "popup.js" ></ script >
</ body >
</ html >
```

popup.js

```
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Reflection.Metadata;
using System;

document.addEventListener('DOMContentLoaded', function() {
  const translateWordButton = document.getElementById("translateWordButton");
  const wordInput = document.getElementById("wordInput");
  const wordTranslationResult = document.getElementById("wordTranslationResult");
  const translatePageButton = document.getElementById("translatePageButton");
  if (!translateWordButton || !wordInput || !wordTranslationResult || !translatePageButton)
  {
    console.error("Елементи не знайдено на сторінці!");
    return;
  }

  translateWordButton.addEventListener("click", () => {
    const word = wordInput.value.trim();

    if (!word)
    {
      wordTranslationResult.textContent = "Будь ласка, введіть слово для перекладу.";
      return;
```

```
        }
        fetchTranslation(word);
    });
    translatePageButton.addEventListener("click", () => {
    chrome.tabs.query({ active: true, currentWindow: true }, (tabs) => {
    const tab = tabs[0];
    if (tab)
    {
        chrome.scripting.executeScript({
        target: { tabId: tab.id },
                function: translatePage
            });}
        });
    });
});
function fetchTranslation(word)
{
    fetch('https://localhost:44322/api/Translate', {
    method: 'POST',
        headers:
        {
            'Content-Type': 'application/json',
            'Accept': '*/*'
        },
        body: JSON.stringify({ text: word })
    })
    .then(response => response.json())
    .then(data => {
        wordTranslationResult.textContent = data.translated ? `Переклад: ${ data.translated}` :
"Помилка перекладу.";
    })
    .catch(error => {
        wordTranslationResult.textContent = "Помилка при запиті до API.";
    });
}

function translatePage()
{
    const pageContent = document.body.innerHTML;
    fetch('https://localhost:44322/api/Translate', {
    method: 'POST',
        headers:
        {
            'Content-Type': 'application/json',
            'Accept': '*/*'
        },
```

```
      body: JSON.stringify({ text: pageContent })
    })
    .then(response => response.json())
    .then(data => {
      if (data.translated)
      {
        document.body.innerHTML = data.translated;
      }
    })
    .catch(error => {
      console.error("Помилка перекладу сторінки:", error);
    });
}
```

content.js

```
const pageContent = document.body.innerHTML;
fetch('https://localhost:44322/api/Translate', {
method: 'POST',
   headers:
   {
      'Content-Type': 'application/json',
      'Accept': '*/*'
   },
   body: JSON.stringify({ text: pageContent })
})
.then(response => response.json())
.then(data => {
   if (data.translated)
   {
      document.body.innerHTML = data.translated;
   }
})
.catch(error => {
   console.error("Помилка перекладу сторінки:", error);
});
```

background.js

```
using System;
chrome.runtime.onInstalled.addListener(function() {
   console.log("Розширення успішно встановлено!");
});

chrome.action.onClicked.addListener(function(tab) {

   console.log("Іконка натиснута на сторінці: " + tab.url)});
```

Додаток Г - Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНОГО ЗАСОБУ ДЛЯ АВТОМАТИЗОВАНОГО ПЕРЕКЛАДУ ТЕКСТІВ У ВЕБ-ДОДАТКАХ НА ОСНОВІ НЕЙРОМЕРЕЖ

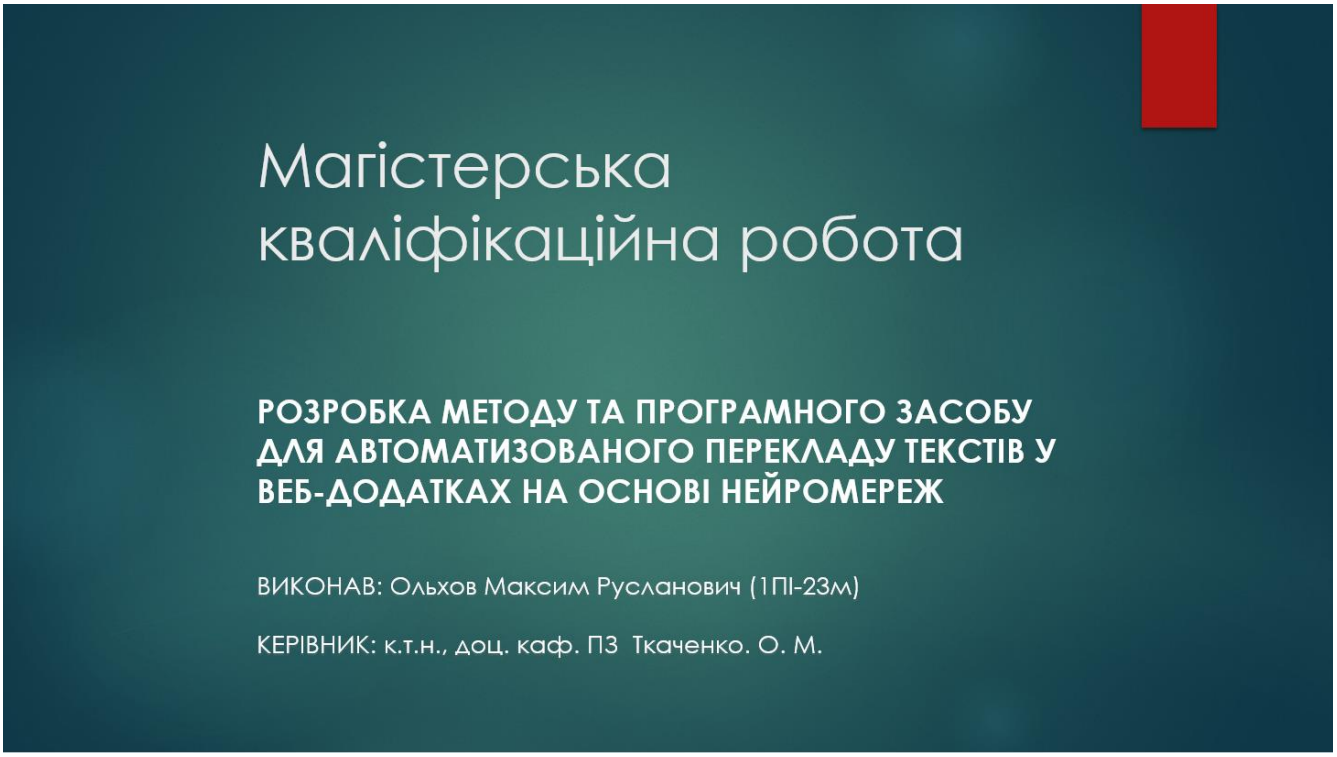


Рисунок Г.1 – Титульний слайд

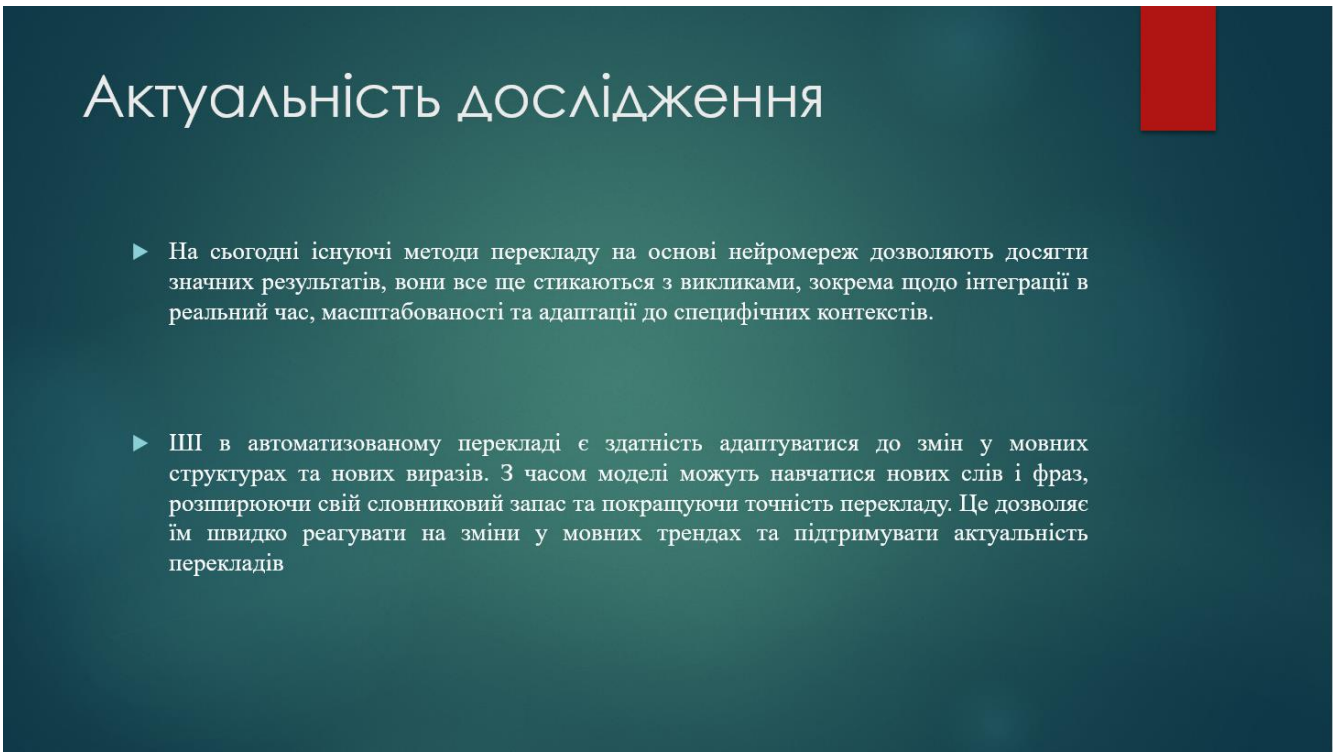


Рисунок Г.2 – Актуальність дослідження

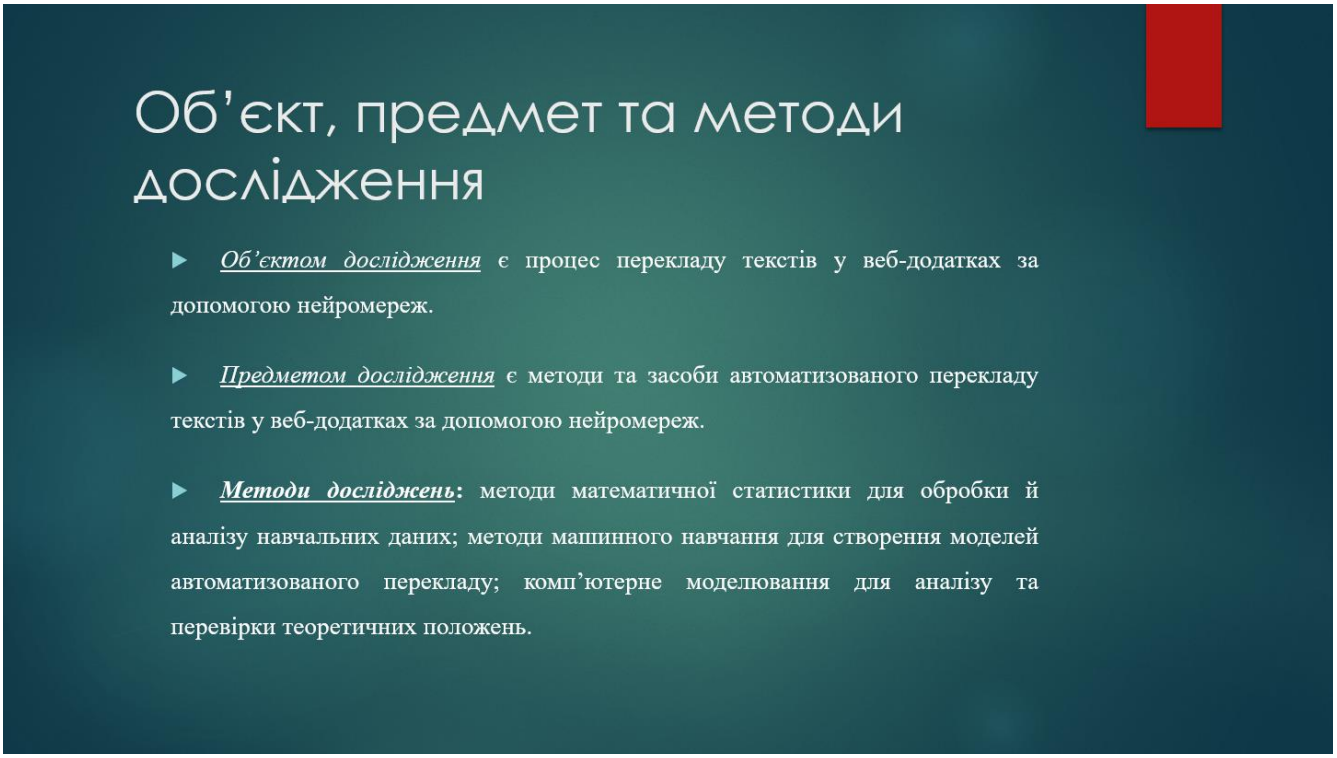


Рисунок Г.3 – Об’єкт, предмет та методи дослідження

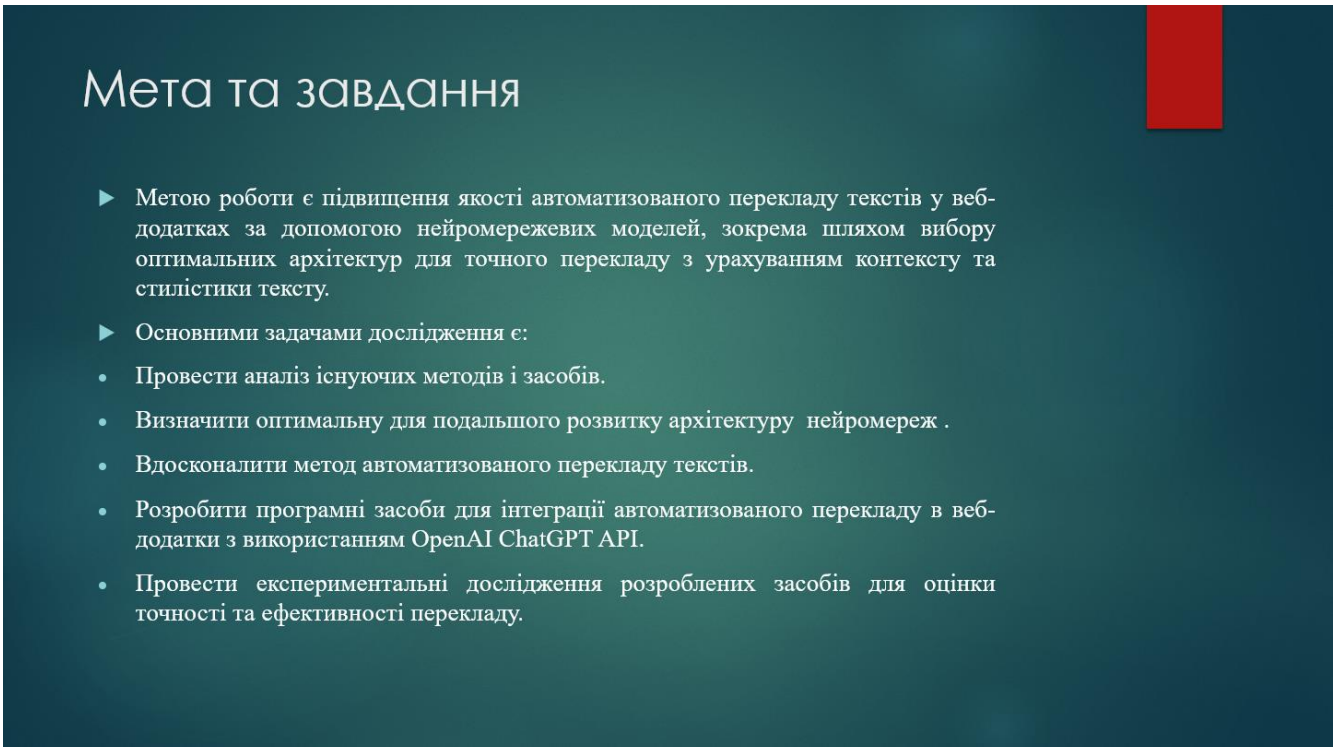


Рисунок Г.4 – Мета та завдання

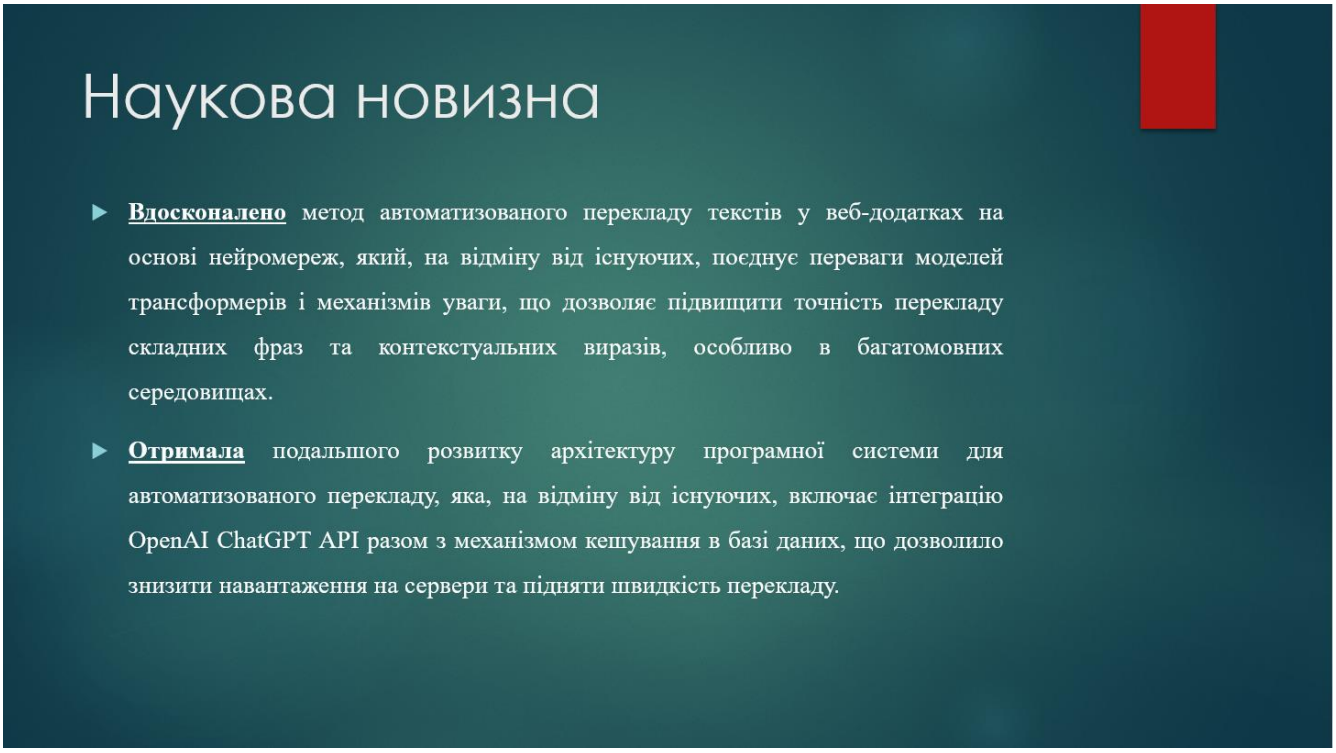


Рисунок Г.5 – Наукова новизна

Рисунок Г.6 – Аналоги

Рисунок Г.7 – Порівння аналогів

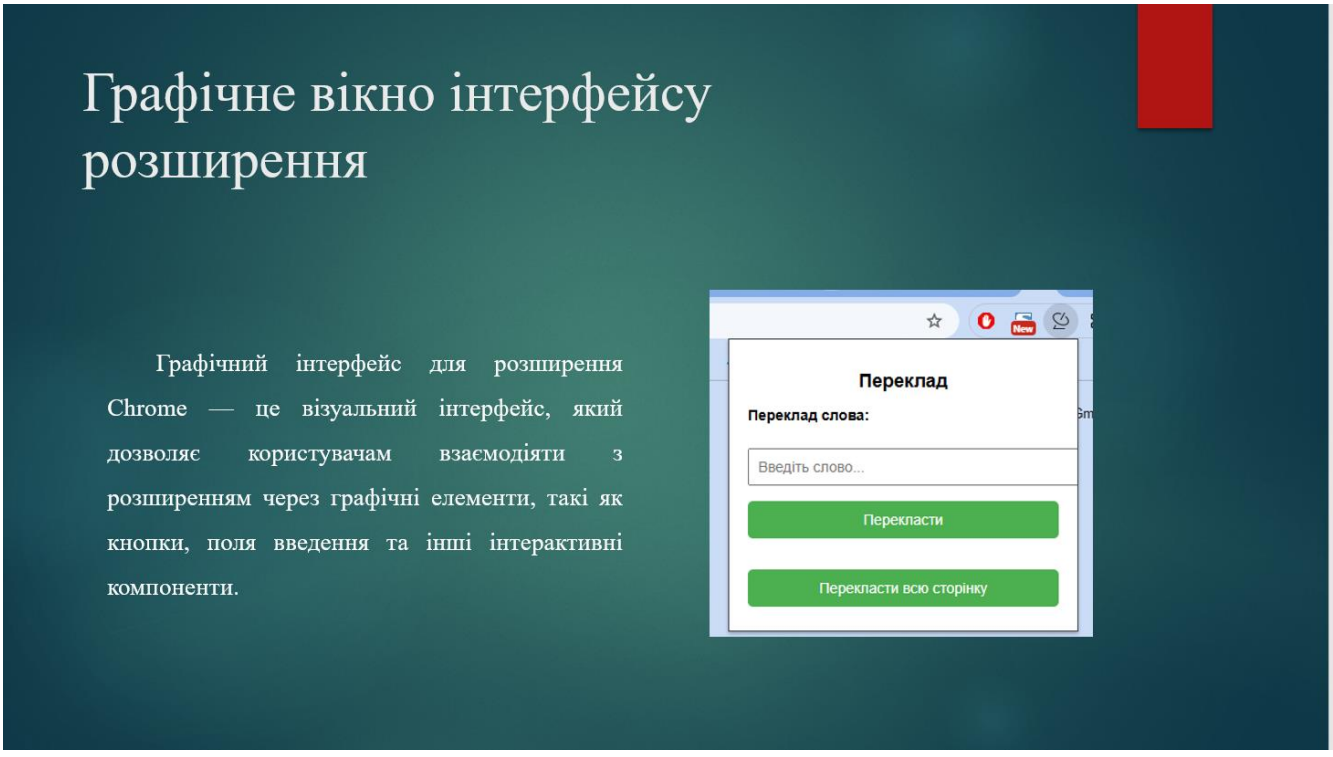
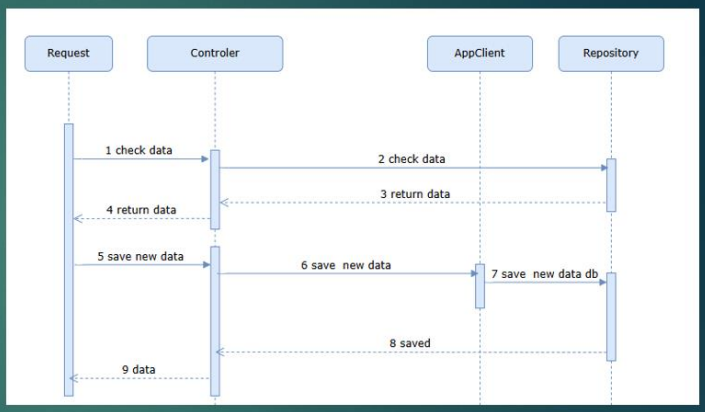


Рисунок Г.8 – Графічне вікно інтерфейсу розширення

Алгоритму роботи

Для цього потрібно створити механізм перевірки наявності перекладу в кеші перед виконанням запиту до API. Якщо результат перекладу для певного тексту вже є в кеші, система може негайно повернути цей результат без необхідності повторно звертатися до API.



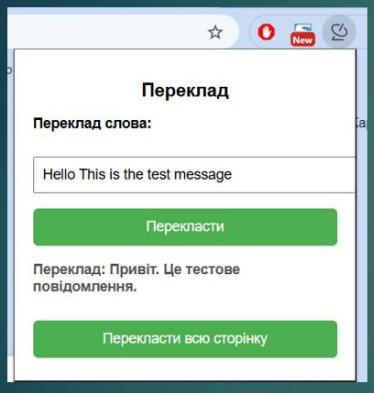
dbo.TranslationCaches [Data] | dbo.TranslationCaches [Design] | dbo.__EFMigrat...istor

Max Rows: 1000

Id	OriginalText	TranslatedText	CachedAt
1	string	рядок	07.11.2024 14:3...
2	пробний текст	Test Message	07.11.2024 14:3...
3	Рядок	String	07.11.2024 14:3...
4	Hello this is the...	Привіт, це тестове повідомлення	07.11.2024 14:3...

Рисунок Г.9 – Алгоритму роботи

Тестування програмної частини



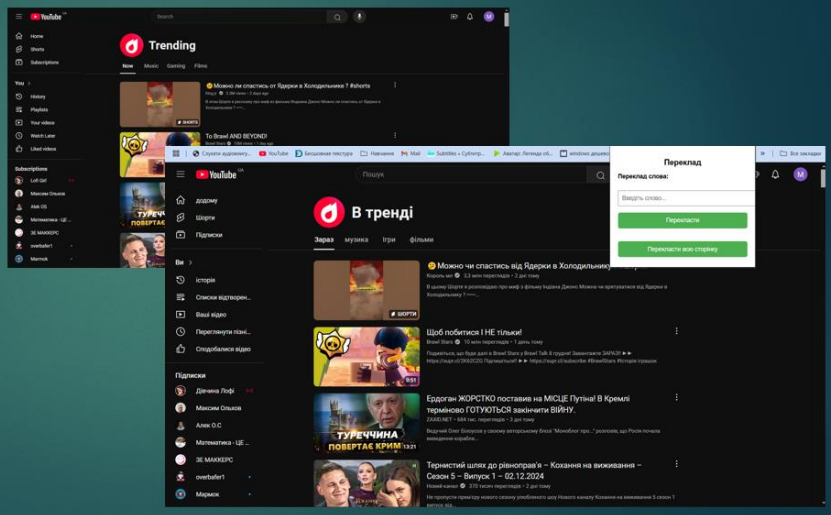


Рисунок Г.10 – Тестування програмної частини

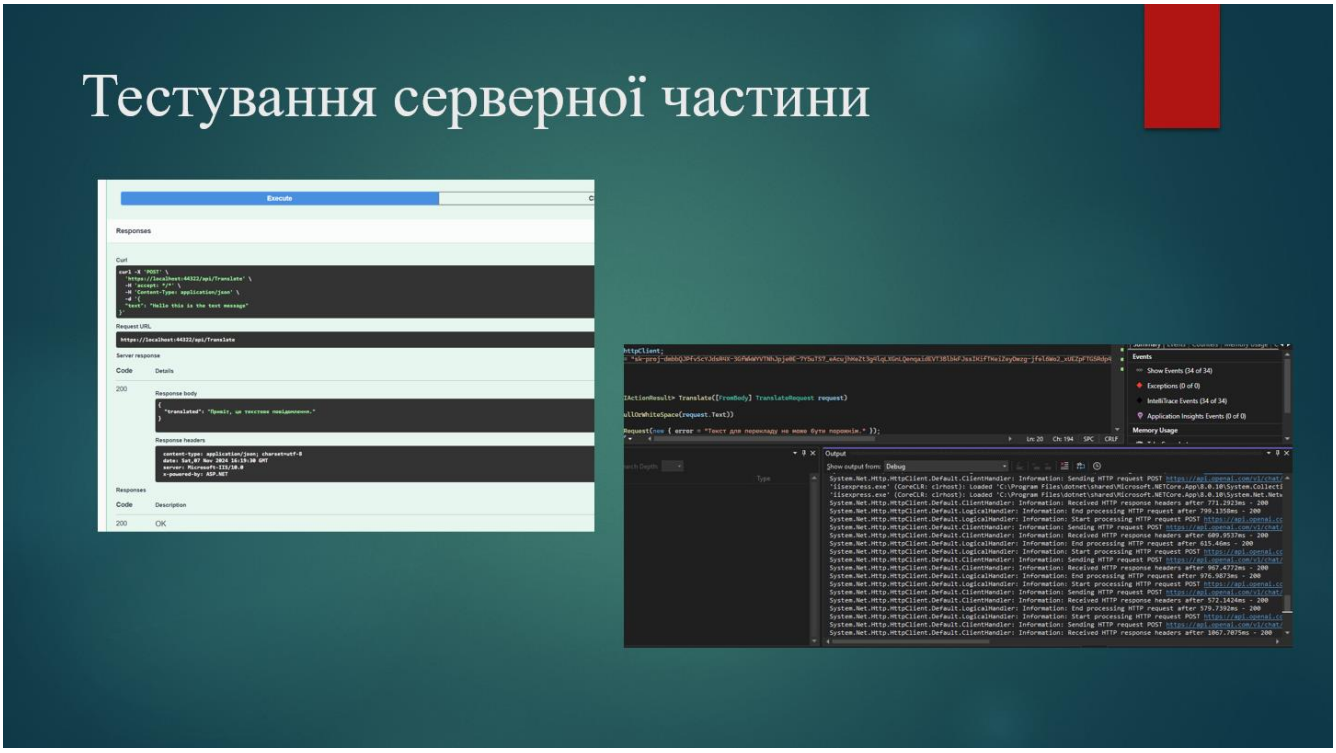


Рисунок Г.11 – Тестування серверної частини

Засіб	Точність	Швидкість обробки	Інтеграція та масштабованість
Розроблений засіб	Висока точність для технічних текстів та окремих фраз	Швидкість вища середньої	Широка інтеграція
Google Translate	Висока для основних мовних пар, але можуть бути помилки для спеціалізованих термінів	Швидка обробка текстів	Інтеграція через API в різні платформи
DeepL	Найвища точність для технічних текстів	Швидкість середня	Обмежена інтеграція
Microsoft Translate	Висока точність для стандартних умов і технічних текстів	Швидкість середня	Інтеграція з продуктами Microsoft

Рисунок Г.12 – Висновки

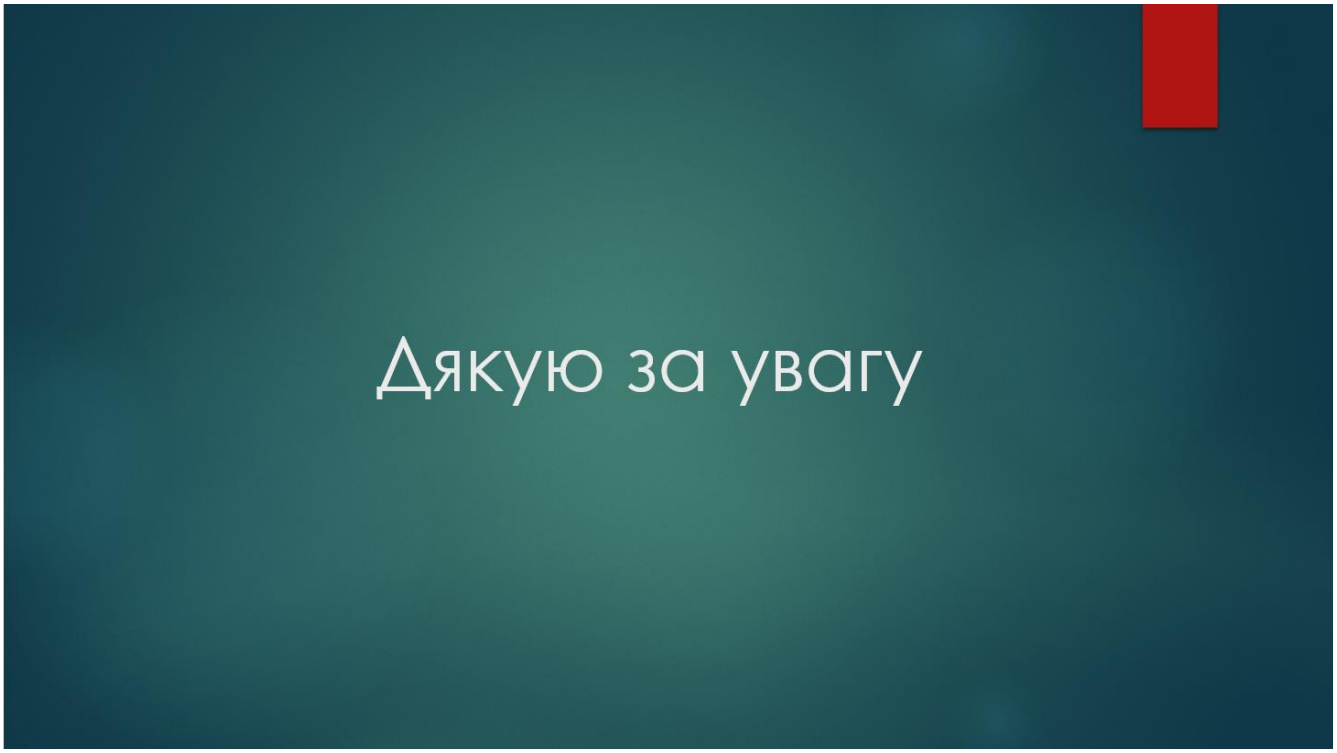


Рисунок Г.12 – Дякую за увагу