Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ДЛЯ ОБРОБКИ НИЗЬКО-РЕСУРСНИХ МОВНИХ ПАР
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-2
Боденчук-Пастухов Є. В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник асист. Кобилін І. О.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Боденчуку-Пастухову Єгору Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтелектуальної системи для обробки низько-ресурсних мовних пар

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 27 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали наукових статей, дані інтернет-мережі, бібліотека нейронних мереж Hugging Face Transformers, проект паралельних корпусів Tatoeba, бібліотека для розробки сервісної частини Spring Boot, бібліотека для розробки інтерфейсів користувача React, середовище для розгортки баз даних Docker.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд моделей обробки тексту та машинного перекладу.
2. Методи оцінювання машинного перекладу.
3. Розробка вебзастосунку.
4. Розгортка нейронних мереж.
5. Налаштування гіперпараметрів нейронних мереж на власному наборі даних.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми, постановка задачі, зображення оцінок метрик, методи оцінювання машинного перекладу, моделі обробки низько-ресурсних мовних пар.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-12.04.25	
4	Аналіз технічних засобів	12.04.25-13.04.25	
5	Розгортка нейронних мереж	13.04.25-25.04.25	
6	Програмна реалізація	25.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ асист. Кобилін І. О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 67 с., 8 табл., 35 рис., 30 джерел.

JAVA, SPRING BOOT, REACT JS, REACT, TRANSFORMERS, SEQUENCE-TO-SEQUENCE, НИЗЬКО-РЕСУРСНІ МОВНІ ПАРИ, ПЕРЕКЛАД, МЕХАНІЗМ УВАГИ, POSTGRESQL, MATERIAL UI.

Об’єктом роботи є низько-ресурсні тексти для обробки та перекладу.

Метою роботи є розробка багато сервісного інтелектуального вебзастосунку, що дозволяє обробити та перекладати тексти з японської, корейської та китайської мов на українську мову.

У роботі проведено аналіз та оцінку доступних open-source нейронних моделей на архітектурі Transformer та Sequence-to-Sequence, які підтримують обрані пари мов. Були використані сучасні технології для розробки сервісів та розгортування нейромереж, метрики для визначення ефективності кожної з моделей.

У результаті роботи здійснена програмна реалізація системи для читання та паралельного перекладу текстів з японської, китайської та корейської мов на українську мову.

JAVA, SPRING BOOT, REACT JS, REACT, TRANSFORMERS, SEQUENCE-TO-SEQUENCE, LOW-RESOURCE LANGUAGE PAIRS, TRANSLATION, ATTENTION MECHANISM, POSTGRESQL, MATERIAL UI.

Object of the work low-resource texts for processing and translation.

The aim of the work is to develop a multi-service intelligent web application that enables the processing and translation of texts from Japanese, Korean, and Chinese into Ukrainian.

The work includes analysis and evaluation of available open-source neural models based on the Transformer and Sequence-to-Sequence architectures that support the selected language pairs. Modern technologies were used to develop the services and deploy the neural networks, as well as metrics to assess the effectiveness of each model.

As a result of the work, a software system was implemented for reading and parallel translation of texts from Japanese, Chinese, and Korean into Ukrainian.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

BLEU – Bilingual Evaluation Understudy (двомовне оцінювання)

METEOR – Metric for Evaluation of Translation with Explicit Ordering (оцінка машинного перекладу з урахуванням порядку слів)

TER – Translation Edit Rate (рівень редагування перекладу)

ChrF – Character F-score (символьна міра *F*)

NLP – Natural Language Processing (обробка природньої мови)

UI – User Interface (інтерфейс користувача)

REST – Representational State Transfer (передача стану представлення)

API – Application Programming Interface (інтерфейс програмування застосунку)

ШІ – штучний інтелект

MNMT – Multimodal Neural Machine Translation (мультимодальна нейронна система машинного перекладу)

PE – Positional Encoding (позиційне кодування)

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

LSTM – Long Short-Term Memory (мережа з довгою короткочасною пам’яттю)

BPE – Byte Pair Encoding (парне кодування байтів)

MBPE – Multilingual Byte Pair Encoding (багатомовне парне кодування байтів)

mBART – Multilingual Bidirectional and Auto-Regressive Transformer (багатомовний двонаправлений та автоматично-регресивний перетворювач)

BART – Bidirectional and Auto-Regressive Transformer (двонаправлений та автоматично-регресивний перетворювач)

НМП – нейронний машинний переклад

VDOM – Virtual Document Object Model (віртуальна об’єктна модель документа)

HTTP – Hypertext Transfer Protocol (протокол передачі гіпертексту)

CORS – Cross-Origin Resource Sharing (спільне використання ресурсів між джерелами)

CPU – Central Processing Unit (центральний процесор)

GPU – Graphical Processing Unit (графічний процесор)

ВСТУП

Обробка та переклад низько-ресурсних мовних пар залишається серйозною проблемою в сучасній обробці природної мови (NLP, Natural Language Processing). Незважаючи на величезну кількість текстових даних, доступних в Інтернеті, знайти достатній і високоякісний набір даних для навчання моделей машинного перекладу для таких мов все ще важко.

Більше 85% мов по всьому світу можна вважати низько-ресурсними. Інформації у відкритому доступі для таких мов менше ніж для інших. Через це, виникає важкість у знаходженні даних для таких мов, які б допомагали у тренуванні нейронних моделей або створені якісних наборів даних.

Метою цього проєкту є розробка інтелектуального вебзастосунку, який зможе оброблювати та перекладати з японської, китайської та корейської мов на українську мову. Такі пари мов вважаються низько-ресурсними і одним із складніших у перекладі та обробці. В основу системи закладено мікросервісна архітектура з використанням інтерфейсу програмування застосунків на основі передачі стану (REST API, Representation State Transfer Application Programming Interface), сучасні моделі у відкритому доступі для обробки та генерації тексту, технології для розгортування баз даних та технології для динамічного інтерфейсу користувача (UI, User Interface).

Актуальність роботи полягає у оцінці та уніфікації обробки та перекладу з японської, китайської та корейської мов на українську мову, адже досліджень саме із такими парами мов не так багато. Така система має заохочувати людей до зацікавлення та вивчення представлених мов та дає підґрунтя для стартапів та малих бізнесів для легкої інтеграції подібного функціоналу на свої ресурси.

1 ОГЛЯД ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Характеристика низько-ресурсних мовних пар

Низько-ресурсні мовні пари – це такі мовні пари, які мають значно менше вмісту, доступного в Інтернеті. З приблизно 7 тис. мов, якими розмовляють у всьому світі, 90% знаходяться під загрозою зникнення до наступного століття, що класифікує їх як низько-ресурсні мови. Прикладами мов із низьким ресурсом є хінді, албанська, баскська, зулу, шотландська гельська, галісійська, ірландська, фінська словацька, японська, корейська і саме головне українська.

Існує і протилежність низько-ресурсних мовних пар – високо-ресурсні мовні пари. Високо-ресурсні мовні пари – це такі мовні пари, які мають велику присутність у медіа просторі. З великою упевненістю можна сказати, що майже усі мовні пари, в яких присутня англійська мова вважаються високо-ресурсними через те, що англійська мова широко розповсюджена не тільки у живому спілкуванні, але й в медійному просторі. В одному тільки Інтернеті існує чи мало документів, відеоматеріалів та аудіоматеріалів на англійській мові.

Слід зауважити, що мови на яких розмовляє більшість людей також можуть вважатися низько-ресурсними, наприклад хінді. На цій мові розмовляє велика частина людей у світі, але багато публікацій та документів у Індії доступні в першу чергу на англійській мові, адже вона являється другої національною мовою країни. Ось чому в Інтернеті ви знайдете порівняно менш якісний двомовний вміст із гінді як вихідною або цільовою мовою.

Саме тому дослідження в галузі обробки низько-ресурсних мов стають дедалі актуальнішими. Вони сприяють збереженню мовної різноманітності та

забезпечують доступ до інформації для мовних спільнот, які раніше були виключені з цифрового простору.

1.2 Критерії визначення мовної пари як низько-ресурсна

Як вже було зазначено, в основному мовну пару називають «низько-ресурсною», коли вона має обмежені дані в медіа просторі, але це не єдиний спосіб кваліфікації.

Однією з основних причин призначення мовної пари до «низько-ресурсної» є наявності лінгвістичних інструментів для їх обробки. Під поняття лінгвістичних інструментів підпадають наступні поняття: токенізатори, морфологічні аналізатори, синтаксичні аналізатори. Якщо корпус мовної пари не має в наявності перелічених інструментів або їх кількість та якість не задовольняє потреби то таку мовну пару також можна назвати «низько-ресурсною».

Якщо для корпусу відсутня офіційна підтримка від популярних фреймворків та бібліотек для роботи з NLP таких як spaCy, NLTK, Stanza або Hugging Face Transformers, це також є ознакою. Також слід зазначити, що відсутність державної підтримки оцифрування мови також є індикатором низько-ресурсного статусу.

Наведені критерії враховуються при розробці мультилингвістичних систем обробки та перекладу тексту для таких моделей, як MarianMT, mBART та M2M100, спеціалізація яких базується на роботі із низько-ресурсними мовами через методи глибокого навчання, архітектури Transformer, методів трансферного навчання та багатомовного навчання [1].

1.3 Найкращі та найгірші мови для машинного перекладу

1.3.1 Де машинний переклад кращий?

Переклад англійською мовою вважається найліпшим для машинного перекладу, адже багато систем надзвичайно добре обробляють англійську мову. Галузеві звіти, як-от «State of Machine Translation» від компанії «Intento», підтверджують, що переклад англійською мовою є ключовою перевагою більшості перекладачів на основі штучного інтелекту (ШІ) [2].

Слід також зазначити західноєвропейські мови, наприклад французька, німецька або іспанська. Міцні політичні та економічні зв'язки між західноєвропейськими країнами призвели до великої кількості багатомовних даних, що значно підвищило якість перекладів між цими мовами.

Романські мови (французька, італійська, португальська, іспанська) також є одними з ліпших для тренування власних нейронних моделей, адже романські мови мають спільну латинську спадщину, що робить їх структурно подібними. Їхній словниковий запас і граматика, що збігаються, дозволяють виконувати дуже точні переклади між ними.

Машинний переклад має перевагу і між скандинавськими мовами (данською, норвезькою, шведською) тому, що ці мови тісно споріднені, що призводить до досить надійних перекладів з однієї на іншу.

Крім того, варто окремо згадати про слов'янські мови, зокрема польську, чеську та словацьку. Попри те, що вони мають складні граматичні структури та велику кількість відмінків, переклад між цими мовами також демонструє прийнятну якість, особливо завдяки їхній морфологічній подібності. Вони часто розглядаються як перспективні для тренування багатомовних моделей у межах регіональних проєктів перекладу.

У контексті Азії, японська, китайська та корейська мови вирізняються високою складністю перекладу через ієрогліфічні системи письма, суттєві відмінності у синтаксисі та стилістичні особливості. Проте завдяки великому обсягу відкритих паралельних корпусів, зібраних для цих мов, сучасні моделі

демонструють помітне покращення якості перекладу, особливо у напрямку з цих мов на англійську.

Також слід зауважити, що машинні перекладачі досягають кращих результатів у мовах з фіксованим порядком слів та меншою кількістю морфологічних змін.

1.3.2 Мови, що становлять виклик для машинного перекладу

В першу чергу це азіатські мови (китайська, японська, корейська), хоча якість перекладу для цих мов значно покращилася за останні роки, їхні структурні відмінності від європейських мов – зокрема, у граматиці, синтаксисі та системі письма – все ще становлять серйозні проблеми.

Далі можна зазначити африканські мови. Африканські мови залишаються значною мірою недостатньо представленими в дослідженнях NLP. Багато з них не є широко вивченими чи задокументованими, що обмежує розробку якісних засобів перекладу. Є надія, що в майбутніх дослідженнях ці мови будуть пріоритетними, особливо ті, яким загрожує зникнення.

Українська – це ще одна мова, у звіті «Intento» про стан машинного перекладу вказується, що доступних ресурсів для перекладу не так багато, незважаючи на велику кількість носіїв [2].

Окрему категорію становлять мови з високим ступенем діалектної варіативності, як-от арабська. Суттєві відмінності між розмовними варіантами арабської та її літературною формою ускладнюють навчання універсальної перекладацької моделі. Це ж стосується й мов з сильною залежністю від контексту, таких як тайська або бірманська, де значення слова може змінюватися залежно від інтонації, соціального статусу мовця або контексту розмови.

Також значні труднощі виникають під час обробки аглютинативних мов, таких як турецька, угорська або фінська. У таких мовах окреме слово може включати довгі ланцюжки афіксів, що кодують різноманітну граматичну інформацію. Через це словоформи рідко повторюються в корпусах, що ускладнює побудову точних статистичних і нейронних моделей.

1.4 Аналіз досліджень у галузі обробки низько-ресурсних мовних пар

1.4.1 Нова архітектура для нейронних мереж від дослідників Google

По-перше варто згадати, що значну зміну у сфері обробки низько-ресурсних мовних пар відіграла поява архітектури Transformer від дослідників Google у статті «Attention is all you need» [3]. На рисунку 1.1 зображено архітектуру Transformer.

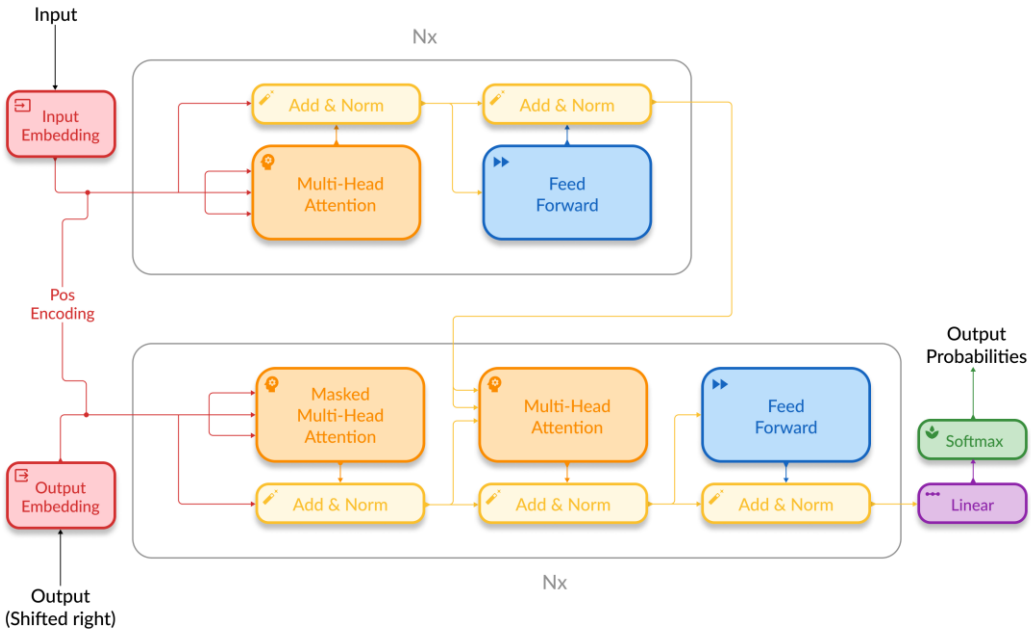

Рисунок 1.1 – Візуальне зображення архітектури Transformer [3]

Такий архітектурний підхід значно збільшив обробку та генерацію взагалі будь-якої мови у світі та дозволив якісно й паралельно оброблювати великі потоки текстової інформації. Проблема полягає у тому, що тренування особистої моделі потребує значних комп'ютерних ресурсів, великих об'ємів даних та часу.

1.4.2 Дослідження у тренуванні мультилингвістичних нейронних машинних перекладів за допомогою адаптерів мовних сімейств

Група дослідників у 2023 році запропонувала оброблювати та перекладати мовні сімейства за допомогою адаптерів на прикладі нейронних моделей mBART-50 та OPUS-100. Наведені моделі являються мультилингвістичними, які використовують Sequence-to-Sequence архітектуру та здатні для перекладу великого корпусу низько-ресурсних мовних пар.

Адаптер відіграє важливу роль у даному досліджені, так як він являється ключовим елементом для підвищення двомовного оцінювання (BLEU, Bilingual Evaluation Understudy) на мовних сімействах. Сама архітектура адаптеру була взята із архітектурної моделі Transformer, який складається із слою нормалізації, проєкції та функції активації ReLU. Адаптер можна записати наступною формулою:

$$Adapter_i(z_i) = U \times ReLU(D \times LN(z_i)) + z_i, \quad (1.1)$$

де z_i – вхідний вектор;

$LN(z_i)$ – шар нормалізації;

D – лінійне зменшення розмірності у вигляді матриці;

$ReLU$ – активаційна функція.

У цьому досліджені українська мова була визначена до балтійсько-словацького сімейства та була оцінена на моделях OPUS-100 та mBART-50 без використання запропонованої технології поєднання адаптеру із архітектурою Sequence-to-Sequence. Були отримані оцінки BLEU на сімейство у даному випадку 15,1 від OPUS-100 та 22,2 від mBART-50. Українська мова не взяла участі у фінальній оцінці, але у балтійсько-словацького сімейства середнє значення BLEU вийшло 21,3 з використанням адаптеру [4].

1.4.3 Дослідження налаштування гіперпараметрів для моделей на архітектурі Transformer

Дослідження «Transformers for Low-Resource Languages: Is Feidir Linn!» наглядно демонструє налаштування гіперпараметрів для моделі, що використовує архітектуру Transformer для низько-ресурсної пари англійської мови до ірландської мови. Модель являється базовою імплементацією OpenNMT 2.0 у бібліотеці Pytorch. В даному випадку було взято наступні метрики: двомовна оцінка (BLEU, Bilingual Evaluation Understudy), рівень редагування перекладу (TER, Translation Edit Rate), символьна міра F (ChrF, Character F-score). Гіперпараметри, які були взяті для налаштування представлені у наступній таблиці (табл. 1.1).

Важливо зауважити, що значний акцент було зроблено на адаптації під специфіку ірландської мови, яка, подібно до інших низько-ресурсних мов, має нестандартну синтаксичну структуру, обмежену кількість анотованих корпусів та значну морфологічну варіативність. Це вимагало ретельного підходу до токенізації вхідних даних – зокрема, використання словникових схем, що дозволило зменшити кількість рідковживаних токенів і підвищити узгодженість векторного представлення.

Таблиця 1.1 – Гіперпараметри, які використовувалися для налаштування [5]

Гіперпараметри	**Значення**
Швидкість навчання	0,1; 0,01; 0,001; 2
Розмір партії	1024, 2048, 4096, 8192
Кількість голів уваги	2, 4, 8
Кількість шарів	5, 6
Розмірність прямого поширення	2048
Розмір ембедингів	128, 256, 512
Згладжування міток	0,1; 0,3
Пропускання нейронів	0,1; 0,3
Вимкнення шарів уваги	0,1
Середнє згасання	0, 0.0001

У результаті дослідження було отримано наступні метрики, які представлені у наступній таблиці (табл. 1.2).

Таблиця 1.2 – Продуктивність моделі на 52k DGT наборі даних [5]

Архітектура	**BLEU**	**TER**	**ChrF**	**Кроки**
dgt-trans-base	53,4	0,41	0,72	55тис.
dgt-trans-bpe8k	59,5	0,34	0,77	200тис.
dgt-trans-bpe16k	60,5	0,33	0,78	180тис.
dgt-trans-bpe32k	59,3	0,35	0,77	100тис.
dgt-trans-unigram	59,3	0,35	0,77	125тис.

Виходить, що найліпший результат вийшов у моделі із 2-ма механізмами уваги. Усі інші мали 8 таких механізмів.

Дане дослідження демонструє, як можна ефективно робити налаштування гіперпараметрів для моделей, які використовують архітектуру

Transformer, для підвищення результату обробки та перекладу низько-ресурсної мовної пари [5] Окрім того, отримані висновки можна адаптувати до інших мов із подібними викликами, що відкриває перспективи для розширення застосування Transformer-архітектур у контексті глобальної мовної різноманітності.

1.4.4 Використання мультимодальних підходів

Дедалі більше уваги привертають мультимодальні нейронні системи машинного перекладу (MNMT, Multilingual Neural Machine Translation), які поєднують текстову та візуальну інформацію. Такий підхід є особливо цінним для мов із обмеженою кількістю лінгвістичних ресурсів, адже зображення можуть слугувати додатковим джерелом семантики для покращення якості перекладу.

У дослідженні «Enhancing Neural Machine Translation of Low-Resource Languages: Corpus Development, Human Evaluation and Explainable AI Architectures» було продемонстровано ефективність використання зображень для навчання MNMT. Проблемою, з якою стикнулися автори, стала відсутність трьохстороннього паралельного корпус, необхідного для повноцінного навчання такої системи.

Для подолання цієї проблеми автори створили синтетичний корпус навчальних даних, в якому англомовні підписи з наявного корпусу зображень були перекладені хінді за допомогою автоматичного перекладу та побудували ручний референтний набір для тестування та розробки, який містить оригінальні зображення та підписи англійською з точним людським перекладом на хінді.

Результати дослідження демонструють, що у випадку низько-ресурсної мовної пари можна використати залучення синтетичних даних і візуального

контексту. Таким чином, зображення відіграють роль «семантичного посередника», що компенсує нестачу текстових ресурсів [6].

1.5 Потенційна користь та соціальний ефект

На сьогодні українським студентам, викладачам і дослідникам у сфері сходознавства, історії, філософії, лінгвістики та культурології бракує україномовного доступу до оригінальних джерел з Китаю, Японії та Кореї. Багато цінних наукових і культурних праць доступні лише на мовах оригіналу, або, в кращому випадку, англійською.

Створення інструменту, що дозволяє читати, перекладати й інтерпретувати ці тексти в українському контексті, значно підвищить якість вищої освіти та досліджень в Україні.

Система може використовуватись для оцифрування та перекладу східних книг, що відкриває шлях до створення публічних електронних бібліотек або навіть навчальних платформ з унікальними матеріалами.

Окрім наукової і літературної цінності, така система може лягти в основу інструментів для локалізації культурного контенту. Це відкриває можливості для креативних індустрій в Україні.

Дослідження має перспективу для поточного інтегрування в малі підприємства, бізнеси, наукові застосунки, освітні застосунки.

1.6 Проблеми та перспективи перекладу на українську мову в системах машинного перекладу

Незважаючи на офіційний статус української мови, її значну кількість носіїв та наявність академічного і публічного мовлення, в контексті машинного перекладу вона продовжує залишатися мовою із частково

низьким рівнем ресурсного забезпечення. Це зумовлює низку фундаментальних викликів при розробці інтелектуальних систем, які працюють з українською як цільовою мовою. Це зумовлено кількома факторами:

– обмежена кількість високоякісних паралельних корпусів, особливо в поєднанні з азійськими мовами, такими як японська, китайська чи корейська;

– невисока присутність в офіційних мовних моделях великих компаній як Google, Meta;

– складна морфологія та гнучкий порядок слів української мови створює додаткові труднощі для генеративних моделей.

Крім недостатнього обсягу даних, специфіка української мови як синтетичної, флективної, з вільним порядком слів, створює додаткове навантаження на механізми attention у трансформерних архітектурах. Моделі схильні втрачати синтаксичну та семантичну узгодженість при перекладі довгих речень, де порядок слів є критичним для збереження значення. Це особливо помітно при перекладі з аналітичних мов, таких як китайська, де контекст передається ієрогліфічними структурами та позиційним синтаксисом, що кардинально відрізняється від українських мовних норм.

Незважаючи на це, останні роки показали позитивні зрушення:

– зростає кількість відкритих паралельних корпусів на таких проєктах як Tatoeba, OPUS, CCAligned, naprklad;

– з’являються ініціативи з перекладу великих мовних моделей на українську та fine-tuning open-source моделей, таких як mBART, NLLB.

У світлі сучасних досліджень, зокрема робіт Meta AI з моделлю NLLB-200, спостерігається позитивна динаміка в напрямі залучення малоресурсних мов до глобальних мовних моделей. Проте ці системи все ще демонструють помітно нижчі метрики BLEU, ChrF та у метриці оцінки машинного перекладу з урахуванням порядку слів (METEOR, Metric for Evaluation of Translation with Explicit Ordering) для української в парі з неєвропейськими мовами.

Варто також звернути увагу на те, що сама природа української мови вимагає від систем машинного перекладу глибшого розуміння контексту, ніж це є необхідним у випадках з мовами, де структура речення суворо фіксована. Під впливом інтонації, авторського стилю чи жанрових особливостей, один і той самий мовний зворот може набувати відтінків. Особливо гостро ця проблема проявляється у перекладі художніх, публіцистичних чи емоційно забарвлених текстів.

1.7 Постановка задачі

Таким чином, обробка та переклад низько-ресурсних мовних пар залишається сучасною проблемою, яка стосується більшості мов, включаючи українську мову та її паралельні корпуси із азіатськими мовами. Тому для паралельної обробки обираються японська, китайська та корейська мови та ставиться задача дослідження різних методів обробки і перекладу з обраних мов на українську, їх оцінка та використання в уніфікованому застосунку для перекладу текстів.

Об’єктом роботи є низько-ресурсні тексти для обробки та перекладу.

Метою роботи є розробка багато сервісної інтелектуального вебзастосунку, що дозволяє обробити та перекласти тексти з японської, корейської та китайської мов на українську мову.

Для досягнення мети необхідно вирішити такі завдання:

- аналіз існуючих сучасних методів обробки та перекладу;
- розробити fine-tuning алгоритм для обраних моделей;
- реалізувати багатосервісну архітектуру для вебзастосунку;
- реалізувати розгортування баз даних;
- реалізувати виконання генерації тексту на графічному процесорі;
- провести оцінку метрик BLEU, TER, ChrF, METEOR;
- розробити компактний UI для застосунку.

2 МАТЕМАТИЧНЕ ОБГРУНТУВАННЯ МЕТОДІВ ТА МОДЕЛЕЙ

2.1 Опис архітектури Transformer

Нейронні мережі архітектури Transformer складаються з двох основних блоків: кодувальника та модуля декодування. Ядром цієї нейронної мережі є механізм багатоголової уваги, який значною мірою базується на механізмі самоуваги. Самоувага – це алгоритм, що обчислює матрицю, яка відображає взаємозв’язки між словами в послідовності [7].

Архітектура Transformer вирішує проблему захоплення довготривалих залежностей у послідовностях – завдання, з яким важко справляються рекурентна нейронна мережа (RNN, Recurrent Neural Network) та мережа з довгою короткочасною пам’яттю (LSTM, Long Short-Term Memory) через їхню послідовну природу. На відміну від RNN, де кожне слово повинно оброблятися послідовно, механізм самоуваги дозволяє архітектурі Transformer розглядати всі слова в послідовності одночасно, що робить його значно ефективнішим у моделюванні віддалених зв’язків між словами.

Ембединг вхідних даних є ключовим етапом у процесі обробки моделі Transformer. Воно полягає у представленні кожного токена у вхідній послідовності у вигляді вектора одноразового кодування, який множиться на матрицю ембедингів (E), щоб отримати вхідні вбудовування (X). Ця матриця вбудовування є параметром, що навчається під час тренування моделі. Також до вхідних вбудовувань додається позиційне кодування, щоб надати моделі інформацію про порядок слів у послідовності. Це забезпечує ефективну обробку послідовностей змінної довжини – одну з головних переваг архітектури Transformer. Ці вбудовування передаються на вхід кодувальника, де проходять через шари багатоголової уваги та нейронної мережі прямого поширення. Завдяки паралельній обробці та механізмам

нормалізації, Transformer забезпечує високу продуктивність і стабільність під час навчання на великих обсягах даних [7].

$$X = E \times I, \tag{2.1}$$

де X – вхідні вбудовування;

E – матриця вбудовування;

I – вхідні one-hot вектори.

Позиційне кодування (PE) додається до вхідних вбудовувань, щоб надати моделі інформацію про порядок слів [7]. Це значення для позиції pos і виміру i обчислюється за наступними формулами:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \tag{2.2}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right). \tag{2.3}$$

Багатоголова увага – один з ключових елементів архітектури Transformer, який забезпечує моделі здатність одночасно фокусуватися на різних частинах вхідної послідовності. Вона складається з декількох одноголових механізмів уваги, які можуть обробляти послідовності паралельно. Така паралельна обробка дозволяє виявляти різні типи залежностей: наприклад один потік може зосередитися на граматичних зв’язках, а інший – на семантичних.

Кожен одноголовий механізм самоуваги обчислює матрицю уваги, яка відображає ступінь впливу кожного слова в послідовності на інші слова, який досягається шляхом обчислення подібності між вектором кожного слова та всіма іншими словами через скалярний добуток після відповідного

перетворення в матриці ключів, матриці запитів та матриці значень [7]. Механізм уваги можна описати наступною формулою:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_{model}}})V\ , \tag{2.4}$$

де Q – матриця запитів;

K – матриця ключів;

V – матриця значень;

d_k – розмірність ключів.

Кожен рядок у вихідній матриці відображає не лише значення слова (згідно з його емебдингом), а й його позицію в реченні (через позиційне кодування) та взаємодію з іншими словами в послідовності. Матриці Q, K та V зазвичай мають розмір $n \times d_{model}$, де n це довжина послідовності, а d_{model} це розмір вбудовувань моделі.

У формулі уваги використовується функція softmax для нормалізації рядків результуючої матриці, забезпечуючи, щоб сума кожного рядка дорівнювала 1. Крім того, функція softmax перетворює великі від'ємні значення (наприклад, -∞) у 0, фактично ігноруючи їх. Це дозволяє моделі зосередитися на найбільш релевантних словах у контексті.

Функція softmax визначається наступним чином:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_i}}, \tag{2.5}$$

де z_i – елемент вектору.

Тож самоувага допомагає сформувати матрицю взаємозв'язків між словами для кращого розуміння контексту тексту. Проте одна самоувага менш ефективна, ніж багатоголова, оскільки в останньому випадку ми маємо

кілька «голів». Кожна голова обчислює зв’язки між словами незалежно, а потім результати об’єднуються в одну матрицю.

Перевага використання кількох «голів» полягає в тому, що кожна з них може фокусуватися на різних аспектах вхідної послідовності. Це дозволяє моделі захоплювати різноманітні типи зв’язків між словами – як синтаксичні, так і семантичні – покращуючи загальну ефективність механізму уваги [7].

Багатоголову увагу можна описати наступною формулою:

$$MultiHead(Q,K,V) = Concat(head_1 ... head_h)W^o, \quad (2.6)$$

де $head$ – це результат обчислення з формули самоуваги;

h – кількість голів;

W^0 – вагова матриця останньої лінійної проєкції після конкатенації всіх голів.

Після багатоголової уваги отриманий вихід передається через прямий нейронний шар, який складається з двох лінійних перетворень із активаційною функцією ReLU між ними [7].

Математично це можна описати наступною формулою:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2, \quad (2.7)$$

де W_1 – вагова матриця першого лінійного шару;

b_1 – зміщення для першого лінійного шару;

W_2 – вагова матриця другого лінійного шару;

b_2 – зміщення для другого лінійного шару.

Після кожного шару прямої нейронної мережі та багатоголової уваги застосовується залишкове з’єднання та шар нормалізації Залишкове з’єднання допомагає уникнути проблеми зникнення градієнта, що критично для глибоких моделей, а нормалізація прискорює та стабілізує процес навчання.

На кожному етапі дані проходять наступну обробку:

$$LayerNorm(x + Sublayer(x)), \tag{2.8}$$

де $Sublayer(x)$ – багатоголова увага, або feed-forward нейронна мережа.

Після проходження всіх блоків енкодера або декодера, дані передаються через лінійний шар і функцію softmax. На цьому етапі ми отримуємо матрицю ймовірностей вихідних токенів. Щоб обрати остаточний результат, можна використати жадібний алгоритм, який на кожному кроці вибирає токен із найвищою ймовірністю.

2.2 Опис метрик для згенерованого тексту

2.2.1 Опис двомовного оцінювання

Двомовне оцінювання (BLEU, Bilingual Evaluation Understudy) – це алгоритм оцінки якості тексту, який був перекладений з однієї природної мови на іншу [8]. Якість вважається відповідністю між результатом роботи машини та результатом роботи людини.

Таким чином основна ідея BLEU – чим ближчий машинний переклад до професійного людського перекладу, тим він кращий, що лежить в основі BLEU, який був одним із перших показників, який стверджував про високу кореляцію з людськими оцінками якості, і залишається одним із найпопулярніших автоматизованих та недорогих показників.

BLEU працює на основі підрахунку збігів *n*-грам – послідовностей слів у перекладеному реченні, які також зустрічаються в референтних перекладах. Він також включає модифікований коефіцієнт точності для запобігання переоцінці часто повторюваних слів.

Оцінки розраховуються для окремих перекладених сегментів – зазвичай речень – шляхом порівняння їх з набором якісних еталонних

перекладів. На наступному кроці ці оцінки усереднюють по всьому корпусу, щоб отримати оцінку загальної якості перекладу. Ні зрозумілість, ні граматична правильність не враховуються.

$$BLEU = BP \times \exp(\sum_{n=1}^{N} w_n \log(p_n)), \tag{2.9}$$

де BP – штраф за стислість, який карає занадто короткі переклади;

p_n – точність $n-gram$, яка вимірює перекриття між згенерованими та еталонними перекладами;

w_n – вага, яка присвоєна кожному $n-gram$;

n – максимальний порядок $n-gram$, що розглядаються.

Точність $n-gram$ обчислюється наступним чином:

$$p_n = \frac{\sum_{n-grams} count\_match}{\sum_{n-grams} count\_gen}, \tag{2.10}$$

де $count\_match$ – кількість $n-grams$, згенеровані у перекладі, які з'являлись у довідковому перекладі.

$count\_gen$ – кількість $n-grams$, які були згенеровані під час перекладу.

Штраф за стислість забезпечує балансування оцінки BLEU, адже без нього короткі переклади могли б отримувати несправедливо високі оцінки через більший відсоток n-gram, що збігаються з референтом. Така система оцінювання особливо важлива в тих випадках, коли машинний переклад видає неповні або урізані результати, які хоч і точні, але не передають повного змісту оригінального тексту. Цей штраф застосовується тоді, коли довжина гіпотетичного перекладу менша за довжину еталонного. Його вплив посилюється із зростанням різниці між довжинами, що стимулює моделі до створення повноцінних, змістовно завершених перекладів [8].

Штраф за стисліcть розраховується за наступною формулою:

$$BP = \begin{cases} 1, & c > r \\ e^{(1-r/c)}, & c \le r \end{cases}, \tag{2.11}$$

де c –довжина згенерованого перетворення;

r – довжина найближчого опорного перетворення.

2.2.2 Опис символьної міри *F*

Символьна міра *F* (ChrF, Character *F*-score) – це метрика оцінки машинного перекладу, яка працює на рівні символів, а не слів, що робить її особливо корисною для морфологічно багатих мов [9]. Вона базується на *F*-оцінці з використанням точності та повноти *n*-грам на рівні символів.

ChrF був запропонований як альтернатива BLEU, оскільки останній часто не демонструє високої кореляції з людськими оцінками для мов із складною морфологією або в умовах низько-ресурсного перекладу. Метрика враховує як точність, так і повноту, що дозволяє отримати більш збалансовану оцінку [9].

Слід зазначити, що ChrF менш чутливий до сегментації, що робить його ефективним для мов, де слово існують складнощі з токенізацією та дозволяє активного його використовувати при оцінювання як традиційних, так і нейронних систем NLP та машинного перекладу для складних мовних пар [9].

Крім того, символьний підхід дозволяє точніше враховувати морфемні та суфіксальні зміни, які часто втрачаються при аналізі на рівні слів. Це особливо важливо для аглютинативних та флективних мов, де велика кількість граматичної інформації закодована в афіксах. У таких мовах слова

можуть змінюватися в залежності від відмінків, часу, числа чи роду, що призводить до утворення великої кількості словоформ.

ChrF базується на F-шкалі, яка врівноважує точність та повноту та розраховується за наступною формулою:

$$ChrF = (1+\beta) \times \frac{(\frac{1}{N}\sum_{k=1}^{N} P_k) \times (\frac{1}{N}\sum_{k=1}^{N} R_k)}{\beta^2 \times (\frac{1}{N}\sum_{k=1}^{N} P_k) + (\frac{1}{N}\sum_{k=1}^{N} R_k)}, \quad (2.12)$$

де N – максимум послідовності $n-gram$;

β – ваговий параметр, який зазвичай встановлюється як 2, щоб трохи більше надавати перевагу запам’ятовування, ніж точності;

P_k – точність для $n-grams$ довжиною k;

R_k – повнота для $n-grams$ довжиною k.

Метрика базується на *F*-оцінці, яка інтегрує два важливих аспекти якості перекладу – точність та повноту.

Точність та повнота рахується за наступною формулою:

$$P_k = \frac{\sum_{k-grams} count\_match}{\sum_{k-grams} count\_gen}, \quad (2.13)$$

$$R_k = \frac{\sum_{k-grams} count\_match}{\sum_{k-grams} count\_ref}, \quad (2.14)$$

де $count\_match$ – кількість збігів символів між згенерованим та еталонним перекладами;

$count\_gen$ – загальна кількість згенерованих символів під час перекладу;

$count\_ref$ – загальна кількість символів у довідковому перекладі.

2.2.3 Опис метрики оцінки машинного перекладу з урахуванням порядку слів

Метрика оцінки машинного перекладу з урахуванням порядку слів (METEOR, Metric for Evaluation of Translation with Explicit Ordering) – це метрика оцінки машинного перекладу, яка покращується шляхом врахування точності та запам'ятованості, синонімії та кореневої структури, штрафів за порядок слів [10]. На відміну від BLEU, METEOR гнучко вирівнює слова, враховуючи точні збіги, кореневі збіги та синоніми, що робить його надійнішим для морфологічно багатих мов.

Загальна формула для обчислення наступна:

$$METEOR = F_{mean} \times (1 - Penalty), \tag{2.15}$$

де F – це гармонійне середнє значення точності та повноти;

$Penalty$ – пояснює помилки порядку слів.

Точність та повноту можна обчислити наступним чином:

$$F_{mean} = \frac{(1+\beta^2) \times P \times R}{\beta^2 \times P + R}, \tag{2.16}$$

де β – константа, яка зазвичай дорівнює 9 для того щоб надати більше ваги повноті;

P – кількість збігів слів, поділена на загальну кількість слів у варіанті перекладу;

R – це кількість збігів слів, поділена на загальну кількість слів у довідковому перекладі.

Штраф у METEOR вводиться для зменшення оцінки в разі наявності багатьох фрагментованих відповідностей між гіпотезою та референтом, дозволяючи «покарати» переклади, в яких порядок слів значно відрізняється

від людського перекладу, що може негативно впливати на зрозумілість [10]. Штраф можна розрахувати за формулою:

$$Penalty = \gamma \times (\frac{ch}{m})^{\theta}, \tag{2.17}$$

де ch – кількість фрагментів;

m – загальна кількість збігів слів;

γ, θ – змінні параметрами (часто встановлюються на 0,5 та 3 відповідно).

Кінцеве значення METEOR формується шляхом обчислення балансу між точністю і повнотою перекладу з урахуванням можливих штрафів за порушення послідовності слів. Менше значення штрафу забезпечує вищу підсумкову оцінку, в той час як серйозні відхилення у порядку передавання змісту можуть суттєво знизити результат. Завдяки цьому формується здатність враховувати лексичну варіативність і гнучкість в оцінюванні якості перекладу навіть за наявності морфологічних змін [10].

2.2.4 Опис рівня редагування перекладу

Рівень редагування перекладу (TER, Translation Edit Rate) – це метрика, яка вимірює кількість редагувань, необхідних для перетворення перекладу, згенерованого системою, на еталонний переклад. На відміну від METEOR або BLEU, TER штрафує редагування, тобто нижчі бали кращі. Нижчий бал TER означає, що потрібно менше редагувань, що вказує на кращий переклад.

Метрика дозволяє прямо оцінити зусилля, яке повинен докласти людина-редактор для виправлення машинного перекладу. Особливістю TER є можливість використання перестановки слів як окремої операції редагування, що покращує точність оцінки в умовах, коли порядок слів у мовах суттєво різниться.

TER можна розрахувати за наступною формулою:

$$TER = \frac{E}{R}, \tag{2.18}$$

де E – кількість необхідних редагувань;

R – кількість слів у довідковому перекладі.

Особливістю TER є те, що операція переміщення розглядається окремо і зазвичай враховується як одна редагувальна дія, навіть якщо за інших умов така трансформація вимагала б послідовних операцій видалення та вставки.

Наприклад, для приведення машинного перекладу «Cat is sitting on mat» до референтної версії «The cat sat on the mat» необхідно виконати наступні редагування: додати артикль «The» на початку речення, замінити «is sitting» на «sat», а також вставити артикль «the» перед словом «mat». Таким чином, кількість редагувань у цьому випадку складатиме щонайменше три операції.

Отже, значення E відображає реальні витрати зусиль на виправлення машинного перекладу, а зменшення кількості редагувань безпосередньо свідчить про підвищення якості автоматичного перекладу.

2.3 Опис обраних моделей

2.3.1 Опис моделі Many-to-Many 100

M2M100 (Many-to-Many 100) – багатомовна модель, яка була розроблена дослідниками Facebook AI у 2020 році. Ця модель стала першою, яка підтримувала переклад між будь-якою парою із 100 мов без використання англійської мови як проміжної. Для низько-ресурсних мов, де англійська не є природною проміжною мовою, такий підхід дозволяє значно покращити якість перекладу завдяки прямому навчанню на паралельних корпусах.

M2M100 базується на архітектурі Transformer із використанням принципу кодувальника та модулю декодування, однак така модель має спільний словник із 128 тис. частин, побудований методом багатомовного парного кодування байтів (MBPE, Multilingual Byte Pair Encoding) із урахуванням символів понад 100 мов [11, 12]. Також навчання здійснювалось на 7,5 мільярдах речень із 100 мов, а на початку кожного речення додається спеціальний токен, що вказує на цільову мову. Також слід зазначити, що корпус формувався так, щоб уникнути перенавчання моделі на високо-ресурсних мовних парах, а особливо таких паралельних корпусах, в яких присутня англійська мова.

Загальний принцип роботи починається з токенізації вхідного тексту, який відбувається за допомогою загального словника парного кодування байтів (BPE, Byte Pair Encoding). На вхід подається спеціальний мовний токен цільової мови. Кодувальник обробляє джерельну послідовність та створює контекстні представлення. Модуль декодування генерує переклад ітеративно, враховуючи попередні токени та контекст. Універсальний словник BPE дозволяє повторно використовувати структури слів між мовами. У таблиці 2.1 наведено структуру моделі M2M100.

Таблиця 2.1 – структура M2M100 [11]

Компонент	**Параметри**
Шари кодувальника	12
Шари модулю декодування	12
Кількість прихованих шарів	1024
Кількість голів уваги	16
Розмір ембедингів	1024
Параметри	~418 млн.

Завдяки архітектурі Many-to-Many та великій кількості тренувальних даних, M2M100 демонструє високу якість перекладу не лише для популярних мовних пар, але й для низько-ресурсних, зменшуючи залежність від англійської як мостової мови.

Переклад у моделі відбувається згідно з класичною умовною ймовірністю:

$$P(Y \mid X, L_T) = \prod_{t=1}^{T} P(y_t \mid y_1, ..., y_{t-1}, X, L_T), \tag{2.19}$$

де X – вхідна послідовність токенів джерельною мовою;

y – послідовність токенів перекладу;

L_T – вказівка цільової мови.

2.3.2 Опис моделі Small100

Small100 – багатомовна модель машинного перекладу, яка була створена для підтримки 100 мовних пар у форматі моделі Many-to-Many. Ця модель була створена з метою експериментів та досліджень у сфері багатомовного перекладу [13].

Модель Small100 використовує архітектуру Transformer із використанням кодувальника та модулю декодування, однак із меншою кількістю параметрів для більшої легкості моделі.

Попри свою компактність, модель демонструє конкурентні результати при перекладі між низько-ресурсними мовами, особливо у випадках, коли великі моделі не можуть бути використані через технічні обмеження. Small100 також може слугувати відправною точкою для подальшого навчання на специфічних мовних парах або доменах. У таблиці 2.2 наведено структуру моделі Small100.

Таблиця 2.2 – структура Small100 [13]

Компонент	**Параметри**
Шари кодувальника	4-6
Шари модулю декодування	4-6
Кількість прихованих шарів	~512
Кількість голів уваги	8
Розмір ембедингів	512
Параметри	~60-90 млн.

Модель адаптована під BPE. Розмір вбудованого словника приблизно дорівнює 32 тис. токенів. Для цієї моделі було використано наступні набори даних: OpenSubtitles, CCMatrix, WikiMatrix, Tatoeba, JW300.

Модель можна описати наступною формулою:

$$P(y_1, y_2, ..., y_T \mid X, L_T) = \prod_{t=1}^{T} P(y_t \mid y_{<t}, Enc(X, L_T)), \quad (2.20)$$

де $Enc(X, L_T)$ – контекстна репрезентація вхідного речення X, з урахуванням цільової мови L_T.

2.3.3 Опис моделі Multilingual Bidirectional and Auto-Regressive Transformer

mBART (Multilingual Bidirectional and Auto-Regressive Transformer) – багатомовна модель машинного перекладу, розроблена Facebook AI як розширення оригінальної моделі mBART, що підтримує 50 мов та переклад між будь-якими парами . На відміну від моделей типу BART (Bidirectional and Auto-Regressive Transformer), які були призначені для

обробки лише англійського тексту, mBART створено з урахуванням багатомовної пре-тренування і переносу знань між мовами [14-16].

Модель mBART побудована на основі архітектури Sequence-to-Sequence Transformer з шумовим автоматичним кодувальником. У таблиці 2.3 наведено структуру моделі mBART.

Таблиця 2.3 – Структура mBART [14]

Компонент	**Параметри**
Шари кодувальника	12
Шари модулю декодування	12
Кількість прихованих шарів	1024
Кількість голів уваги	16
Розмір ембедингів	1024
Параметри	~610 млн.

Модель використовує токенізатор SentencePiece з ULM (Unigram Language Model). Словник моделі становить 250 тис. частин. Модель підтримує 50 мов, включаючи японську, корейську, китайську та українську [14-16]. На початку речення присутній маркер, який вказує вхідну мову.

Модель можна описати наступною формулою:

$$L_{NMT} = -\sum_{t=1}^{T} \log P(y_t \mid y_{<t}, X, L_T), \tag{2.21}$$

де X – токенізоване джерельне речення;

L_T – код цільової мови;

y_t – послідовність токенів перекладу.

3 РЕАЛІЗАЦІЯ ПРОТОТИПУ ЗАСТОСУНКУ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи було розроблено вебзастосунок, в який було інтегровано та розгорнуто нейронні моделі, які були налаштовані та оцінені на власному набору даних для обробки та машинного перекладу текстів з японської, корейської та китайської мов на українську мову. З урахуванням складності проєкту, який поєднує в собі як класичні програмні шари інтерфейсу користувача та серверної частини, так і обчислювальні компоненти пов’язані з ШІ, було обрано три ключові інструменти: IntelliJ IDEA як основне інтегроване середовище розробки (IDE, Integrated Development Environment) для роботи з Java серверною частиною і JavaScript частиною інтерфейсу користувача, середовище Lightning AI для організації, обгортання і запуску моделей машинного навчання, а також середовище Docker для ізоляції, розгортання та масштабування всіх компонентів проєкту.

Інтегроване середовище розробки IntelliJ IDEA було обрано для реалізації серверної частини через його глибоку інтеграцію з екосистемою Spring, підтримку Java, Maven, а також велику кількість розширень для автоматизації типових завдань розробника. Зокрема, можливість автоматичного створення конфігурацій, генерації коду, інспекції залежностей, перевірки синтаксису в реальному часі, а також перевірка коду суттєво пришвидшили процес написання стабільного та чистого коду. Вбудовані інструменти для профілювання, роботи з базами даних, REST API та контейнерами Docker дозволили звести до мінімуму потребу в сторонніх інструментах. Крім того, IntelliJ IDEA забезпечує комфортне середовище для написання коду, завдяки якому можна безболісно змінювати структуру коду навіть у великих проєктах, не побоюючись втратити зв’язки між класами чи порушити логіку. Приклад інтерфейсу середовища IntelliJ IDEA показано на рисунку 3.1.

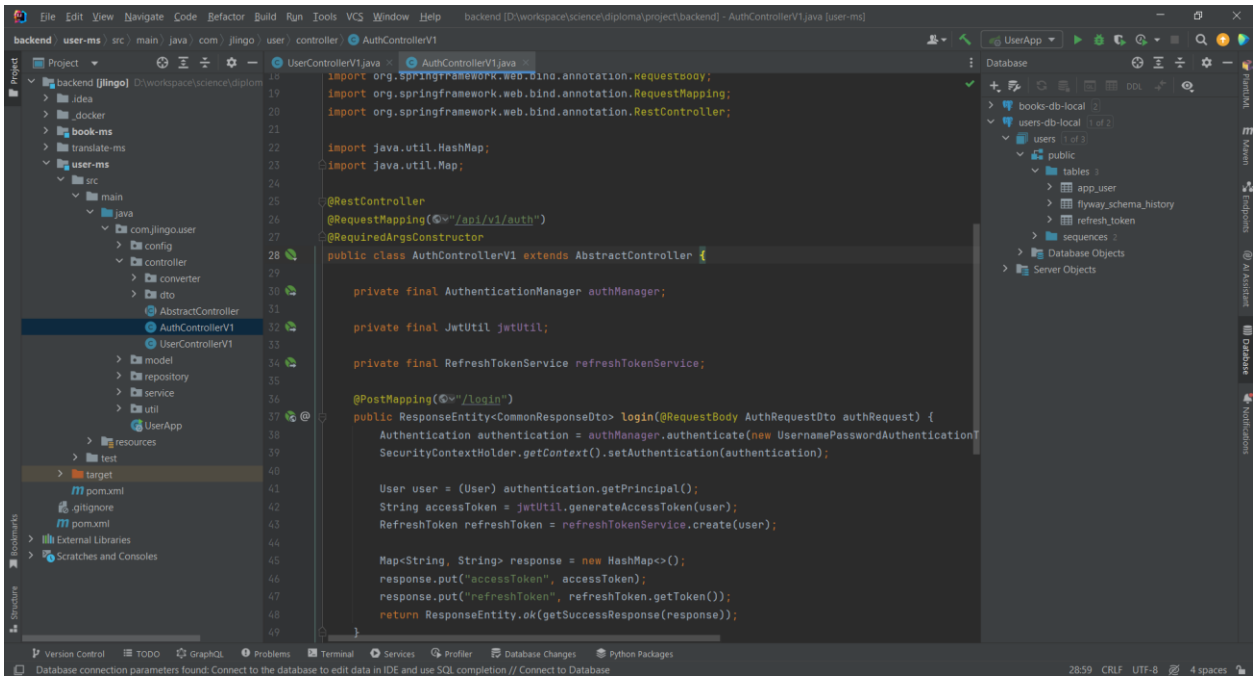

Рисунок 3.1 – Інтерфейс середовища IntelliJ IDEA

Для роботи з нейронними моделями та машинним навчанням було обрано Lightning AI – середовище, що дозволяє стандартизувати тренування, збору відміток, розгортання та управління нейронних мереж у виробничому середовищі. Завдяки модульності та високому рівню абстракції, це середовище дало змогу легко поєднати попередньо навчені моделі з власним набором даних, оптимізувати гіперпараметри, організувати логіку процесу отримання результатів від моделі на основі вхідних даних й водночас зберегти контроль над низько-рівневими аспектами.

Особливо цінним є те, що Lightning AI дозволяє зручно розмежувати навчання, перевірку та тестування, автоматизуючи рутинні процеси й підвищуючи повторюваність експериментів, що критично важливо в контексті кваліфікаційної роботи. Крім того, використання Lightning AI дозволило уникнути надмірної фрагментації коду, що часто трапляється при побудові експериментального середовища вручну. Усі етапи – від завантаження даних до розрахування метрик – було інтегровано в єдину логічну структуру, що значно спростило відстеження змін у процесі навчання та повторне відтворення результатів. Такий підхід забезпечив не лише

чистоту архітектури, а й можливість швидко масштабувати модель або змінювати конфігурацію, не переписуючи значну частину коду. На рисунку 3.2 зображено інтерфейс Lightning AI.

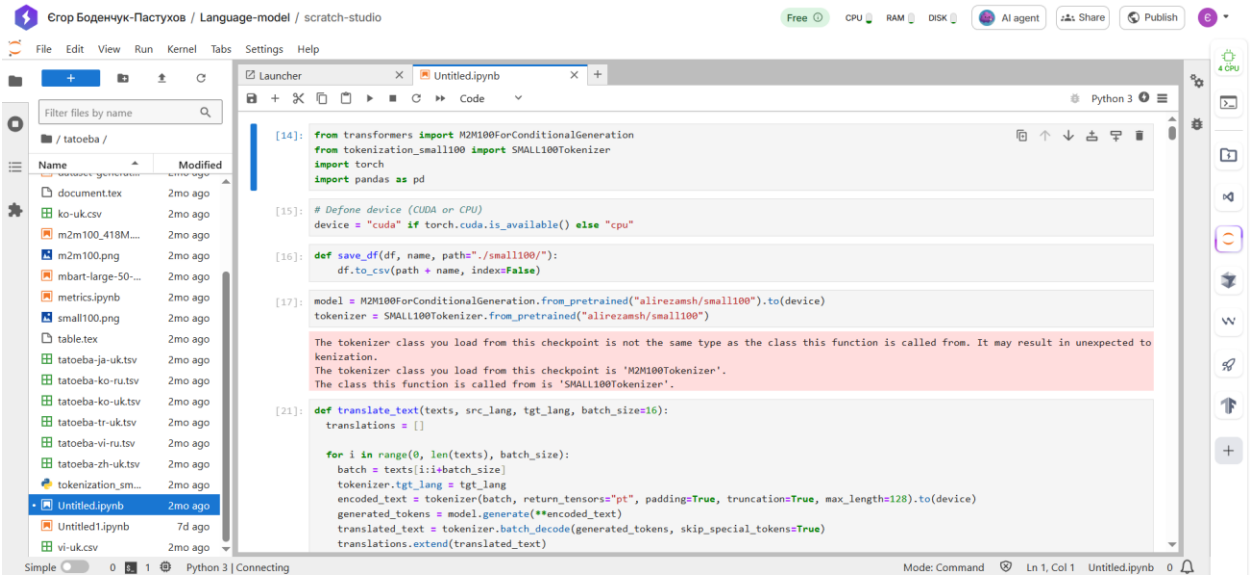

Рисунок 3.2 – Інтерфейс середовища Lightning AI

Для ізоляції середовищ, уникнення конфліктів залежностей та забезпечення можливості переносу застосунку між різними платформами було впроваджено Docker. Завдяки контейнеризації вдалося досягти єдиного стандарту розгортання: незалежно від середовища розробки чи цільового сервера, кожен компонент має власний Dockerfile і конфігурацію. Це дозволяє запускати застосунок як локально, так і в хмарному середовищі без необхідності вручну відтворювати залежності або середовище виконання.

Крім того, використання Docker дало змогу автоматизувати взаємодію між сервісами – що особливо зручно у випадках, коли одночасно мають бути активні декілька контейнерів з відкритими портами [17]. Також варто зазначити, що завдяки Docker було досягнуто повної повторюваності середовища, що є критично важливим у контексті тестування та подальшого масштабування системи. Кожен розробник або користувач може розгорнути повноцінну копію проєкту за допомогою однієї команди, без потреби налаштовувати середовище вручну, що суттєво зменшує ризик людських

помилок. Це особливо актуально для систем, які включають як класичні вебкомпоненти, так і машинне навчання, де сумісність бібліотек і драйверів часто стає джерелом проблем [17]. На рисунку 3.3 зображено інтерфейс Docker.

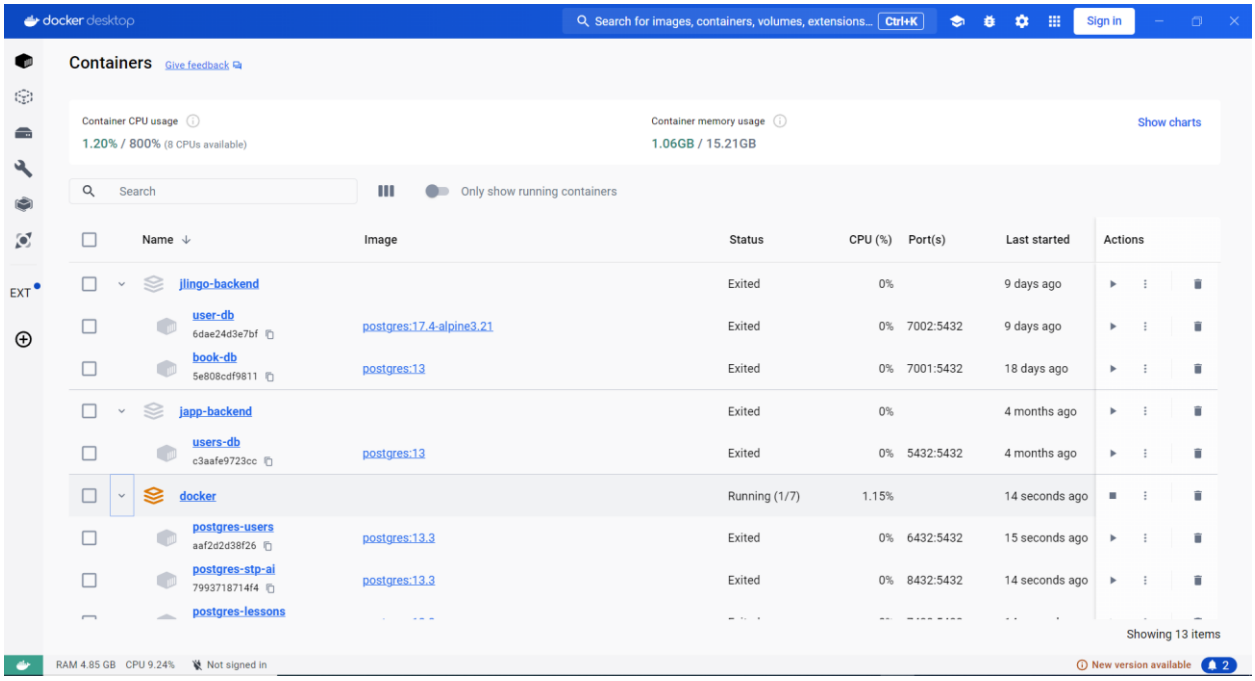

Рисунок 3.3 – Інтерфйс середовища Docker

3.2 Обґрунтування реалізації архітектури застосунку

3.2.1 Обґрунтування реалізації інтерфейсу користувача

Для реалізації інтерфейсу користувача, було обрано мову програмування JavaScript, в основу, якої лягла бібліотека React. Бібліотека React забезпечує компонентно-орієнтований підхід до побудови інтерфейсів, що дозволяє розділяти функціональність на ізольовані, багаторазово використовувані елементи. Завдяки віртуальній об’єктній моделі документа (VDOM, Virtual Document Object Model), React оптимізує оновлення інтерфейсу, забезпечуючи високу продуктивність навіть при динамічній зміні даних. Крім того, екосистема React включає численні додаткові інструменти, зокрема бібліотеку React Router для маршрутизації та бібліотеки управління

станом, які полегшують розробку масштабованих застосунків з однією сторінкою (SPA, Single Page Application).

Бібліотека React також має високу гнучкість щодо інтеграції з API сторонніх сервісів, що дозволяє безперешкодно отримувати дані з серверу, виконувати динамічну обробку введення користувача та синхронізувати стан інтерфейсу з результатами обробки та перекладу. Завдяки підтримці розширення мови TypeScript, React може використовуватися як у динамічно, так і у статично типізованому середовищі, що додатково підвищує надійність коду. Також React підтримує бібліотеку MaterialUI, яка стала основою у побудуванні каркасу для сторінок застосунку. Завдяки механізму повторного використання компонентів, розробка стає більш структурованою, що особливо важливо при масштабуванні застосунку або додаванні нових функцій.

У межах реалізованого застосунку бібліотека React стала основою для побудови інтуїтивно зрозумілого, швидкого та адаптивного інтерфейсу, що дозволив користувачам взаємодіяти з моделлю перекладу в режимі реального часу.

3.2.2 Обґрунтування реалізації серверної частини

Для реалізації серверної частини було обрано мову програмування Java, яка зарекомендувала себе як одна з найнадійніших, масштабованих та безпечних мов у галузі розробки корпоративного програмного забезпечення. Основною технологічною основою стала платформа Spring Boot, яка забезпечує зручний та стандартизований підхід до побудови REST API. Вибір Spring Boot обумовлений його гнучкістю, потужним інструментарієм та широкою підтримкою модульної архітектури, що дозволило організувати чіткий розподіл відповідальності у коді.

Однією з ключових переваг Spring Boot є можливість швидкого створення самодостатніх вебсервісів завдяки механізму автоматичної конфігурації та вбудованому серверу, а саме бібліотеки Tomcat, що значно зменшує час розгортання проєкту. Крім того, Spring Boot має розвинену екосистему, зокрема Spring Security для реалізації авторизації та автентифікації, Spring Data для зручної роботи з базами даних, та Spring Web, який забезпечує ефективну обробку запитів на протоколі передачі гіпертексту (HTTP, Hypertext Transfer Protocol). У межах кваліфікаційної роботи ці компоненти дозволили реалізувати стабільну, безпечну та легко масштабовану серверну частину застосунку.

Використання Java та Spring Boot також забезпечує високу продуктивність за рахунок обробки запитів, яка підтримує багато потоків, і суворої типізації, що мінімізує ризики логічних помилок ще на етапі компіляції. Завдяки цьому сервер здатен ефективно обробляти запити користувача, координувати взаємодію з моделлю машинного перекладу, здійснювати авторизацію доступу, а також взаємодіяти з базою даних для збереження текстів для обробки та перекладу, моделей, конфігурації моделей та профілів користувачів.

Також важливим фактором є активна підтримка спільноти мови Java, яка забезпечує доступ до великої кількості готових рішень, прикладів і розширень. Це дозволяє не лише пришвидшити розробку, але й підвищити надійність та чіткість коду в довгостроковій перспективі.

3.2.3 Обґрунтування реалізації інтегрування нейронних моделей

Для реалізації модуля машинного перекладу, який взаємодіє з користувачем у режимі реального часу, було прийнято рішення використати мову програмування Python разом із бібліотекою HuggingFace Transformers та бібліотекою Flask. Python є де-факто стандартом у галузі штучного інтелекту

завдяки своїй простоті, читабельності, а також широкому набору бібліотек для обробки природної мови, глибокого навчання та роботи з текстовими даними. Його синтаксис дозволяє зосередитися на обробці даних, а не на технічних деталях, що особливо важливо при інтеграції та тестуванні моделей нейронного машинного перекладу (НМП).

Бібліотека HuggingFace Transformers надала інструментарій для завантаження, тонкого налаштування та використання багатомовних моделей перекладу, зокрема mBART, M2M100 та Small100, що були адаптовані під задачі перекладу тексту з азійських мов на українську. Обрана бібліотека підтримує попередньо навчені моделі, що дозволяє запускати їх локально без потреби у великій кількості ресурсів. Це дозволило досягти високої точності перекладу без значних витрат на обчислювальні потужності, використовуючи лише локальний графічний процесор (GPU, Graphical Process Unit) або навіть центральний процесор (CPU, Central Process Unit).

Особливу увагу було приділено налаштуванню гіперпараметрів моделей. Зокрема, експериментально було визначено оптимальні значення для максимальної довжини та довжини штрафу, які суттєво впливають на якість генерації токенів перекладу. Для забезпечення балансу між продуктивністю та глибиною обробки, також проводилась оптимізація таких параметрів, як рання зупинка та штраф за повторення. Ці налаштування дозволили уникнути як надмірного скорочення вихідного тексту, так і небажаних повторів, що часто зустрічаються при генерації послідовностей у великих мовних моделях.

Для обгортки моделі в REST API було використано бібліотеку Flask, який завдяки своїй мінімалістичній структурі дозволив швидко створити серверну обробку запитів на переклад. Кожен запит до API передбачав надсилання тексту та параметрів перекладу, після чого модель повертала оброблений результат у вигляді JSON-об'єкта. Завдяки простоті інтеграції Flask із орієнтованим сервером на Java, був реалізований чіткий поділ

відповідальності: обробка HTTP-запитів та авторизації залишалася за Java-сервером, а обчислювальні задачі – на Python-сервісі з нейронною моделлю.

На практиці така конфігурація виявилась ефективною як з точки зору гнучкості, так і масштабованості. При необхідності Python-сервіс може бути винесений в окремий контейнер, що дозволяє легко оновлювати модель, змінювати параметри або замінювати модель на більш сучасну без втручання в основну архітектуру застосунку.

3.3 Обґрунтування вибору моделей та їх оцінка

У процесі розробки інтелектуального модуля машинного перекладу особливу увагу було зосереджено на виборі відповідних моделей, здатних забезпечити належну якість обробки тексту в умовах багатомовного середовища, зокрема при роботі з низько-ресурсними мовними парами. Враховуючи поставлені цілі – побудову системи, що працює у режимі реального часу з допустимою точністю, – вибір моделей здійснювався не лише на основі технічних характеристик, а й з урахуванням попередніх досліджень, доступності паралельних корпусів, підтримки спільнотою та можливості адаптації.

Паралельно з теоретичним обґрунтуванням було проведено емпіричне тестування кількох моделей, зокрема mBART-large-50, M2M100-418M та Small100, кожна з яких має свої сильні сторони в контексті багатомовного перекладу. З метою отримання об’єктивної оцінки продуктивності, для кожної моделі було використано набір метрик, що відображають якість перекладу з різних точок зору: від точності до семантичної повноти.

Серед розглянутих моделей першою було проаналізовано mBART-large-50 – багатомовну модель перекладу, що поєднує у собі двонапрямлене кодування та автогенеративне декодування. Її перевагою є здатність працювати з 50 мовами без потреби в англійській як проміжній, а також

ефективне багатомовне попереднє навчання на великих корпусах. У тестових перекладах з японської, корейської та китайської мов на українську mBART демонструвала стабільно результати за метриками BLEU, ChrF, METEOR, TER, особливо в парах із високою морфологічною складністю. Проте варто зазначити, що час генерації в цій моделі дещо вищий у порівнянні з легшими архітектурами, що може впливати на загальну продуктивність при використанні в обмежених обчислювальних умовах [18].

M2M100-418M, у свою чергу, була розроблена з урахуванням перекладу між будь-якими парами зі 100 мов, без англійської як посередника. Ця модель продемонструвала конкурентні результати на тих же мовних парах, однак у низці випадків – з кращою узгодженістю між фразами. Особливо помітно це було в перекладі довших речень з японської, де M2M100 краще зберігала порядок слів і граматичну логіку. Модель також має менше параметрів, ніж mBART-large-50, що робить її дещо швидшою в роботі. Водночас її результати в умовах сильного домену – наприклад, спеціалізованих термінів – були менш точними без додаткового навчання [18].

Останньою в аналізі стала small100 – компактна модель, що розроблена переважно для експериментів та освітніх завдань. Незважаючи на значно меншу кількість параметрів, small100 виявилася ефективною у простих фразах і невеликих текстових фрагментах. Проте при перекладі довших або стилістично насичених речень вона виявляла схильність до узагальнення, іноді опускаючи другорядні, але змістовні частини речення. Її основна перевага – легкість, висока швидкість обробки та низькі вимоги до ресурсів – дозволяє використовувати її в умовах обмеженого обладнання [18].

На рисунках 3.4 – 3.7 показано графік метрик BLEU, ChrF, METEOR, TER для кожної моделі відповідно. Також на рисунках 3.8 – 3.10 показано різницю між результуючими перекладами та перекладами, які були взяті за основу в наборі дані для оцінювання та валідації моделей.

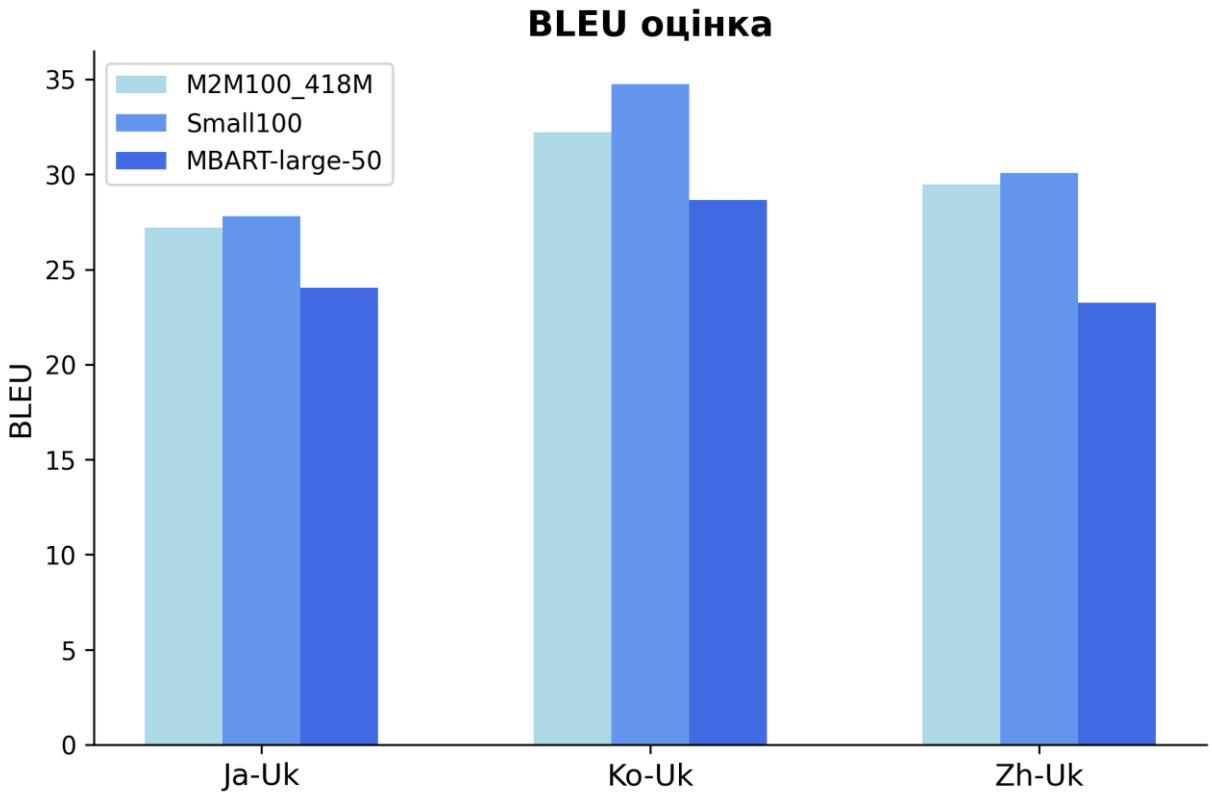

Рисунок 3.4 – Графік BLEU оцінки для моделей M2M100_418M, Small100, mBART-large-50

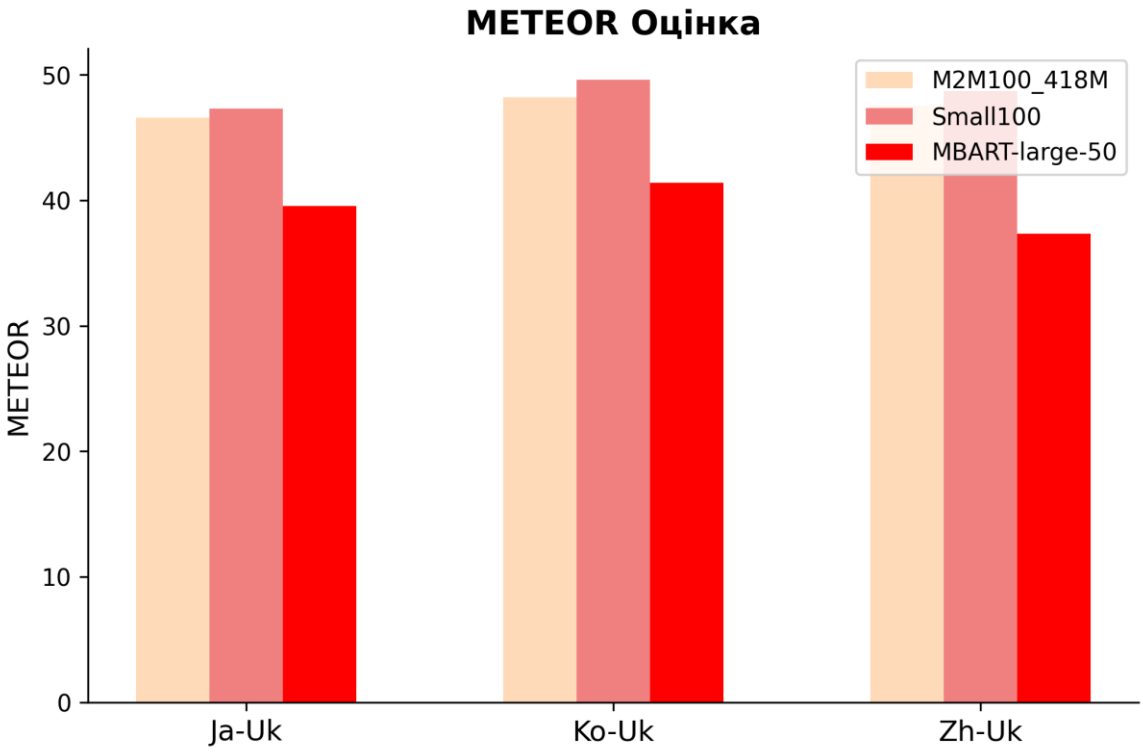

Рисунок 3.5 – Графік METEOR оцінки для моделей M2M100_418M, Small100, mBART-large-50

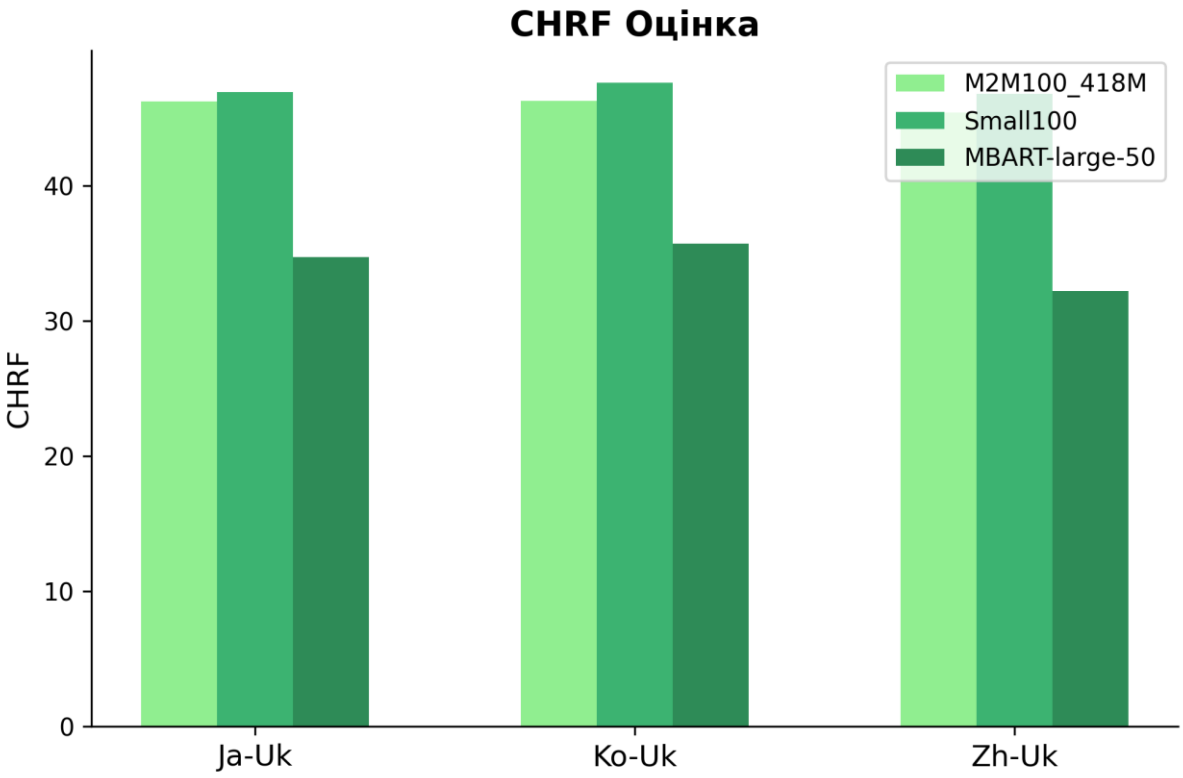

Рисунок 3.6 – Графік ChrF оцінки для моделей M2M100_418M, Small100, mBART-large-50

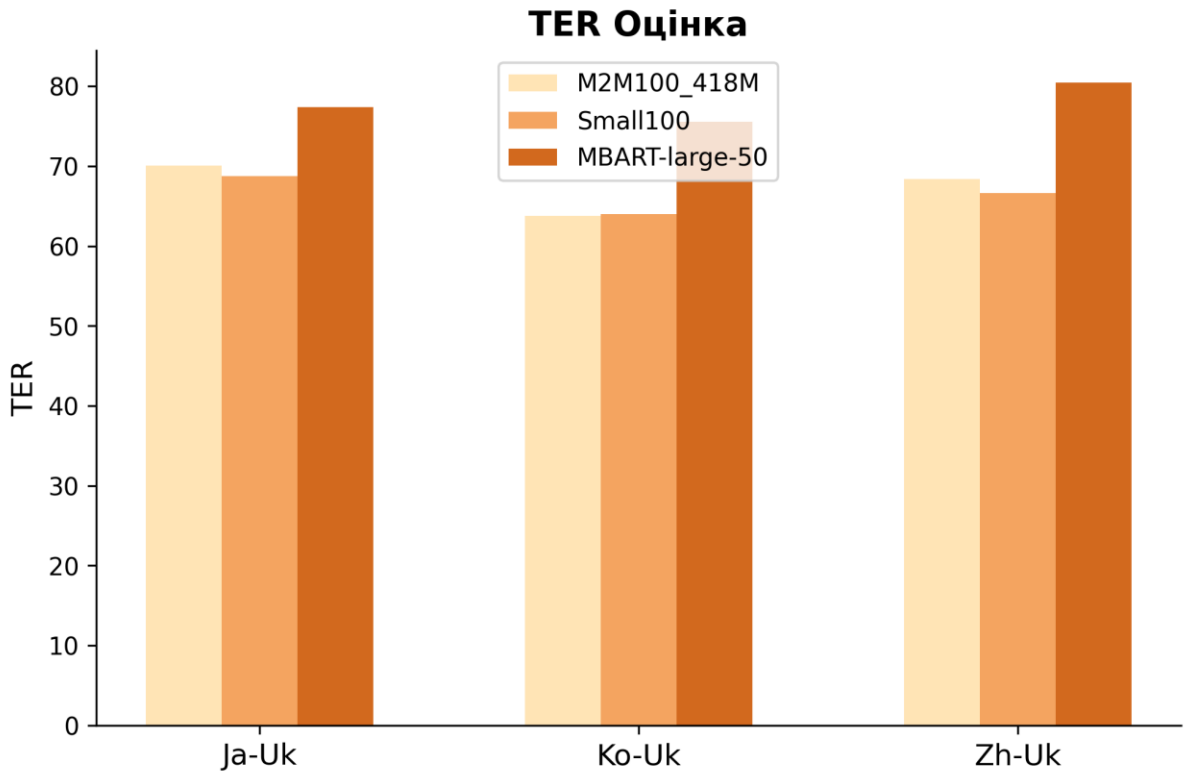

Рисунок 3.7 – Графік TER оцінки для моделей M2M100_418M, Small100, mBART-large-50

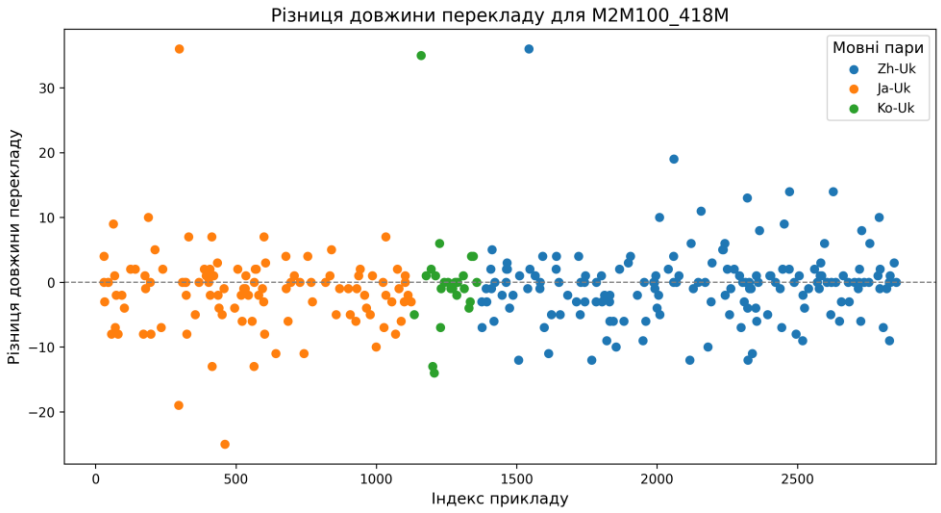

Рисунок 3.8 – Графік різниці довжини перекладу для моделі M2M100_418M

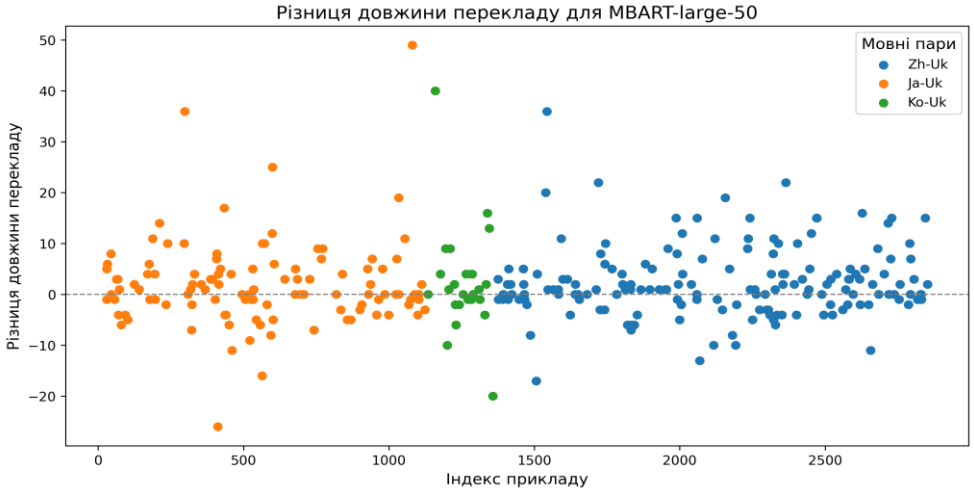

Рисунок 3.9 – Графік різниці довжини перекладу для моделі mBART-50-large

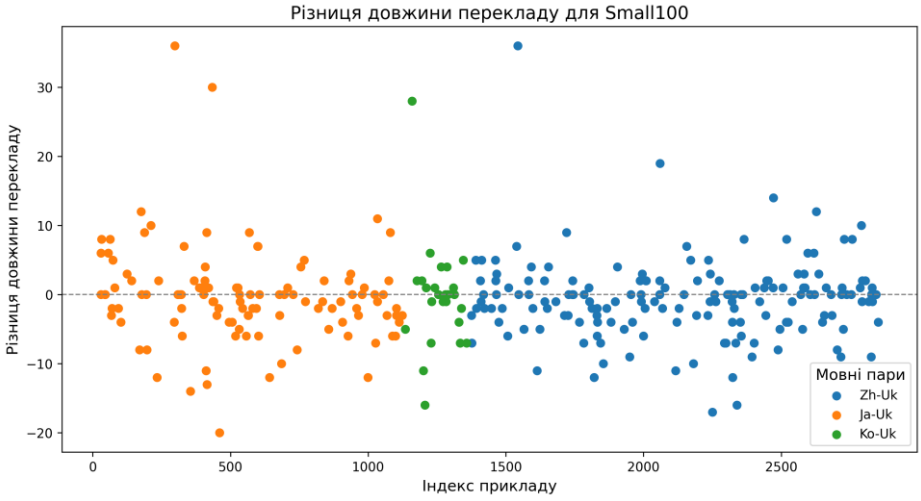

Рисунок 3.10 – Графік різниці довжини перекладу для моделі Small-100

Графіки було перенесено до таблиць 3.1 – 3.3 для більш точного розуміння ситуації по кожній з метрик.

Таблиця 3.1 – Метрики для японсько-українського корпусу

Модель	**BLEU**	**ChrF**	**TER**	**METEOR**
M2M100	27,20	46,19	70,06	46,58
Small100	27,79	46,91	68,73	47,31
mBART-large-50	24,03	34,70	77,37	39,54

Таблиця 3.2 – Метрики для корейсько-українського корпусу

Модель	**BLEU**	**ChrF**	**TER**	**METEOR**
M2M100	32,22	46,27	63,81	48,19
Small100	34,75	47,61	63,99	49,61
mBART-large-50	28,64	35,71	75,57	41,38

Таблиця 3.3 – Метрики для китайсько-українського корпусу

Модель	**BLEU**	**ChrF**	**TER**	**METEOR**
M2M100	29,47	45,37	68,41	47,53
Small100	30,08	46,77	66,62	48,70
mBART-large-50	23,24	32,18	80,47	37,33

Аналіз отриманих результатів демонструє, що для всіх трьох мовних пар найбільш ефективною виявилася модель Small100, яка переважала конкурентів за більшістю метрик. Помітно, що модель mBART-large-50, незважаючи на більший розмір та теоретично більшу кількість параметрів, показує нижчі результати.

3.4 Програмна реалізація

3.4.1 Реалізація шару баз даних

Для реалізації шару баз даних було використано технологію docker-compose для того, щоб зібрати да розгорнути контейнери на віртуальній системі. Кожен контейнер має свої параметри, а також набір даних, які зберігаються, та віртуальний порт та локальний порт. Для кожного контейнера використовувався образ postgres:17.4-alphine3.21 так як цей образ найновіший із доступніших та як й усі образи типу alphine малий у розмірах та зручний у використанні для серверів із малими процесорними ресурсами. На рисунках 3.11 та 3.12 можна побачити програмну реалізацію зборки та розгортування всіх баз даних, для кожного сервісу, та їх налаштування.

```yaml
version: '3.8'

services:
  book-database:
    container_name: book-db
    image: postgres:17.4-alpine3.21
    environment:
      POSTGRES_DB: books
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: password
    ports:
      - "7001:5432"
    volumes:
      - book_data:/var/lib/postgresql/data

  user-database:
    container_name: user-db
    image: postgres:17.4-alpine3.21
    environment:
      POSTGRES_DB: users
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: password
    ports:
      - "7002:5432"
    volumes:
      - user_data:/var/lib/postgresql/data
```

Рисунок 3.11 – Код розгортування баз даних для серверів User та Book

```
  model-database:
    container_name: model-db
    image: postgres:17.4-alpine3.21
    environment:
      POSTGRES_DB: models
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: password
    ports:
      - "7003:5432"
    volumes:
      - model_data:/var/lib/postgresql/data

volumes:
  book_data:
  user_data:
  model_data:
```

Рисунок 3.12 – Код розгорутвання бази даних для сервера Model

Так на рисунку 3.13 було реалізовано код для автоматичного запуску завантаження всіх образів та створення на машині docker у відповідній папці усіх контейнерів зазначених у docker-compose.

```
docker-compose -p jlingo-backend down --remove-orphans
docker-compose -p jlingo-backend build --no-cache
docker-compose -p jlingo-backend up -d --force-recreate

docker volume prune -f
docker image prune -af
docker builder prune -af

if [ ! -f docker-compose.yml ]; then
  echo "docker-compose.yml not found!"
  exit 1
fi

mkdir -p logs data/db uploads
touch .env

docker-compose -p jlingo-backend ps
```

Рисунок 3.13 – Код автоматичного запуску розгортування контейнерів

На рисунку 3.14 можна побачити результат запуску усіх контейнерів.

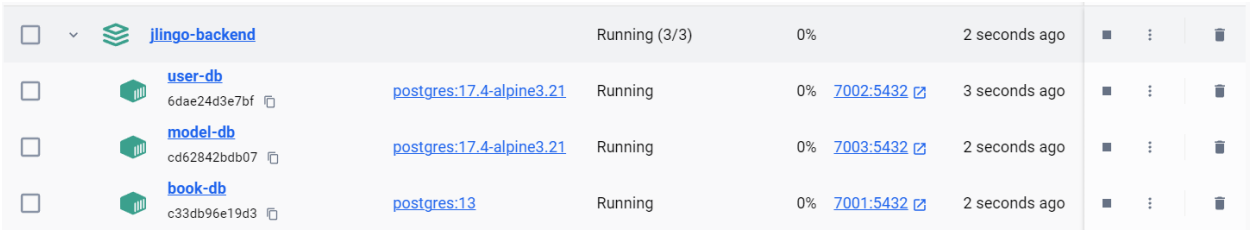

Рисунок 3.14 – Результат запуску усіх контейнерів

На рисунку 3.15 можна побачити результат тестування всіх контейнерів.

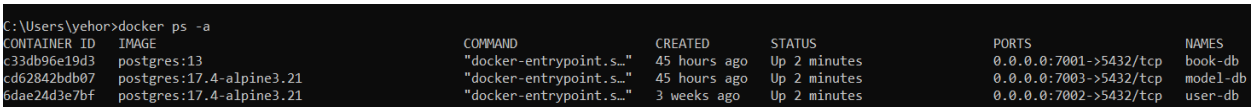

Рисунок 3.15 – Результат тестування, що всі контейнери запущені

3.4.2 Реалізація шару мікросервісів

Під час реалізації архітектурного шару мікросервісів було прийнято рішення застосувати підхід розділення відповідальності за функціональні домени в межах окремих модулів. Це дозволило побудувати систему, в якій кожен сервіс відповідає за чітко визначену частину бізнес-логіки, забезпечуючи при цьому гнучкість у розгортанні, тестуванні та масштабуванні.

Кожен мікросервіс було винесено в окремий модуль з використання інструменту для зборки maven із власною структурою, залежностями, налаштуваннями безпеки, об’єктами передачі даних (DTO, Data Transfer Object) та контролерами. Зокрема, сервіс user-ms обробляє усі запити, пов’язані з автентифікацією, керуванням користувачами, а також роботою з токенами оновлення. Сервіс book-ms забезпечує створення, збереження,

редагування та перегляд книжок і сторінок, а також має можливість інтегруватися з сервісами автоматичного перекладу. Модуль gateway-ms виконує роль єдиного вхідного шлюзу до всієї системи, делегуючи запити на відповідні сервіси на основі правил маршрутизації. На рисунках 3.16 та 3.17 можна побачити реалізацію маршрутизатора, використовуючи бібліотеку Spring Cloud.

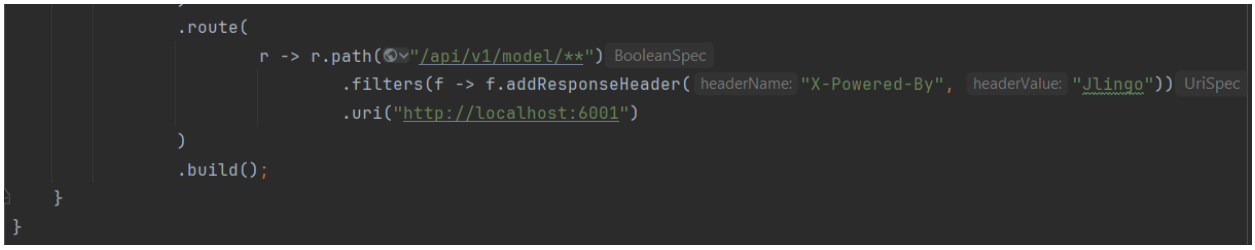

Рисунок 3.16 – Реалізація збірки конфігурації для маршрутизатора

Рисунок 3.17 – Реалізація маршрутизатора на Spring Cloud

Конфігурації з’єднання, профілі середовищ та політики спільного використання ресурсів між джерелами (CORS, Cross-Origin Resource Sharing) задаються в окремих файлах, що забезпечує прозорість налаштувань і

полегшує адаптацію під різні умови розгортання. На рисунку 3.18 можна побачити реалізація політики CORS для усієї серверної частини.

Рисунок 3.18 – Реалізація політики CORS

Особливу увагу було приділено питанням безпеки. Всі запити, що потребують авторизації, проходять через фільтр JSON вебтокену (JWT, JSON Web Token), реалізований у сервісі user-ms. Завдяки використанню Spring Security та власних компонентів, сервіс автентифікації може не лише перевіряти токени, а й розширювати контекст користувача для подальшого використання в бізнес-процесах.

До того ж, підтримується механізм оновлення токенів, що зменшує кількість повторних логінів та покращує користувацький досвід. Варто також зазначити, що в ході реалізації було запроваджено єдині підходи до обробки помилок, валідації вхідних даних та відповіді API, що дало змогу знизити технічну заборгованість і прискорити розробку UI. На рисунку 3.19 можна побачити реалізація механізму генерації JWT-токену.

Крім того, під час розробки значну увагу приділили питанням продуктивності та стабільності сервісу авторизації. Зокрема, було реалізовано механізм зберігання токенів на основі Redis, завдяки чому значно скоротився час на повторні запити до бази даних. Це рішення дозволило не лише прискорити обробку запитів, але й знизити навантаження на сховище,

забезпечуючи стабільність роботи системи при великій кількості одночасних користувачів.

```
YehorBodenchuk
@PostConstruct
public void init() {
    this.key = Keys.hmacShaKeyFor(secret.getBytes(StandardCharsets.UTF_8));
}

3 usages  YehorBodenchuk
public String generateAccessToken(User user) {
    final long expiration = 15 * 60 * 10000;

    Map<String, Object> claims = new HashMap<>();
    claims.put("firstName", user.getFirstName());
    claims.put("lastName", user.getLastName());
    claims.put("age", user.getAge());
    claims.put("email", user.getEmail());
    claims.put("photo", user.getPhoto());

    return Jwts.builder()
            .subject(user.getEmail())
            .claims(claims)
            .issuedAt(new Date())
            .expiration(new Date(System.currentTimeMillis() + expiration))
            .signWith(key, algorithm)
            .compact();
}
```

Рисунок 3.19 – Реалізація механізму генерації JWT-токену

На рисунку 3.20 можна побачити діаграму класів всієї серверної частини.

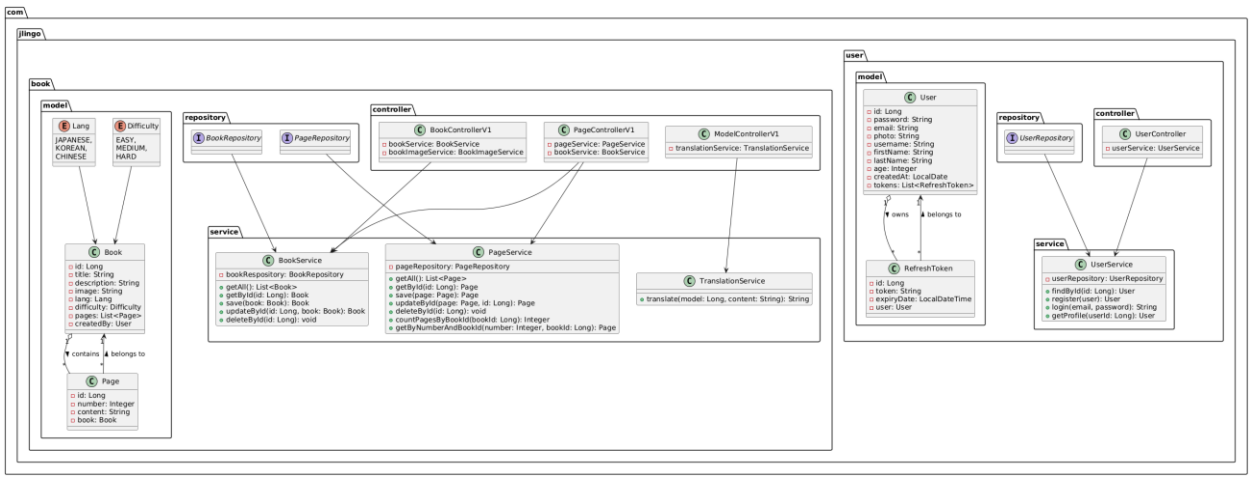

Рисунок 3.20 – Діаграма класів всієї серверної частини

3.4.3 Реалізація шару обробки та перекладу тексту з використанням нейронних моделей

З метою забезпечення якісного узгодженого машинного перекладу в межах архітектури системи було реалізовано окремий шар, відповідальний за попередню обробку текстових запитів і їхню трансформацію за допомогою нейронних моделей. Цей компонент не лише забезпечує функціональність автоматичного перекладу, а й виконує роль проміжної ланки між UI та мовною моделлю, побудованою на основі архітектури Transformer.

Архітектурно реалізація була здійснена за допомогою бібліотеки FastAPI, який відзначається своєю легкістю, швидкодією та підтримкою сучасних асинхронних операцій. Сам API був розгорнутий у середовищі, до якого можна отримати зовнішній доступ через тунелювання з використанням сервісу ngrok. Такий підхід дозволив уникнути складної серверної інфраструктури під час етапу тестування та водночас забезпечити зовнішні HTTP-запити до локального сервера.

На рисунку 3.21 можна побачити реалізацію основних маршрутів для обробки і перекладу тексту.

```
28 class TranslationRequest(BaseModel):
29     text: str
30     src_lang: str
31     tgt_lang: str
32 
33 @app.post("/translate/mbart")
34 def translate(req: TranslationRequest):
35     inputs = tokenizer(req.text, return_tensors="pt", padding=True, truncation=True)
36     generated_tokens = model.generate(
37         **inputs,
38         forced_bos_token_id=tokenizer.lang_code_to_id[req.tgt_lang]
39     )
40     translated = tokenizer.batch_decode(generated_tokens, skip_special_tokens=True)
41     return {"translation": translated[0]}
42 
43 @app.post("/translate/m2m")
44 def translate(req: TranslationRequest):
45     m2m_tok.src_lang = req.src_lang
46     encoded_text = m2m_tok(req.text, return_tensors="pt", padding=True, truncation=True, max_length=128)
47     generated_tokens = m2m_model.generate(**encoded_text, forced_bos_token_id=m2m_tok.get_lang_id(req.tgt_lang))
48     translated_text = m2m_tok.batch_decode(generated_tokens, skip_special_tokens=True)
49     translated = m2m_tok.batch_decode(generated_tokens, skip_special_tokens=True)
50     return {"translation": translated[0]}
51 
```

Рисунок 3.21 – Реалізація маршрутів для обробки та перекладу тексту

На рисунку 3.22 можна побачити реалізацію завантаження моделей та токенізаторів з платформи Hugging Face.

```python
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

def load_model(model_name: str):
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
    return tokenizer, model
```

Рисунок 3.22 – Реалізація завантаження моделей та токенізаторів з платформи Hugging Face

3.5 Ілюстрація роботи вебзастосунку

Заходячи до вебзастосунку користувача зустрічає сторінка реєстрації та авторизації. На рисунках 3.23 та 3.24 показано сторінки авторизації та реєстрації.

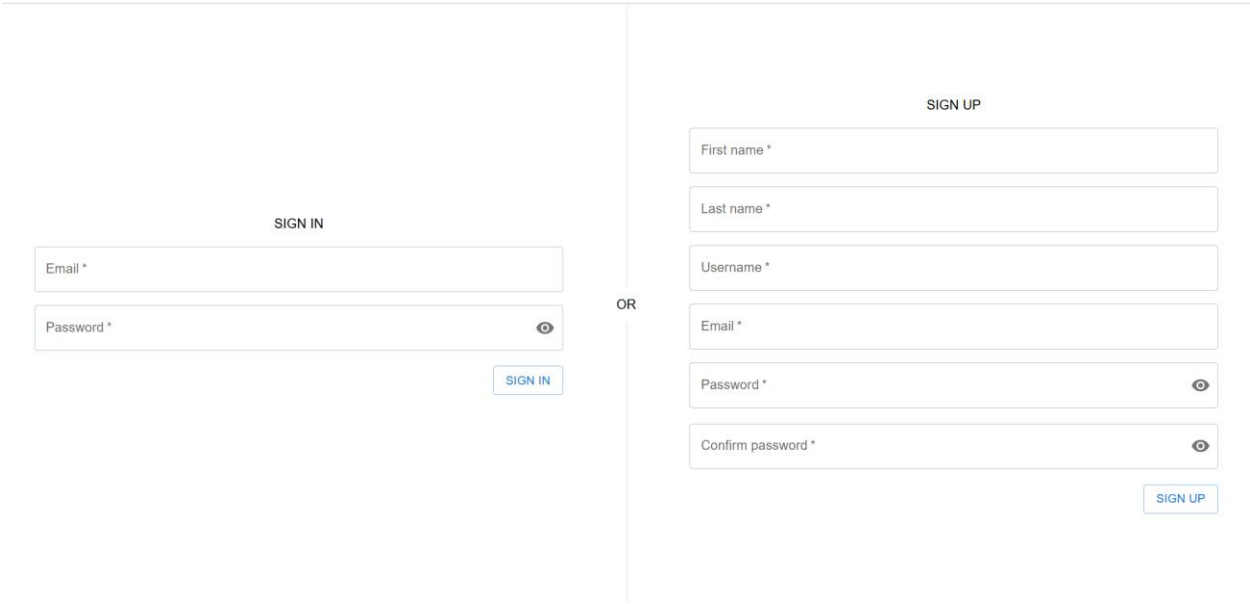

Рисунок 3.23 – Сторінка реєстрації та авторизації

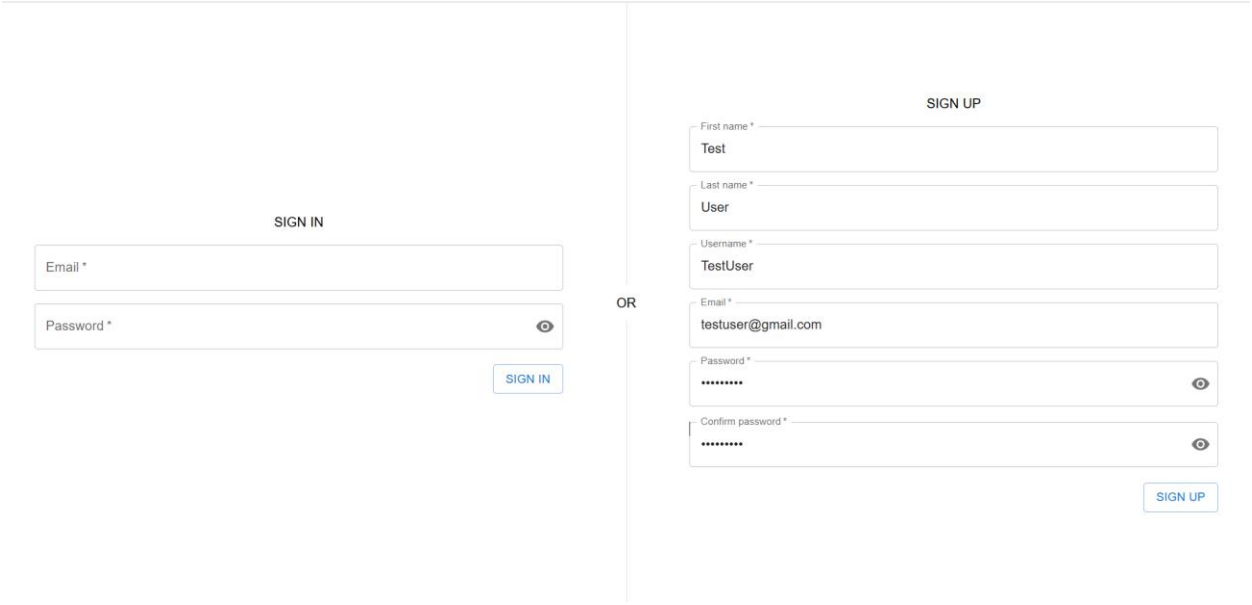

Рисунок 3.24 – Створення нового користувача в застосунку

Створивши користувача, або успішно авторизувавшись, користувач потрапляє на головну сторінку, де він може обрати будь, який текстовий документ (книжка, журнал, стаття) або сам створити його натиснувши на кнопку «Add Book» (рис. 3.25).

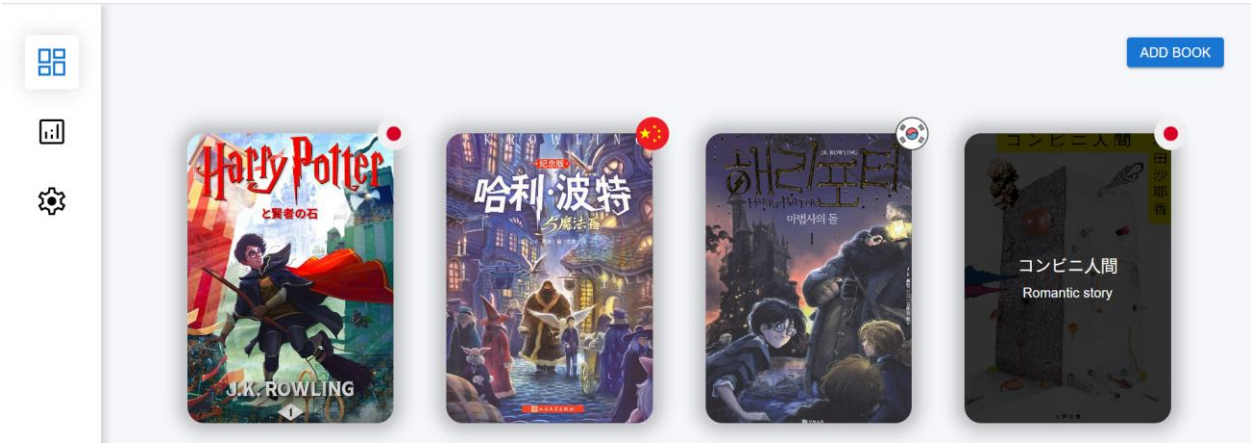

Рисунок 3.25 – Головна сторінка

Обравши будь-який текстовий документ, користувач потрапляє на сторінку де він може переключатись між сторінками цього документу та читати текст або додати сторінку натиснувши кнопку «Add Page» (рис. 3.26).

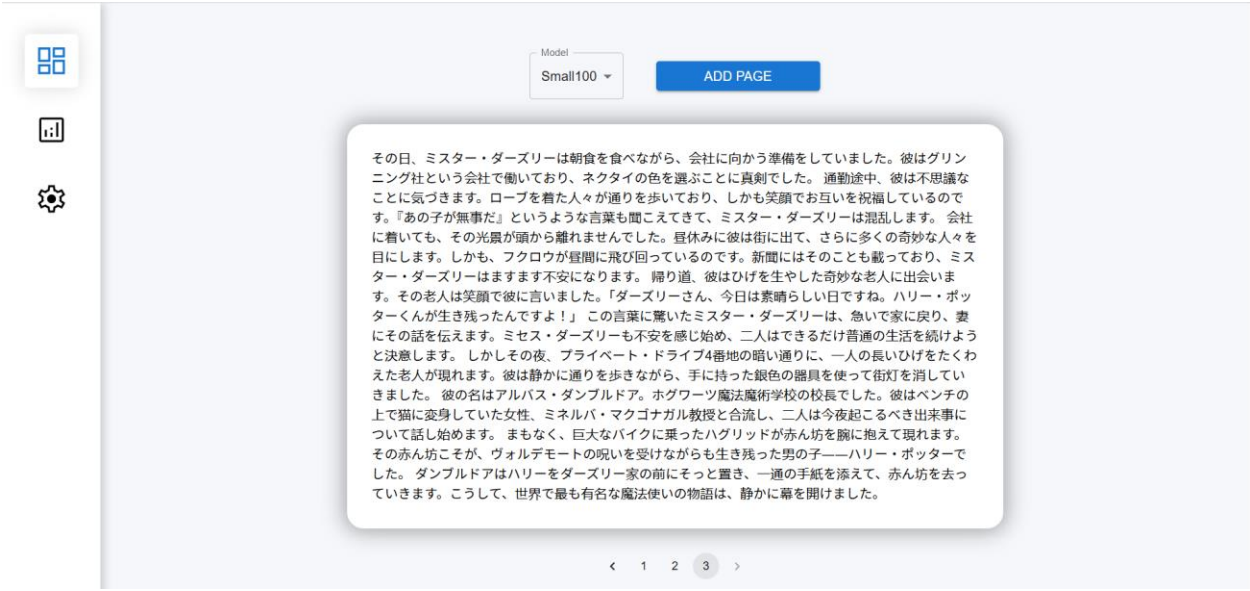

Рисунок 3.26 – Сторінка для перегляду текстового документу

У перемикачі «Model» користувач може обрати вже розгорнуті нейронні моделі та починати переводити текст на українську мову просто виділивши її. У перемикачі доступні моделі Small100, M2M100_418, mBART, які були оцінені (рис. 3.4 – 3.10), навчені на власному наборі даних, та у яких були налаштовані гіперпараметри. На рисунках 3.27 – 3.29 можна побачити демонстрацію перекладу моделі Small100 на всіх доступних мовах (японської, корейської та китайської).

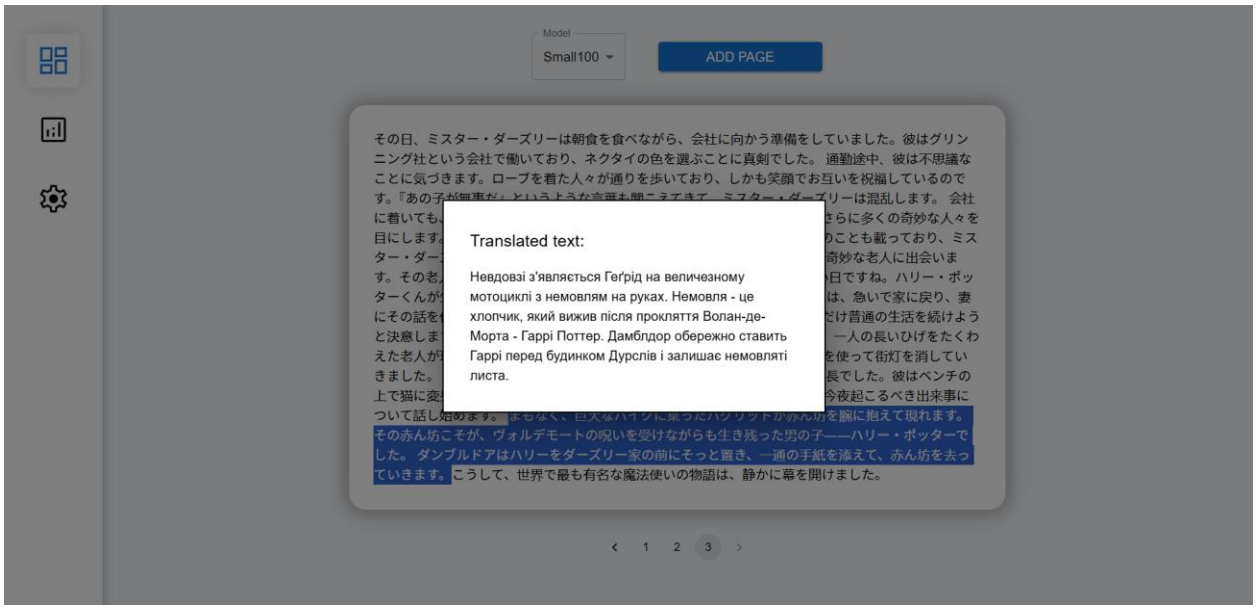

Рисунок 3.27 – Приклад перекладу моделі Small100 з японської мови

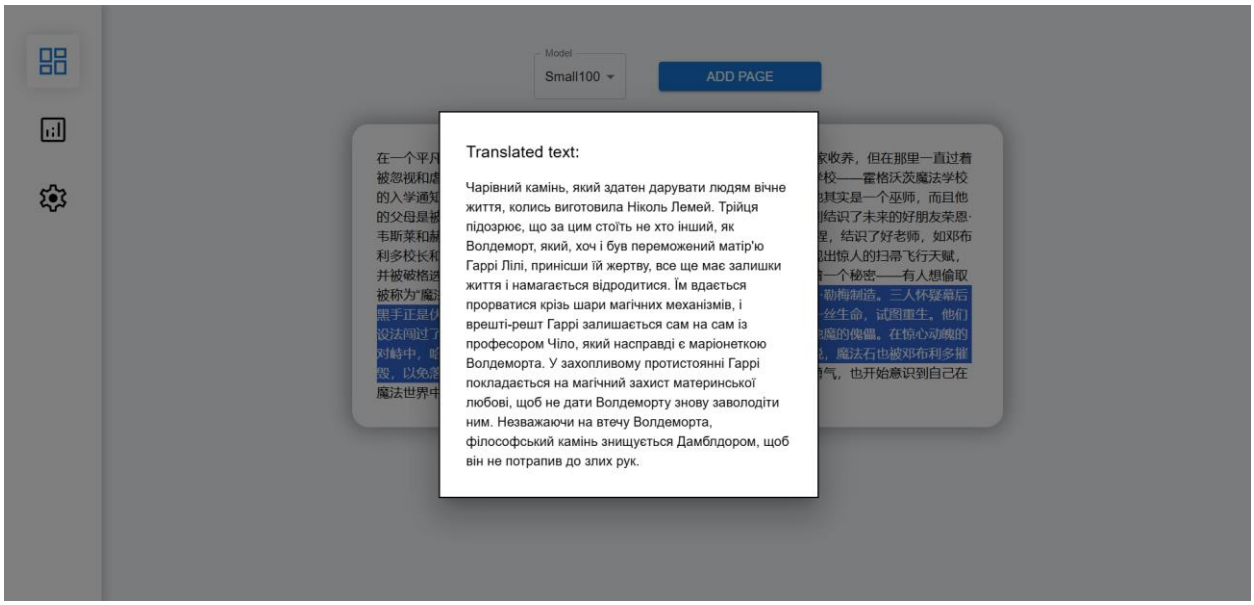

Рисунок 3.28 – Приклад перекладу моделі Small100 з китайської мови

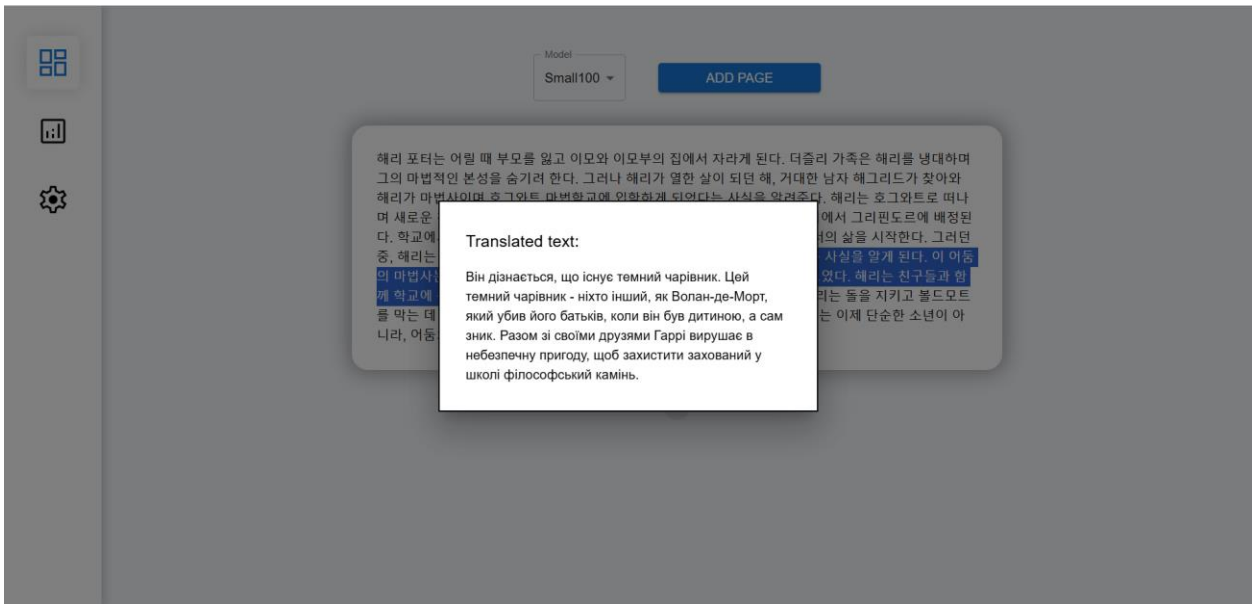

Рисунок 3.29 – Приклад перекладу моделі Small1100 з корейської мови

Моделі здатні перекладати не тільки речення зі змістом, алей й окремі солова. Така гнучкість особливо цінна для навчальних або референтних систем, де користувачі мають змогу швидко з’ясувати значення конкретного фрагмента тексту. На рисунках 3.30 – 3.32 можна побачити, як всі моделі перекладають виділене слово.

Рисунок 3.30 – Переклад моделі Small100

Рисунок 3.31 – Переклад моделі M2M100_418

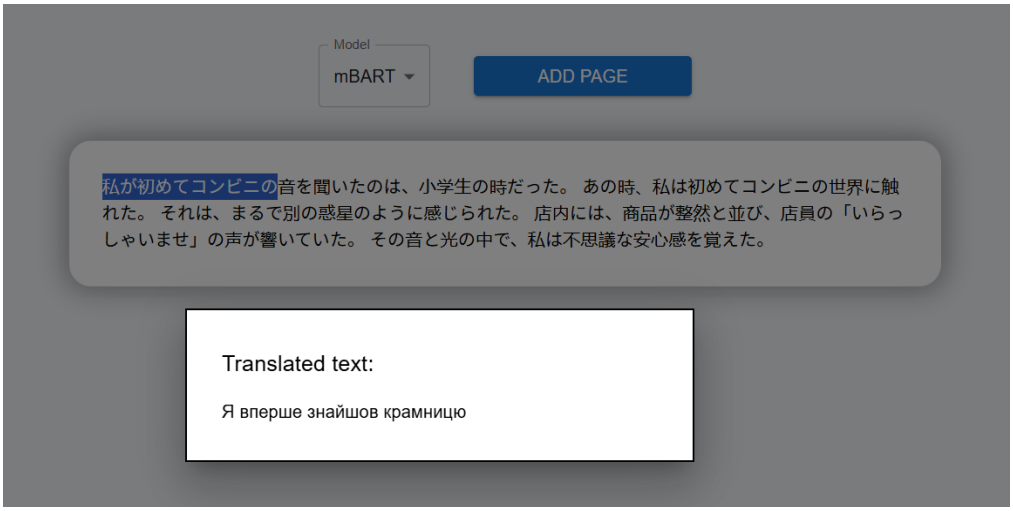

Рисунок 3.32 – Переклад моделі mBART

На рисунку 3.33 зображено сторінку форми створення текстового документу.

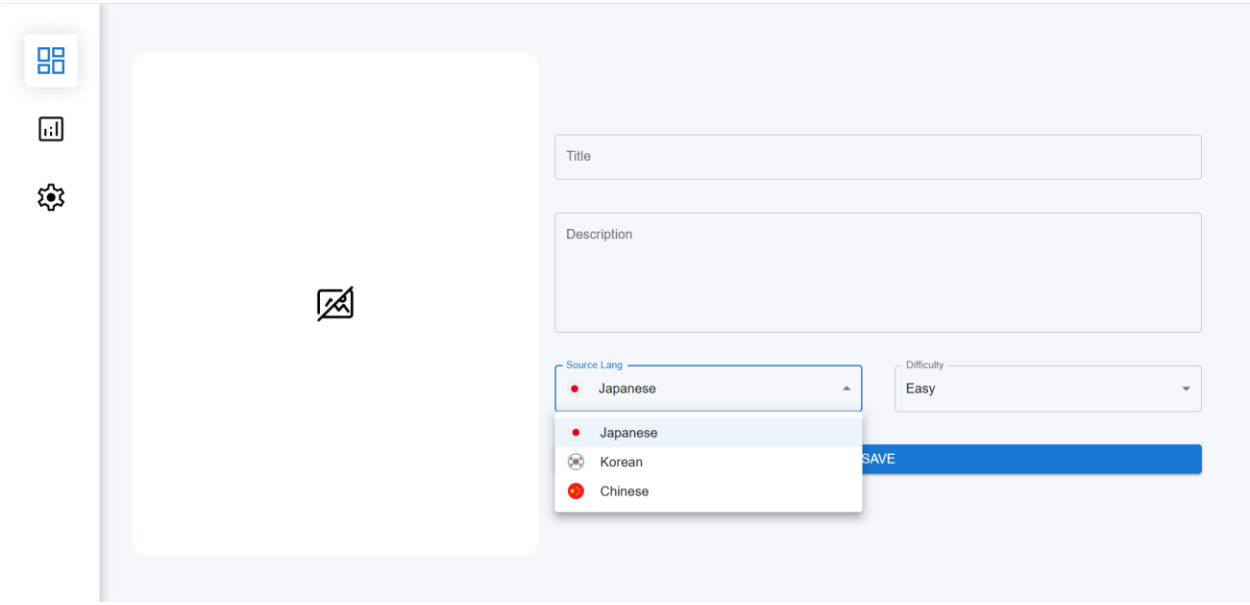

Рисунок 3.33 – Сторінка форма створення текстового документу

На рисунку 3.34 зображено форму створення сторінок для обраного текстового документу.

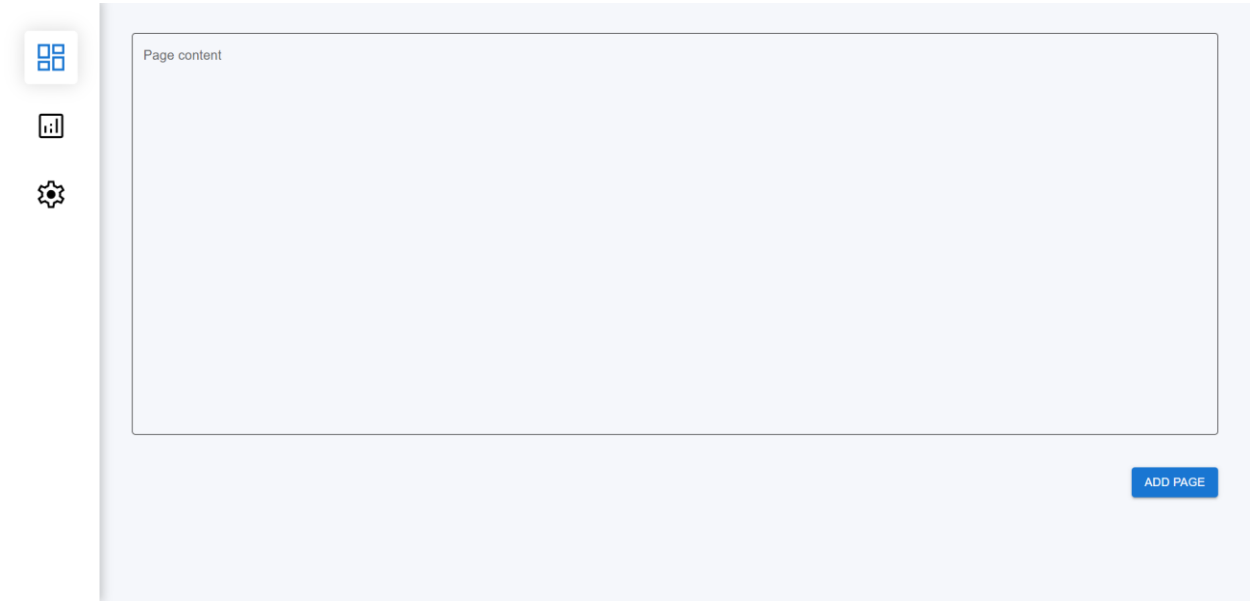

Рисунок 3.34 – Форма створення сторінко для обраного текстового документу

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та реалізовано уніфікований вебзастосунок, який оброблю та перекладає тексти з японської, корейської та китайської мов на українську мову, з використанням відкритих нейронних мереж з платформи Hugging Face, які були навчені на власному наборі даних та розгорнуті на віртуальному сервері.

Для створення застосунку були вирішенні наступні завдання:

- проведено аналіз існуючих сучасних методів обробки та перекладу;
- розроблено fine-tuning алгоритм для обраних моделей;
- реалізовано багатосервісну архітектуру для вебзастосунку;
- реалізовано розгортування баз даних;
- реалізовано виконання генерації тексту на графічному процесорі;
- проведено оцінку метрик BLEU, TER, ChrF, METEOR;
- розроблено компактний UI для застосунку.

Під час роботи було використано сучасні технології для розгортування баз даних, розробки мікросервісної архітектури, навчання та оцінка нейронних моделей, розробка UI.

Результатом роботи є уніфікована система, яка надає гнучкий функціонал для зручного опрацювання текстів.

Систему можна розширити актуальними тенденціями в області інтелектуальної обробки даних і комп’ютерного зору, зокрема методи кластеризації потоків даних [19–22], алгоритми аналізу зображень [23–28], а також сучасні підходи до класифікації візуальних об’єктів [24–30].

Результати роботи апробовано у вигляді 2 тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [18], І Всеукраїнської науково-практичної студентської конференції «ІТ-ПРОСТІР СЬОГОДЕННЯ: ТЕНДЕНЦІЇ, ІННОВАЦІЇ ТА ПЕРСПЕКТИВИ РОЗВИТКУ» [7].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., ... & Joulin, A. (2021). Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, *22*(107), pp. 1-48.

2. Martindale, M., Campbell, J., Savenkov, K., & Goel, S. (2024, September). Proceedings of the 16th Conference of the Association for Machine Translation in the Americas (Volume 2: User Track). In *Proceedings of the 16th Conference of the Association for Machine Translation in the Americas (Volume 2: User Track)*.

3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

4. Chronopoulou, A., Stojanovski, D., & Fraser, A. (2022). Language-family adapters for low-resource multilingual neural machine translation. *arXiv preprint arXiv:2209.15236*.

5. Lankford, S., Afli, H., & Way, A. (2024). Transformers for Low-Resource Languages: Is F\'eidir Linn!. *arXiv preprint arXiv:2403.01985*.

6. Lankford, S. (2024). Enhancing Neural Machine Translation of Low-Resource Languages: Corpus Development, Human Evaluation and Explainable AI Architectures. *arXiv preprint arXiv:2403.01580*.

7. Bodenchuk-Pastukhov, Y. V., & Kobylin, I. O. APPLICATION OF MULTI-HEAD ATTENTION MECHANISM IN SOFTWARE TOOLS FOR MACHINE TRANSLATION WITHIN INTELLIGENT DATA PROCESSING. *MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE VN KARAZIN KHARKIV NATIONAL UNIVERSITY*, 99.

8. Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).

9. Popović, M. (2015, September). chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation* (pp. 392-395).

10. Banerjee, S., & Lavie, A. (2005, June). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization* (pp. 65-72).

11. Maltas, S. I. (2024). *Efficient finetuning strategies for multilingual neural machine translation* (Master's thesis, Universitat Politècnica de Catalunya).

12. Keita, M. K., Homan, C., Zampieri, M., Bremang, A., Alfari, H. A., Ibrahim, E. A., & Owusu, D. (2024). Grammatical Error Correction for Low-Resource Languages: The Case of Zarma. *arXiv preprint arXiv:2410.15539*.

13. Mohammadshahi, A., Nikoulina, V., Berard, A., Brun, C., Henderson, J., & Besacier, L. (2022). SMaLL-100: Introducing shallow multilingual machine translation model for low-resource languages. *arXiv preprint arXiv:2210.11621*.

14. Chipman, H. A., George, E. I., McCulloch, R. E., & Shively, T. S. (2022). mBART: multidimensional monotone BART. *Bayesian Analysis*, *17*(2), 515-544.

15. Cao, K., Cheng, W., Hao, Y., Gan, Y., Gao, R., Zhu, J., & Wu, J. (2024). DMSeqNet-mBART: a state-of-the-art adaptive-DropMessage enhanced mBART architecture for superior Chinese short news text summarization. *Expert Systems with Applications*, *257*, 125095.

16. Dabre, R., & Chakrabarty, A. (2021, August). NICT-5's submission to WAT 2021: MBART pre-training and in-domain fine tuning for indic languages. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)* (pp. 198-204).

17. Кобилін, І. О., & Ніколайчук, А. І. (2024). MONITORING AND DIAGNOSING FAULTS IN ONLINE MODE USING TIME SERIES DATA. *Системи обробки інформації*, (3 (178)), 27-32.

18. Bodenchuk-Pastukhov Y.V. (2025) Evaluating language models on low-resource pairs. *Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків 16-19 квітня 2025 р.).* Харків: ХНУРЕ, 2025. Т. 7. С. 17-19.

19. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

20. Bodyanskiy, Y., Vynokurova, O., Szymański, Z., Kobylin, I., & Kobylin, O. (2016, August). Adaptive robust models for identification of nonstationary systems in data stream mining tasks. In *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)* (pp. 263-268). IEEE.

21. Бодянський, Є., Винокурова, О., Кобилін, І., & Мулеса, П. (2017). Адаптивна матрична нейро-фаззі самоорганізовна мережа для кластеризації багатовимірних потоків даних. *Вісник Національного університету Львівська політехніка. Комп'ютерні науки та інформаційні технології*, (864), 314-319.

22. Бодянський, Є. В., Винокурова, О. А., Ізонін, І. В., Кобилін, І. О., & Мулеса, П. П. (2017). Кластеризація багатовимірних часових рядів на основі адаптивної матричної нейро-фаззі самоорганізовної мережі. *Intellectual Systems For Decision Making and Problems of Computational Intelligence*, 247.

23. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. *Telecommunications and Radio Engineering*, *79*(10).

24. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description. *Advances in Electrical and Electronic Engineering*, *21*(1), 19.

25. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for fast metric data search in structural methods for image classification. *IEEE Access*, *10*, 124738-124746.

26. Daradkeh, Y. I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L. A., & Ahmad, N. (2021). Development of effective methods for structural image recognition using the principles of data granulation and apparatus of fuzzy logic. *IEEE Access*, *9*, 13417-13428.

27. Kobylin, O., & Putiatina, O. (2025). Some aspects of real-time image denoising influenced by shot noise and compound Poisson noise. In *CEUR Workshop Proceedings* (pp. 109-117).

28. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. *Advanced Information Systems*, *9*(1), 5-12.

29. Кобилін, І. О., & Харченко, А. І. (2024). CLASSIFICATION TECHNIQUES FOR COMPUTER VISION. *Системи обробки інформації*, (3 (178)), 33-41.

30. Kharchenko, A. I., & Kobylin, I. O. PERFORMANCE ANALYSIS OF SUPPORT VECTOR MACHINE FOR VEHICLE CLASSIFICATION. *MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE VN KARAZIN KHARKIV NATIONAL UNIVERSITY*, 101.