МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Голова циклової комісії,

спеціаліст в/к

_____ С.О. Сабанов

ВИПУСКНА РОБОТА МОЛОШОГО СПЕЦІАЛІСТА РОЗРОБКА ТЕЛЕГРАМ БОТА РОЗПІЗНАВАЧА ЯПОНСЬКИХ КОМІКСІВ НА МОВІ PYTHON

Виконав

ст. гр. ІПЗ – 119к9 _____ О.Ю. Андрієнко

Керівник

викл. _____ Д.О. Костерной

Запоріжжя

2023

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ЗАТВЕРДЖУЮ

Голова циклової комісії,

спеціаліст в/к

_____ С.О. Сабанов

З А В Д А Н Н Я

НА ВИПУСКНУ РОБОТУ МОЛОДШОГО СПЕЦІАЛІСТА

студенту гр. ІПЗ-119К9 ,

спеціальності 121 - «Інженерія програмного забезпечення»

Андрієнко Олександрі Юріївні

1.Тема: Розробка Телеграм бота розпізнавача японських коміксів на мові Python

затверджена наказом № ___ –__ від _____

2. Термін здачі студентом закінченої роботи: _____

3. Перелік питань, що підлягають розробці:

1. Провести огляд літератури, що присвячена тематиці досліджень.
2. Провести аналіз існуючих технологій перекладу та розпізнавання зображень.
3. Створення плану розробки.
4. Розробка архітектури та функціоналу бота.
5. Створення бази даних.
6. Розробка тестового набору та оцінка ефективності бота.
7. Розгортання та налаштування бота на хмарному сервісі.
8. Оформити звіт за результатами роботи

4. Календарний графік підготовки випускної роботи молодшого спеціаліста

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Формулювання (корегування) теми випускної роботи молодшого спеціаліста та збір практичного матеріалу за темою випускної роботи			
2	І атестація І розділ випускної роботи молодшого спеціаліста			
3	ІІ атестація ІІ розділ випускної роботи молодшого спеціаліста			
4	ІІІ атестація ІІІ розділ випускної роботи молодшого спеціаліста, висновки та рекомендації, додатки, реферат			
5	Перевірка випускної роботи молодшого спеціаліста програмою «Антиплагіат»			
6	Доопрацювання випускної роботи молодшого спеціаліста, підготовка презентації, отримання відгуку керівника і рецензії			
7	Попередній захист випускної роботи молодшого спеціаліста			
8	Подача випускної роботи молодшого спеціаліста на кафедру	за 3 дні до захисту		
9	Захист випускної роботи молодшого спеціаліста			

Керівник _____ Д.О. Костерной
(підпис) (ініціали, прізвище)

«____» _____ 2023 р.

Завдання отримав до виконання _____ О.Ю. Андрієнко
(підпис студента) (ініціали, прізвище)

«____» _____ 2023 р.

РЕФЕРАТ

Пояснювальна записка складається з: 56 с., 79 рисунків, 24 джерела, 7 додатків.

Об’єкт дослідження – Телеграм бот розпізнавач японських коміксів.

Мета роботи – Розробка телеграм бота у середовищі VS Code мовою Python із застосуванням технології Telegram bot API та подійно-орієнтованого програмування.

Розглянуто опис поточного стану в області перекладу та розпізнавання зображень. Зокрема, розглянуто застосування різних видів перекладачів, розпізнавачів, довідників та технологій, що їх забезпечують.

Проводиться аналіз засобів перекладу з використанням Телеграм. Обґрунтовується вибір технології та середовища розробки додатку.

БОТ, ТЕЛЕГРАМ, КЛІЄНТ-СЕРВЕР, API, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, ПЕРЕКЛАД.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна назва	Пояснення/переклад
OCR	Optical Character Recognition	Оптичне розпізнавання символів
API	Application Programming Interface	Прикладний програмний інтерфейс
	Манґа	Комікси, що створені в Японії або японською мовою
	Канджі	Ієрогліфічне письмо, складова частина японської писемності.
PIL	Python Imaging Library	Бібліотека обробки зображень Python
JSON	JavaScript Object Notation	Запис об'єктів JavaScript
MongoDB	Mongo Data Base	База даних Монґо
IDE	Integrated Development Environment	Інтегроване середовище розробки
БД	База даних	
ПЗ	Програмне забезпечення	
	Чат-бот	Віртуальний співрозмовник, програма, яка створена для імітації поведінки людини при спілкуванні з одним або декількома співрозмовниками.
	Месенджер	Клієнтська програма системи миттєвих повідомлень
	Система миттєвих повідомлень	Телекомунікаційна служба для обміну текстовими повідомленнями між комп'ютерами або іншими пристроями користувачів через комп'ютерні мережі

ВСТУП

Тема дипломної роботи - "Телеграм бот розпізнавач японських коміксів на мові Python".

Метою цієї дипломної роботи є розробка ефективного та надійного телеграм бота, який зможе розпізнавати японський текст та перекладати його на обрану мову. Крім того, метою є створення бази даних перекладених ієрогліфів для вивчення їх методом повторення та тестування запам'ятовування. Результатом дослідження буде готовий телеграм бот та база даних ієрогліфів, які можуть бути використані для вивчення японської мови.

Об’єктом дослідження є область комп’ютерних наук, а саме створення телеграм бота, який здійснює розпізнавання японських коміксів і переклад їх тексту на іншу мову. Предметом дослідження є розробка та реалізація алгоритмів розпізнавання та перекладу японського тексту, а також створення бази даних для зберігання перекладених ієрогліфів.

Тема моєї дипломної роботи є важливою і актуальною з кількох причин:

- Зараз манґа (японські комікси) є дуже популярною формою розваг у світі, і мільйони людей читають її щодня. Однак, не всім доступні японські видання манґи, і навіть якщо їм вони і доступні то не усім зрозуміла мова, та і не кожен може зрозуміти складний ієрогліфічний письмовий знак, яким викладена манґа. Моє дослідження поєднає в собі функції розпізнавання японської мови та перекладу ієрогліфів на зрозумілу більшості мову, що забезпечить доступ до японських коміксів для більш широкої аудиторії.
- Моя дипломна робота відповідає потребам технологічного розвитку, а саме розширює можливості телеграм ботів на прикладі розпізнавання та перекладу японської мови. Це може знайти своє використання у різних сферах, таких як туризм, ділові зустрічі, навчання та багато іншого.
- Створення телеграм-бота, який зможе розпізнавати японський текст на сторінках манґи та перекладати його, може стати важливим інструментом для людей, які навчаються японської мови або

зацікавлені в її вивченні. Крім того, бот зможе збирати дані та зберігати їх у базі даних для того, щоб користувачі могли вивчати японські ієрогліфи методом повторення та тестування запам'ятованих.

– Створення такого бота дозволить автоматизувати процес перекладу манґи, що зекономить час та зусилля перекладачів та дозволить їм сконцентруватись на більш важливих завданнях.

За останні роки, дослідження у галузі розпізнавання та перекладу текстів японських коміксів стали дуже популярні.

Одним з основних напрямків роботи є використання комп'ютерного зору та обробки зображень для виявлення та розпізнавання японського тексту. Для цього застосовуються алгоритми, такі як розрізнення контурів, сегментація та класифікація зображень.

Для перекладу тексту зі сторінок манґи було запропоновано кілька підходів. Один з них - використання машинного перекладу на основі нейронних мереж. Однак, цей підхід не завжди дає точний результат через особливості японської мови та складну структуру речень.

Ще одним напрямком розробок є використання методів оптичного розпізнавання символів (OCR) для розпізнавання японських ієрогліфів. Однак, цей підхід не завжди дозволяє точно розпізнати символи, особливо якщо вони написані в різних стилях та шрифтах.

Проблеми, що залишаються невирішеними в цій галузі, включають точність та швидкість розпізнавання японського тексту, особливо у складних умовах, таких як зміщення, нахил, та використання різних шрифтів та стилів.

У цій дипломній роботі я досліджувала тему розробки телеграм бота для розпізнавання коміксів на зображеннях. Моя робота має великий практичний потенціал, оскільки може бути використана для швидкого та точного розпізнавання коміксів, що є важливим у багатьох сферах, таких як маркетинг, аналіз соціальних мереж та культурних тенденцій, наукові дослідження та багато іншого. Я впевнена, що моя робота стане корисною для широкого кола користувачів та сприятиме вирішенню актуальних проблем у цій галузі.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ТЕХНОЛОГІЙ

1.1 Пошук та аналіз існуючих телеграм ботів для розпізнавання коміксів

Проведений мною пошук не надав жодних результатів щодо існуючих ботів перекладачів або розпізнавачів коміксів. Однак існують окремі додатки, що можуть це робити. Наприклад:

- Manga Translator – доповнення до браузера Chrome, що використовує технології штучного інтелекту для перекладу коміксів, але перекладає тільки англійською.
- Scan Translator – ще одне доповнення для браузера Chrome, що може перекладати комікси на 50 мов, використовуючи технології штучного інтелекту, але має обмеження у 20 перекладів на тиждень.
- Cotrans Manga/Image Translator – онлайн перекладач, який перекладає комікси на будь-яку мову, за допомогою штучного інтелекту, але переклад займає багато часу, адже вебсайт додає кожен запит на переклад до черги, яка зазвичай не коротка.
- Також на ресурсі github.com було знайдені проєкти перекладачів такі як, manga-translator, JMTrans, Manga-MTL-Website, comic-translator, роботу яких не вдалось перевірити.

1.2 Огляд існуючих бібліотек та інструментів для роботи з зображеннями та обробки мови

Існує безліч бібліотек та інструментів для роботи з зображеннями та обробки мови. Огляд деяких з них наведено нижче:

- OpenCV

OpenCV - це бібліотека комп'ютерного зору з відкритим кодом, яка дозволяє працювати з зображеннями та відео.

OpenCV застосовується в багатьох галузях, включаючи комп'ютерний зір, робототехніку, автоматичне керування, медицину та інші. Завдяки своїй відкритості та потужній функціональності, OpenCV є однією з найпопулярніших бібліотек для обробки зображень та комп'ютерного зору.

– TensorFlow

TensorFlow - це відкрите програмне забезпечення з вільним кодом для створення та навчання моделей машинного навчання.

TensorFlow використовується в багатьох галузях, включаючи комп'ютерний зір, обробку природної мови, аналіз даних та інші. Завдяки своїй потужній функціональності та відкритості, TensorFlow є однією з найпопулярніших бібліотек для машинного навчання.

– PyTorch

PyTorch - це бібліотека з відкритим кодом для машинного навчання та глибокого навчання, розроблена компанією Facebook.

Однією з особливостей PyTorch є те, що вона дозволяє використовувати GPU для прискорення обчислень, що дозволяє розробникам машинного навчання працювати з великими обсягами даних та розробляти складні моделі швидше.

– NLTK (Natural Language Toolkit)

NLTK (Natural Language Toolkit) - це бібліотека для обробки природної мови (NLP) в середовищі Python.

NLTK використовується в багатьох галузях, включаючи аналіз соціальних медіа, машинний переклад, обробку документів та інші. Ця бібліотека стала популярною серед дослідників та розробників машинного навчання, які працюють з текстовими даними.

– spaCy

spaCy - це бібліотека для обробки природної мови (NLP) в середовищі Python.

Загалом, spaCy є потужним та ефективним інструментом для обробки природної мови, який дозволяє розробникам легко та швидко розробляти програми для аналізу та обробки тексту на багатьох мовах.

– PIL (Python Imaging Library)

PIL (Python Imaging Library) - це бібліотека для роботи з зображеннями в середовищі Python.

Загалом, PIL є потужною та простою використанні бібліотекою для роботи з зображеннями в середовищі Python. Вона надає широкі можливості для обробки та маніпулювання зображеннями, що дозволяє розробникам легко та швидко створювати програми для обробки зображень.

– scikit-image

Scikit-image - це бібліотека для обробки зображень в середовищі Python.

Загалом, scikit-image є потужною та розширеною бібліотекою для обробки зображень в середовищі Python. Вона містить розширений набір інструментів для обробки зображень та машинного навчання, що дозволяє розробникам легко створювати програми для обробки зображень в різних областях, таких як медична діагностика, розпізнавання обличчя, комп'ютерний зір та багато інших.

– Keras

Keras - це високорівнева бібліотека для машинного навчання, яка працює в середовищі Python та надає інтерфейс для розробки та тренування нейронних мереж.

Загалом, Keras є потужною та зручною бібліотекою для розробки та тренування нейронних мереж. Вона надає високорівневий API для роботи з мережами та має багато інструментів для тренування та оцінки моделей. Keras є частиною екосистеми Tensorflow та може використовуватися разом з іншими бібліотеками машинного навчання

– DeepL

DeepL є однією з провідних онлайн-платформ для машинного перекладу, яка використовує глибоке навчання

Загалом, DeepL є потужним та досить точним інструментом машинного перекладу, який дозволяє отримувати якісні результати перекладу різноманітних текстів різної складності та формату.

– Pytesseract

Pytesseract - це бібліотека для оптичного розпізнавання символів (OCR) у зображеннях та документах за допомогою Tesseract OCR Engine.

Загалом, Pytesseract є потужним та простим у використанні інструментом, що дозволяє виконувати оптичне розпізнавання символів з високою точністю та швидкістю.

– Googletrans

Googletrans - це бібліотека машинного перекладу для мови Python, яка використовує веб-служби Google Translate для автоматичного перекладу тексту з однієї мови на іншу.

Загалом, Googletrans - це потужна та проста у використанні бібліотека машинного перекладу для мови Python, що дозволяє виконувати переклад текстів з високою точністю та швидкістю.

1.2 Висновки за розділом

Після аналізу існуючих бібліотек та інструментів для обробки зображень та мови, було прийнято рішення використати Googletrans, Pytesseract та PIL для реалізації проекту. Googletrans використовує веб-служби Google Translate для автоматичного перекладу тексту з однієї мови на іншу, Pytesseract - це OCR-бібліотека для розпізнавання тексту зображень, а PIL - бібліотека для роботи з зображеннями у Python. Використання цих інструментів дозволить розпізнавати та перекладати текст зображень, що є ключовою функціональністю проекту. Крім того, ці інструменти дуже популярні та мають добру документацію та підтримку спільноти, що забезпечить простоту використання та відладку. Отже, використання Googletrans, Pytesseract та PIL є розумним вибором для реалізації проекту з розпізнавання та перекладу коміксів.

РОЗДІЛ 2
ІНСТРУМЕНТИ РОЗРОБКИ

2.1 Загальний огляд компонентів

Під час створення цього додатка були використані наступні інструменти:

- для клієнтської частини програми: Telegram bot API;
- для організації роботи програми: dotenv, PIL, pytesseract, deep_translator, re, os, typing, jisho_api, json, random;
- для оформлення взаємодії з базою даних: pymongo.

Допоміжні бібліотеки та тривіальні рішення не є темою даної дипломної роботи, але коротко розглянемо детальніше використані основні технології.

2.2 Telegram Bot API

Telegram Bot API – це набір інструментів, який надається Telegram для розробників, щоб створювати та керувати своїми власними телеграм ботами. Цей API дозволяє розробникам взаємодіяти з Telegram-платформою, отримувати та надсилати повідомлення, керувати користувачами та ботами, обробляти медіафайли та виконувати багато інших операцій.

Основні можливості Telegram Bot API включають:

– Взаємодія з повідомленнями: API дозволяє отримувати повідомлення від користувачів та відправляти їм відповіді. Це може бути текстове повідомлення, зображення, відео, аудіо, стікери та багато іншого. Розробники можуть обробляти отримані повідомлення та виконувати необхідну логіку відповідно до них.

– Керування клавіатурою та кнопками: API дозволяє створювати клавіатури та кнопки для взаємодії з користувачем. Це дозволяє створювати інтерактивні боти зі спеціальними кнопками для виконання певних дій.

– Управління користувачами: API надає можливість отримувати інформацію про користувачів, їхній статус та взаємодіяти з ними. Розробники можуть

керувати списком користувачів, блокувати або розблоковувати їх, встановлювати адміністративні привілеї та багато іншого.

- Робота з медіафайлами: API дозволяє обробляти та відправляти різні типи медіафайлів, такі як фотографії, відео, аудіо, документи тощо. Розробники можуть створювати ботів, які автоматично обробляють та відповідають на отримані медіафайли.
- Взаємодія з групами та каналами: API дозволяє керувати групами та каналами, надсилати повідомлення та отримувати їх, додавати та видаляти учасників групи чи каналу, налаштовувати права доступу тощо.

Це лише декілька основних можливостей Telegram Bot API. Він надає розробникам потужний інструментарій для створення різноманітних телеграм ботів зі спеціальною функціональністю та взаємодією з користувачами.

2.3 Бібліотека програмування dotenv

dotenv (скорочено від "Dotenv") - це бібліотека для мови програмування Python, яка дозволяє завантажувати змінні оточення з файлу .env у процес вашої програми. Файл .env - це текстовий файл, в якому зберігаються конфігураційні дані, такі як паролі, ключі API, налаштування з'єднання з базою даних тощо. Використання dotenv дозволяє вам зберігати ці дані окремо від коду програми, що полегшує управління конфігурацією та забезпечує безпеку.

Бібліотека dotenv зчитує змінні оточення з файлу .env і робить їх доступними у вашій програмі як звичайні змінні середовища. Це дозволяє вам зручно використовувати ці значення в коді програми без необхідності хардкодити їх безпосередньо.

Основні переваги використання dotenv:

- Зручність: Завантаження змінних оточення з файлу .env дозволяє зберігати конфігурацію вашої програми в одному місці, що спрощує її управління.
- Безпека: Змінні оточення, які зберігаються в файлі .env, можна виключити з системи контролю версій, що забезпечує безпеку конфіденційної інформації, такої як паролі або ключі API.

- Портативність: Ви можете легко переносити вашу програму на інші середовища, такі як розробка, тестування та продакшн, просто змінивши файли .env, без необхідності внесення змін у код програми.
- Сумісність: Бібліотека dotenv підтримується в різних середовищах програмування, включаючи Python, що дозволяє використовувати її в широкому спектрі проектів.

dotenv – це корисна бібліотека для управління конфігурацією вашої програми та збереження конфіденційної інформації в безпечному місці. Вона спрощує налаштування середовища вашої програми та полегшує її перенесення на різні середовища розробки.

2.4 Бібліотека програмування PIL

PIL (Python Imaging Library) – це бібліотека для мови програмування Python, яка надає широкий набір функцій для роботи з зображеннями. Вона дозволяє відкривати, зберігати, змінювати розмір, обрізати, маніпулювати пікселями та здійснювати багато інших операцій над зображеннями.

Основні можливості PIL включають:

- Завантаження та збереження зображень: PIL підтримує різні формати зображень, включаючи JPEG, PNG, BMP, GIF і TIFF. Вона дозволяє відкривати зображення з файлу, зберігати зображення у файл або працювати з зображеннями, що знаходяться у пам'яті.
- Маніпуляції з зображеннями: PIL надає різні методи для зміни розміру зображення, обрізання, обертання, зміни контрастності та яскравості, налаштування кольорів, застосування фільтрів тощо. Ви можете виконувати складні операції над зображеннями за допомогою простих методів, які надає PIL.
- Робота з пікселями: PIL дозволяє доступатися до окремих пікселів зображення та змінювати їх значення. Це дозволяє вам виконувати детальні маніпуляції з кожним пікселем зображення, наприклад, змінювати його кольорову складову або застосовувати ефекти.

– Робота з масками та альфа-каналами: PIL дозволяє працювати з масками, що дозволяють вибрати певні області зображення для обробки. Ви також можете керувати альфа-каналом зображення, що використовується для представлення прозорості пікселів.

PIL є потужним інструментом для роботи з зображеннями в мові програмування Python. Вона широко використовується у різних областях, таких як обробка зображень, комп'ютерне зорове сприйняття, графіка, наука про дані та багато інших.

2.5 Бібліотека програмування Pytesseract

Pytesseract – це бібліотека для мови програмування Python, яка надає простий спосіб розпізнавання тексту з зображень за допомогою відкритого OCR-движку Tesseract. OCR (Optical Character Recognition) – це технологія, яка дозволяє комп'ютеру розпізнавати текст на зображеннях або сканованих документах.

Pytesseract забезпечує зручний інтерфейс для використання Tesseract в Python. Вона дозволяє завантажувати зображення і отримувати розпізнаний текст з цих зображень у вигляді рядка. Бібліотека підтримує багато мов, включаючи англійську, французьку, німецьку, іспанську, китайську та багато інших.

Основні переваги використання pytesseract:

– Простота використання: Завдяки простому інтерфейсу pytesseract, використання Tesseract для розпізнавання тексту стає легким завданням. Вам потрібно лише завантажити зображення та викликати функцію pytesseract.image_to_string() для отримання розпізнаного тексту.
– Підтримка багатьох мов: Pytesseract підтримує розпізнавання тексту на різних мовах, що дозволяє використовувати його для різноманітних проектів, які потребують розпізнавання тексту різних мов.
– Налаштування параметрів: Pytesseract дозволяє налаштувати різні параметри розпізнавання, такі як мова, розмір шрифту, детекція

розділових знаків тощо. Це дає вам більшу гнучкість у виборі налаштувань для досягнення кращих результатів розпізнавання.

- Відкритість: Pytesseract базується на Tesseract, який є відкритим проектом з вільною ліцензією Apache 2.0. Це означає, що ви можете використовувати та модифікувати код pytesseract та Tesseract згідно з вашими потребами.

Pytesseract є потужним інструментом для розпізнавання тексту з зображень у програмах Python. Вона забезпечує легку інтеграцію OCR-функціональності в ваші проекти та відкриває багато можливостей для автоматичного аналізу та обробки текстової інформації.

2.6 Бібліотека програмування Deep_translator

Deep_translator – це бібліотека для мови програмування Python, яка надає простий спосіб перекладати тексти між різними мовами за допомогою різних машинних перекладачів. Вона дозволяє легко інтегрувати функціональність машинного перекладу в ваші програми та додатки.

Основні можливості deep_translator включають:

- Машинний переклад: Бібліотека підтримує різні машинні перекладачі, такі як Google Translate, Microsoft Translator та інші. Вона надає зручний спосіб використовувати ці сервіси для перекладу тексту між різними мовами.
- Підтримка багатьох мов: Deep_translator підтримує широкий спектр мов, що дозволяє використовувати її для перекладу тексту у багатьох мовних комбінаціях. Ви можете легко налаштувати бібліотеку для роботи з конкретними мовами, використовуючи відповідні параметри.
- Простий використання: Deep_translator надає простий та зрозумілий інтерфейс для перекладу тексту. Вам потрібно лише викликати відповідну функцію для перекладу тексту, вказавши вхідну мову та цільову мову. Результат перекладу повертається у зручному форматі, що дозволяє використовувати його у вашому коді.
- Розширені можливості: Deep_translator також надає можливість виконувати інші операції, пов'язані з машинним перекладом, такі як

визначення мови тексту, отримання доступних мовних пар перекладу, виявлення мовних кодів та інше. Це дозволяє вам розширити функціональність вашого додатку, пов'язаного з перекладом тексту.

Deep_translator є корисною бібліотекою для реалізації машинного перекладу в програмах Python. Вона спрощує процес перекладу тексту та відкриває широкі можливості для розробки додатків, що вимагають функціональності перекладу між різними мовами.

2.7 API онлайн-словника jisho_api

jisho_api - це необхідний інструмент для доступу до API Jisho, яке є онлайн-словником і перекладачем японської мови. Цей API надає можливість отримувати інформацію про слова, канї (китайські позначення для японських канджі) та інші аспекти японської мови. За допомогою jisho_api можна автоматизувати процес отримання перекладів, визначення частин мови та інших лінгвістичних даних для японських слів і канджі.

За допомогою jisho_api можна робити наступні запити до API Jisho:

- Пошук за словом або фразою: Ви можете шукати переклади або додаткові відомості про слово або фразу в японській мові. Запит повертає список результатів, які містять інформацію про переклади, визначення, частини мови, приклади вживання та інше.
- Пошук за канджі: Ви можете шукати інформацію про певний канджі (китайський ієрогліф, що використовується в японській мові). Запит повертає дані про канджі, такі як його значення, частини мови, приклади вживання та інші атрибути.
- Пошук за ромаджі: Ви можете шукати японські слова або канджі за їх ромаджі (латинська транслітерація японських символів). Це зручно, коли ви не знаєте канджі для певного слова.

jisho_api дозволяє отримувати детальну інформацію про слова та канджі в японській мові, що робить його цінним інструментом для розробки додатків, ботів або веб-сайтів, пов'язаних з японською мовою.

2.8 Бібліотека програмування pymongo

pymongo є офіційною бібліотекою для підключення та взаємодії з MongoDB базами даних у мові програмування Python. Вона надає простий та зручний спосіб працювати з MongoDB, дозволяючи виконувати операції збереження, вставки, оновлення та видалення даних.

Основні можливості та функціональність, надана pymongo, включають:

- Підключення до MongoDB: pymongo дозволяє підключитися до MongoDB сервера за допомогою URL-адреси бази даних або параметрів підключення, таких як хост, порт та інші налаштування.
- Взаємодія з колекціями: pymongo дозволяє виконувати операції збереження, вставки, оновлення, видалення та отримання даних з колекцій MongoDB. Ви можете використовувати різні методи та опції для виконання потрібних запитів.
- Запити до бази даних: pymongo дозволяє виконувати різні типи запитів до бази даних MongoDB, такі як прості запити, запити з фільтрацією, сортуванням, проекцією, агрегацією та багато іншого. Ви можете використовувати потужну мову запитів MongoDB (MongoDB Query Language) для отримання потрібних результатів.
- Індексація та оптимізація: pymongo дозволяє створювати та керувати індексами в MongoDB, що допомагає покращити швидкодію та ефективність запитів до бази даних.
- Робота з даними у вигляді об'єктів Python: pymongo надає можливість зберігати та отримувати дані у вигляді об'єктів Python, таких як словники, списки та інші структури даних.

pymongo є потужним інструментом для взаємодії з MongoDB базами даних у Python. Вона дозволяє легко та зручно виконувати операції збереження, запитів та оптимізації даних у MongoDB, що робить її популярним вибором для розробки Python додатків, які використовують MongoDB.

2.9 Інші інструменти розробки

Модуль re є вбудованим модулем мови програмування Python, який надає можливості роботи з регулярними виразами. Регулярні вирази – це механізм для пошуку та обробки тексту за певними шаблонами.

Модуль os є вбудованим модулем мови програмування Python, який надає функціональність для взаємодії з операційною системою. Він дозволяє вам працювати з файловою системою, отримувати інформацію про середовище виконання програми, взаємодіяти з процесами та багато іншого.

Модуль typing є вбудованим модулем мови програмування Python, який надає можливості для статичного типування в Python 3.6 і новіших версіях. Він дозволяє вам вказувати типи даних для змінних, аргументів функцій та повертає значення функцій, що полегшує розробку, розуміння та підтримку коду.

Бібліотека json в мові програмування Python надає засоби для роботи з форматом даних JSON (JavaScript Object Notation). JSON є легким, текстовим форматом обміну даними, який широко використовується для передачі та збереження структурованих даних.

Бібліотека json надає функціональність для серіалізації (конвертації об'єктів Python у рядки JSON) та десеріалізації (конвертації рядків JSON у об'єкти Python) даних. Основні функції та класи, що надаються бібліотекою json, включають:

Модуль random є вбудованим модулем мови програмування Python, який надає функції для генерації випадкових чисел, вибору випадкових елементів та інших операцій, пов'язаних з випадковістю.

2.10 Висновки за розділом

У цьому розділі були розглянуті усі інструменти, що були використані під час розробки додатку. Вони дозволили розробити потужний та функціональний Telegram бот для розпізнавання японських коміксів та надання довідки по словах.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО КОДУ БОТА

3.1 Архітектура додатку

Архітектура додатку включатиме серверну частину, клієнтську частину та базу даних. Клієнтська частина буде відповідальна за інтерфейс користувача та взаємодію з серверною частиною. Серверна частина буде обробляти запити від клієнтів та здійснювати необхідну логіку додатку. База даних буде використовуватися для зберігання та управління даними додатку.

У даній архітектурі використовується методика Long Polling для передачі оновлень від клієнта до сервера. Long Polling є технікою, коли клієнтська частина робить запит до сервера, а сервер тримає цей запит в очікуванні, поки не буде наявна нова інформація для відповіді. Це дозволяє серверу відповідати на запити клієнта миттєво, як тільки відбуваються оновлення.

Серверна частина буде взаємодіяти з базою даних за допомогою API бази даних, зокрема з використанням MongoDB API. MongoDB є документ-орієнтованою базою даних, яка дозволяє зберігати дані у вигляді JSON-подібних документів. Вона надає розширений набір функцій для зберігання, оновлення та запитування даних.

Така архітектура додатку забезпечує ефективну комунікацію між клієнтом та сервером, зручне управління даними за допомогою бази даних та можливість розширення та модифікації функціональності додатку. Застосування методики Long Polling та використання MongoDB API допоможуть забезпечити швидку та надійну роботу додатку, забезпечуючи задоволення потреб користувачів.

3.2 Серверна частина

Розглянемо поступово структуру сервера на рисунку 3.1:

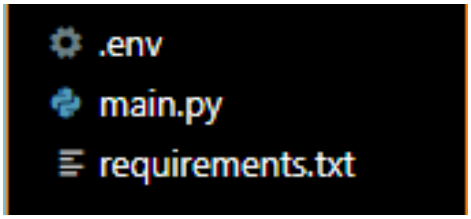

Рис. 3.1 — Структура сервера

Файл .env містить ключі взаємодії з базою даних та ботом. На рисунку 3.2 зображено вигляд структури файлу:

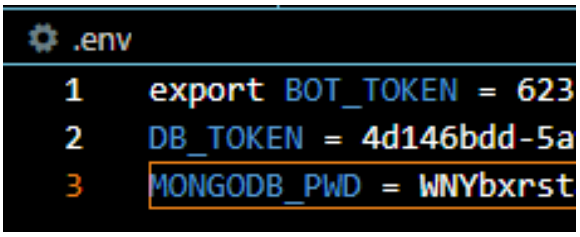

Рис. 3.2 — Структура файлу .env

Файл main.py містить увесь код, що забезпечує роботу серверної частини додатку та є основним скриптом цього проєкту.

Файл requirements.txt є текстовим файлом, який використовується в екосистемі Python для визначення залежностей проекту. Він містить перелік пакетів (і їх версій), які необхідно встановити, щоб проект працював належним чином.

Основна мета файлу requirements.txt – забезпечити легку та повторювану установку всіх залежностей проєкту. Коли створюється або виконується робота над Python-проєктом, зазвичай використовуються сторонні бібліотеки або модулі, які не є частиною стандартної бібліотеки Python. requirements.txt допомагає вказати ці залежності та їх версії, щоб встановити їх одночасно і автоматично.

3.3 Клієнтська частина

Клієнтська частина реалізована засобами Телеграм. За допомогою бота, що відповідає за усі дії щодо телеграм ботів, було налаштовано:

– Ім’я, опис, зображення профілю бота, що зображені на рисунку 3.3:

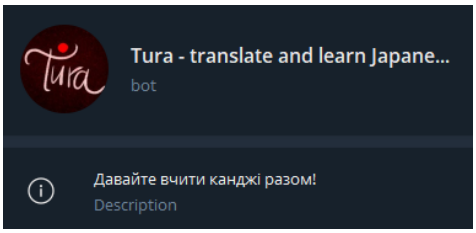

Рис. 3.3 — Зовнішній вигляд профілю бота у Телеграм

– Список команд, що зображений на рисунку 3.4:

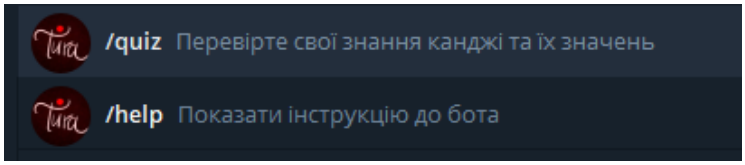

Рис. 3.4 — Список команд

Приклад роботи з ботом наведено на рисунках 3.5 – 3.10:

Рис. 3.5 — При першому відкритті чату з ботом буде відображено довідкове повідомлення

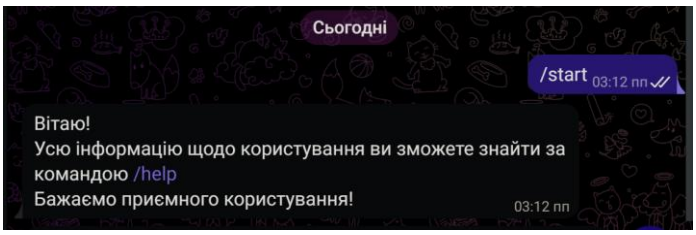

Рис. 3.6 — При натисканні кнопки «Розпочати» буде викликано команду «start»

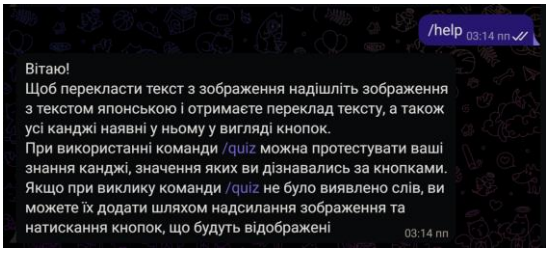

Рис. 3.7 — Результат виконання команди «help»

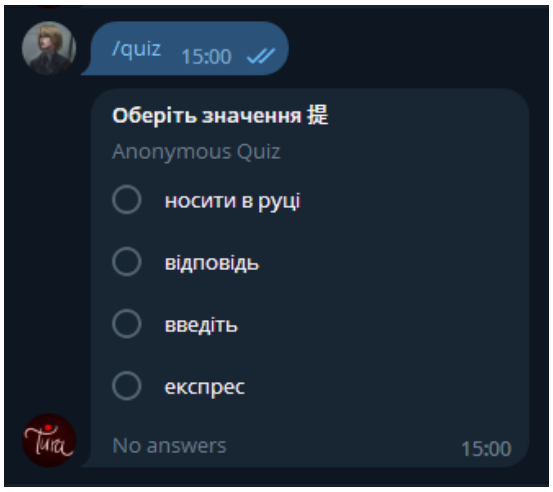

Рис. 3.8 — Результат виконання команди «quiz»

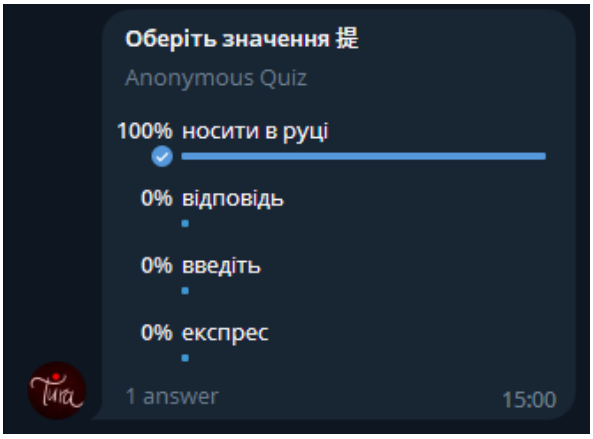

Рис. 3.9 — Результат отримання правильної відповіді

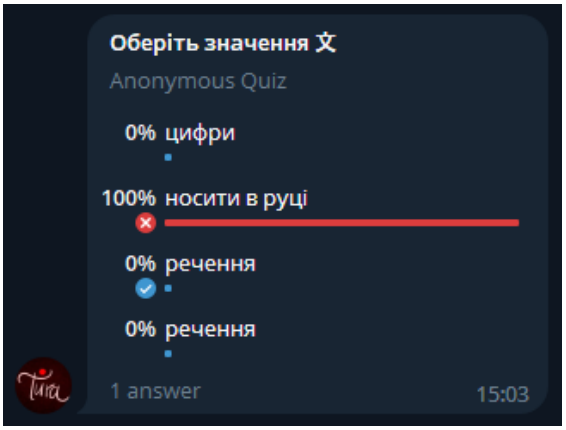

Рис. 3.10 — Результат отримання неправильної відповід

3.4 Розробка алгоритму розпізнавання коміксів на зображеннях

Перед розпізнаванням тексту зображення треба обробити, що буде виконуватися з використанням модуля Image. Розпізнавання тексту відбувається за допомогою модуля Pytesseract, заснованого на технологіях Tesseract.

Для початку необхідно провести підключення модулів:

```
from PIL import Image
import pytesseract
```

Далі відкривається файл з зображенням та розпізнається з зображення текст:

```
image = Image.open(file)
    text = pytesseract.image_to_string(image, config='-l jpn')
```

, де config='-l jpn' зазначає те, що мова тексту на зображеннях буде японська.

У результаті виконання коду, описаного вище, отримується змінна, вмістом якої є увесь текст, що було зчитано з зображення.

Наступний крок – це переклад та обробка тексту, отриманого у попередньому кроці. Це виконується використовуючи функцію is_kanji та модулі такі, як: re та GoogleTranslator.

Функція is_kanji перевіряє чи переданий їй символ є канджі, виконуючи перевірку за діапазоном Unicode кодів символів, до якого належать усі канджі, і повертає істину, якщо символ є канджі, та хибу – якщо ні:

```
def is_kanji(character):
    kanji_range = (0x4E00, 0x9FBF)
    character_code = ord(character)
    return kanji_range[0] <= character_code <= kanji_range[1]
```

Далі необхідно підключити модулі:

```
from deep_translator import GoogleTranslator
import re
```

Після цього треба виконати видалення перенесень рядків та пробілів на початку та на кінці задля реалізації нормалізації отриманого з зображення тексту, яка сприятиме уникненню зайвих помилок при подальшій роботі з текстом:

```
new_text = re.sub(r"[\n\r]+", " ", text).strip()
```

Наступним кроком у роботі програми є переклад тексту:

```
translated_text = GoogleTranslator(source='ja', target='uk').translate(new_text)
```

Далі з тексту вибираються канджі. Обрані канджі додаються до списку та проводиться перевірка на наявність повторюваних символів. Адже даний список пізніше буде використовуватись для формування клавіатури кнопок, які користувач зможе використовувати для отримання довідки про кожен символ, що був у розпізнаному тексті:

```
kanji_list = []
    for i in new_text:
        if is_kanji(i):
            kanji_list.append(i)
    no_dupes_kanji = []
    for i in kanji_list:
        if i not in no_dupes_kanji:
            no_dupes_kanji.append(i)
```

Наступний крок – розробка функції отримання інформації про кожний окремий канджі. Для цього використовуватимуться функції is_a_valid_symbol і value та модулі: jisho_api, json та GoogleTranslator.

Функція value перевіряє, чи отриманий текст є порожнім і, якщо дана умова виконується, повертає символ пробілу, інакше – значення переданої їй змінної:

```
def value(variable):
    if variable is None:
        return " "
    else:
        return variable
```

Функція is_a_valid_symbol перевіряє, чи текст є порожнім або числом, аби забезпечити безпомилкову роботу перекладача, та повертає істину лише у разі невиконання і першої, і другої умов:

```
def is_a_valid_symbol(a:str):
    a = a.strip()
    if not a.isnumeric():
        if not (a == ""):
            return True
        else:
            return False
    else:
            return False
```

Для забезпечення роботи наступної функції необхідно провести підключення необхідних модулів:

```
from jisho_api.kanji import Kanji
import json
from deep_translator import GoogleTranslator
```

Далі виконується запит до словника японської мови jisho та розбивається отримана відповідь на змінні, щоб дані було зручно використати у наступних етапах розробки – див. Додаток А

Наступний крок – це забезпечення взаємодії з базою даних. Для цього використовується база даних MongoDB, модулі dotenv, MongoClient, os та функції insert_into_doc та find_words.

Функція insert_into_doc виконує операцію додавання запису до бази даних:

```
def insert_into_doc(user_data_doc, update:Update):
    collection = accounts_db.Translations
    inserted_id = collection.insert_one(user_data_doc).inserted_id
```

Функція find_words виконує пошук по записам у базі даних та повертає отримані результати:

```
def find_words(update: Update):
```

```
    id = update.message.from_user.id
    results = acc_collection.find({"user":id})
    return results
```

Наступним кроком виконується підключення необхідних модулів:

```
from pymongo import MongoClient
from dotenv import load_dotenv, find_dotenv
import os
```

Далі виконується авторизація у системі MongoDB, проводиться підключення до бази даних за допомогою логіна та пароля, що були введені при реєстрації на вебсайті сервісу, та зберігаються у файлі .env та здійснюється обрання колекції записів, з якою буде працювати даний додаток:

```
load_dotenv(find_dotenv())
password = os.environ.get('MONGODB_PWD')
connection_string = f"mongodb+srv://Oleksa:{password}@cluster0.bjdsloy.mongodb.net/?retryWrites=true&w=majority"
client = MongoClient(connection_string)
dbs = client.list_database_names()
accounts_db = client.Translations
collections = accounts_db.list_collection_names()
acc_collection = accounts_db.Translations
```

Після того виконується внесення даних зі змінних, отриманих раніше, до бази даних з полями для кожного канджі, його основних значень та ідентифікатора користувача, до якого прив’язані записані слова:

```
kanji_data = {
    "kanji" : kanji,
    "main meanings" : tr_meanings,
    "user": query.from_user.id
}
insert_into_doc(kanji_data, update)
```

Далі проводиться отримання слів за ідентифікатором користувача:

results = list(find_words(update))

Наступний крок – це розробка алгоритму створення навчальних тестів по отриманих з бази даних словах. Для цього використовуватиметься модуль random.

Виконується підключення даного модуля:

import random

Далі зі списку слів обирається те канджі, про яке буде поставлене питання. Зі значень цього канджі обирається те значення, яке буде одним з варіантів відповідей та вважатиметься правильною відповіддю. Далі значення усіх слів, прив’язаних до користувача, заносяться у один список. З цього списку обиратимуться три інші варіанти відповіді, що вважатимуться неправильними. Після цього генерується випадкове число від одного до чотирьох, що позначатиме порядковий номер правильної відповіді серед усіх можливих відповідей. У результаті отримується змінна, що містить питання, список, що містить варіанти відповідей та змінна, яка зберігає порядковий номер правильної відповіді – див. Додаток Б

3.5 Реалізація функцій бота для взаємодії з користувачем та відображення результатів

Для реалізації роботи бота з Телеграм необхідно підключити такі модулі як: InlineKeyboardButton, InlineKeyboardMarkup, Update, CallbackQuery, Application, CallbackQueryHandler, CommandHandler, MessageHandler, filters, ContextTypes, load_dotenv, find_dotenv, os, Final де:

- InlineKeyboardButton – клас, що відповідає за одну кнопку, прикріплену до повідомлення.
- InlineKeyboardMarkup – клас, що відповідає за відображення усіх кнопок разом.

- Update – клас, об’єкт якого містить отримане оновлення від клієнта/користувача.
- CallbackQuery – клас, що відповідає за інформацію про взаємодію користувача з кнопками.
- Application – клас, що містить сам додаток, тобто відповідає за отримання оновлень та передачу їх відповідним обробникам. Також об’єкт цього класу є стартовою точкою роботи додатку.
- CallbackQueryHandler – клас, що відповідає за обробку об’єктів класу CallbackQuery.
- CommandHandler – клас, що відповідає за обробку отриманих команд.
- MessageHandler – клас, що відповідає за обробку отриманих повідомлень.
- Filters – модуль, що відповідає за сортування повідомлень за типами вмісту.
- ContextTypes – клас, що відповідає за сортування оновлень за типами причини їх виникнення.
- load_dotenv – функція, що відповідає за отримання даних про усі змінні з файлу .env.
- find_dotenv – функція, що виконує пошук файлу типу dotenv та повертає шлях до файлу у разі успішного виконання.

Для подальшої роботи виконується підключення модулів:

```
from typing import Final
from telegram import InlineKeyboardButton, InlineKeyboardMarkup, Update, CallbackQuery
from telegram.ext import Application, CallbackQueryHandler, CommandHandler, MessageHandler, filters, ContextTypes
import os
from dotenv import load_dotenv, find_dotenv
```

Спочатку необхідно виконати створення об’єкту класу Application, через який відбуватиметься використання усіх функцій Telegram bot API, та відповідні обробники команд для усіх типів взаємодій користувача з ботм, на які відповідатиме бот та почати отримання оновлень за методом long polling:

```
load_dotenv(find_dotenv())
TOKEN: Final = os.environ.get('BOT_TOKEN')
app = Application.builder().token(TOKEN).build()
   app.add_handler(CommandHandler('start', start_command))
   app.add_handler(CommandHandler('quiz', translate_command))
   app.add_handler(CommandHandler('help', help_command))
   app.add_handler(MessageHandler(filters.PHOTO, translate_msg))
   app.add_handler(CallbackQueryHandler(button))
   app.add_error_handler(error)
   app.run_polling(poll_interval=3)
```

Далі ініціалізуються функції, яким обробники подій передаватимуть оновлення. Функція, що відповідає за надсилання довідки і викликається командою help:

```
async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Вітаю!\nЩоб перекласти текст з зображення надішліть зображення з текстом японською і отримаєте переклад тексту, а також усі канджі наявні у ньому у вигляді кнопок.\nПри використанні команди /quiz можна протестувати ваші знання канджі, значення яких ви дізнавались за кнопками.\nЯкщо при виклику команди /quiz не було виявлено слів, ви можете їх додати шляхом надсилання зображення та натискання кнопок, що будуть відображені")
```

Функція, що відповідає за надсилання привітального повідомлення і викликається командою start:

```
async def start_command(update: Update, context:ContextTypes.DEFAULT_TYPE):
```

await update.message.reply_text("Вітаю!\nУсю інформацію щодо користування ви зможете знайти за командою /help\nБажаємо приємного користування!")

Функція, що відповідає за надсилання тестів по словах і викликається командою quiz – див. Додаток В.

Функція, що відповідає за отримання зображення, його завантаження та створення кнопок, за допомогою яких користувач може вказати, чи текст зображення вертикальний чи горизонтальний, та при натисканні кнопок передає інформацію іншій функції і викликається при надсиланні боту зображення – див. Додаток Г

Функція, що відповідає за надання інформації по тексту, зчитаному з зображення, його переклад та створює набір кнопок, що містять кожне канджі, що є у отриманому тексті і викликається при надсиланні боту зображення і викликається обробником оновлень після натискання кнопок – див. Додаток Д.

Ця функція використовує іншу функцію для завантаження файлів для подальшого використання:

```
async def downloader(update: Update, context: ContextTypes.DEFAULT_TYPE):
    new_file = await update.message.effective_attachment[-1].get_file()
    file = await new_file.download_to_drive()
    return file
```

Також використовується функція, що розділяє список на рівні частини, щоб забезпечити те, що будь-яка кількість кнопок може бути відображена коректно:

```
def chunks(lst, n):
    for i in range(0, len(lst), n):
        yield lst[i:i + n]
```

Функція, що відповідає за надання розробнику інформації про помилки під час виконання програми і викликається при виникненні помилок:

```
async def error(update: Update, context: ContextTypes.DEFAULT_TYPE):
```

```
    print(f'Update {update} caused error {context.error}')
```

Функція, що відповідає за обробку взаємодій з кнопками і викликається при будь-якій взаємодії з кнопкою:

```
async def button(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    query = update.callback_query
    await query.answer()
    if query.data.startswith('Vertical') or query.data.startswith('Horizontal'):
        await recognise(update, context, query.data)
    else:
        await kanji_meaning(query.data, query, update)
```

Ця функція використовує іншу функцію для обробки даних отриманих після натискання кнопки – див. Додаток Е.

3.6 Висновки за розділом

Отже, на цьому завершується огляд скриптової частини додатку. Був розглянутий процес створення клієнт-серверного Телеграм бота для розпізнавання японських коміксів та надання довідки по словах. Для розробки такого бота були використані різні технології та інструменти, включаючи Python, Visual Studio Code, Telegram Bot API, dotenv, PIL, pytesseract, deep_translator, re, os, typing, jisho_api та pymongo.

Архітектура додатку включала серверну та клієнтську частини, а також використання бази даних MongoDB для збереження даних. За допомогою методики Long Polling клієнтська частина бота передавала оновлення серверній, що дозволяло реагувати на запити користувачів у реальному часі.

За допомогою Telegram Bot API було реалізовано взаємодію бота з платформою Telegram, що дозволяло користувачам взаємодіяти з ботом та отримувати необхідну інформацію.

РОЗДІЛ 4

РОЗРОБКА ТЕСТОВОГО НАБОРУ ТА ОЦІНКА ЕФЕКТИВНОСТІ БОТА

3.1 Побудова тестового набору зображень коміксів та позначення їх класів

Для тестування було обрано 4 види зображень з текстом японською:

– Звичайна манґа, тобто найпоширеніший вид коміксів. Приклади, що використовуватимуться наведені на рисунках 3.1 – 3.4:

Рис. 4.1 — Сторінка з манґи Oshi No Ko

Рис. 4.3 — Сторінка з манґи Vanitas no Carte

Рис. 4.2 — Сторінка з манґи Dr STONE

Рис. 4.4 — Сторінка з манґи Hunter x Hunter

– Дитяча манґа, тобто комікси з більш легкими словами для дітей. Приклади, що використовуватимуться наведені на рисунках 3.5 – 3.6:

Рис. 4.5 — Сторінка з манґи Doraemon

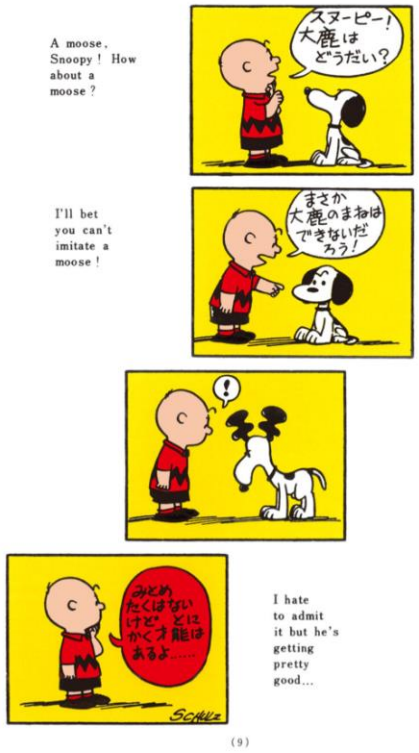

Рис. 4.6 — Сторінка з коміксу Peanuts

– Веб-сторінки – у багатьох випадках користувачам може знадобитись інформація з японських веб-сторінок, наприклад перегляд японських версій вже відомих веб-сторінок є хорошим методом вивчення мови. Приклади, що використовуватимуться наведені на рисунках 3.7 – 3.9:

Рис. 4.7 — Веб-сторінка BBC

Рис. 4.8 — Веб-сторінка прогнозу погоди від AccuWeather

Рис. 4.9 — Веб-сторінка Zoom

Рис. 4.10 — Веб-сторінка Wikipedia

3.2 Проведення тестування бота на тестовому наборі та оцінка ефективності

Далі буде проведено тестування розробленого бота на тестовому наборі зображень з попереднього пункту, а також оцінено його ефективність. Після успішного створення бота та побудови тестового набору зображень, настав час перевірити, наскільки добре він виконує свої функції і чи задовольняє вимоги проєкту.

При тестуванні розробленого бота на тестовому наборі будуть проведені такі кроки:

- Вирізання частини зображення з текстом: З кожного зображення тестового набору буде вирізана частина, яка містить текст для розпізнавання. Це допоможе зосередитися на самому важливому елементі для подальшого аналізу.
- Надсилання зображення боту: Вирізана частина зображення буде надіслана до бота, щоб отримати результат, який користувач побачив би при використанні бота. Бот буде виконувати розпізнавання тексту та надавати довідку по словах на зображенні.
- Аналіз результатів: Після отримання результату від бота буде проведений аналіз ефективності. Буде оцінено, наскільки точно бот розпізнає текст з

зображення та надає довідку по словах. Також буде виміряно час, який займає виконання розпізнавання та надання довідки.

Цей процес дозволить оцінити якість та ефективність розробленого бота на основі реальних тестових даних. Аналіз результатів допоможе виявити можливі проблеми, недоліки та зробити необхідні корективи для подальшого вдосконалення та покращення бота перед його фінальним використанням.

Результат отримання довідки по символу наведено у додатку Ж

- Канджі:
- 間
- Основні значення:
- Інтервал, простір
- Основні кун читання:
- あいだ, ま, あい
- Основні он читання:
- カン, ケン
- Для 間 「あいだ」(кун) є такі значення:
- пробіл (між), розрив, інтервал, відстань, розтягуватись, період часу (протягом поки), тривалість, інтервал, між (двома сторонами або речами), серед (групи), відносини (між), відносини, середня точка, середній, на півдорозі, середина, через, тому що
- Для 間中 「あいだじゅう」(кун) є такі значення:
- протягом
- Для 候間 「そうろうあいだ」(кун) є такі значення:
- як...
- Для ついこの間 「ついこのあいだ」(кун) є такі значення:
- буквально днями, зовсім недавно
- Для 間 「ま」(кун) є такі значення:
- час, пауза, простір, кімната
- Для 間際 「まぎわ」(кун) є такі значення:

- точка перед..., суть (робити), край (з), напередодні(безпосередньо перед), краєм
- Для 茶の間 「ちゃのま」(кун) є такі значення:
- Вітальня (в японському стилі).
- Для 床の間 「とこのま」(кун) є такі значення:
- токонома (альков, де виставлено мистецтво або квіти)
- Для 間 「あいだ」(кун) є такі значення:
- пробіл (між), розрив, інтервал, відстань, розтягуватись, період часу (протягом, поки), тривалість, інтервал, між (двома сторонами або речами), серед (групи), відносини (між), відносини, середня точка, середній, на півдорозі, середина, через, тому що
- Для 間中 「あいだじゅう」(кун) є такі значення:
- протягом
- Для 波間 「なみま」(кун) є такі значення:
- інтервал між хвилями, проміжок між хвилями
- Для 幕間 「まくあい」(кун) є такі значення:
- антракт (між діями), інтерлюдія
- Для 間 「カン」(он) є такі значення:
- інтервал, проміжок часу, серед, між, між-, хороша можливість, шанс, відчуження, розбрат, відчуження, шпигун, таємний агент
- Для 間隔 「カンカク」(он) є такі значення:
- простір, інтервал, космічний характер, пробіл
- Для 短期間 「タンキカン」(он) є такі значення:
- короткий термін, короткий час
- Для 右中間 「ウチュウカン」(он) є такі значення:
- між правим і центральним польовими гравцями (центр)
- Для 間 「ケン」(он) є такі значення:
- ken (6 shaku прибл. 1,818 м), лічильник, який використовується для нумерації проміжків між стовпами

– Для 間数 「ケンスウ」(он) є такі значення:
– кількість кен (в довжину або ширину)
– Для 世間 「セケン」(он) є такі значення:
– світ, суспільства, Люди, громадського
– Для 六百間 「ロッピャッケン」(он) є такі значення:
– roppyakken (type of game)

3.3 Висновки за розділом

Отже, результати тестування показали, що розпізнавання тексту зображень відбувається добре та достатньо швидко. Бот успішно розпізнає текст, якщо він має стандартний шрифт та чітко виділений на зображенні. Такі типи тексту, наприклад, друкований текст або текст із зрозумілими літерами, розпізнаються без проблем.

Однак, виникають певні проблеми у випадках, коли текст на зображенні є рукописним, має незвичний шрифт або колір. У таких ситуаціях точність розпізнавання може знизитися, і бот може зробити помилки або не розпізнати текст повністю. Це може бути пов'язано зі складністю визначення літер або незнайомими шрифтовими структурами.

З іншого боку, з довідкою по словах проблем немає. Бот успішно надає довідку по словах, які були розпізнані на зображенні, і виконує це безперебійно та швидко. Це свідчить про ефективність алгоритму обробки тексту та доступ до потрібної лексиконної бази даних.

Загалом, результати тестування свідчать про те, що бот ефективно виконує свої функції розпізнавання тексту та надання довідки. Проте, є потреба у подальшому вдосконаленні алгоритмів розпізнавання для поліпшення точності та здатності розпізнавати текст з незвичайними шрифтами або ускладненими умовами.

ВИСНОВКИ

При виконанні даної дипломної роботи була виконана розробка ефективного та надійного телеграм бота, який зможе розпізнавати японський текст та перекладати його на обрану мову. Крім того, було проведене створення бази даних перекладених ієрогліфів для вивчення їх методом повторення та тестування запам'ятовування. Результатом роботи став готовий телеграм бот та база даних ієрогліфів, які можуть бути використані для вивчення японської мови.

Переклад та інтерпретація іншомовного тексту, зчитування тексту з зображення, пошук інформації про кожний символ – усе це може здаватись дуже складною задачею. Однак, розумне використання різних бібліотек та інструментів може значно спростити ці задачі.

Згодом, для вдосконалення додатку можуть бути додані наступні функції:

- можливість вибору мов, з яких та на які перекладати;
- статистика правильних/неправильних відповідей на тести та відповідний до результатів підбір слів для тестування;
- можливість редагувати список слів;
- аналіз інших частин речення
- наведення довідки не тільки щодо використаних слів, а ще й щодо граматичної складової тексту
- можливість автоматичного вибору усього тексту зі сторінки коміксу та потім додавання перекладеного тексту на відповідні місця.

Під час аналізу доступних джерел було проведено дослідження поняття телеграм бот, під час якого була проведена класифікація. Додатково було складено алгоритм розробки бота. Були проаналізовані популярні засоби розробки. При аналізі існуючих розробок, було проведено їх порівняння і виділені їхні переваги і недоліки. В ході аналізу стало ясно, що при розробці бота варто звернути увагу на можливі відсутні значення деяких даних. Це потрібно для того, щоб зменшити кількість можливих помилок, і у результаті збільшити стійкість та зручність додатку.

Ґрунтуючись на отриманій в ході дослідження інформації, було вирішено розробити веб-чат на основі Telegram API з використанням бази даних MongoDB. Таке рішення було прийнято з двох причин:

- Telegram наразі є одним з найпопулярніших засобів спілкування у світі та має безкоштовний доступ до API.
- MongoDB також є безкоштовним, а також відсутність усталеної схеми таблиць надає можливість вільно записувати у базу даних дані будь-якої довжини та вмісту, що забезпечує легкість розробки та більш ефективну роботу з даними.

Після вибору засобів розробки було розпочато вивчення бібліотеки Telegram bot API, а також розробка самого проєкту. В ході розробки були отримані наступні знання та вміння:

- збереження та зчитування інформації у базі даних;
- використання API;
- налаштування серверної і клієнтської частин;
- робота з файлами у Телеграм та використання зображень у розробці;
- робота з типуванням у Python;
- використання бібліотеки dotenv.

Освоєння Python і бібліотек dotenv та Telegram bot API несе важливий характер, так як індустрія розробки спеціалізованих ботів все більше поширюється в нашому суспільстві. Месенджери перестали бути лише інструментом для комунікації, і тепер використовуються і у інших областях, наприклад, у сфері підтримки користувачів, онлайн чат ботів, що можуть відповідати на поширені питання та навіть використання технологій штучного інтелекту. Тому розвиток в даному напрямку можна вважати одним з найважливіших в сучасному суспільстві.

Таким чином, в ході реалізації проєкту були виконані наступні завдання:

- вивчено особливості і стан сучасної індустрії чат ботів у світі;
- обрано жанр, вид та платформа для бота;
- підготовлені необхідні для бота бібліотеки та засоби реалізації;
- реалізовано робочого бота.

ПЕРЕЛІК ПОСИЛАНЬ

1. Офіційна документація по Python [Електронний ресурс] / Режим доступу www. URL: https://docs.python.org/3/.
2. Офіційна документація по Telegram bot API [Електронний ресурс] / Режим доступу www. URL: https://core.telegram.org/bots/api.
3. Офіційна документація по dotenv [Електронний ресурс] / Режим доступу www. URL**:** https://github.com/motdotla/dotenv.
4. Офіційна документація по PIL [Електронний ресурс] / Режим доступу www. URL: https://omz-software.com/pythonista/docs/ios/PIL.html**.**
5. Офіційна документація по pytesseract [Електронний ресурс] / Режим доступу www. URL: https://github.com/madmaze/pytesseract.
6. Офіційна документація по Tesseract [Електронний ресурс] / Режим доступу www. URL: https://pytesseract.readthedocs.io/en/latest/.
7. Офіційна документація по deep_translator [Електронний ресурс] / Режим доступу www. URL: https://deep-translator.readthedocs.io/en/latest/.
8. Офіційна документація по jisho-api [Електронний ресурс] / Режим доступу www. URL: https://github.com/pedroallenrevez/jisho-api.
9. Офіційна документація по PyMongo [Електронний ресурс] / Режим доступу www. URL: https://pymongo.readthedocs.io/en/stable/.
10. Провотар О.І. Особливості та проблеми віртуального спілкування за допомогою чат-ботів / О.І. Провотар, Х.А. Клочко // Наукові праці ВНТУ: Інформаційні технології та комп'ютерна техніка. – 2013. – № 3. – 6 с.
11. Muldowney O. Chatbots. An Introduction And Easy Guide To Making Your Own O. Muldowney. – Dublin: Curses & Magic, 2017. – 69 с.
12. Історія чат ботів [Електронний ресурс] / Режим доступу www. URL: https://onlim.com/en/the-history-of-chatbots/.
13. What is a chatbot? [Електронний ресурс] / Режим доступу www. URL: https://www.ibm.com/topics/chatbots.
14. Telegram Bot Features [Електронний ресурс] / Режим доступу www. URL: https://core.telegram.org/bots/features.

15. Bots: An introduction for developers [Електронний ресурс] / Режим доступу www. URL: https://core.telegram.org/bots.
16. Tesseract [Електронний ресурс] / Режим доступу www. URL: https://uk.wikipedia.org/wiki/Tesseract.
17. Japanese Letter Pattern Recognition Application with Tesseract Engine [Електронний ресурс] / Режим доступу www. URL: https://www.researchgate.net/publication/338360169_Japanese_Letter_Pattern_Recognition_Application_with_Tesseract_Engine/fulltext/5e0f37baa6fdcc28375368a2/Japanese-Letter-Pattern-Recognition-Application-with-Tesseract-Engine.pdf.
18. A. R. Kardian, “Pengantar Pengolahan Citra,” in Pengolahan Citra Digital, Jakarta, 2012.
19. R. Smith, “Tesseract OCR engine What it is, where it came from,where it is going,” 2007.
20. E. and C. L. Tsujita, Mahir Bahasa Jepang dalam Sepekan. Jakarta: Kesaint Blanc, 2005.
21. A. Hasnan, “Pengantar Belajar Bahasa Jepang,” 2009. . [5] S. Center, “Hiragana,” 2006.
22. Ласкаво просимо до документації aiogram! [Електронний ресурс]. URL: https://docs.aiogram.dev/en/latest/
23. Чат-бот для бізнесу: плюси й мінуси впровадження роботів [Електронний ресурс]. www. URL: https://nikopolnews.net/chat-bot-dlya-biznesupljusi-j-minusi-vprovadzhennya-robotiv/
24. Використання мови Python для розробки науково-технічного програмного забезпечення [Електронний ресурс]. www. URL: https://dou.ua/lenta/articles/python-for-science

ДОДАТОК А

ЗАПИТ ДО СЛОВНИКА ЯПОНСЬКОЇ МОВИ JISHO

```
        response =
Kanji.request(symbol).json()
            response_json =
json.loads(response)
            kanji =
value(response_json["data"]["kanji"])
            main_meanings =
value(response_json["data"]["main_meanings
"])
            main_readings_kun =
value(response_json["data"]["main_readings"
]["kun"])
            main_readings_on =
value(response_json["data"]["main_readings"
]["on"])
            ex_readings_kun =
value(response_json["data"]["reading_exampl
es"]["kun"])
            ex_readings_kun_m = ""
            for i in
range(len(ex_readings_kun)):
                ex_readings_kun_m =
ex_readings_kun_m + "\n\nДля " +
ex_readings_kun[i]["kanji"] + " 「" +
ex_readings_kun[i]["reading"] + "」(кун) є
такі значення:"
                for m in
ex_readings_kun[i]["meanings"]:
                    if is_a_valid_symbol(m):
                        m =
GoogleTranslator(source='en',
target='uk').translate(m)
                    ex_readings_kun_m =
ex_readings_kun_m + "\n" + m
            ex_readings_on =
value(response_json["data"]["reading_exampl
es"]["on"])
            ex_readings_on_m = ""
            for i in range(len(ex_readings_on)):
                ex_readings_on_m =
ex_readings_on_m + "\n\nДля " +
ex_readings_on[i]["kanji"] + " 「" +
ex_readings_on[i]["reading"] + "」(он) є такі
значення:"
                for m in
ex_readings_on[i]["meanings"]:
                    if is_a_valid_symbol(m):
                        m =
GoogleTranslator(source='en',
target='uk').translate(m)
                    ex_readings_on_m =
ex_readings_on_m + "\n" + m
            main_meanings_reply = ""
            main_readings_kun_reply = ""
            main_readings_on_reply = ""
            tr_meanings = []
            for i in main_meanings:
                if is_a_valid_symbol(m):
                    i =
GoogleTranslator(source='en',
target='uk').translate(i)
                tr_meanings.append(i)
                main_meanings_reply =
main_meanings_reply + "\n" + i
            for i in main_readings_kun:
                main_readings_kun_reply =
main_readings_kun_reply + "\n" + i
            for i in main_readings_on:
                main_readings_on_reply =
main_readings_on_reply + "\n" + i
```

ДОДАТОК Б

ФОРМУВАННЯ ЗМІСТУ ТЕСТУ

```
        results = list(find_words(update))
        n = len(results)
        right = random.randint(0, n-1)
            question = "Оберіть значення " +
results[right]["kanji"]
            correct_options =
results[right]["main meanings"]
            n2 = len(correct_options)
            cor_op_id = random.randint(0, n2-
1)
            correct_option =
correct_options[cor_op_id]
            all_options = []
            for a in results:
                for b in a["main meanings"]:
                    all_options.append(b)
            n1 = len(all_options)
            options = []
            for i in range(3):
                option_id = random.randint(0,
n1-1)
                option = all_options[option_id]
                options.append(option)
            correct_id = random.randint(0, 3)
            options.insert(correct_id,
correct_option)
```

ДОДАТОК В

СТВОРЕННЯ НАДИСЛАННЯ ТЕСТІВ ПО СЛОВАХ

```
async def translate_command(update: Update,
context:ContextTypes.DEFAULT_TYPE):
    results = list(find_words(update))
    if not results:
        await update.message.reply_text("Не знайдено жодних вивчених слів, за допомогою зверніться до команди /help")
        return
    n = len(results)
    if n<4:
        await update.message.reply_text("Виявлена недостатня кількість слів, будь ласка, додайте хоча б 4 слова до вашого спику\nЯкщо виникають проблеми зверніться до /help")
        return
    right = random.randint(0, n-1)
    question = "Оберіть значення " + results[right]["kanji"]
    correct_options = results[right]["main meanings"]
    n2 = len(correct_options)
    cor_op_id = random.randint(0, n2-1)
    correct_option = correct_options[cor_op_id]
    all_options = []
    for a in results:
        for b in a["main meanings"]:
            all_options.append(b)
    n1 = len(all_options)
    options = []
    for i in range(3):
        option_id = random.randint(0, n1-1)
        option = all_options[option_id]
        options.append(option)
    correct_id = random.randint(0, 3)
    options.insert(correct_id, correct_option)
    print(f"Correct option: {correct_option}")
    await update.message.reply_poll(type="quiz", question=question, options=options, correct_option_id=correct_id)
```

ДОДАТОК Г

ОБРОБКА ОТРИМАНОГО ЗОБРАЖЕННЯ

```
        async def translate_msg(update:
Update, context:
ContextTypes.DEFAULT_TYPE):
            if (
                not update.message
                or not update.effective_chat
                or (
                   not update.message.photo
                   and not
update.message.video
                   and not
update.message.document
                   and not
update.message.sticker
                   and not
update.message.animation
                )
              ):
                return
            file = await downloader(update,
context)
            if not file:
                await
update.message.reply_text("щось пішло не
так, спробуйте ще раз")
                return

            vert_b=[]

vert_b.append(InlineKeyboardButton("Верти
кальний", callback_data="Vertical"+ ",,"+
str(file)))

vert_b.append(InlineKeyboardButton("Гориз
онтальний", callback_data="Horizontal"+
",,"+ str(file)))
            vert_k=[]
            vert_k.append(vert_b)

vert_r=InlineKeyboardMarkup(vert_k)
            await
update.message.reply_text("Текст на
зображенні вертикальний чи
горизонтальний?",reply_markup=vert_r)
```

ДОДАТОК Е

ОБТРИМАННЯ ДОВІДКИ ПО КАНДЖІ

```
        async def kanji_meaning(symbol: str, query: CallbackQuery, update:Update):
            await query.message.reply_text("Запит виконується, зачекайте")
            response = Kanji.request(symbol).json()
            response_json = json.loads(response)
            kanji = value(response_json["data"]["kanji"])
            main_meanings = value(response_json["data"]["main_meanings"])
            main_readings_kun = value(response_json["data"]["main_readings"]["kun"])
            main_readings_on = value(response_json["data"]["main_readings"]["on"])
            ex_readings_kun = value(response_json["data"]["reading_examples"]["kun"])
            ex_readings_kun_m = ""
            for i in range(len(ex_readings_kun)):
                ex_readings_kun_m = ex_readings_kun_m + "\n\nДля " + ex_readings_kun[i]["kanji"] + " 「" + ex_readings_kun[i]["reading"] + "」 (кун) є такі значення:"
                for m in ex_readings_kun[i]["meanings"]:
                    if is_a_valid_symbol(m):
                        m = GoogleTranslator(source='en', target='uk').translate(m)
                    #t = GoogleTranslator(source='en', target='uk').translate(m)
                    ex_readings_kun_m = ex_readings_kun_m + "\n" + m
            ex_readings_on = value(response_json["data"]["reading_examples"]["on"])
            ex_readings_on_m = ""
            for i in range(len(ex_readings_on)):
                ex_readings_on_m = ex_readings_on_m + "\n\nДля " + ex_readings_on[i]["kanji"] + " 「" + ex_readings_on[i]["reading"] + "」 (он) є такі значення:"
                for m in ex_readings_on[i]["meanings"]:
                    if is_a_valid_symbol(m):
                        m = GoogleTranslator(source='en', target='uk').translate(m)
                    #t = GoogleTranslator(source='en', target='uk').translate(m)
                    ex_readings_on_m = ex_readings_on_m + "\n" + m
            main_meanings_reply = ""
            main_readings_kun_reply = ""
            main_readings_on_reply = ""
            tr_meanings = []
            for i in main_meanings:
                if is_a_valid_symbol(m):
                    i = GoogleTranslator(source='en', target='uk').translate(i)
                    tr_meanings.append(i)
                    main_meanings_reply = main_meanings_reply + "\n" + i
            for i in main_readings_kun:
                main_readings_kun_reply = main_readings_kun_reply + "\n" + i
            for i in main_readings_on:
                main_readings_on_reply = main_readings_on_reply + "\n" + i
            kanji_data = {
                "kanji" : kanji,
                "main meanings" : tr_meanings,
                "user": query.from_user.id
            }
            insert_into_doc(kanji_data, update)
            await query.message.reply_text("Канджі:\n" + kanji + "\n\nОсновні значення:" + main_meanings_reply + "\n\nОсновні кун читання:" + main_readings_kun_reply + "\n\nОсновні он читання:" + main_readings_on_reply + ex_readings_kun_m +  ex_readings_on_m)
```

ДОДАТОК Д

НАДАННЯ ІНФОРМАЦІЇ ПО ТЕКСТУ, ЙОГО ПЕРЕКЛАД ТА СТВОРЕННЯ КНОПОК

```
async def recognise(update: Update, context: ContextTypes.DEFAULT_TYPE, mode:str):
    s=mode.split(",,")
    type=s[0]
    file=s[1]
    image = Image.open(file)
    if type=="Vertical":
        text = pytesseract.image_to_string(image, config='-l jpn_vert --oem 1 --psm 1')
    else:
        text = pytesseract.image_to_string(image, config='-l jpn --oem 1 --psm 1')
    if not text:
        await update.callback_query.message.reply_text("Щось пішло не так, спробуйте ще раз")
        return
    new_text = re.sub(r"[\n\r]+", " ", text).strip()
    translated_text = GoogleTranslator(source='ja', target='uk').translate(new_text)
    kanji_list = []
    for i in new_text:
        if is_kanji(i):
            kanji_list.append(i)
    no_dupes_kanji = []
    for i in kanji_list:
        if i not in no_dupes_kanji:
            no_dupes_kanji.append(i)
    new_kanji_list = list(chunks(no_dupes_kanji, 8))
    keyboard = []
    for a in new_kanji_list:
        temp_list = []
        for b in a:

            temp_list.append(InlineKeyboardButton(b, callback_data=b))
        keyboard.append(temp_list)
    reply_markup = InlineKeyboardMarkup(keyboard)

    await update.callback_query.message.reply_text(new_text, reply_markup=reply_markup)
    await update.callback_query.message.reply_text(translated_text)
```

ДОДАТОК Ж

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Рис. 1

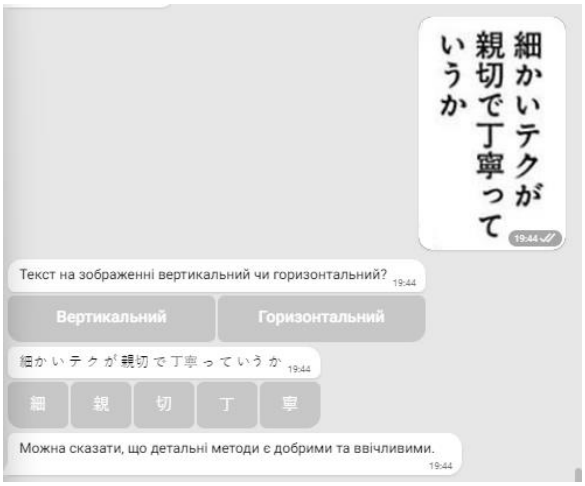

Рис. 2

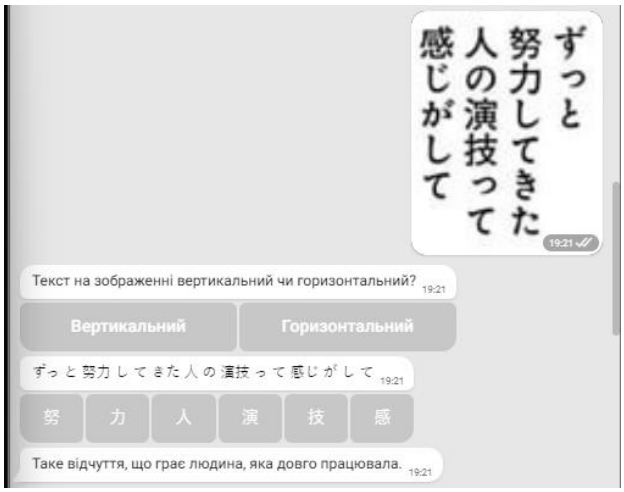

Рис. 3

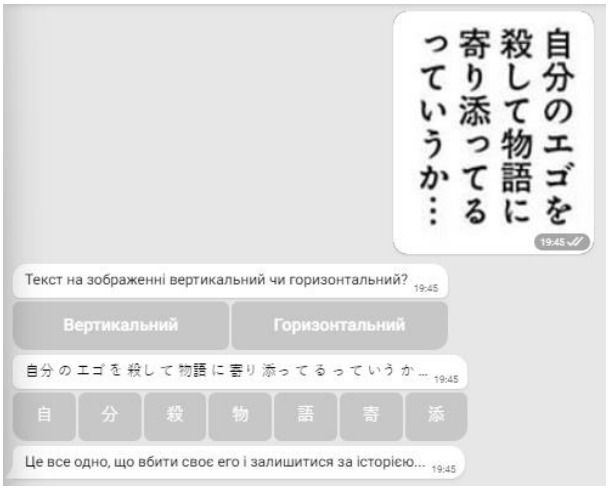

Рис. 4

Рис. 5

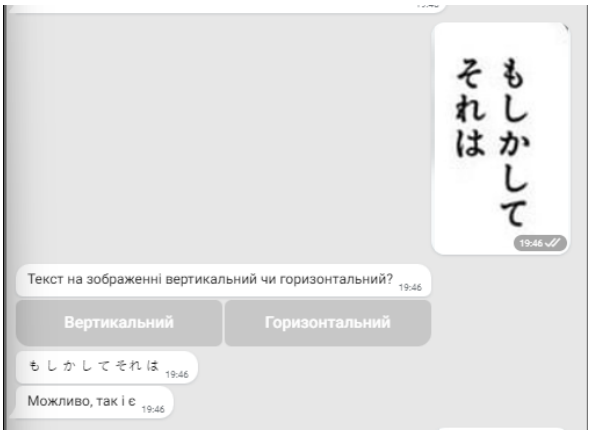

Рис. 6

Рис. 7

Рис. 8

Рис. 9

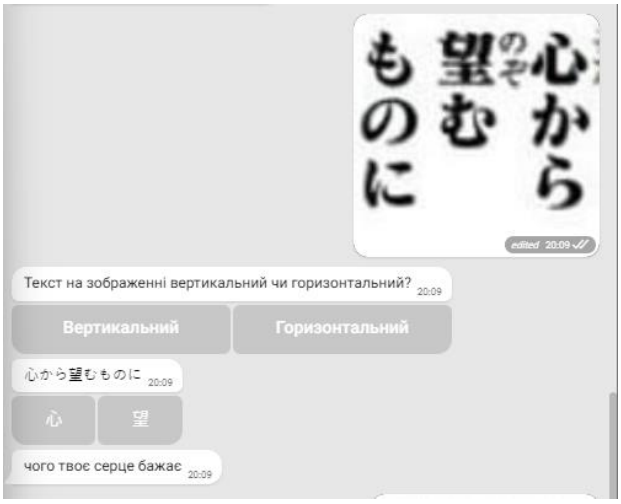

Рис. 10

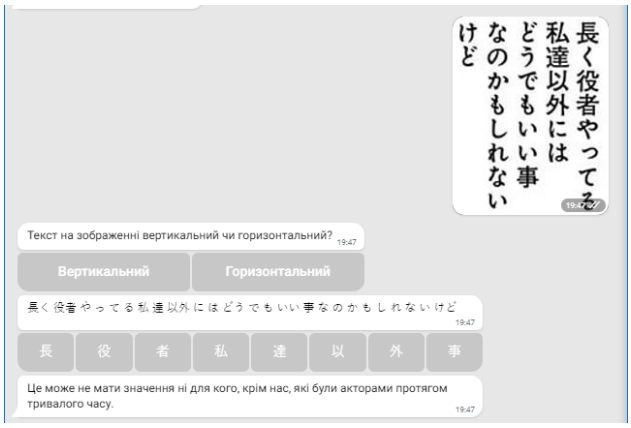

Рис. 11

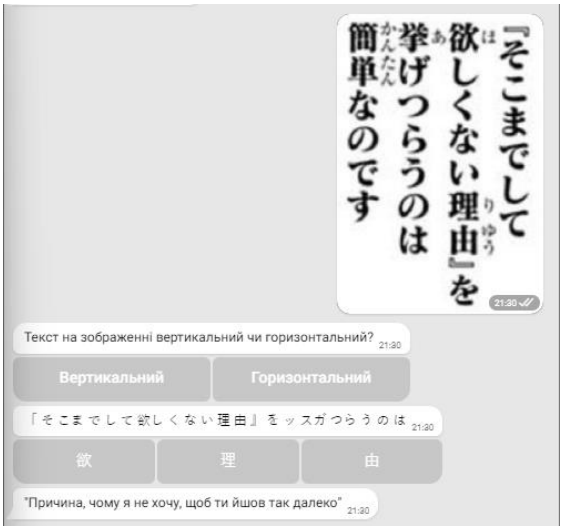

Рис. 12

Рис. 13

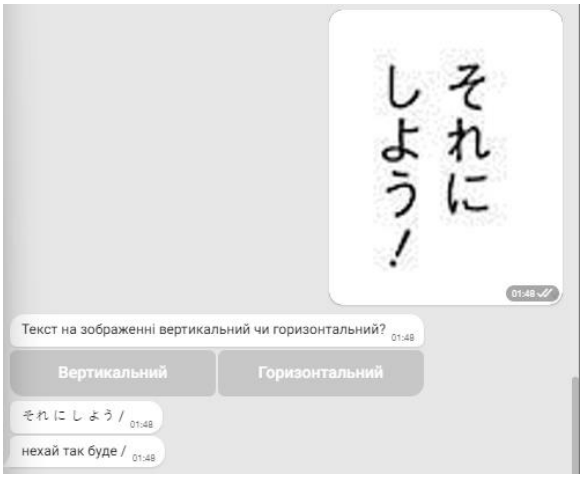

Рис. 14

Рис. 15

Рис. 16

Рис. 17

Рис. 18

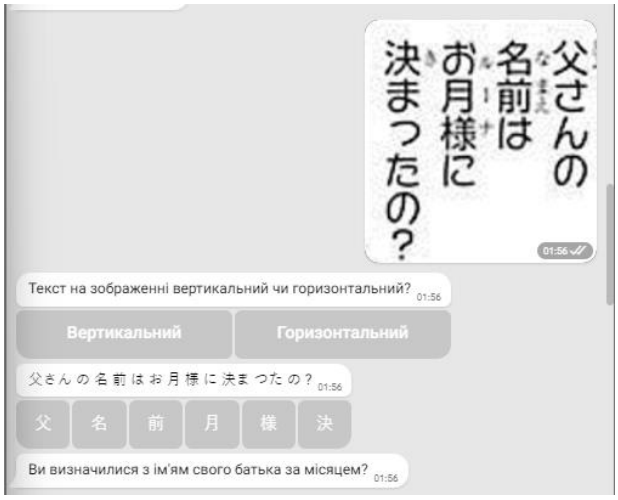

Рис. 19

Рис. 20

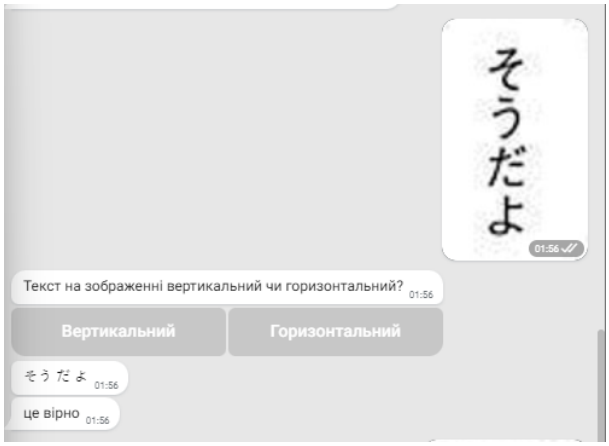

Рис. 21

Рис. 22

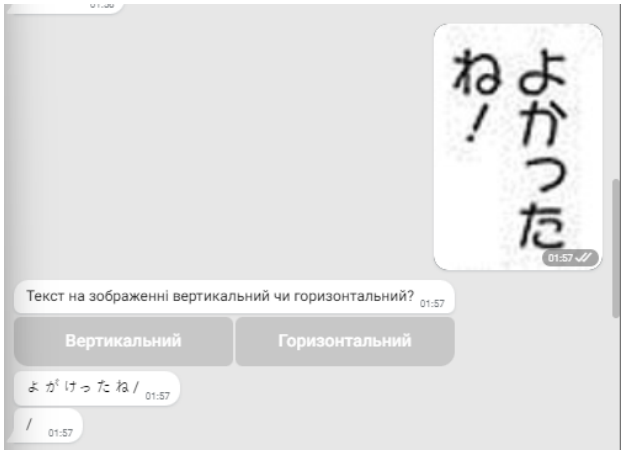

Рис. 23

Рис. 24

Рис. 25

Рис. 26

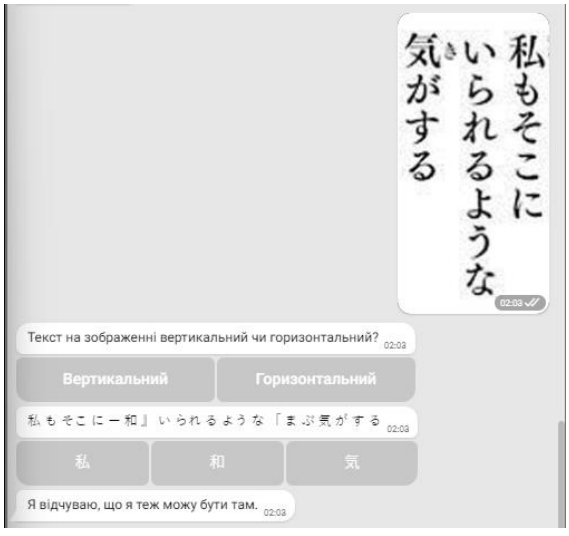

Рис. 28

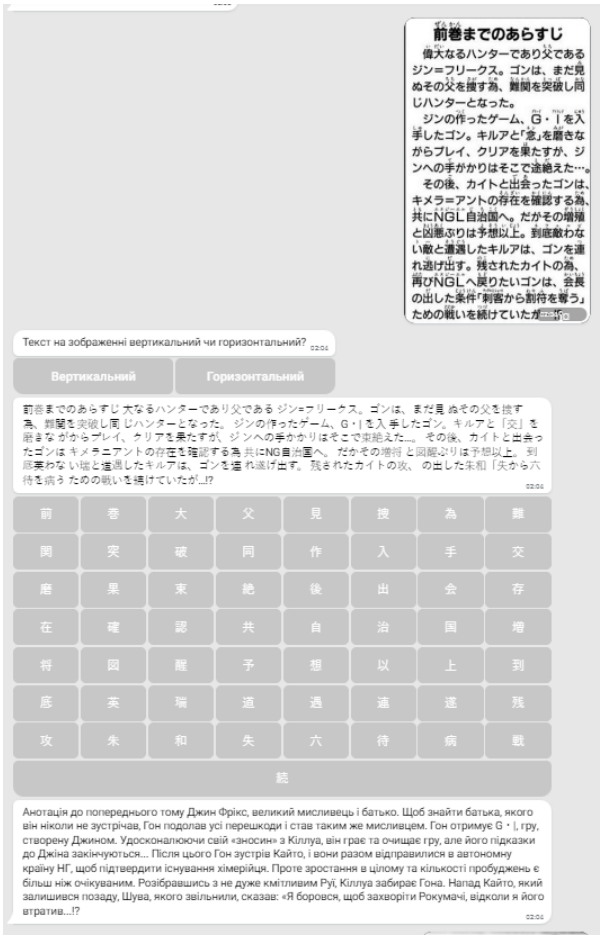

Рис. 27

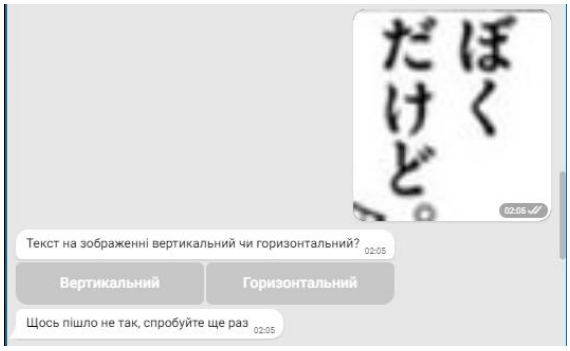

Рис. 29

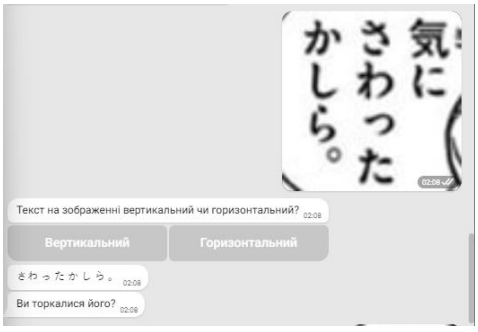

Рис. 30

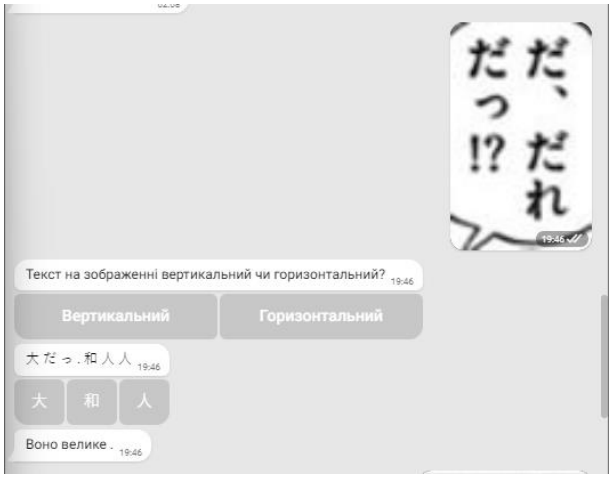

Рис. 31

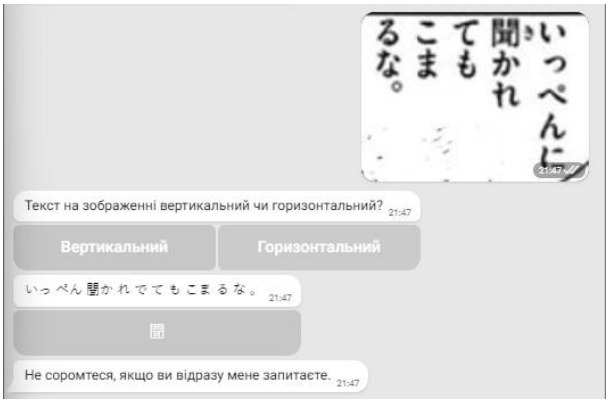

Рис. 32

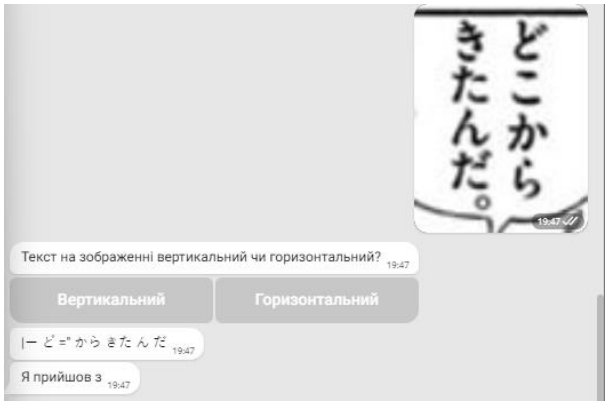

Рис. 34

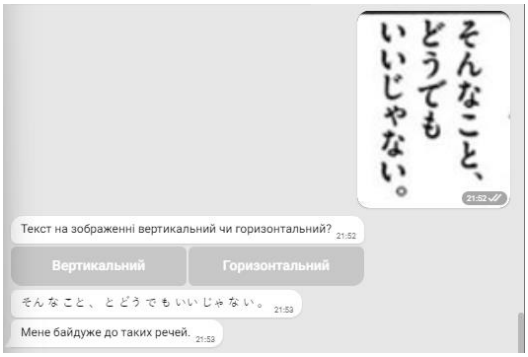

Рис. 33

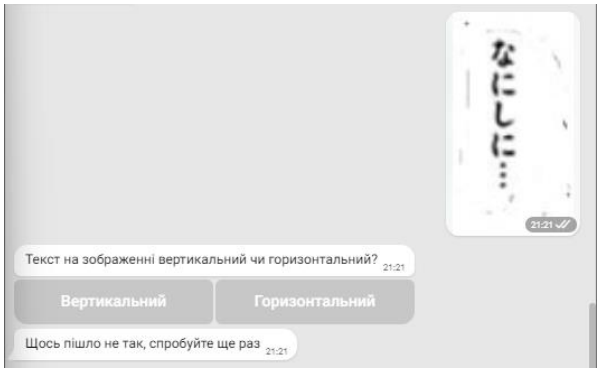

Рис. 36

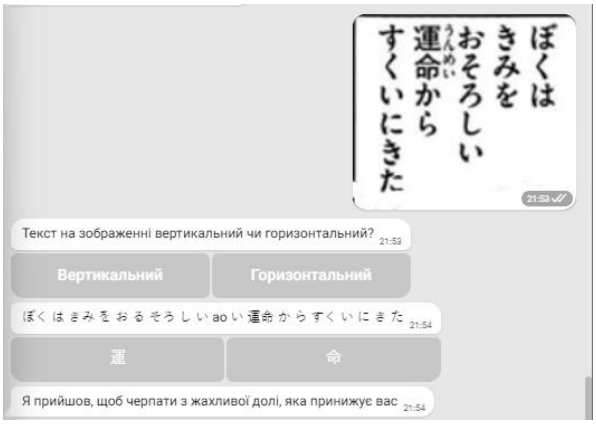

Рис. 35

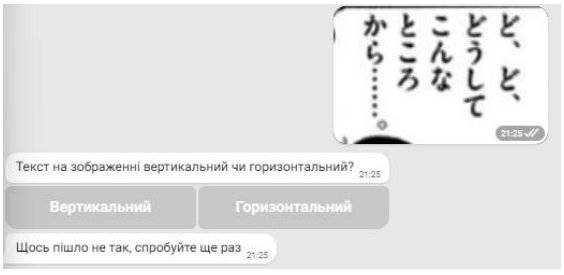

Рис. 37

Рис. 38

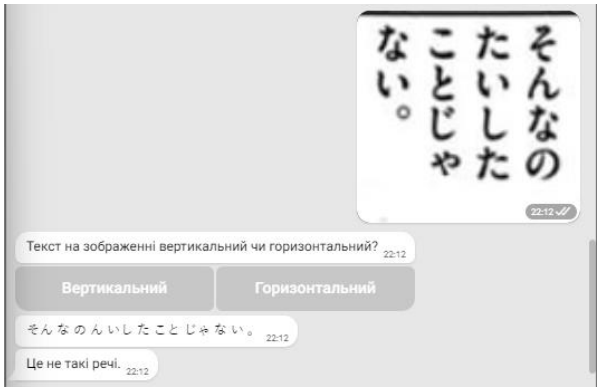

Рис. 39

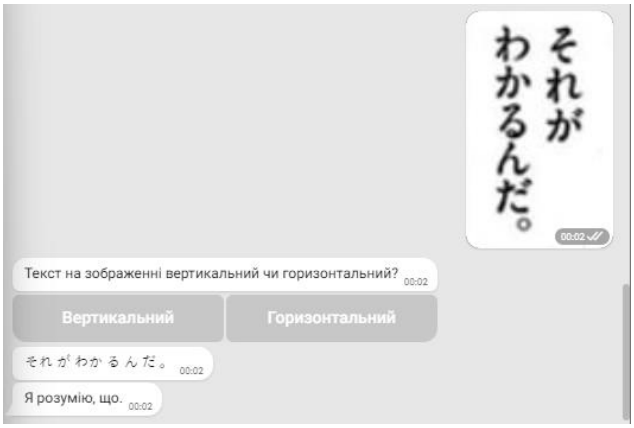

Рис. 40

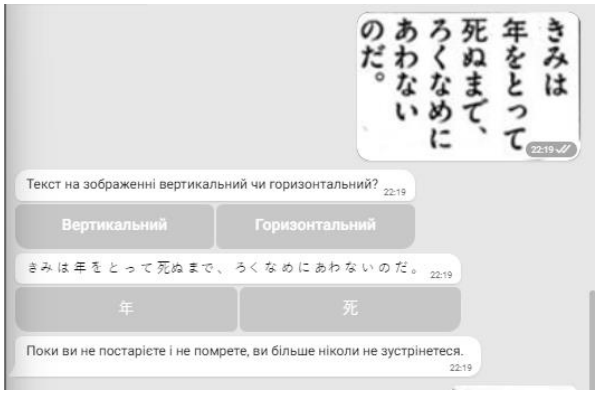

Рис. 41

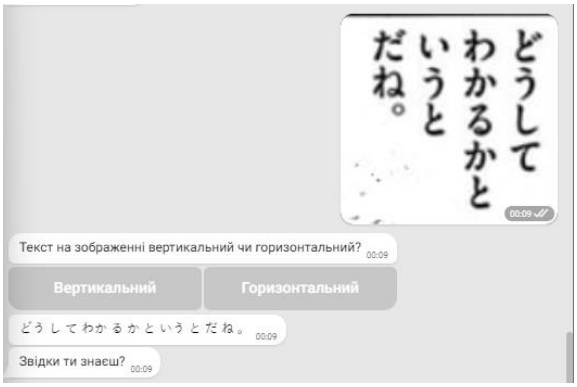

Рис. 42

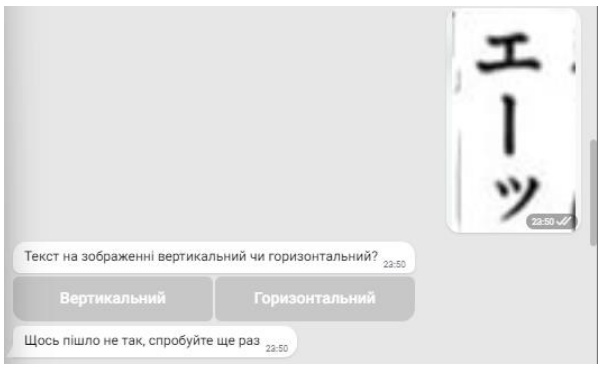

Рис. 43

Рис. 44

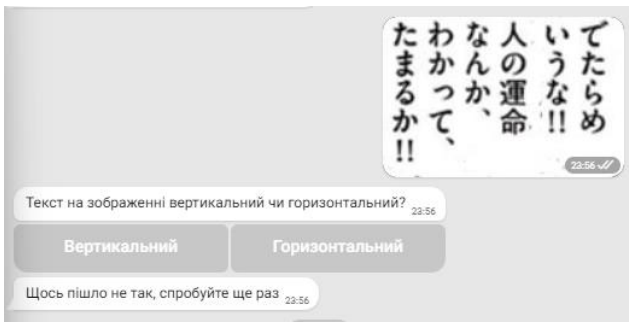

Рис. 45

Рис. 46

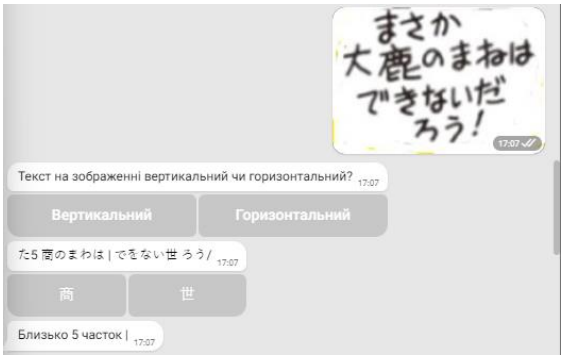

Рис. 47

Рис. 48

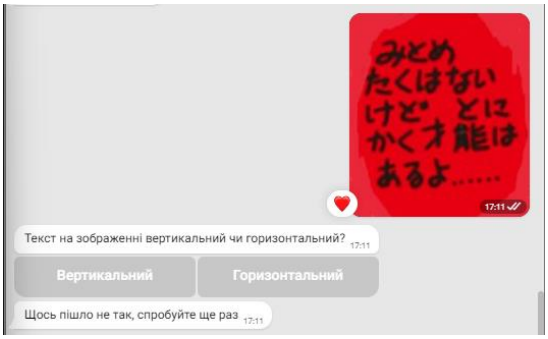

Рис. 49

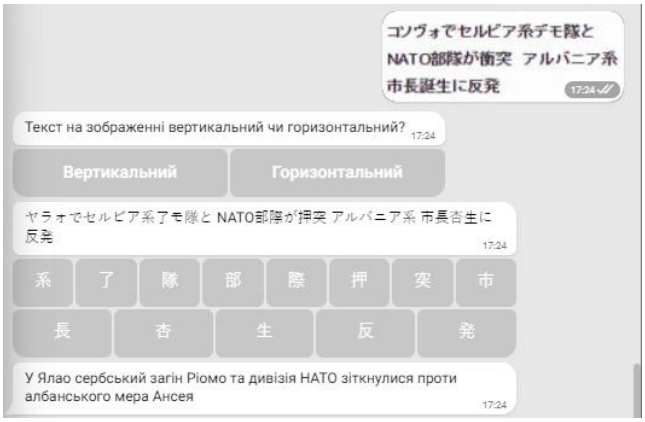

Рис. 50

Рис. 51

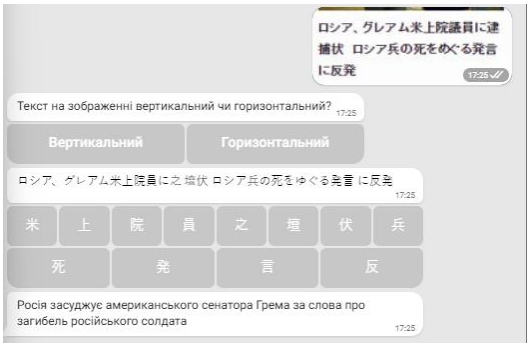

Рис. 52

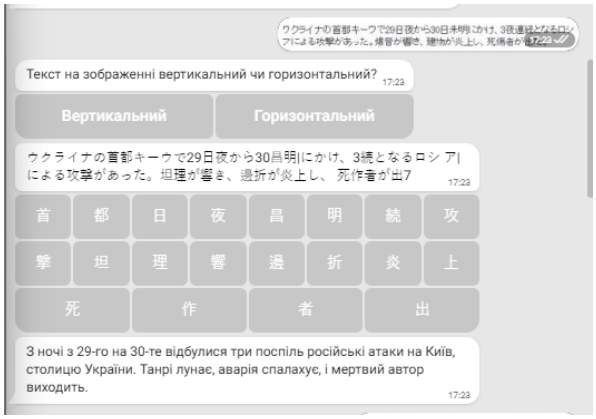

Рис. 53

Рис. 54

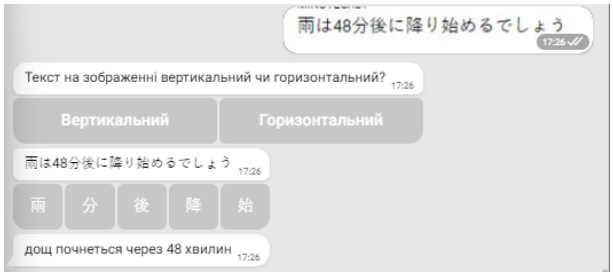

Рис. 55

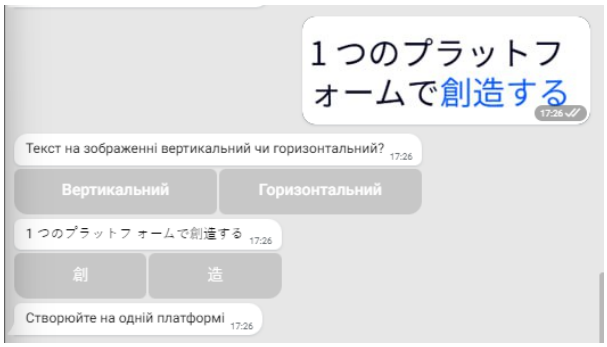

Рис. 56

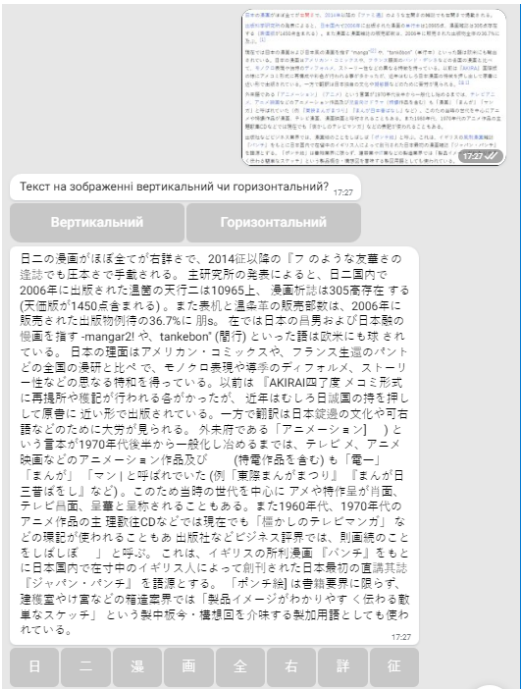

Рис. 57

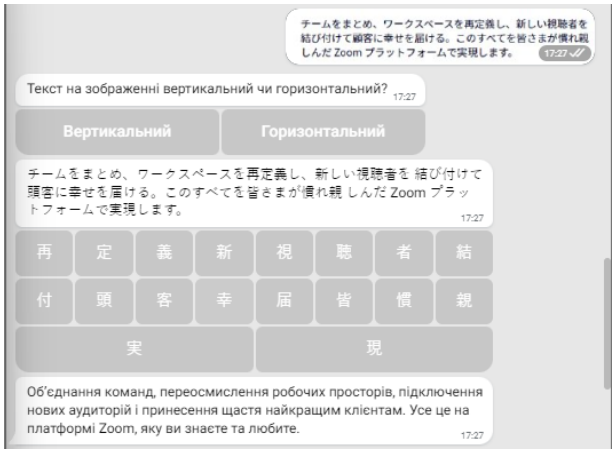

Рис. 58

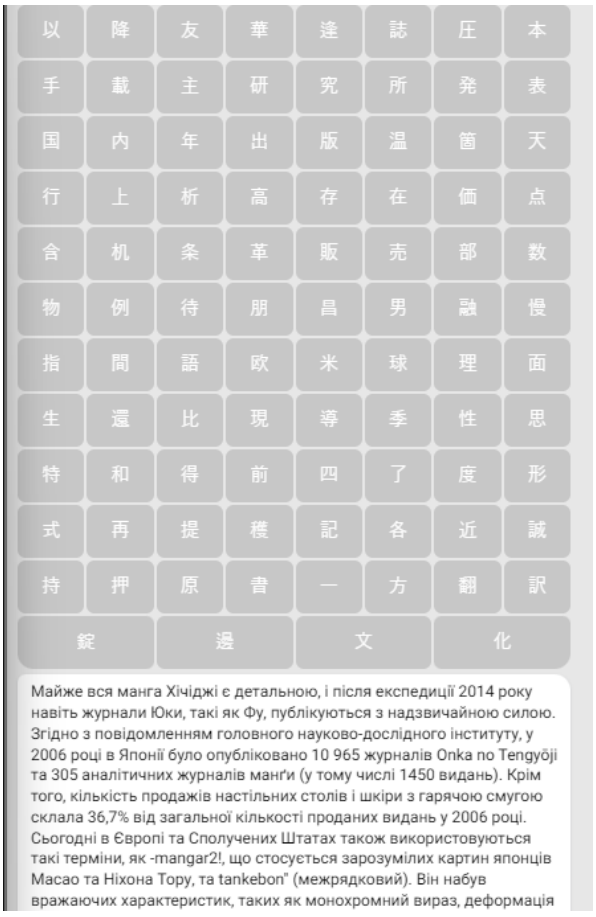

Рис. 59