Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Система автоматичного перекладу української жестової мови в текст

(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-5

Дарина Зайцева
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект

(повна назва освітньої програми)
Керівник ас. Дмитро Водяницький
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Зайцевій Дарині Михайлівні
(прізвище, ім'я, по батькові)

1. Тема роботи Система автоматичного перекладу української жестової мови в текст

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 червня 20 25 р.

3. Вихідні дані до роботи Мови програмування C++, Python, Qt/Qml; бібліотки TensorFlow, MediaPipe, OpenCV; середа розробки Visual Studio Code, система керування базами даних Portgress, онлайн-сервіс розробки інтерфейсів Figma, генератор сценаріїв складання CMake, Відомості про існуючі рішення

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Проєктування системи
3) Вибір технологій та програмних засобів
4) Розробака додатку

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	19.05.2025 – 21.05.2025	виконано
3	Аналіз існуючих рішень	22.05.2025 – 23.05.2025	виконано
4	Постановка задачі	24.05.2025 – 25.05.2025	виконано
5	Визначення функціональних вимог	26.05.2025 – 27.05.2025	виконано
6	Вибір та проєктування архітектури застосунку	28.05.2025 – 31.06.2025	виконано
7	Проєктування бази даних	01.06.2025 – 02.06.2025	виконано
8	Вибір технологій та програмних засобів для	03.06.2025 – 05.06.2025	виконано
	реалізації		
9	Розробка застосунку	05.06.2025 – 07.06.2025	виконано
10	Оформлення пояснювальної записки	08.06.2025 – 09.06.2025	виконано
11	Попередній захист	10.06.2025	виконано
12	Захист перед ДЕК	18.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Дмитро Водяницький
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 61 с., 18 рис., 1 табл., 3 дод., 23 джерела.

ЖЕСТОВА МОВА, КОМП’ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ, МОВЛЕННЯ, ПЕРЕКЛАД, ТЕКСТ, УКРАЇНСЬКА ЖЕСТОВА МОВА, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – переклад української жестової мови у текст.

Предмет дослідження – процес автоматичного розпізнавання жестів та перетворення їх у текстовий та голосовий формат.

Метою кваліфікаційної роботи є розробка підходу до автоматизованого перекладу жестів української жестової мови у текст шляхом застосування сучасних методів комп'ютерного зору, машинного навчання та глибинних нейронних мереж, з урахуванням граматичних, лексичних і семантичних відмінностей між жестовою та українською усною/письмовою мовами.

Методи дослідження – експериментальне моделювання системи перекладу української жестової мови у текст на основі підходу, що поєднує методи комп’ютерного зору, обробки відеопотоку, алгоритми машинного та глибинного навчання, з подальшим аналізом точності.

Кваліфікаційна робота включає дослідження предметної галузі, огляд та аналіз переваг/недоліків існуючих рішень, визначення вимог до власного застосунку, чітке формулювання задачі, визначення функціональних вимог, вибір архітектури проєкту, опис обраних інструментів, проєктування та розробку додутку, навчання моделі перекладу української жестової мови. Запропоноване рішення вирізняється підтримкою української жестової мови, функціонуванням без використання додаткових сенсорів та можливістю перетворення жестів в текст та звук.

ABSTRACT

Bachelor’s thesis contains: 61 pp., 18 fig., 1 tabl., 3 ann., 23 references.

ARTIFICIAL INTELLIGENCE, COMPUTER VISION, MACHINE LEARNING, SIGN LANGUAGE, SPEECH, TEXT, TRANSLATION, UKRAINIAN SIGN LANGUAGE.

Object of the research – translation of Ukrainian Sign Language into text.

Subject of the research – the process of automatic gesture recognition and conversion into textual and voice format.

The aim of this qualification work is to develop an approach for the automated translation of Ukrainian Sign Language gestures into text by applying modern methods of computer vision, machine learning, and deep neural networks, taking into account grammatical, lexical, and semantic differences between sign language and spoken/written Ukrainian.

Research methods – experimental modeling of a Ukrainian Sign Language translation system based on an approach that combines computer vision techniques, video stream processing, machine learning, and deep learning algorithms, followed by accuracy analysis.

The qualification work includes research of the subject domain, a review and analysis of the advantages and disadvantages of existing solutions, definition of requirements for the custom application, clear formulation of the task, identification of functional requirements, selection of the project architecture, description of selected tools, design and development of the application, and training of the Ukrainian Sign Language translation model.
The proposed solution is distinguished by its support for Ukrainian Sign Language, operation without additional sensors, and the ability to convert gestures into both text and speech.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АЖМ – американська жестова мова;

БД – база даних;

ОС – операційна система;

СУБД – система управління базами даних;

УЖМ – українська жестова мова;

ACID – Atomicity, Consistency, Isolation, Durability – атомарність, узгодженість, ізоляція, довговічність;

LSTM – Long Short-Term Memory – довга короткочасна пам'ять;

SQL – Structured Query Language – структурована мова запитів;

VS Code – Visual Studio Code – засіб для створення та редагування програмного коду;

1D-CNN – One-Dimensional Convolutional Neural Network – одновимірна згорткова нейронна мережа.

ВСТУП

У сучасному суспільстві важливим напрямом є створення інклюзивних технологій, що послаблюватимуть бар'єри у спілкуванні між людьми з особливими потребами та рештою суспільства. Відсутність ефективних засобів перекладу української жестової мови ускладнює доступ до послуг освіти, працевлаштування та соціальної взаємодії.

Жестова мова – це повноцінний засіб комунікації для мільйонів осіб із порушеннями слуху. Але бар'єри між тими, хто нею володіє, та іншими є серйозною соціальною проблемою. Штучний інтелект, машинне навчання та комп'ютерний зір дають нові можливості для створення інструментів автоматичного перекладу жестової мови. Це сприяє інклюзивності, рівному доступу до інформації та можливостей. Розробка системи перекладу української жестової мови (УЖМ) – це не тільки інженерна задача, а й важлива соціальна місія. Більшість наявних рішень зорієнтовані на американську або міжнародну жестову мову, ігноруючи потреби українських користувачів.

В межах даного проєкту поставлено задачу розробити автономний застосунок, здатний розпізнавати жести з використанням камери та перетворювати їх на текст чи голос. Акцентовано, що проєкт не потребує дорогих пристроїв, спеціальних рукавиць чи сенсорів. Це потенційно розширює коло користувачів. Під час реалізації особливу увагу зосереджено не тільки на технічних аспектах, але й на дослідженні предметної галузі, аналізі існуючих рішень, обґрунтуванні вибору архітектури, інтерфейсів, мов програмування та бібліотек. Таким чином, створення такої системи має наукову та практичну цінність для розвитку інклюзивних технологій в Україні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

Системи автоматичного перекладу жестової мови текст – це один з напрямків досліджень у галузі штучного інтелекту, комп'ютерного зору та машинного навчання. Основною метою таких систем є розробка технологій автоматичного розпізнавання жестів і перекладу їх у текст. Це сприятиме інтеграції осіб з вадами слуху у соціальну та професійну діяльності, а також сприятиме покращенню здатності спілкуватися з людьми, які не користуються жестовою мовою.

Соціальна інтеграція людей з порушеннями слуху є важливим чинником забезпечення інклюзивності та рівних можливостей у сучасному суспільстві. Відсутність ефективних засобів перекладу української жестової мови ускладнює доступ до послуг освіти, працевлаштування та соціальної взаємодії. Автоматизовані системи перекладу жестової мови можуть подолати ці бар'єри, надаючи можливість перекладати жести в текст та/або аудіо, яке може зрозуміти людина, що не володіє жестовою мовою.

Перші системи перекладу жестової мови в текст були засновані на механічному та електронному розпізнаванні жестів рук. Однак такі системи були неточними через складність різних жестів та швидкості руху. Точність розпізнавання також обмежувалася відсутністю великих масивів даних (системи зазвичай були навчені лише на алфавіті жестів) та ефективних алгоритмів [1].

Пізніше, щоб удосконалити ці системи, для аналізу жестів почали використовували камери та сенсори. Такі системи могли фіксувати тривимірні жести, але їм бракувало гнучкості та адаптивності до різних умов, таких як освітлення, фізіологічні особливості (розмір рук, форма пальців, колір шкіри, тощо). Також системи мали проблеми з фоном та можливими рухами об’єктів на задньому плані [1].

Перші системи, що використовують комп'ютерний зір і алгоритми машинного навчання для розпізнавання жестів, з'явилися в 2010-х роках. Точність розпізнавання таких моделей є кращою за рахунок навчання на великих масивах даних. Такі системи, на відміну від їх попередників, здатні розпізнавати не лише літери, але й цілі слова та речення. Наприклад, Google MediaPipe [2] пропонує потужний інструмент розпізнавання жестів у реальному часі, який використовується в сучасних перекладацьких проєктах [3].

1.2 Аналіз існуючих рішень

1.2.1 Система SingAll

Одним з найбільш відомих, вже існуючих рішень, є система від компанії SingAll [4], що зосереджена на перекладі американської жестової мови (АЖМ) в текст.

Проєкт був заснований у 2016 році як стартап, що мав на меті створити систему, яка здатна перекладати мову жестів в текст у реальному часі. Спочатку для відстеження руху використовували спеціальні рукавички з кольоровими маркерами, але розвиток технологій MediaPipe дозволив перейти на використання звичайних камер, які не потребують рукавичок.

На рисунку 1.1 зображено інтерфейс програми [5]. У центрі камери розміщена людина, що говорить мовою жестів. Унизу знаходиться білий фон, на якому з'являється текст. Зліва бачимо перемикач режиму Fingerspelling, що вмикається за потреби сказати число чи ім’я, власні назви або інші слова, що потребують говоріння по буквах. Зправа бачимо позначку повідомлення – це кнопка, яку використовує той, що говорить для позначення закінчення свого повідомлення. Після завершення говоріння, з метою виключення помилок, система перепитує людину, надаючи варіанти сказаного нею (рисунок 1.2).

Рисунок 1.1 – Інтерфейс SingAll (1)

Рисунок 1.2 – Інтерфейс SingAll (2)

Система SingAll має високу точність завдяки використанню декількох камер і глибокому машинному навчанню, інтегрована з MediaPipe, тому не потребує спеціальних рукавичок, також може використовуватися як для перекладу, так і для навчання жестової мови, але система орієнтована на АЖМ, а інші жестові мови, включаючи українську, поки що не підтримує.

Також система не фокусується на перетворені жестів у звук що ускладнює спілкування з людьми, що не бачать.

1.2.2 Система KinTrans

Ще однією, не менш розповсюдженою системою є KinTrans [6].

KinTrans була заснована в 2013 році і використовує датчики руху для розпізнавання мови жестів і перетворення її в текст або звук в режимі реального часу. Система підтримує декілька мов жестів, таких як арабська, англійська та китайська. Метою системи, що розглядається є покращення комунікації між нечуючими та чуючими людьми у сфері обслуговування, освіти та охорони здоров'я.

Технологія KinTrans особливо популярна в Північному Техасі та на Близькому Сході, де компанія працює з бізнесом над впровадженням своїх рішень. Система використовує датчики на кшталт Microsoft Kinect, що дозволяє виявити навіть найменший рух руки або пальців, що робить її надзвичайно точною.

На рисунку 1.3 бачимо інтерфейс KinTrans. По середині розміщена людина, що говорить мовою жестів, зліва програма виводить звернення до до доповідача, а праворуч відображається сказане жестовою мовою та автоматично озвучується. Внизу розташована кнопка виходу з режиму спілкування.

Ця система підтримує декілька жестових мов, завдяки використання датчиків є високоточною. Але така система для нормальної роботи потребує зовнішні сенсори, що може обмежити її використання зі звичайними пристроями, такими як ноутбуки та смартфони. Також система не підтримує української жестової мови (УЖМ).

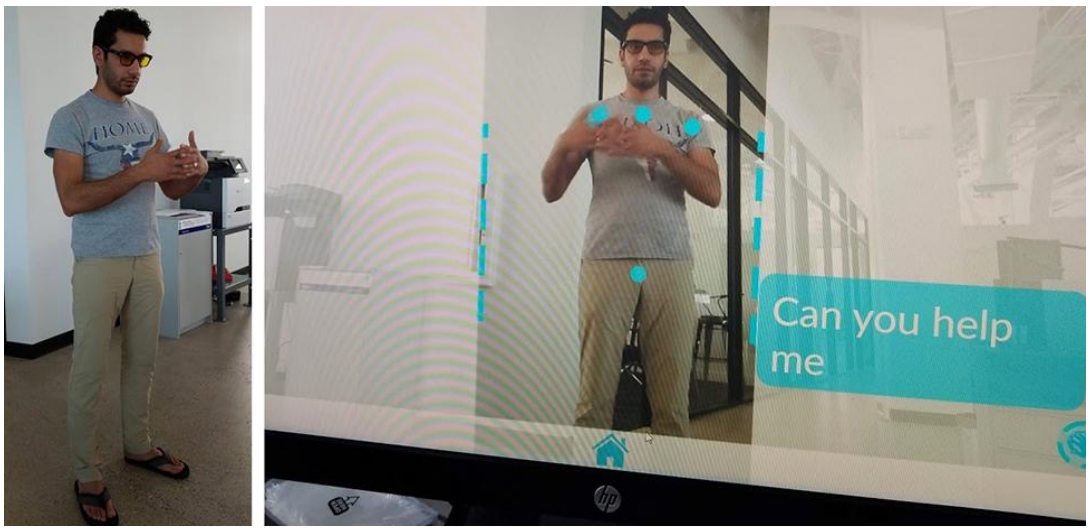

Рисунок 1.3 – Інтерфейс KinTrans

1.2.3 Пристрій MotionSavvy

Наступне рішення MotionSavvy [7] запропонували студенти з Рочестерського інституту технологій (США).

MotionSavvy – це проєкт, який використовує технологію Leap Motion для відстеження рухів рук і пальців.

Компанія була заснована у 2014 році групою студентів. Метою MotionSavvy була розробка портативного пристрою, який перекладає мову жестів у текст і голос, щоб допомогти людям з вадами слуху спілкуватися на щоденній основі.

На рисунку 1.4 зображено інтерфейс пристрою. Пристрій фіксує рухи рук та пальців, відображаючи їх у центрі екрана у вигляді схематичного зображення, що дозволяє користувачеві бачити, як саме система сприймає виконувані жести. Зліва відображається інтерпретований текст сказаного мовою жестів, який пристрій автоматично перетворює у голосове повідомлення за допомогою вбудованого синтезатора мовлення. Справа виводиться текстова версія того, що було сказано голосом співрозмовника, що забезпечує зворотну комунікацію між двома сторонами – користувачем, який спілкується жестами, і особою, яка спілкується усною мовою.

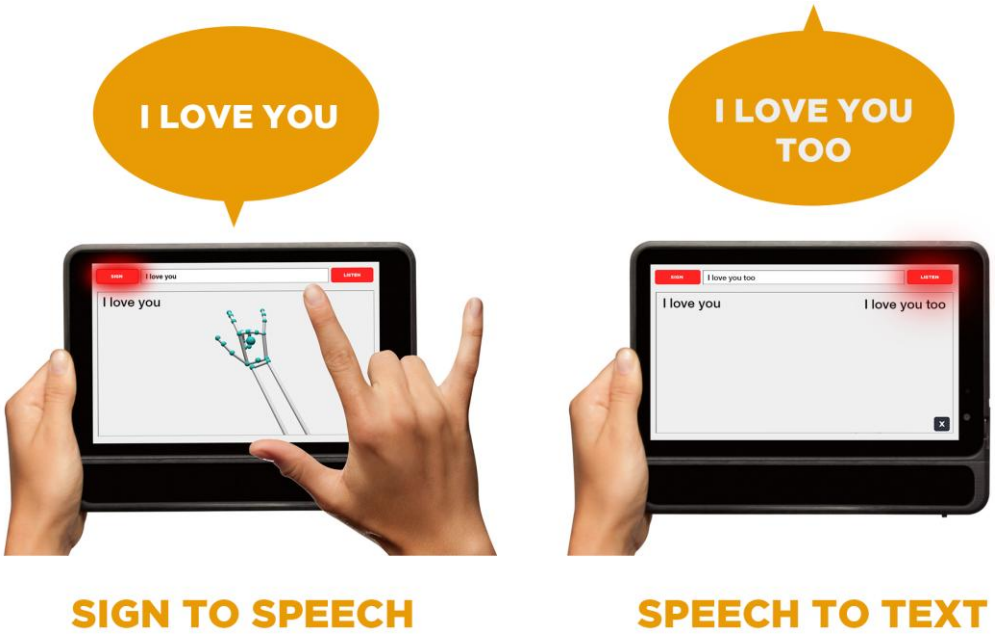

Рисунок 1.4 – Інтерфейс MotionSavvy

Запропонована система сприяє покращенню комунікації осіб з порушенням слуху у повсякденному житті; пристрій є портативним і, завдяки датчикам Leap Motion може точно розпізнавати рухи. Але система підтримує лише АЖМ та поки не вийшла у масове виробництво.

1.2.4 Система Google MediaPipe

Компанія Google також працює над експериментальними проєктами для підтримки перекладу жестової мови [8].

У 2019 році Google запустив систему MediaPipe, яка може розпізнавати позу та жести рук у реальному часі. Цей проєкт відкриває можливості для розробників, які працюють над перекладом мови жестів у текст та звук. Наразі компанія працює над інтеграцією перекладу АЖМ у свої проєкти.

На рисунку 1.5 бачимо як це повинно виглядати у додатках Google.

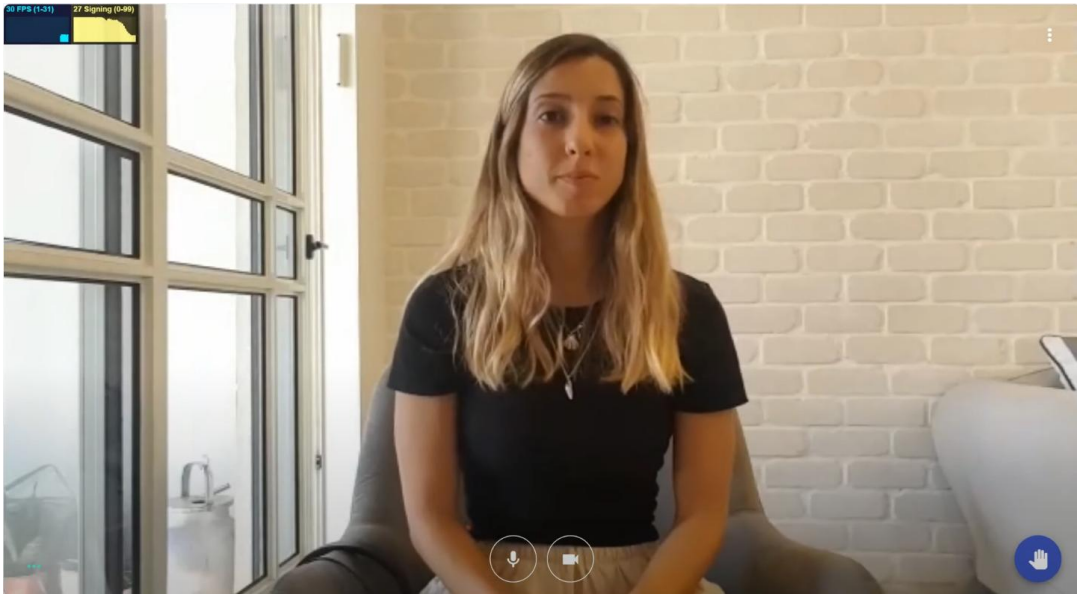

Рисунок 1.5 – Інтерфейс Google Translate for Sign Language

Враховуючи вплив Google, такі системи можуть широко використовуватися в майбутньому, в тому числі для перекладу в режимі реального часу на смартфонах та інших пристроях. Так як компанія використовує свою розробку MediaPipe така система може бути точною без використання датчиків чи спеціальних рукавичок. Але ця система знаходиться на стадії розробки та експериментів та поки підтримує лише АЖМ.

1.3 Порівняльний аналіз

Перед реалізацією проєкту слід провести порівняльний аналіз між системою, що розробляється, та вже існуючими рішеннями. Такий аналіз дозволяє виявити сильні та слабкі сторони доступних технологій, а також обґрунтувати доцільність розробки нового програмного продукту. Для цього було обрано такі ключові критерії оцінювання: підтримка української жестової мови (УЖМ), потреба в додаткових апаратних датчиках (рукавичках, сенсорах тощо), наявність функціоналу для

перетворення жестів у текст, а також можливість озвучування розпізнаних жестів, тобто перетворення жестів у голос.

Крім того, при порівнянні враховувалися такі аспекти, як зручність у використанні, доступність (вартість та відкритість технології), точність розпізнавання, а також наявність підтримки або можливості навчання нових жестів. У результаті проведеного аналізу стало очевидно, що більшість існуючих рішень або орієнтовані на англійську/американську жестову мову, або потребують спеціального обладнання, що суттєво знижує їхню доступність для широкого кола користувачів в Україні.

Результати аналізу продемонстровані у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз систем

	Підтримка УЖМ	Потреба в датчиках	Перетворення жестів у текст	Перетворення жестів у голос
SingAll	–	–	+	–
KinTrans	–	+	+	+
MotionSavvy	–	+	+	+
Google Trans. for Sign Lang.	–	–	+	+/–
Кваліфікаційний проєкт	+	–	+	+

З порівняльного аналізу, представленого в таблиці 1.1, можна зробити висновок, що система, яка розробляється, має низку суттєвих переваг над іншими рішеннями. Основною відмінністю є підтримка УЖМ, що є унікальною на ринку, оскільки інші системи не мають такої можливості. Ще однією перевагою є те, що жести можуть бути переведені не тільки в текст, але і в мову, що значно розширює можливості використання системи в сфері комунікації, наприклад, серед людей з вадами слуху і тих, хто не може бачити або не володіє жестовою мовою.

У порівнянні з іншими рішеннями, такими як SingAll, що забезпечує лише переклад у текст, система, що розробляється, буде здатна на як текстовий переклад так і на переклад у мову.

Такі рішення, як KinTrans і MotionSavvy, використовують додаткові датчики для аналізу жестів, але обмежені потребою в спеціалізованому обладнанні, що збільшує витрати на впровадження. На відміну від них, система, що розробляється, не потребує додаткових датчиків, що робить її більш доступною та універсальною для широкого кола користувачів.

Завдяки цим перевагам система, що розробляється, має великий потенціал в Україні для використання в навчальних закладах, офісах, медичних установах та інших сферах, де важлива швидка і точна комунікація між носіями жестової мови та людьми, що не володіють мовою жестів.

1.4 Постановка задачі

Метою проєкту є розробка застосунку для покращення комунікації людей з вадами слуху з людьми, що не володіють мовою жестів, який перекладатиме жестову мову в текст та голос. Такий інструмент сприятиме усуненню бар’єрів у спілкуванні, підвищенню інклюзивності та полегшенню доступу до повноцінного соціального життя для людей з порушеннями слуху.

Розроблений застосунок повинен мати можливість розпізнавати жести УЖМ за допомогою звичайної камери, без потреби у використанні спеціального обладнання, що зробить його більш доступним для широкого кола користувачів. Система повинна автоматично аналізувати жести, перекладати їх у текстову форму, а також, за потреби, озвучувати результат за допомогою синтезатора мовлення.

Користувач повинен мати модливість вмика/вимика функцію голосового перекладу.

Успішне втілення поставленої задачі передбачає виконання наступних завдань:

- проведення аналізу предметної галузі;
- проведення аналізу існуючих рішень;
- проєктування архітектури застосунку;
- вибір мови програмування та технології;
- пошук/створення набору даних;
- розробка моделі;
- тестування точності моделі;
- виправлення помилок/покращення точності моделі;
- розробка інтерфейсу застосунку;
- тестування працездатності застосунку.

Таким чином, додаток надаватиме зручний та ефективного інструмент для перекладу УЖМ в текст і мовлення, покращуючи таким чином комунікацію для людей, що мають вади слуху, з тими, хто не володіє жестовою мовою. Додаток повинен бути не тільки функціональним, але й інтуїтивно зрозумілим та адаптованим до потреб користувача, зокрема, він повинен мати можливість вмикати та вимикати голосовий переклад.

Крім того, важливо забезпечити високу точність розпізнавання жестів, враховуючи різні індивідуальні особливості користувачів, такі як швидкість виконання жестів, варіації у формі рук, а також можливі відхилення в жестах, зумовлені віком чи фізичними обмеженнями. Додаток має працювати в реальному часі, з мінімальною затримкою, щоб забезпечити природний та безперервний характер спілкування.

Особливу увагу слід приділити інтерфейсу: він має бути доступним для людей з різними рівнями технічної підготовки, з можливістю персоналізації – наприклад, зміни мови інтерфейсу. Також доцільно передбачити функціонал зворотного зв’язку, що дозволить користувачам повідомляти про помилки в розпізнаванні або пропонувати нові жести для додавання до словника системи.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Функціональні вимоги до проєкту

Функціональними вимогами до виконаного проєкту є перелік ключових можливостей, які повинна забезпечувати система для коректної та ефективної роботи відповідно до поставленої мети. До таких вимог належать:

– захоплення відеопотоку з камери та обробка заздалегідь записаного відео;

– підтримка стандартних форматів відеофайлів;

– використання алгоритмів машинного навчання для класифікації жестів та алфавіту УЖМ;

– переклад УЖМ на текст та/або голос у реальному часі;

– відображення розпізнаного тексту в режимі реального часу;

– можливість редагування отриманого тексту користувачем;

– експорт тексту у форматах TXT, DOCX, PDF;

– зручний інтерфейс;

– система повинна працювати на ОС Windows.

Реалізований проєкт повинен надавати користувачу такі можливості:

– увійти/зареєструватися;

– перевірити роботу камери;

– почати/зупинити переклад у реальному часі;

– вмикати/вимикати функцію голосового перекладу;

– переглядати та виправляти перекладений текст;

– переглядати історію перекладу;

– залишати вікгук щодо коректності роботи програми;

– переглядати переклад.

2.2 Вибір архітектури проєкту

Шаблон програмної архітектури задає загальну будову та організацію програмного забезпечення, описує основні складові, принци взаємодії та загальну структуру системи. Архітектурні шаблони впливають на рішення щодо масштабованості, ефективності роботи та зручності супроводу [9].

Одна з розглянутих архітектур це layered pattern. Така система є однією з найбільш поширених архітектурних парадигм у проєктуванні програмного забезпечення. У системі структуровано поділене програмне забезпечення на горизонтальні шари, кожен з яких виконує конкретні задачі: наприклад, представлення інформації користувачу, обробка логіки чи управління даними. Це підхід, що є зручним у розробці та подальшому розширенні програми. Недоліками такого типу архітектуру можуть бути зайві витрати на зв’язок між рівнями та складність при наявності великої кількості рівнів. Зазвичай layered pattern використовується у веб-застосунках [10].

Наступною архітектурою, що була розглянута є client-server arhitecture. Така система базується на взаємодії клієнта та сервера. Клієнт надсилає запит, який отримує та обробляє сервер. Потім сервер пересилає готовий результат клієнту. Сервер має здатність обробляти численні запити одночасно. Клієнтно-серверні зазвичай використовуються у веб-додатках та мають безліч переваг, а саме: мінімальні витрати ресурсів клієнта, гнучкість та модульність, логування та моніторинг, тощо. Але є і недоліки, такі як: проблеми з оновлюваністю, якщо на сервері великий трафік, такі додатки програють у швидкості автономним програмам [11].

Також була розглянута MVC (Model-View-Controller) architecture. Така система поділена на три компоненти, модель, інтерфейс та контролер. Модель та інтерфейс є незалежними між собою, а контролер є містком для обміну інформацією між цими двома компонентами. Такі системи широко застосовуються у веб-розробці, мобільних додатках, настільних програмах,

іграх, корпоративних системах та IoT. Перевагами MVC є легка підтримка коду, зручна масштабованість та гнучкість, а недоліками складність реалізації, велика кількість файлів та коду, складність тестування [12].

Перш ніж обирати архітектуру проєкту слід визначити буде програма автономною чи буде використовувати сервер для обчислень, перекладу, обробки даних, тощо.

Розглянемо плюси та мінуси автономної програми та серверної системи. Автономні програми зазвичай працюють швидше так як не вимагають часу для звернення до серверу, не потребують підключення до інтернету та не загрожують конфіденційності даних, проте такі системи можуть вимагати обладнання певної потужності, потребують ручного оновлення користувачем та розширення автономних програм ускладнене. Серверні системи можуть використовувати як потужні так і слабкі пристрої так як майже усі обчислення виконує сервер, такі системи є легко розширюваними та не потребують ручного оновлення користувачем.

Для даного проєкту метою є розробити автономну програму, щоб користувач, навіть без доступу в інтернет міг користуватися додатком, тож такі типи архітектури проєкту, як client-server далі не розглядаються.

У системі планується реалізувати логіку за допомогою C++, модель машинного навчання використовуючи python та інтерфейс користувача з інструментами Qt/qml, також система передбачає базу даних для можливості користувачів реєструватися та авторизуватися у додатку, тощо. Таку систему логічно реалізовувати використовуючі MVC архітектуру, але у такому випадку слід чітко відокремити три компоненти: модель, інтерфейс та контроллер, що не є дуже зручним у зазначеному випадку. Тож, доречніше використати багаторівневу архітетектуру, що буде подібною до MVC. Попередньо виокремимо такі рівні: логіка (C++ частина), інтерфейс (Qt/qml частина), модель машинного навчання (python частина) та база даних.

2.3 Проєктування архітектури застосунку

На етапі проєктування архітектури застосунку ключове значення має побудова діаграм, що візуалізують складові системи та взаємодію елементів. Структурні схеми дають змогу чітко пояснити заплановану структуру системи, зв’язки та взаємодію, полегшити проєктування, розуміння та передачу ідей, що закладені у проєкт розробником.

Для розроблюваного додатку були розроблені UseCase (рисунок 2.1), UserFlow (додаток А) та Class (рисунок 2.4) діаграми.

UseCase є поверхневими діаграмами, що демонструють вимоги до проєкту, вказують на очікувану поведінку, але не на точний спосіб її здійснення [13]. Така схема зазвичай будується першою серед усіх та має бути простою та зрозумілою для будь-кого, хто приймає участь у розробці проєкту.

На рисунку 2.1 зображена Use Case-діаграма для системи перекладу української жестової мови в текст. Дана діаграма ілюструє взаємодію користувача із застосунком та перелік функціональних можливостей, які система надає для забезпечення ефективної роботи. Зліва представлено користувача – основного актора системи, який взаємодіє з програмою через інтерфейс. Справа вказані основні дії (випадки використання), доступні користувачеві.

Серед них – реєстрація нового користувача, що дозволяє створити обліковий запис для персоналізованого використання системи, та авторизація, яка забезпечує доступ до функціоналу після підтвердження особи. Перевірка камери це важливий крок для початку роботи з перекладом, оскільки система потребує відеопотік для аналізу жестів.

Користувач має можливість вмикати та вимикати голосовий переклад, що дозволяє адаптувати спосіб спілкування залежно від конкретної ситуації чи потреб. Крім того, користувач може почати або зупинити процес перекладу жестів у текст, що забезпечує гнучке керування роботою системи.

Результати перекладу відображаються в інтерфейсі, де є можливість переглядати перекладений текст у реальному часі.

Також реалізовано можливість залишити відгук, що сприяє подальшому вдосконаленню програми на основі зворотного зв’язку від реальних користувачів. Нарешті, користувач може переглядати історію перекладу, що дозволяє відслідковувати попередні сесії роботи із застосунком і за потреби повертатися до раніше перекладеної інформації.

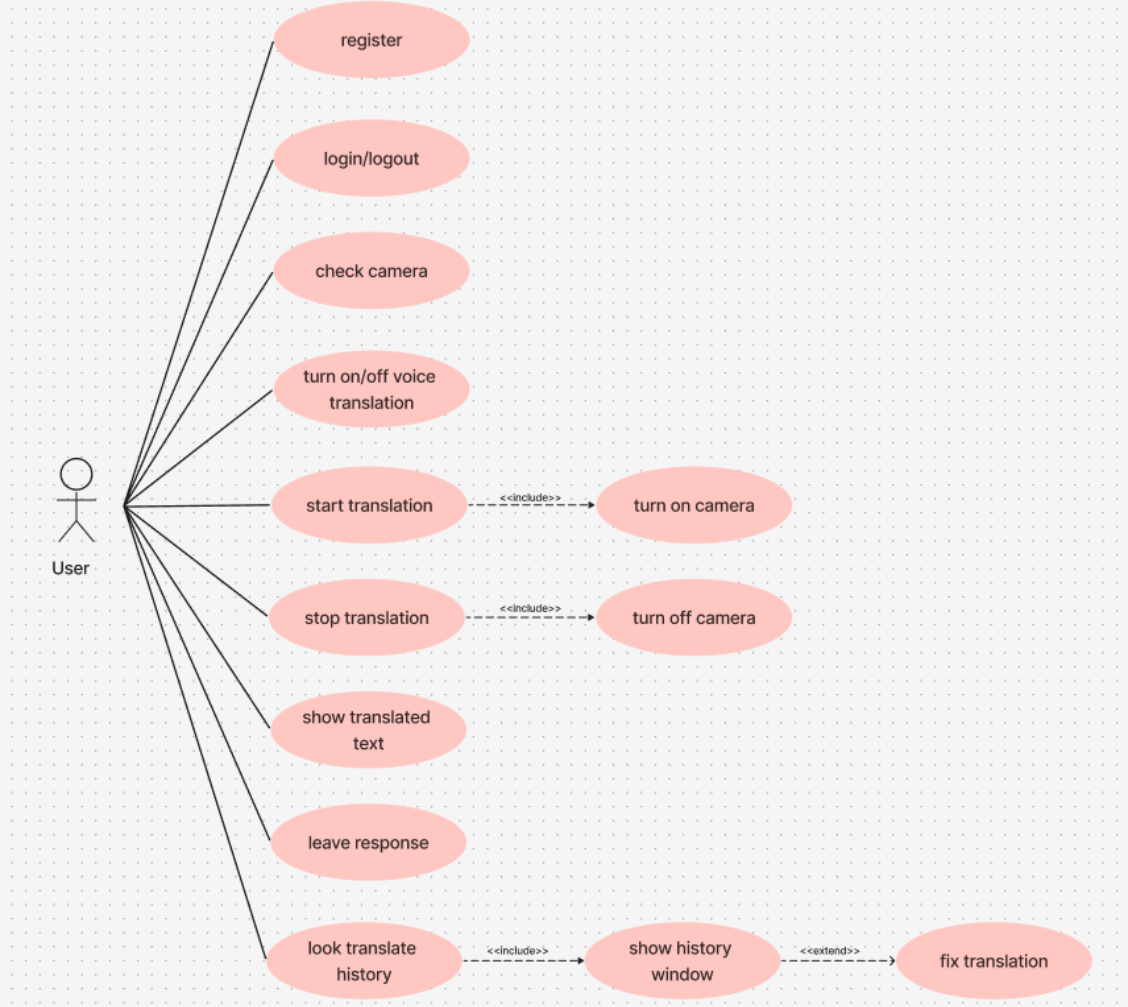

Рисунок 2.1 – UseCase діаграма додатку

Наступною була розроблена UserFlow діаграма.

UserFlow діаграми відображають шлях користувача через рішення. Такі схеми наглядно демострують як користувач досягає своїх цілей в

середині системи. Для побудови та розуміння UserFlow слід розуміти основні принципи та правила створення такого типу діаграм. Базові компонентами схеми зображені на рисунку 2.2.

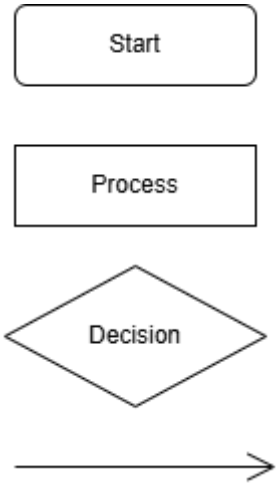

Рисунок 2.2 – Базові компоненти UserFlow

Прямокутник з закругленими кутами, відображає подію, яка визначає початок і кінець шляху користувача (наприклад користувач заходить у додаток, відкриває головну сторінку, закінчує реєстрацію, тощо). Звичайний прямокутник відповідає за кроки, що виконує користувач (наприклад шукає інформацію, виправляє текст, перевіряє камеру, тощо). Ромб відображає місця у програмі де користувач має зробити вибір, що вплине на його шлях. Це майже будь-які моменти у системі де передбачається відповідь від користувача. Стрілки вказують на напрямок користувача у системі. У додатку А можна побачити UserFlow діаграму для системи перекладу української жестової мови в текст [14]. Рисунок А.1 у додатку А демонструє очікувану поведінку користувача та шлях у системі залежно від прийнятих рішень.

Останньою була розроблена діаграма класів. Така діаграма робиться розробниками для розробників для візуалізації об’єктно-орієнтованої системи, представлення структури та зв’язків між класами. Тут важливо

чітко розуміти правила побудови схеми. На рисунку 2.3 зображено приклад діаграми класів.

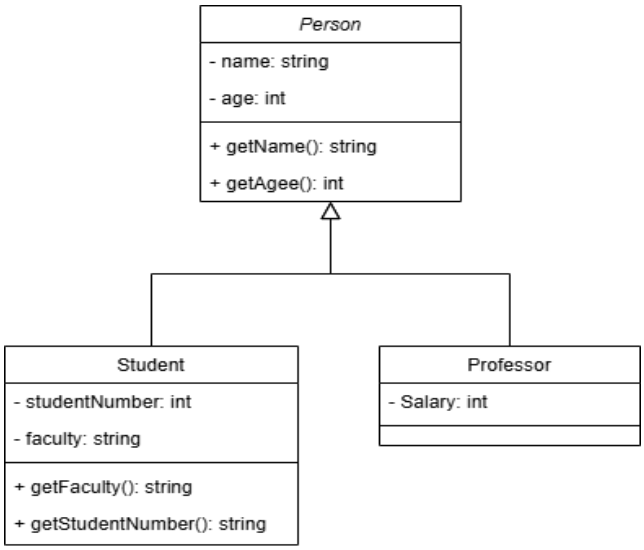

Рисунок 2.3 – Приклад діаграми класів

Класи у таких схемах є квадратом, що містить назву згори (ім’я класу), потім члени та функції класу (назва + тип даних). +/- у подібних схемах відображають модифікатори доступу до члена класу чи функції. Зв’язки між класами позначають стрілками, що зображені на рисунку 2.4.

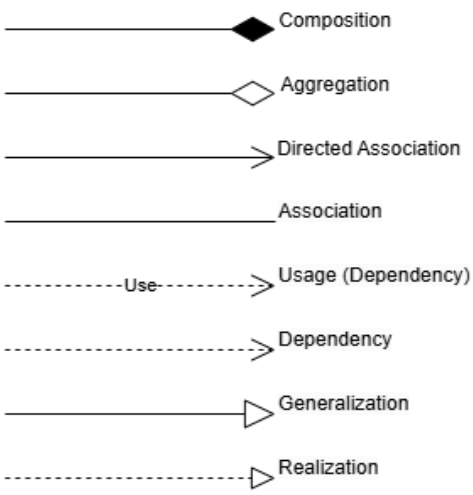

Рисунок 2.4 – Зв’язки у діаграмі класів

У діаграмах класів UML зв’язки між елементами відіграють важливу роль у моделюванні структури програмного забезпечення, оскільки вони визначають характер взаємодії між класами та інтерфейсами. Кожен тип зв’язку має свою семантику, візуальне позначення та практичне значення при проєктуванні об’єктно-орієнтованих систем.

Асоціація (Association) є базовим видом зв’язку між двома класами, що відображає наявність стійкої логічної або фізичної взаємодії між об’єктами відповідних типів. Такий зв’язок встановлюється, коли один клас містить посилання на інший або коли об’єкти обох класів існують водночас і взаємодіють протягом тривалого часу. Асоціація може бути двонаправленою або однонаправленою, а також може мати уточнення у вигляді ролей, кратності та назв, що підвищує експресивність моделі.

Зв’язок використання (Usage), також трактований як залежність, відображає ситуацію, коли один клас тимчасово використовує інший у своїй поведінці – наприклад, як тип аргументу методу, локальну змінну або об’єкт, що створюється динамічно. Він не означає довготривалої або жорсткої прив’язки між класами, але вказує на факт, що для коректного функціонування одного елемента необхідне існування іншого. В UML цей тип залежності позначається пунктирною лінією зі стрілкою в бік залежного елемента.

Поняття залежності (Dependency), подібно до зв’язку використання, визначає слабку форму зв’язку між елементами моделі, яка свідчить про те, що зміни в одному з елементів можуть вплинути на інший. Це означає, що елемент-клієнт залежить від специфікації іншого елемента, але зв’язок між ними залишається нетривким і не накладає обов’язку наявності під час виконання програми. Такий зв’язок є важливим для відстеження впливу змін на архітектуру та сприяє підтримці низького рівня зв’язаності (low coupling) між компонентами.

Узагальнення (Generalization) репрезентує ієрархічне наслідування, при якому один клас (підклас) успадковує структуру та поведінку

іншого (надкласу). Це дозволяє реалізувати механізм повторного використання коду та поліморфізму, оскільки підклас може спеціалізувати або розширювати функціональність, задану в надкласі. Зв’язок узагальнення зображається суцільною лінією з порожнім трикутником, спрямованим у бік загального класу, що чітко вказує на напрям спадкування.

Реалізація (Realization) використовується для моделювання відношення між класом і інтерфейсом, який він реалізує. На відміну від узагальнення, реалізація передбачає, що клас не наслідує реалізацію, а лише дотримується контракту, визначеного інтерфейсом. Такий підхід дозволяє досягати високого рівня абстракції та гнучкості при зміні реалізацій без впливу на клієнтів інтерфейсу. Реалізація в UML позначається пунктирною лінією з порожнім трикутником, подібно до узагальнення, що відображає її концептуальну спорідненість, але водночас відмінність у семантиці.

Діаграма класів для системи перекладу української жестової мови в текст зображена у додатку А на рисунку А.2. Основними класами додатку є TranslateModel, SyntaxModel, Translator, GUI, App та languageManager.

2.4 Проєктування бази даних

База даних є важливим елементом у системі для перетворення української жестової мови у текстову та голосову форми. Вона виступає як організоване сховище даних, забезпечуючи систематизацію, зберігання та подальшу обробку різних видів інформації, включно зі статистичними показниками, обліковою інформацією про користувачів (такою як дані аутентифікації та контактів), а також історією взаємодії з додатком.

Архітектура бази даних будується навколо структурованих об’єктів, зокрема, таблиць, запитів, форм та макросів. Таблиця бази даних – це впорядкований набір даних, структурований у вигляді рядків і стовпців. Кожен стовпець відповідає певному атрибуту, а кожен рядок – окремому

випадку. Щоб забезпечити логічну цілісність та мінімізувати надмірність інформації, структуру бази даних нормалізують, що передбачає тематичний розподіл даних між пов’язаними таблицями. Наприклад, інформація про користувачів зберігається в окремій таблиці, а історія перекладів у наступній.

Запити – це засоби отримання інформації з бази даних, що надають можливість обирати, змінювати, видаляти або записувати дані згідно з чітко визначеними умовами. Створення запитів базується на структурованій мові, що сприяє адаптивності дій та гарантує значний рівень контролю над обробкою даних.

Форми виступають інтерфейсним рівнем, спрощуючи взаємодію з даними завдяки наочному та структурованому відображенню інформації. Форми надають зручний доступ до окремих елементів бази даних, що особливо актуально для інтерактивних програм користувацького рівня.

Макроси використовуються для автоматизації операцій з даними, дозволяючи виконувати попередньо визначені дії у відповідь на певні події, підвищуючи ефективність роботи системи. Макроси реалізують адаптивну логіку обробки, зменшуючи потребу в ручному втручанні та оптимізуючи повторювані процеси.

Процес створення схеми бази даних, зображеної на діаграмі, базується на формалізації логічної структури інформаційної системи. Побудова діаграми починається з аналізу предметної галузі, результатом якого є визначення всіх сутностей, що представляють об'єкти реального світую. Кожна сутність відповідає логічній таблиці бази даних, що візуалізується у вигляді прямокутника зі списком атрибутів. Всередині кожної таблиці виділяється первинний ключ – атрибут або комбінація атрибутів, що однозначно ідентифікує кожен запис. За необхідності встановлення зв’язку з іншими таблицями вказуються зовнішні ключі, з відповідним посилаюнням на відповідні первинні ключі у пов’язаних таблицях.

Наступним етапом є визначення типів зв'язків між сутностями. Візуалізація зв'язків здійснюється за допомогою ліній, що з'єднують відповідні таблиці. Над лініями вказується кардинальність – числові або символьні маркери, які показують кількісні співвідношення між екземплярами сутностей, зокрема, один до одного, один до багатьох або багато до багатьох. У випадку зв'язку багато до багатьох проєктується проміжна таблиця, що забезпечує реалізацію цього зв'язку на фізичному рівні.

У процесі побудови діаграми виконується нормалізація атрибутів, яка полягає в усуненні надлишкових залежностей, дублювання та забезпеченні логічної незалежності даних. Це дозволяє досягти стабільності схеми під час подальших змін та розширень. Якщо є атрибути, які самі можуть виступати як окремі сутності, відбувається їх декомпозиція з утворенням нових таблиць.

Створена діаграма є результатом логічного моделювання, що відображає всю структуру зберігання інформації та слугує фундаментом для фізичної реалізації бази даних. Ця схема виступає не лише технічною документацією, а й засобом узгодженності між розробниками, аналітиками та дизайнерами.

На рисунку 2.5 зображено розроблену схему бази даних (БД) для додатку, що створюється у рамках кваліфікаційної роботи. Ця схема є ключовим компонентом архітектури системи, оскільки забезпечує організоване та структуроване зберігання всієї необхідної інформації, пов’язаної з функціонуванням застосунку.

Схема БД була спроєктована з урахуванням принципів нормалізації, що дозволяє уникати надлишкового дублювання інформації та забезпечити узгодженість даних. Її структура є достатньо гнучкою для подальшого масштабування системи.

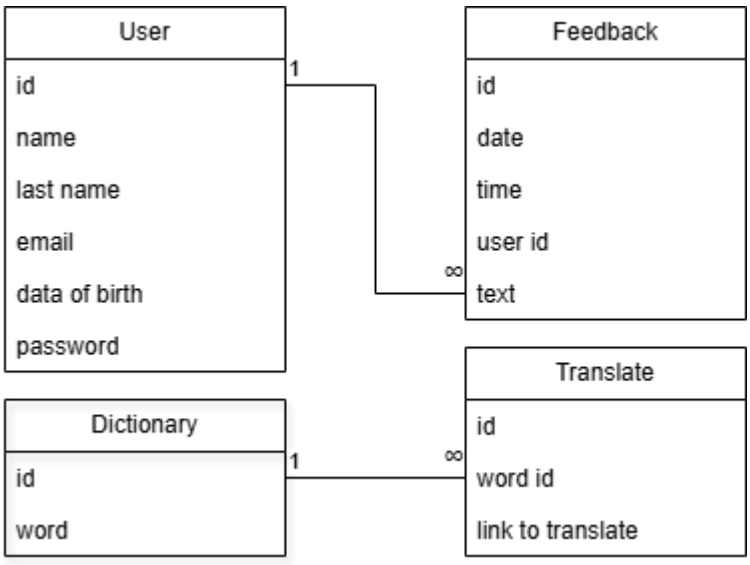

Рисунок 2.5 – Діаграма бази даних системи УЖМ

Основними сутностями застосунку є користувач, слово, словник перекладу та зворотній зв’язок. Кожна з цихсутностей представлена у базі даних окремою таблицею, яка містить унікальний ідентифікатор та відповідну інформацію, що стосується сутності.

Сутність користувача включає особисті дані, зокрема ім’я, прізвище, електронну пошту, дату народження та пароль, що необхідні для реєстрації та подальшої авторизації в системі. Це дозволяє персоналізувати досвід користувача та, за потреби, зберігати його активність.

Сутність зворотного зв’язку дає змогу зберігати відгуки, які залишають користувачі після використання застосунку. У базі даних фіксується сам текст повідомлення, дата і час його надсилання, а також інформація про того, хто його залишив. Це дозволяє розробникам отримувати корисні відомості для подальшого вдосконалення системи.

Словник перекладу реалізує зв’язок між словами та відповідними відеофрагментами жестів. Для кожного слова зберігається текстова форма і посилання на відео, у якому показано, як виконується відповідний жест. Це дозволяє застосунку швидко і точно відображати потрібну візуальну інформацію під час перекладу.

3 ВИБІР ТЕХНОЛОГІЙ ТА ПРОГРАМНИХ ЗАСОБІВ РОЗРОБК

3.1 Мова програмування C++

C++ це об’єктно-орієнтована мова програмування, проте може використовувати і процедурний підхід. Мова розробки знайшла широке застосування в високопродуктивних програмних комплексах, де показники швидкодії, раціональне використання пам'яті та комунікація з фізичними пристроями мають першорядне значення. Саме ці риси зумовлюють актуальність C++ в задачах комп'ютерного зору, опрацювання сигналів та розпізнавання жестів – складових системах перекладу жестової мови [15].

На відміну від мов високого рівня, таких як Python або Java, C++ надає змогу безпосередньо взаємодіяти з пам'яттю та ресурсами системи. Це має критичне значення під час обробки відеопотоків у реальному часі, роботи з зображеннями, або застосування алгоритмів машинного навчання на нижчих рівнях. Наприклад, OpenCV, широко відома бібліотека для комп'ютерного зору, створена саме на C++, забезпечуючи максимальну продуктивність при обробці відео та зображень. Якщо Python ідеально підходить для початкового етапу розробки через свій зручний інтерфейс, то для фінального впровадження системи розпізнавання, здатної працювати з камерою у режимі реального часу, знадобиться більш потужний інструмент, яким є C++.

Окрім цього, C++ дозволяє втілювати багатопотоковість, що є критичним для паралельного розпізнавання жестів рук, обличчя, аналізу контексту та конструювання синтаксичної структури жестів. Адаптивність мови сприяє як приєднанню зовнішніх бібліотек (наприклад, TensorFlow C++ API, MediaPipe, dlib), так і розробці власних оптимізованих алгоритмів.

У зіставленні з мовами, націленими на бізнес-логіку чи розробку інтерфейсів (як-от C# або JavaScript), C++ ліпше пристосований до

створення ядра системи, котре відповідає за обробку візуальної інформації та попереднє кодування мовних елементів.

Відтак, вибір мови C++ у рамках цієї кваліфікаційної роботи видається логічним і обґрунтованим, адже це гарантує необхідний рівень продуктивності, сумісність з інструментами обробки відео та алгоритмами комп'ютерного зору, а також гнучкість для інтеграції з іншими складниками системи.

3.2 Мова програмування Python

Python це мова програмування високого рівня з динамічним типом, яка є популярною завдяки простому синтаксису, гнучкості та величезній кількості готових бібліотек. У сфері комп'ютерного зору, машинного навчання та обробки відео Python часто обирають як першу мову програмування через зручність розробки, активну спільноту та підтримку сучасних фреймворків, таких як TensorFlow, PyTorch, OpenCV, MediaPipe, NumPy та інші.

У контексті впровадження системи перекладу жестової мови в текст Python відіграє важливу роль на етапах розробки, дослідження алгоритмів та обробки даних. Його висока абстракція дає змогу зосередитися на логіці програми, швидко експериментувати з різними моделями розпізнавання та без проблем обробляти великі обсяги відео або координатної інформації жестів. Наприклад, використання бібліотеки MediaPipe, яка підтримує детекцію рук та відстеження положення пальців, у Python дає змогу реалізувати систему розпізнавання жестів, навіть без глибоких знань низькорівневої обробки відео.

Проте, не дивлячись на зручність та велику кількість інструментів, Python поступається C++ в продуктивності, особливо в завданнях реального часу. У випадках, коли необхідно обробляти відеопотік з високою частотою кадрів або реалізовувати багатопотокову обробку, інтерпретована природа

Python може призвести до затримок або перевантаження системи. Саме тому у багатьох сучасних проєктах Python використовують разом з C++: перша мова – для прототипування та роботи з моделями, друга – для продуктивної реалізації ключових обчислювальних модулів.

В кваліфікаційної роботи Python застосовується для створення прототипів розпізнавання жестів, тестування моделей машинного навчання, візуалізації результатів і попередньої обробки даних. Завдяки своїй гнучкості та розвиненій екосистемі він дає можливість швидко перевіряти ідеї, інтегрувати нові алгоритми та забезпечувати подальше розширення системи.

3.3 Декларативна мова програмування Qt/QML

Qt/QML – це потужна платформа для створення міжплатформних графічних інтерфейсів (GUI) та програм. Вона поєднує декларативну мову QML (Qt Meta Language) для опису інтерфейсу та C++ для втілення головної логіки та функціональності.

QML, як декларативна мова, дозволяє розробникам визначати зовнішній вигляд і поведінку інтерфейсу за допомогою структурованого синтаксису, подібного до CSS та JavaScript. Це полегшує процес дизайну та ітерації інтерфейсу, дозволяючи розробникам зосереджуватися на візуальних елементах, їх розміщенні та анімаціях, не потребуючи компіляції коду після кожної зміни. QML містить великий вибір вбудованих елементів інтерфейсу, таких як кнопки, текстові поля, слайдери, а також дозволяє створювати власні кастомні компоненти.

C++, як головна мова програмування Qt, забезпечує високу продуктивність та доступ до низькорівневих можливостей системи. Він використовується для реалізації складної бізнес-логіки, обробки даних, мережевих операцій та інтеграції з іншими бібліотеками та API. Qt надає велику кількість C++ класів і модулів, що охоплюють різні аспекти

розробки, включаючи роботу з файлами, мережею, базами даних, графікою, мультимедіа та багатьом іншим.

Однією з ключових переваг Qt/QML є її кросплатформність. Програми, розроблені за допомогою цього фреймворку, можна компілювати та запускати на різних операційних системах, таких як Windows, macOS, Linux, Android, iOS та вбудовані системи, без суттєвих змін у кодовій базі. Це досягається завдяки абстракції Qt від особливостей кожної конкретної платформи.

Інтеграція між QML та C++ є безшовною. Розробники можуть легко передавати дані та викликати функції між цими двома частинами програми. QML може використовувати об'єкти C++ як моделі даних або для виконання складних обчислень, а C++ може керувати візуальними елементами, визначеними в QML [16].

3.4 Бібліотека TensorFlow

TensorFlow – це потужна бібліотека з відкритим кодом, розроблена Google для машинного навчання. Цей інструмент є фундаментом для створення, навчання та розгортання нейронних мереж, знаходячи застосування як у наукових дослідженнях, так і в індустріальних проєктах. Завдяки своїй гнучкості, TensorFlow надає як низькорівневі інструменти для детального контролю обчислень, так і високоуровневі API (наприклад, Keras), які значно спрощують побудову та навчання моделей [17].

У контексті розробки системи перекладу жестової мови в текст, TensorFlow є ключовим інструментом для створення моделей глибокого навчання, відповідальних за розпізнавання та інтерпретацію рухів рук. Наприклад, TensorFlow дозволяє використовувати згорткові нейронні мережі (CNN) для аналізу відеопотоку, рекурентні нейронні мережі (RNN) або трансформери для обробки послідовностей жестів, а також реалізувати багатокласову класифікацію для розпізнавання окремих жестів.

Перевагою TensorFlow є підтримка роботи як на центральних процесорах (CPU), так і на графічних процесорах (GPU), що дозволяє суттєво прискорити процес навчання моделей. Крім того, фреймворк надає інструменти для оптимізації та експорту моделей у полегшені формати, готові до використання на мобільних пристроях або вбудованих системах (наприклад, TensorFlow Lite або TensorFlow.js). Це дозволяє реалізувати систему розпізнавання жестів не тільки в експериментальному середовищі, але й у реальному застосуванні.

Не менш важливою є активна спільнота TensorFlow, яка забезпечує швидкий доступ до рішень, прикладів та документації. Це суттєво полегшує процес розробки, особливо при інтеграції з бібліотеками, такими як OpenCV або MediaPipe, які можуть надавати попередньо оброблені дані (наприклад, координати суглобів рук) як вхідні дані для нейронної мережі.

В рамках кваліфікаційної роботи TensorFlow використовується для розробки моделі машинного навчання, яка аналізує вхідні жести та перетворює їх у відповідні текстові конструкції. Це дозволяє реалізувати адаптивний, масштабований та ефективний підхід до автоматичного перекладу української жестової мови.

3.5 Бібліотека OpenCV

OpenCV є однією з найвідоміших, що використовується для комп'ютерного зору та обробки зображень. Інструмент широко застосовують в наукових розробках, промисловості, а також в освітніх програмах. Бібліотека містить різноманітні функції: від елементарної обробки зображень до розпізнавання об'єктів, відстеження руху, аналізу динаміки та глибинного навчання. OpenCV сумісна з різними мовами програмування, зокрема C++, Python, Java, і підтримує прискорення за допомогою апаратних засобів [18].

У реалізації системи перекладу жестової мови в текст, OpenCV відіграє ключову роль у попередній обробці відеопотоку. Зокрема, застосовується для детекції рук, фільтрації кадрів, визначення контурів, нормалізації кольору шкіри або автоматичного кадрування необхідних зон. Завдяки можливості роботи в реальному часі, OpenCV дає змогу зчитувати зображення з камери, виконувати базову обробку та передавати очищені дані, наприклад, до моделей TensorFlow чи систем аналізу координатних точок.

Окрім базового функціоналу, OpenCV надає доступ до більш складних алгоритмів, як-от детекція облич, відстеження рухомих об'єктів, робота з картами глибини, перетворення перспективи та інших. У поєднанні з іншими бібліотеками, такими як MediaPipe або NumPy, OpenCV дозволяє ефективно опрацьовувати координати суглобів руки, виділені з відео, та формувати їх як вхідні дані для нейронної мережі.

У кваліфікаційній роботі OpenCV використовується для обробки відеозаписів, виділення та відстеження областей жестів, а також для підготовки вхідних даних для системи розпізнавання.

3.6 Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) є легким, кросплатформенним середовище розробки від Microsoft, яке підтримує широкий спектр мов програмування та інструментів завдяки розширенням.

У проєкті з реалізації системи перекладу жестової мови в текст, Visual Studio Code використовується як основне середовище розробки завдяки своїй гнучкості та адаптивності до різних мов програмування. Зокрема, Python та C++, які широко застосовуються в цій роботі. Завдяки підтримці відповідних розширень, таких як Python Extension, C++ Intellisense, CMake Tools, Code Runner, Jupyter та інші, середовище дозволяє працювати як з моделями машинного навчання, так і з системами комп'ютерного зору.

Особливу цінність VS Code має для багатомовних проєктів, де потрібно швидко перемикатися між різними модулями, реалізованими на різних мовах, інтегрувати сторонні бібліотеки або взаємодіяти з системами керування версіями. Під час розробки системи перекладу жестів це дає змогу зручно редагувати і налагоджувати Python-скрипти для роботи з TensorFlow, обробку відеопотоку через OpenCV або MediaPipe, а також C++-модулі, що відповідають за критичні обчислення.

3.7 Сервіс розробки інтерфейсів Figma

Figma – це популярний серед UI/UX дизайнерів застосунок, що надає можливість працювати з векторною графікою, створювати макети, інтерфейси, зображення, схеми, ілюстрації, тощо.

У рамках виконання кваліфікаційного проєкту, було використано Figma для розробки прототипу графічного інтерфейсу користувача системи перекладу жестової мови. Інтерфейс було спроєктовано з особливою увагою до принципів доступності та легкості в користуванні для людей з вадами слуху. Створено макети основних вікон застосунку, а саме: головне вікно, вікно, що відображає відео з камери, область виводу перекладу тексту та кнопки для керування.

3.8 Система збірки CMake

CMake – це кросплатформенна система автоматизації збірки, котра дозволяє керувати компіляцією та збиранням програмного забезпечення незалежно від платформи чи середовища розробки. Ключове завдання інструменту – полегшити процес створення проєктів, особливо масштабних та багатокомпонентних, що використовують різноманітні бібліотеки, мови програмування та залежності. CMake генерує проєкти для широкого кола середовищ, включно з Makefile, Ninja, Visual Studio, Xcode та інші [19].

В межах проєкту автоматичного перекладу жестової мови в текст, CMake виконує роль центрального інструменту управління збіркою C++ та qt/qml компонентів.

Завдяки використанню CMake, можна легко описати залежності проєкту, підключити зовнішні бібліотеки, такі як OpenCV, TensorFlow C++ API, чи Boost, а також визначити конфігурації збірки – приміром, налагоджувальну чи релізну. Це робить систему гнучкою та зручною в обслуговуванні, особливо при розробці на різних операційних системах або ж в команді з різними середовищами розробки [20].

Ще одним плюсом CMake є сумісність із системами тестування та безперервної інтеграції, що дозволяє підключати юніт-тести або автоматичну збірку моделей, що має велике значення при розробці програм з машинним навчанням.

У кваліфікаційній роботі CMake використовується для автоматизованої збірки C++-модулів системи, котрі відповідають за попередню обробку вхідного відео, роботу з графічним інтерфейсом, збірки qt/qml-модулів nf для інтеграції з зовнішніми бібліотеками.

3.9 Система керування базами даних PostgreSQL

PostgreSQL – це об'єктно-реляційна система управління базами даних, що інтегрує класичну реляційну модель з можливостями об'єктно-орієнтованого програмування [21]. Архітектура цієї СУБД спроєктована з акцентом на підтримку транзакційності, узгодженості, ізольованості та надійності (ACID), гарантуючи високу цілісність даних в умовах паралельного доступу та великого навантаження. Реалізація PostgreSQL включає багаторівневу систему кешування, що дає змогу ефективно обробляти надскладні запити, а також підтримує широкий спектр типів даних, включаючи користувацькі, що розширює функціональні можливості при моделюванні конкретних предметних областей.

Система містить засоби автоматичної оптимізації запитів, що базуються на оцінці вартості виконання, з врахуванням статистики, зібраної для таблиць і стовпців. Реалізовано підтримку індексів різноманітних видів, включно з B-деревами, хеш-структурами, GiST та GIN, що сприяє ефективній обробці пошукових запитів як точкового, так і повнотекстового характеру. Розподілена архітектура, з можливістю горизонтального масштабування та реплікації, гарантує стабільну роботу PostgreSQL за великих навантажень, забезпечуючи високий рівень доступності.

Особливу увагу приділено засобам забезпечення безпеки, зокрема, багаторівневій аутентифікації, гнучкому управлінню правами доступу, підтримці SSL та механізмам шифрування. Підтримка тригерів, процедур і функцій, що виконуються на стороні сервера, дає змогу інтегрувати логіку обробки даних безпосередньо в межах СУБД, що підвищує узгодженість даних і знижує навантаження на прикладні рівні.

PostgreSQL відрізняється високим рівнем стандартизації, забезпечуючи відповідність мові SQL згідно зі специфікаціями ISO/IEC. Система активно розвивається в рамках відкритого проєкту, що гарантує своєчасне оновлення та наявність широкого спектру супутніх інструментів та розширень. Поєднання продуктивності, гнучкості та відповідності стандартам робить PostgreSQL широко використовуваною в академічних, промислових та урядових системах, що висувають підвищені вимоги до надійності та масштабованості.

У кваліфікаційній роботі PostgreSQL використовується для реалізації бази даних з метою зберігання інформації щодо користувачів та їх відгуки про коректність роботи додатку, аунтефікації користувачів. Такий підхід забезпечує приватність користувача, та створює можливість покращення застосунку спираючись на побажання та відгуки клієнтів застосунку.

3.10 Бібліотека Google MediaPipe

Google MediaPipe є багатоплатформеною бібліотекою, призначеною для створення обчислювальних графів потокових мультимедійних даних в реальному часі, орієнтованою на задачі комп'ютерного зору, обробки аудіо та даних з сенсорів. Архітектура бібліотеки ґрунтується на принципі обчислювальних графів, де вузли, котрі називаються «калькуляторами», відповідають за окремі операції обробки або аналізу даних, а зв'язки між ними визначають порядок передачі інформації. Такий підхід забезпечує модульність, можливість повторного використання компонентів та високу продуктивність при обробці потоків даних з камер, мікрофонів чи інших датчиків [22].

MediaPipe підтримує апаратне прискорення обчислень через інтеграцію з графічними процесорами та мобільними нейронними прискорювачами, дозволяючи ефективно втілювати моделі машинного навчання безпосередньо на пристроях користувачів. Це зменшує затримку, зберігає конфіденційність даних та знижує навантаження на серверну інфраструктуру. Інструменти для оптимізації обчислювальних графів включають буферизацію кадрів, асинхронну обробку та можливість синхронізації частоти оновлення між різними потоками даних.

Бібліотека широко застосовується для розв'язання задач розпізнавання облич, відстеження рухів тіла, виявлення об'єктів, оцінки пози користувача та розпізнавання жестів. Системи, побудовані на основі MediaPipe, демонструють високу точність та стабільність в умовах складного знімання, що досягається за рахунок використання попередньо навчених моделей та вбудованих механізмів нормалізації даних. Підтримка C++, Python та платформ Android/iOS забезпечує гнучку інтеграцію в мобільні додатки, десктопні системи та вебсервіси.

4 РОЗРОБКА ДОДАТКУ

4.1 Розробка інтерфейсу

Розробка інтерфейсу користувача відіграє ключову роль у створенні системи автоматичного перекладу української жестової мови в текст. Інтерфейс має бути простим для сприйняття, доступним та комфортним у використанні, особливо для людей з вадами слуху. Під час розробки основна увага приділялася візуальній легкості, раціональному розташуванню елементів управління та загальній інтуїтивності взаємодії.

Для проєктування інтерфейсу використовувався хмарний сервіс Figma, який дав можливість створити інтерактивні макети з можливістю попереднього перегляду реакції інтерфейсу ще до програмування. Робота з Figma дозволила швидко змінювати структуру інтерфейсу, перевіряти зручність навігації та тестувати різні дизайнерські рішення.

Інтерфейс програми представлений кількома основними вікнами, які забезпечують зручну та інтуїтивну взаємодію користувача з системою. Основне вікно (рисунок 4.1) слугує стартовою точкою для запуску основних функцій застосунку. Воно містить меню навігації та базові елементи управління.

Вікно для перевірки відео з камери (рисунок 4.2) дозволяє користувачу налаштувати та протестувати підключену камеру, переконавшись у її коректній роботі перед початком перекладу жестів. Це забезпечує якісний вхідний відеопотік для подальшої обробки.

Вікно перекладу (рисунок 4.3) є центральним у роботі застосунку, тут відбувається безпосереднє відображення текстового перекладу жестів у спеціальному полі. Користувач має змогу в режимі реального часу бачити розпізнаний текст, що підвищує зручність і швидкість спілкування

На рисунку 4.1 зображено основне вікно застосунку.

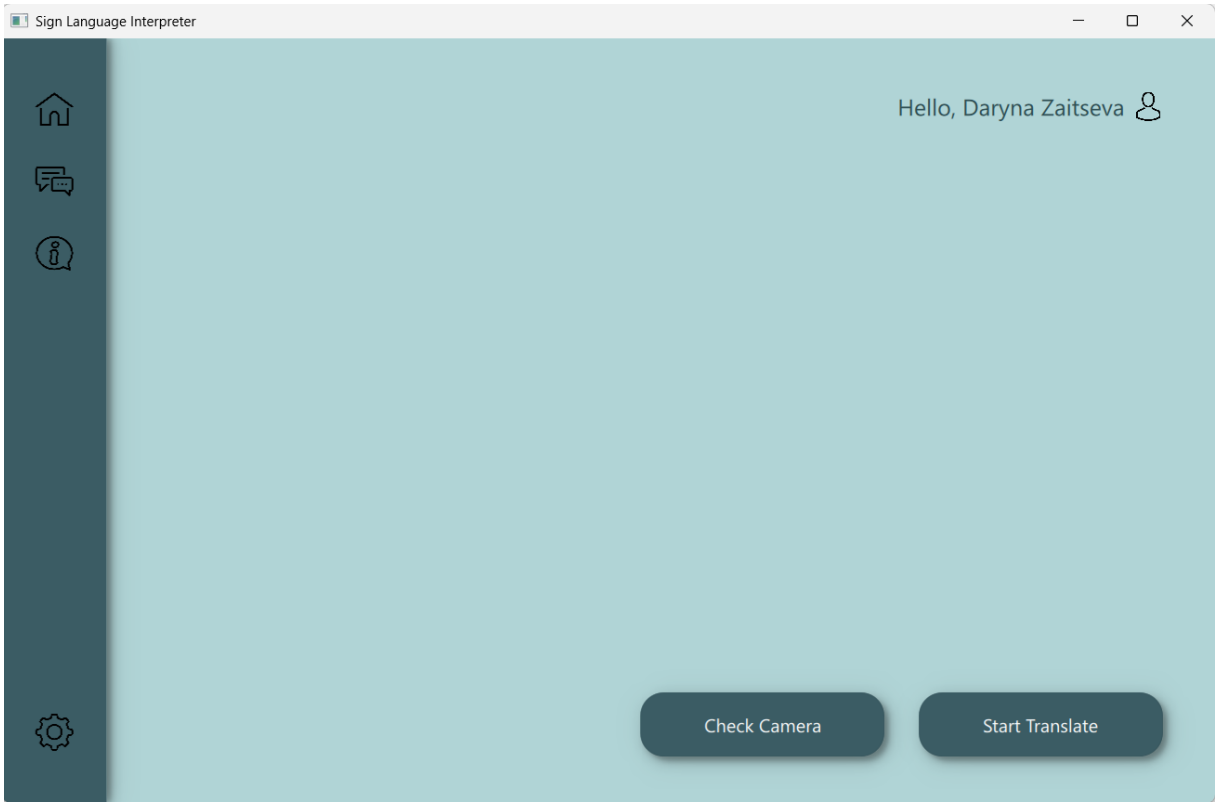

Рисунок 4.1 – Головне вікно застосунку

Зліва розташоване поле навігації по застосунку, яке є постійним елементом інтерфейсу на кожному екрані. Воно забезпечує швидкий доступ до основних розділів системи: головної сторінки, історії перекладів, інформації про застосунок та налаштувань. Така структура навігації сприяє зручності користування і дозволяє легко переходити між різними функціональними блоками без зайвих дій.

Справа вгорі розміщене вітальне повідомлення, яке персоналізує досвід користувача, звертаючись до нього на ім’я або іншим індивідуальним способом. Це створює більш дружню атмосферу та покращує взаємодію з програмою.

У нижній правій частині екрану знаходяться кнопки для виконання ключових дій – перевірки камери та запуску процесу перекладу. Їхнє зручне розташування робить управління основними функціями інтуїтивно

зрозумілим та швидким, що є важливим для користувачів під час роботи з жестовою мовою.

На рисунку 4.2 зображено екран для перевірки камери.

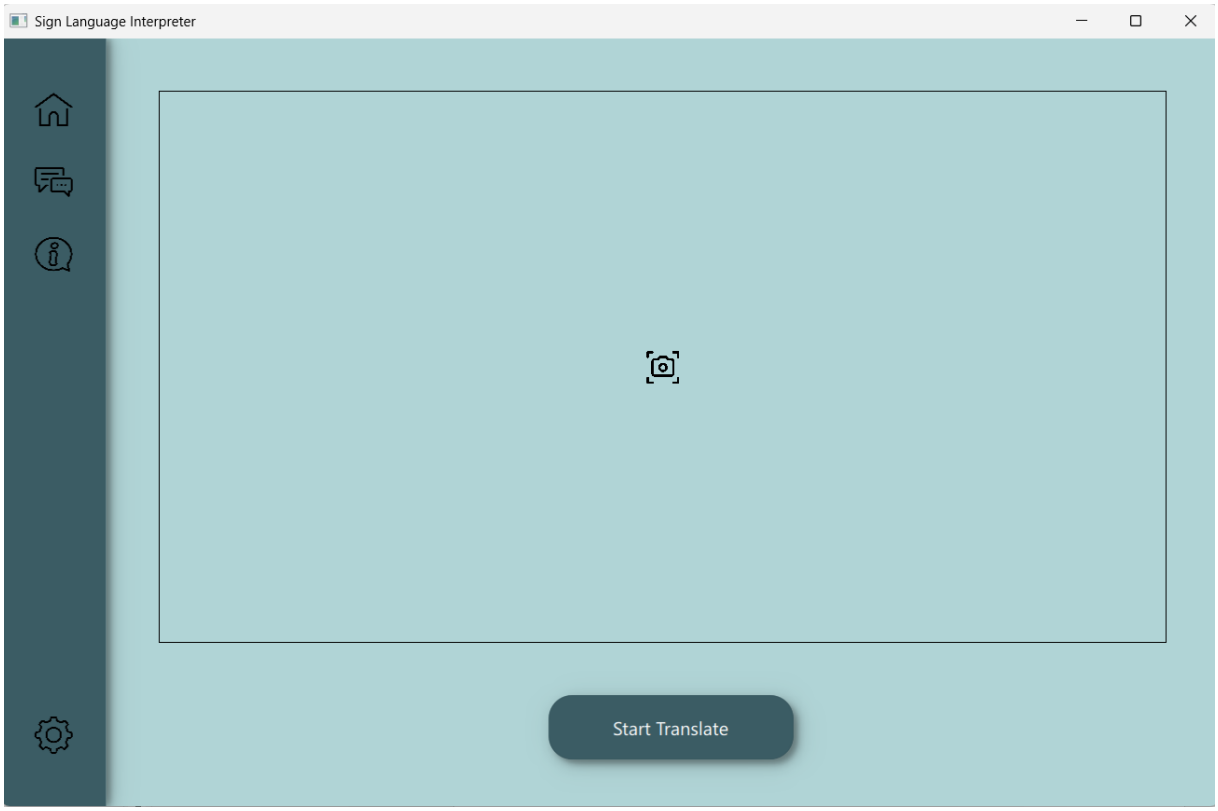

Рисунок 4.2 – Екран перевірки камери

На екрані для перевірки камери зліва розташовано бар навігації, який забезпечує швидкий доступ до інших розділів застосунку. У центрі екрану розміщене велике поле, призначене для відображення відеопотоку з камери, що дозволяє користувачу оцінити якість зображення та переконатися в коректній роботі пристрою. Нижня частина екрану містить кнопку для початку процесу перекладу, що робить запуск основної функції максимально простим і зручним. Така організація інтерфейсу сприяє інтуїтивній взаємодії та ефективному використанню застосунку.

Рисунок 4.3 демонструє сторінку для перекладу.

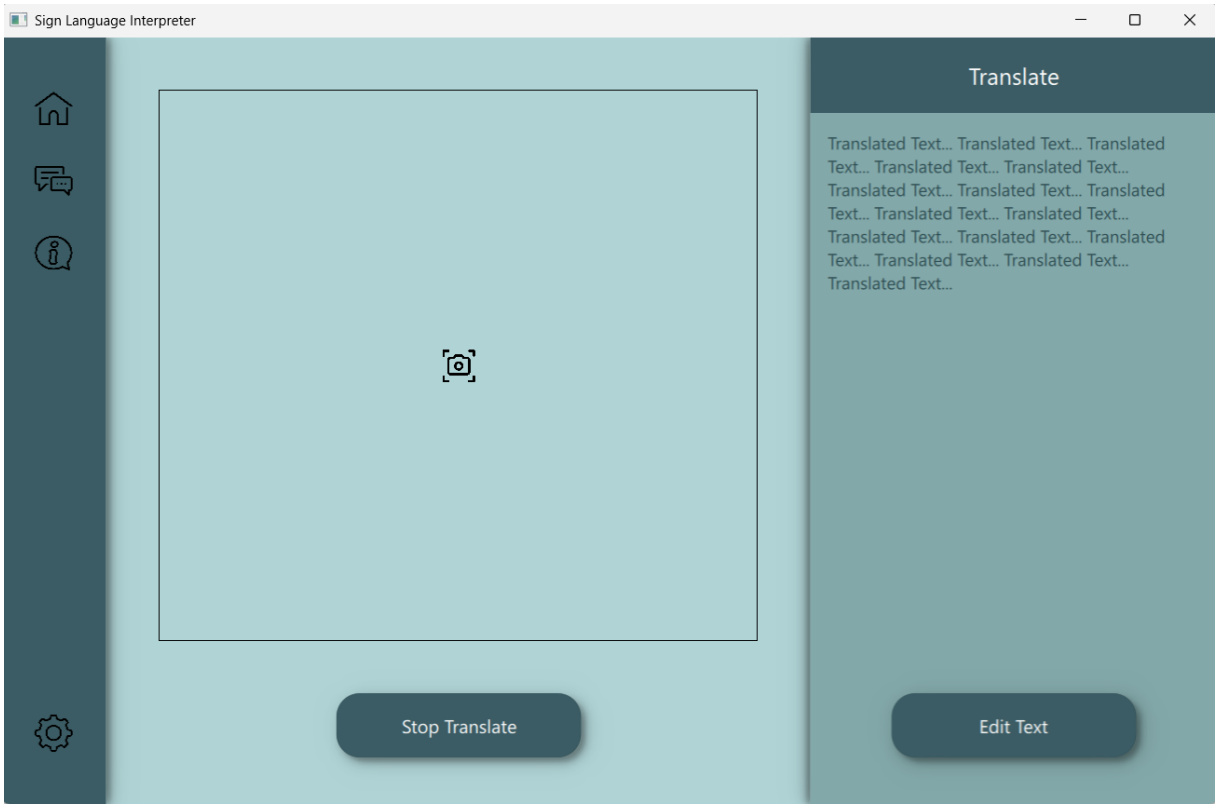

Рисунок 4.3 – Екран перекладу

На екрані перекладу зліва розташований бар навігації, який забезпечує легкий доступ до основних розділів застосунку. По центру розміщене поле, у якому в реальному часі демонструється зображення з камери, що дозволяє користувачу бачити процес розпізнавання жестів. Для зручності та комфорту користувача під відео-полем знаходиться кнопка для зупинки перекладу, що дає можливість контролювати роботу системи у будь-який момент.

Справа розташоване текстове поле, де виводиться переклад жестів у текстовій формі. Користувач має можливість коригувати цей текст, виправляючи помилки або неточності, що підвищує точність та якість комунікації. Такий інтерфейс створений з урахуванням потреб користувача, роблячи процес спілкування максимально зручним і адаптивним.

Візуальні компоненти було дібрано з метою уникнення перевантаження користувача надлишковими візуальними даними та

забезпечення зручного користування системою навіть для людини, що відкрила застосунок вперше. Інтерфейс було розроблено згідно з сучасними принципами інклюзивності, зрозумілості та практичності взаємодії, що гарантує доступність для різноманітної аудиторії.

4.2 Розробка моделі перекладу

Процес розробки моделі перекладу УЖМ супроводжувався низкою технічних, лінгвістичних та організаційних викликів, що вплинули як на вибір архітектури, так і на ефективність кінцевої системи. Основною задачею було створення моделі, здатної ідентифікувати жест за відеофрагментом тривалістю від 1 до 4 секунд та зіставити його з відповідним словом української мови.

Головною перешкодою у розробці такої моделі є відсутність даних для навчання, а саме коротких відео типу жест–переклад, тож першим етапом у розробці моделі є пошук даних. З метою отмання датасету було надіслано листи таким платформам та проєктам як: «Dictio» [23], «SpreadTheSign», «Відчуй» та «Українське товариство глухих». Відповідь та дані для навчання надала лише платформа «Dictio». Від підтримки проєкту було отримано 500 різних відео з жестами, набір є різноманітним але навчання моделі з мінімальною точністю потребує набору з неменше ніж 6 відео на один жест. Решта екземплярів збиралася самостійно. Матеріал фіксувався у стабільних умовах освітлення та з однакового ракурсу, однак варіативність виконання жестів залишалася обмеженою, що ускладнило формування моделі, стійкої до змін середовища та індивідуальних особливостей користувача. Застосовувались методи аугментації відеоданих: змінювався масштаб, дзеркальність, яскравість та інколи частково варіювався фон, щоб моделювати реальніші умови використання системи. Для обробки було обрано та підготовано 13

частовживаних жестів, що начиться розпізнавати перша версія моделі, розроблена у рамках кваліфікаційної роботи.

Другим етам є підготовка даних за допомогою бібліотек TensorFlow та MediaPipe.

Основною архітектурною основою для розпізнавання рухів було обрано конвеєр на базі Google MediaPipe Hands у поєднанні з LSTM-мережею, що обробляла послідовності координат основних точок руки. Такий підхід забезпечив компроміс між швидкістю обробки і здатністю враховувати часову динаміку жесту. У подальших експериментах також випробовувалась 1D-CNN модель для класифікації ряду координат, однак LSTM продемонструвала вищу стабільність при обробці подовжених жестів. Для підвищення точності використовувались техніки нормалізації координат за розміром руки та відносним розташуванням у кадрі. MediaPipe Hands – це модель, яка в реальному часі розпізнає руки на відео і визначає 21 трекінгову точку на кожній руці. Кожна з точок включає координати (x, y, z) для кожного суглоба пальців і зап’ястя. Таким чином було отримано вихід для кожного кадру як:

$$A = 21 \times 3 \times b, \tag{4.1}$$

де 21 – кількість трекінгових точок;

3 – кількість координат;

b – кількість рук, що використовуються у жесті.

Приклад коду реалізації етапу наведено у додатку Б у лістингу Б.1.

Після отримані дані стосовно кожного кадру було сформовано у послідновність (тензор) з 30 елементів та отримано позлідовність елементів, кожен з яких має 63/126 чисел. Приклад коду реалізації етапу наведено у додатку Б у лістингу Б.2.

Наступним етапом є навчання LSTM-мережі розпізнавати жести УЖМ, класифікувати на основі патерну руху. Приклад навчання мережі описано у лістингу 4.1.

Лістинг 4.1 – Програмний код навчання LSTM-мережі

```
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=30,
validation_data=(X_test, y_test))
```

Одним із ключових недоліків моделі виявилася її недостатня узагальнюваність. Жести, виконані іншими людьми або у змінених умовах освітлення, розпізнавалися зі значно нижчою точністю, що свідчить про обмеження в наборі навчальних прикладів та брак варіантності до зовнішніх факторів. Також залишилася невирішеною проблема обробки жестів з контекстною неоднозначністю – модель не враховує лінгвістичний контекст, а отже, не здатна відрізнити багатозначні жести, які мають різне значення залежно від позиції у реченні.

Попри ці обмеження, модель продемонструвала придатність до інтеграції в реальний застосунок, орієнтований на переклад одиничних слів з УЖМ. У подальшому передбачається розширення набору даних, впровадження трансформерних архітектур для врахування контексту, а також використання технік few-shot learning для кращої адаптації моделі до нових користувачів.

4.3 Результати розробки

У результаті проведного дослідження було навчено LSTM-мережу та розроблено дестктопний застосунок для ОС Windows, що здатен перекладати обмежену кількість слів УЖМ, перетворювати окремі слова у

зв’язні речення та, завдяки впровадженому спеціальному жесту, що зображеному на рисунку 4.4 розпізнавати кінець речення.

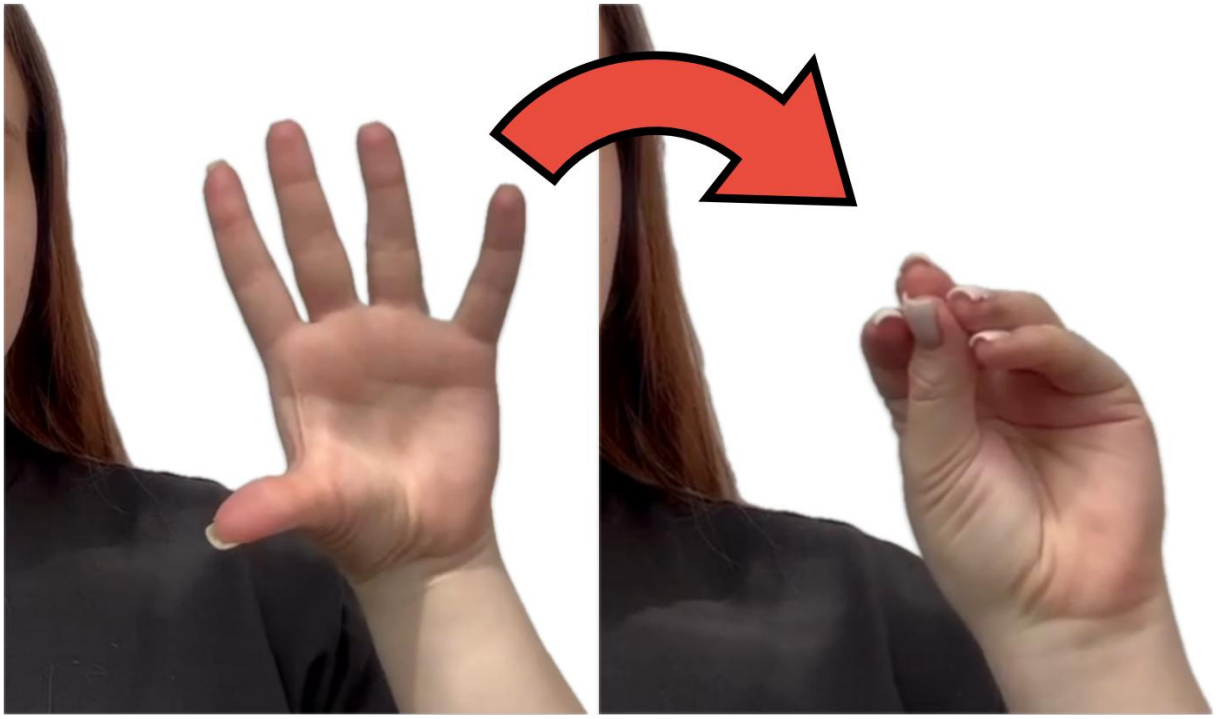

Рисунок 4.4 – Жест позначення кінця речення

Також застосунок має базу даних застосовану для зберігання даних щодо користувачів та їх відгути стосовно роботи застосунку.

Створена програма здатна перекладати наразі 13 часто вживаних жестів, а саме: привіт, буль ласка, дякую, перекладач жестової мови, це, ви, мати (дієслово), питання, я, мене звати, глухий, бувай та вищезазначений жест для позначення кінця речення. Користувач має можливість озвучити перекладений моделлю текст.

Для комунікації та майбутного поліпшення застосунку було реалізовано можливість залишити відрук користувачем, після відправлення повідомлення залишиться у базі даних.

Приклад роботи застосунку зображено на рисунках 4.5 та 4.6.

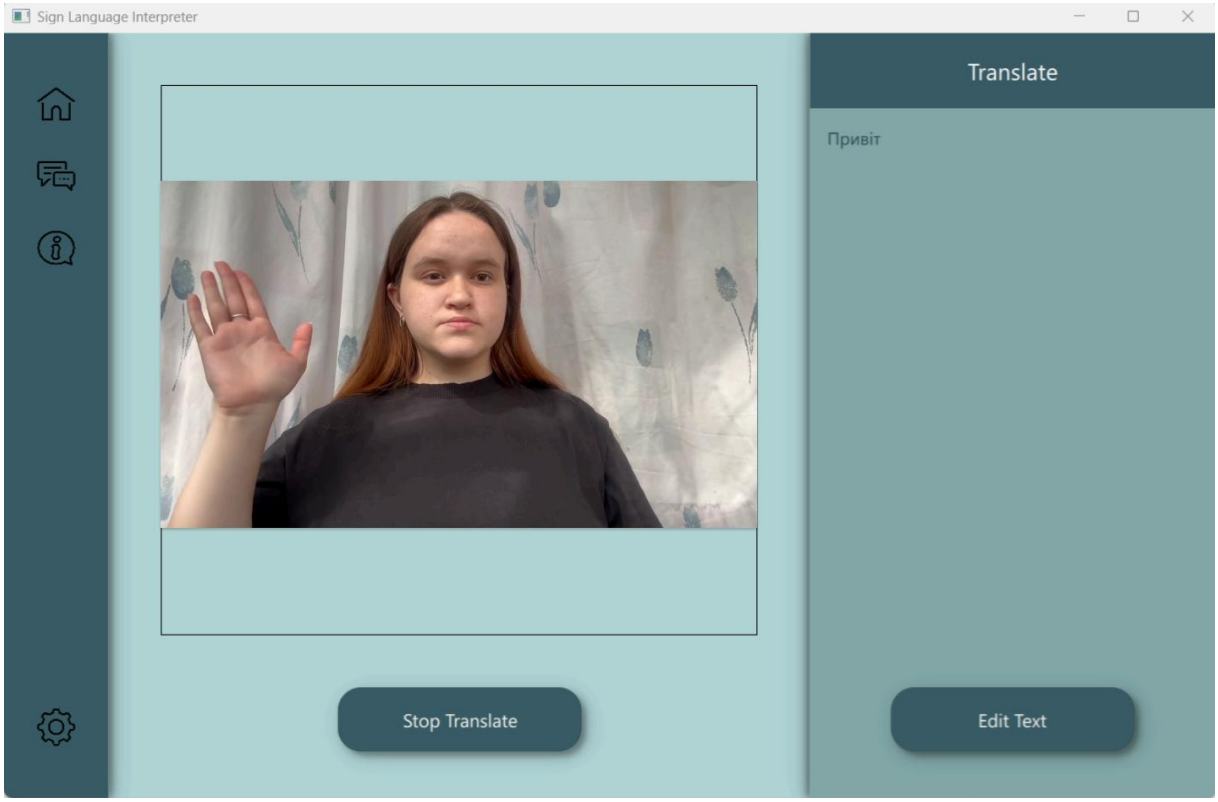

Рисунок 4.5 – Демонстрація роботи застосунку (1)

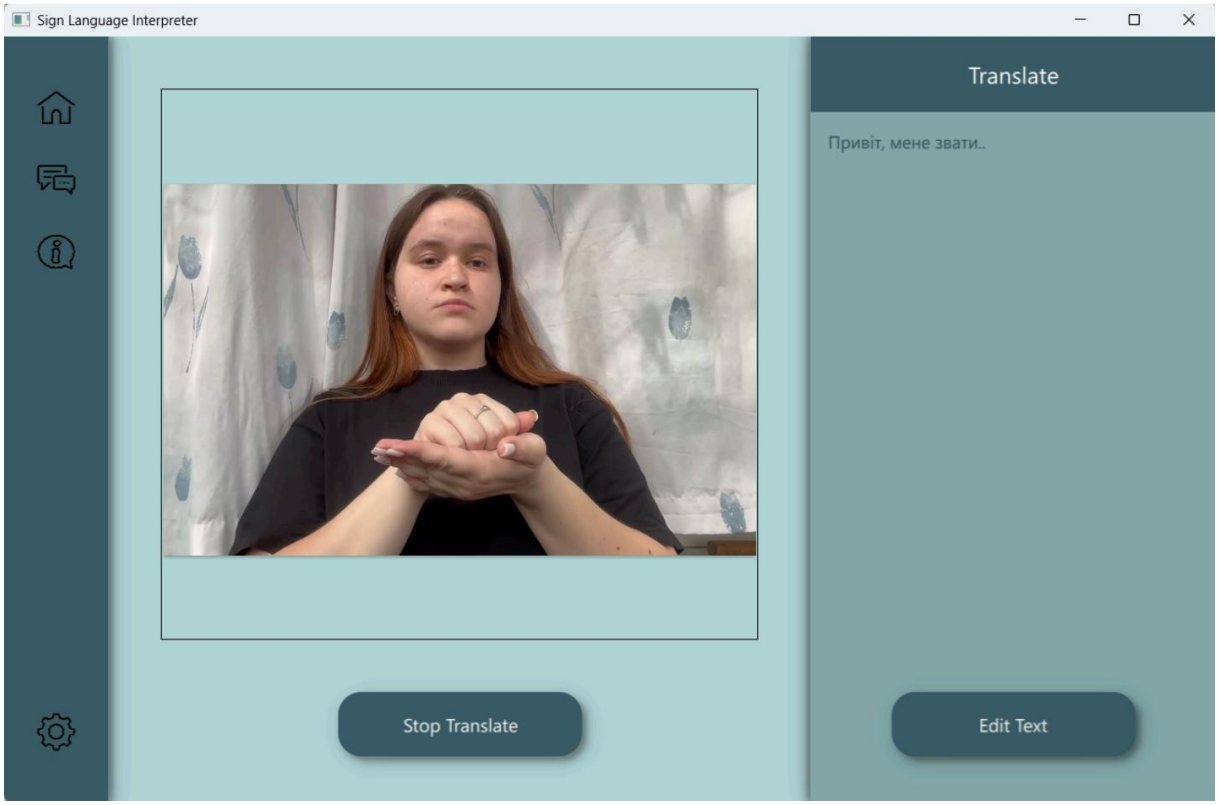

Рисунок 4.6 – Демонстрація роботи застосунку (2)

Як видно з рисунків 4.5 та 4.6, інтерфейс програми є простим і зручним у використанні, що забезпечує інтуїтивну навігацію навіть для користувачів з мінімальним досвідом роботи з подібними системами. Користувач має можливість легко обрати необхідний режим перекладу, а також переглянути розпізнаний текст у спеціальному полі, що значно спрощує процес комунікації. Результати розпізнавання жестів відображаються у чіткій та зрозумілій формі, що дає змогу швидко орієнтуватися у змісті повідомлення та при потребі оперативно коригувати текст. Такий підхід робить застосунок не тільки функціональним, а й максимально комфортним для повсякденного використання.

Функціональність програми дозволяє не лише забезпечити переклад жестів, а й створює основу для інтерактивної взаємодії з користувачем. Завдяки реалізованій системі зворотного зв’язку, користувачі можуть повідомляти про помилки, залишати пропозиції щодо покращення якості розпізнавання або додавання нових жестів, що сприятиме подальшому вдосконаленню застосунку. Зібрані відгуки зберігаються в базі даних і можуть бути використані для аналізу ефективності системи та визначення напрямів її розвитку.

Таким чином, створений застосунок демонструє базову, але водночас функціональну реалізацію системи перекладу української жестової мови, що успішно виконує основні завдання, а саме: розпізнавання жестів, їх трансформацію у текст та голосове відтворення. Застосунок має значний потенціал для подальшого масштабування, включаючи розширення словника жестів, підвищення точності розпізнавання за рахунок вдосконалення моделей, а також інтеграцію з іншими цифровими сервісами і платформами. Це відкриває широкі можливості для покращення комунікації та соціальної адаптації людей з порушеннями слуху, роблячи технології більш доступними і корисними у повсякденному житті.

ВИСНОВКИ

У процесі створення системи автоматичного перекладу української жестової мови в письмову та голосову форми було досягнуто поставленої мети: розроблено програмне забезпечення, яке забезпечує двосторонній зв'язок між людьми з вадами слуху та тими, хто не володіє жестовою мовою. Запропонована система вигідно відрізняється від наявних аналогів саме орієнтацією на українську жестову мову, що є критично важливим для соціальної інтеграції в Україні. Система функціонує автономно, не потребуючи інтернет-з'єднання чи додаткових сенсорів, що дає змогу використовувати її на звичайному персональному комп'ютері, не вдаючись до спеціального обладнання. Важливим є також той факт, що реалізовано переклад жестів не лише в текст, але й у мову, що є важливим кроком у напрямку інклюзії для людей, які мають порушення не лише слуху, але й зору.

Сильними сторонами розробленої системи також є використання сучасних технологій комп'ютерного зору та глибокого навчання, що забезпечує високу точність розпізнавання жестів у реальному часі. Архітектура застосунку розроблена з урахуванням гнучкості, можливості розширення та простоти підтримки, а інтерфейс створено з урахуванням потреб кінцевого користувача.

Водночас, під час роботи над проєктом виявлено деякі обмеження. Система продовжує залежати від якості вхідного відео, освітлення та положення рук, що може впливати на точність розпізнавання в умовах, які важко контролювати. Обмеженою є також кількість жестів, які на даний момент підтримуються, оскільки навчання моделі базувалося на доступному наборі даних, який не охоплює всю повноту української жестової мови. Крім того, процес інтеграції моделі машинного навчання з інтерфейсом та програмною логікою вимагав значної узгодженості між різними технологіями та мовами програмування, що потенційно може

ускладнити майбутнє розширення проєкту без ретельної документації та підтримки.

Загалом, розроблений застосунок демонструє практичну реалізацію ідей, що були закладені в теоретичній частині роботи, та має перспективи для подальшого розвитку, включаючи розширення словника жестів, оптимізацію моделей, адаптацію до мобільних платформ та впровадження нових функцій відповідно до потреб користувачів.

У майбутньому словникова база застосунку буде розширена, моделі оптимізовні до відео поганої якості та недостатнього освітлення. Для більшої мобільності та зручності застосунок варто адаптувати під платформи Android та Apple.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sign language translation: a survey of approaches and techniques. *MDPI*. URL: https://www.mdpi.com/2079-9292/12/12/2678 (дата звернення: 21.09.2024).

2. MediaPipe studio | google AI edge | google AI for developers. *Google AI for Developers*. URL: https://ai.google.dev/edge/mediapipe/solutions/studio (дата звернення: 21.09.2024).

3. Method. *ar5iv*. URL: https://ar5iv.labs.arxiv.org/html/2408.09311 (дата звернення: 21.09.2024).

4. SignAll. *Future of Interface*. URL: https://futureofinterface.org/signall/ (дата звернення: 23.09.2024).

5. SignAll SDK: Sign language interface using MediaPipe is now available for developers- Google Developers Blog. *Home - Google Developers Blog*. URL: https://developers.googleblog.com/en/signall-sdk-sign-language-interface-using-mediapipe-is-now-available-for-developers/ (дата звернення: 24.09.2024).

6. KinTrans movement tech turns motion into voice, text–and translates sign language. *Dallas Innovates*. URL: https://dallasinnovates.com/kintrans-sign-language-tech-translates-movements-text-voice/ (дата звернення: 24.09.2024).

7. MotionSavvy UNI: 1st sign language to voice system. *Indiegogo*. URL: https://www.indiegogo.com/projects/motionsavvy-uni-1st-sign-language-to-voice-system#/ (дата звернення: 24.09.2024).

8. Developing real-time, automatic sign language detection for video conferencing. *Google Research - Explore Our Latest Research in Science and AI*. URL: https://research.google/blog/developing-real-time-automatic-sign-language-detection-for-video-conferencing/ (дата звернення: 24.09.2024).

9. Software Architecture Patterns: What Are the Types and Which Is the Best One for Your Project | Turing. *Turn AGI Research into Real-World*

Impact | Turing. URL: https://www.turing.com/blog/software-architecture-patterns-types (дата звернення: 20.03.2025).

10. Walpita P. Software Architecture Patterns–Layered Architecture. *Medium*. URL: https://priyalwalpita.medium.com/software-architecture-patterns-layered-architecture-a3b89b71a057 (дата звернення: 20.03.2025).

11. Client-server architecture. *Online courses from QATestLab | Main page*. URL: https://en.training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення: 20.03.2025).

12. GeeksforGeeks. MVC Architecture - System Design - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/mvc-architecture-system-design/ (дата звернення: 20.03.2025).

13. Architecture Diagram Basics & Best Practices - vFunction. *vFunction*. URL: https://vfunction.com/blog/architecture-diagram-guide/ (дата звернення: 20.03.2025).

14. User Flows. Як ця техніка допомагає в роботі над проєктами. *DOU*. URL: https://dou.ua/lenta/columns/user-flows/ (дата звернення: 21.03.2025).

15. Publishing P. C++ Programming Language 5th Edition. Independently Published, 2019 (дата звернення: 10.04.2025).

16. Qt Documentation | Home. Qt Documentation | Home. URL: https://doc.qt.io/ (дата звернення: 10.04.2025).

17. TensorFlow. *TensorFlow*. URL: https://www.tensorflow.org/ (дата звернення: 10.04.2025).

18. OpenCV documentation index. *OpenCV documentation index*. URL: https://docs.opencv.org (дата звернення: 10.04.2025).

19. CMake Documentation and Community. *CMake - Upgrade Your Software Build System*. URL: https://cmake.org/documentation (дата звернення: 10.04.2025).

20. Hoffman B., Martin K. Mastering CMake. Kitware, Incorporated, 2015. 706 с.

21. PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org/ (дата звернення: 29.05.2025).

22. MediaPipe Solutions guide | Google AI Edge | Google AI for Developers. *Google AI for Developers*. URL: https://ai.google.dev/edge/mediapipe/solutions/guide (дата звернення: 29.05.2025).

23. Dictio. *Dictio*. URL: https://www.dictio.info/ (дата звернення: 29.05.2025).