ДОДАТОК А

Код програми

```
# main.py

from tts import init_tts_client
from logs import check_logs, get_logs, checknout
import concurrent.futures
import json

# Глобальний клієнт TTS
tts_client = init_tts_client()
logs = get_logs(url="http://127.0.0.1:5000/logs")

def handle_recognized_text(text):
    # Тут обробляйте розпізнаний текст
    print("Розпізнано:", text)

def callback_function(result):
    result_json = json.loads(result)
    if 'text' in result_json:
        recognized_text = result_json['text']

def listen_and_respond():

    with concurrent.futures.ThreadPoolExecutor() as executor:
        future = executor.submit(check_logs)
        result = future.result()  # Отримуємо результат від потоку
```

```
            # Запускаємо checknout, якщо результат check_logs == True
            if result:
                checknout()
        if __name__ == "__main__":

            listen_and_respond()

# Модель для синтезу мови
from google.cloud import texttospeech
import os
import sounddevice as sd
import numpy as np

#шлях до мого облікового запису гугл клауд
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] =
"D:\Projects\IVsSystem\IVS_CODE.json"


# Ініціалізація клієнта Text-to-Speech
def init_tts_client():
    return texttospeech.TextToSpeechClient()
def text_to_speech(client, ssml_text):
    # Використання SSML для включення пауз
    # ssml_text = "<speak>" + text.replace(".", ".<break time='1000ms'/>") +
"</speak>"
    synthesis_input = texttospeech.SynthesisInput(ssml=ssml_text)
    # Конфігурація голосу
    voice = texttospeech.VoiceSelectionParams(language_code="uk-UA",
ssml_gender=texttospeech.SsmlVoiceGender.NEUTRAL)
    # Конфігурація аудіо (зі змінною швидкості мовлення)
```

```
    audio_config =
texttospeech.AudioConfig(audio_encoding=texttospeech.AudioEncoding.LINEAR
16, speaking_rate=0.8, pitch=1.0)
    # Запит на синтез мовлення
    response = client.synthesize_speech(input=synthesis_input, voice=voice,
audio_config=audio_config)
    # Конвертація аудіо відповіді в формат NumPy
    audio_data = np.frombuffer(response.audio_content, dtype=np.int16)
    # Відтворення аудіо
    sd.play(audio_data, 24000)# 24000 - частота дискретизації
#
    sd.wait()   # Чекаємо завершення відтворення



# logs.py



  import requests
  import time
  from threading import Event
  from input_voice import va_listen
  import Fuzz
  from tts import text_to_speech, init_tts_client
  import kommanden

  # Глобальна подія для сигналізації про завершення
  log_thread_finished = Event()

  # URL маршруту логів на вашому Flask-сервері
```

```
log_url = "http://127.0.0.1:5000/logs"


def process_recognized_text(text):
    category = Fuzz.process_command(text)
    if category:
        kommanden.execute(category)
    else:
        print("Команда не розпізнана.")

def get_logs(url):
    try:
        response = requests.get(url)
        response.raise_for_status()  # Перевірка на наявність помилок HTTP
        return response.json()  # Повернення логів у форматі JSON
    except requests.RequestException as e:
        print(f"Помилка при запиті: {e}")
        return None

def checknout(): #Функція для обробки останнього логу, прослуховування мікро та виведення інформації щодо вирішення помилки
    va_listen(process_recognized_text)


def check_logs():
    error_detected = False

    # Оновлення логів до тих пір, поки printing не стане False
    while error_detected == False:
```

```
        global log_thread_finished
        tts_client = init_tts_client()
        logs = get_logs(log_url)

        if logs is not None:
            # Перевірка останнього запису в логах
            latest_log = logs[-1]
            printing_status = latest_log.get("printing")
            error_status = latest_log.get("error")
            current_tape_length_status = latest_log.get("length")

           # print(f"Latest log - Printing status: {printing_status},"
            #      f" Error status: {error_status},"
              #   f" length: {current_tape_length_status}")

            if current_tape_length_status == "50":
                # Якщо залишилось 10 відсотків
                text_to_speech(tts_client, f"Залишилось 50%:
{current_tape_length_status}")
                print("Залишилось 50%")

            # Переривання циклу, якщо error має помилку
#
            if logs and any(log.get("printing") is False for log in logs):
                # Зупинка слухання та виведення питання



                response = input("Сталася помилка, чи хочете ви продовжити?
(Y/N): ")
                if response.lower() == 'y':
```

```
                if error_status == "Tape out":
                    # Якщо скінчилась стрічка
                    text_to_speech(tts_client, f"Скінчилась стрічка:
{current_tape_length_status}")




                if error_status == "Error":
                    # Якщо є помилка, відправити повідомлення через TTS
                    text_to_speech(tts_client, f"Помилка принтера:
{error_status}")

                    error_detected = True
                    return True
            else:
                log_thread_finished.set()
                return False



            # elif response.lower() == 'n':
            #    log_thread_finished.set()

        else:
            print("No logs received.")

        if error_detected == False:
            time.sleep(5)  # Затримка перед наступним запитом, якщо помилка
```

```
не була знайдена

    #   log_thread_finished.set()



def check_logs():
    error_detected = False

    # Оновлення логів до тих пір, поки printing не стане False
    while error_detected == False:

        global log_thread_finished
        tts_client = init_tts_client()
        logs = get_logs(log_url)

        if logs is not None:
            # Перевірка останнього запису в логах
            latest_log = logs[-1]
            printing_status = latest_log.get("printing")
            error_status = latest_log.get("error")
            current_tape_length_status = latest_log.get("length")

           # print(f"Latest log - Printing status: {printing_status},"
            #      f" Error status: {error_status},"
              #   f" length: {current_tape_length_status}")

            if current_tape_length_status == "50":
                # Якщо залишилось 10 відсотків
```

```
            text_to_speech(tts_client, f"Залишилось 50%:
{current_tape_length_status}")
            print("Залишилось 50%")

        # Переривання циклу, якщо error має помилку
#
        if logs and any(log.get("printing") is False for log in logs):
            # Зупинка слухання та виведення питання


            response = input("Сталася помилка, чи хочете ви продовжити?
(Y/N): ")
            if response.lower() == 'y':

                if error_status == "Tape out":
                    # Якщо скінчилась стрічка
                    text_to_speech(tts_client, f"Скінчилась стрічка:
{current_tape_length_status}")




                if error_status == "Error":
                    # Якщо є помилка, відправити повідомлення через TTS
                    text_to_speech(tts_client, f"Помилка принтера:
{error_status}")

                    error_detected = True
                    return True
            else:
```

```
            log_thread_finished.set()
            return False


from tts import text_to_speech, init_tts_client
import requests


log_url = "http://127.0.0.1:5000/logs"

def get_logs(url):
    try:
        response = requests.get(url)
        response.raise_for_status()  # Перевірка на наявність помилок HTTP
        return response.json()  # Повернення логів у форматі JSON
    except requests.RequestException as e:
        print(f"Помилка при запиті: {e}")
        return None

def execute(category):
    global log_thread_finished
    tts_client = init_tts_client()

    logs = get_logs(log_url)

    # Перевірка останнього запису в логах
    latest_log = logs[-1]
    error_status = latest_log.get("error")
    current_tape_length_status = latest_log.get("length")
```

```
    if category == "Помилка":

        if current_tape_length_status == "0":
            # Якщо скінчилась стрічка
            text_to_speech(tts_client, f"Треба замінити стрічку:
{current_tape_length_status}")

        if error_status == "Error":
            # Якщо є помилка, відправити повідомлення через TTS
            text_to_speech(tts_client, f"Помилка принтера. {error_status} Ось три
найчастіші проблеми:  "
                           f"1) Проблеми з підключенням та мережею:
Перевірте мережеві кабелі та обладнання, перезавантажте мережеві
пристрої, перенастроюйте принтер у мережі."
                           f"2) Засмічення або знос обладнання: Проведіть
чистку принтера, зокрема друкуючої головки, замініть зношені частини,
використовуйте рекомендовані витратні матеріали."
                           f"3) Збій у програмному забезпеченні:
Перезавантажте принтер, встановіть доступні оновлення ПЗ, за потреби
перевстановіть ПЗ або зверніться до техпідтримки.")


from flask import Flask, jsonify, request, render_template
from flask_cors import CORS
import time
import threading
#

#   curl http://127.0.0.1:5000/start_printing
```

```
#   curl http://127.0.0.1:5000/status

#   curl http://127.0.0.1:5000/stop_printing

# curl http://127.0.0.1:5000/logs



app = Flask(__name__)
CORS(app)
# Список для логів
logs = []

# Початкові параметри принтера
printer_state = {
    "total_tape_length": 200, # загальна довжина стрічки в метрах
    "length": 200, # Актуальна довжина стрічки
    "printing_speed": 40, # Швидкість друку у метрах про хвилину
    "printing": False, # Стан принтера
    "error": None # Помилка, якщо є
}


print_timer = None

last_sent_update = 0  # Змінна для відстеження останнього відправленого оновлення

def print_job():
    global print_timer, last_sent_update
```

```
    while printer_state["printing"]:
        printer_state["length"] -= printer_state["printing_speed"]
        current_usage_percentage = (1 - printer_state["length"] /
printer_state["total_tape_length"]) * 100

        # Перевірка чи використано більше 24 метрів стрічки  (настройка
значень при яких зупиняється принтер)
        if printer_state["total_tape_length"] - printer_state["length"] > 200:
            printer_state["printing"] = False
            printer_state["error"] = "Error"

        # Якщо закінчилась стрічка то ми отримуемо помилку
        if printer_state["length"] <= 0:
            printer_state["length"] = 0
            printer_state["printing"] = False
            printer_state["error"] = "Tape out"
            break

        # Інформація о стрічці кожні 5 секунд
        if print_timer is None:
            print_timer = threading.Timer(5, print_tape_info)
            print_timer.start()

        time.sleep(5)

# Функция для додавання логів
def add_log():
    while True:
        log_entry = {
```

```
            "time": time.ctime(),
            "length": printer_state["length"],
            "printing": printer_state["printing"],
            "error": printer_state["error"]
        }
        logs.clear()  # Очистка попередніх логів
        logs.append(log_entry)
        time.sleep(5)

# Запуск потока для регулярного додавання логів
log_thread = threading.Thread(target=add_log)
log_thread.start()
def print_tape_info():
    global print_timer
    used_length = printer_state["total_tape_length"] - printer_state["length"]
    log_entry = {
        "time": time.ctime(),
        "message": f"Використано {used_length} метрів стрічки",
        "length": printer_state["length"],
        "printing": printer_state["printing"]
    }
    logs.append(log_entry)
    print_timer = None



@app.route('/start_printing')
def start_printing():
    if not printer_state["printing"] and printer_state["length"] > 0:
        printer_state["printing"] = True
        printer_state["error"] = None
```

```
        threading.Thread(target=print_job).start()
        return jsonify({"message": "Printing started"})
    else:
        return jsonify({"error": "Cannot start printing"})

@app.route('/stop_printing')
def stop_printing():
    printer_state["printing"] = False
    return jsonify({"message": "Printing stopped"})

@app.route('/status')
def status():
    return jsonify({
        "length": printer_state["length"],
        "printing": printer_state["printing"],
        "error": printer_state["error"]
    })

@app.route('/logs')
def get_logs():
    # Повертаєм останні 20 записів лога
    return jsonify(logs)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

```
import sounddevice as sd
import queue
from vosk import Model, KaldiRecognizer
import sys
import time

model = Model("small_model")
samplerate = 16000
device = 1

def va_listen(callback, listen_duration = 3):
    q = queue.Queue()

    def q_callback(indata, frames, time, status):

        if status:
            print(status, file=sys.stderr)
        q.put(bytes(indata))
#
    with sd.RawInputStream(samplerate=samplerate, blocksize=8000,
device=device, dtype='int16', channels=1, callback=q_callback):
        rec = KaldiRecognizer(model, samplerate)
        print("Запис почався")
        start_time = time.time()  # Записуємо час початку прослуховування
       # global keep_listening
       # keep_listening = True

        while True:
            current_time = time.time()
            if current_time - start_time > listen_duration:
```

```
            data = q.get()
            if rec.AcceptWaveform(data):
                callback(rec.Result())
                print(rec.Result())


from fuzzywuzzy import process

# Словник можливих запитань

komtegory = {

    "Помилка": [
        "Що таке", "Що сталося", "У чому проблема", "Чому стоїмо",
"Розповідай", "Що будемо робити", "Що знову"
                ]
}
def process_command(command_text):

    best_match = None
    highest_score = 0

    for category, command_variants in komtegory.items():
        match, score = process.extractOne(command_text, command_variants)
        if score > highest_score:
            highest_score = score
            best_match = category

    return best_match if highest_score >= 60 else None
```

ДОДАТОК Б

Демонстраційний матеріал

Формат	Позиція	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	Примітки
			Текстові документи		
А4	1	ГЮІК 501310.043 ПЗ	Пояснювальна записка	114	с.
			Інші документи		
	2		Додаток А	15	с.
			Код програми		
	3		Додаток Б	1	с.
			Демонстраційний		
			матеріал		

					ГЮІК 501310.043 ПЗ
Змін	Арк.	№докум.	Підпис	Дата	
Розробив		Терещук Д.О.			Розробка інтелектуальної голосової системи підтримки прийняття рішень у виробничому приміщені
Перевірив		МаксимоваС.С.			
Норм. контроль		СтародубцевН.Г.			
Затв.		НевлюдовІ.Ш.			

Літера	Аркуш	Аркуш
Н	1	1

Кафедра КІТАР ХНУРЕ