J



**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH**

**VIỆN ĐÀO TẠO CHẤT LƯỢNG CAO**

BÁO CÁO NGÔN NGỮ LẬP TRÌNH PYTHON

**Đề Tài: Voice Assistant (Trợ lý giọng nói)**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn : | Th.S Huỳnh Thanh Việt |
| Sinh viên thực hiện : | Nguyễn Minh Nguyệt – 079304044741  Nguyễn Thị Hồng Đào –  Nguyễn Xuân Tân  Trương Tuấn Thi |
|  |  |

Tp Hồ Chí Minh, ngày 29 tháng 3 năm 2025

**Mục Lục**

[**Lời mở đầu** 3](#_Toc199970740)

[**CHƯƠNG 1: TỔNG QUAN ỨNG DỤNG** 4](#_Toc199970741)

[**1.** **Giới thiệu ứng dụng trợ lý giọng nói (voice assistant)** 4](#_Toc199970742)

[**2.** **Thư viện sử dụng** 4](#_Toc199970743)

[**3.** **Luồng hoạt động** 4](#_Toc199970744)

[**CHƯƠNG 2: THÀNH PHẦN ỨNG DỤNG** 4](#_Toc199970745)

[**1.** **Speech-to-Text (STT) – speech\_recognition** 4](#_Toc199970746)

[**2.** **Mô hình ngôn ngữ lớn (LLM) – Ollama** 5](#_Toc199970747)

[**2.1 Bộ điều phối (wrapper)** 5](#_Toc199970748)

[**2.2 Bộ xử lý chính** 5](#_Toc199970749)

[**2.2.1 Khởi tạo** 5](#_Toc199970750)

[**2.2.2 Quản lý phiên cuộc trò chuyện** 6](#_Toc199970751)

[**2.2.3 Xử lý câu hỏi thời gian** 6](#_Toc199970752)

[**2.2.4 Xử lý câu hỏi chung** 8](#_Toc199970753)

[**2.2.5 Quản lý bộ nhớ hội thoại** 10](#_Toc199970754)

[**3.** **Text-to-Speech (TTS) – gTTS** 11](#_Toc199970755)

[**4.** **Lưu trữ cuộc trò chuyện – MongoDB** 11](#_Toc199970756)

[**4.1 Khởi tạo** 12](#_Toc199970757)

[**4.2 Hàm tạo cuộc hội thoại** 12](#_Toc199970758)

[**4.3 Hàm tạo tin nhắn vào cuộc hội thoại** 12](#_Toc199970759)

[**4.4 Hàm xóa phiên cuộc trò chuyện** 13](#_Toc199970760)

[**4.5 Hàm đóng kết nối** 13](#_Toc199970761)

[**5.** **Giao diện người dùng (GUI)** 13](#_Toc199970762)

[**CHƯƠNG 3: ĐÁNH GIÁ & KẾT QUẢ THỬ NGHIỆM** 14](#_Toc199970763)

[**CHƯƠNG 4: HƯỚNG PHÁT TRIỂN TƯƠNG LAI** 14](#_Toc199970764)

# **Lời mở đầu**

Ngày trước con người chúng ta chỉ biết tới công nghệ thông tin như một vực mới lạ và phức tạp. Nhưng khoảng những năm đổ lại đây, sự bùng nổ AI (trí tuệ nhân tạo) vào cuối những năm 2010 và bắt đầu phổ biến với toàn thế giới vào cuối năm 2022 và sự phát triển xử lý ngôn ngữ tự nhiên (NLP). Sự ra đời của những mô hình ngôn ngữ lớn GPT-3, PaLM, Gemini, LlaMA… Càng giúp việc tương tác giữa con người và máy tính đã trở nên tự nhiên và thân thiện hơn bao giờ hết.

Trong bài báo cáo này, nhóm chúng em sẽ đưa ra ứng dụng trợ lý giọng nói (Voice Assistant) được phát triển nhằm tạo ra một trợ lý ảo có khả năng hiểu và phản hồi bằng giọng nói tiếng Việt, giúp người dùng có thể tương tác một cách tự nhiên thông qua các cuộc hội thoại. Đây là một bước tiến quan trọng trong việc ứng dụng công nghệ AI vào cuộc sống hàng ngày, mang lại trải nghiệm người dùng thân thiện và hiệu quả.

# **CHƯƠNG 1: TỔNG QUAN ỨNG DỤNG**

## **Giới thiệu ứng dụng trợ lý giọng nói (voice assistant)**

Voice Assistant là một ứng dụng trợ lý ảo được phát triển bằng Python.

Có khả năng:

* Nhận diện giọng nói tiếng Việt từ người dùng
* Xử lý và hiểu ngữ nghĩa câu hỏi thông qua mô hình AI
* Đưa ra câu trả lời thông minh và phù hợp
* Chuyển đổi văn bản thành giọng nói để phản hồi
* Lưu trữ và quản lý lịch sử hội thoại

Ứng dụng được thiết kế với giao diện đồ họa thân thiện, cho phép người dùng dễ dàng tương tác thông qua việc nhấn nút và nói chuyện tự nhiên.

## **Thư viện sử dụng**

Xử lý giọng nói:

* speech\_recognition (Speech – to – text): Nhận diện giọng nói
* gTTS (Google Text-to-Speech): Chuyển đổi văn bản thành giọng nói
* playsound: Phát âm thanh

Xử lý AI và NLP:

* langchain\_ollama: Tích hợp với mô hình Ollama
* pymongo: Kết nối và quản lý cơ sở dữ liệu MongoDB

Giao diện người dùng:

* tkinter: Xây dựng GUI desktop
* threading: Xử lý đa luồng để tránh block UI

Tiện ích:

* datetime: Xử lý thời gian
* uuid: Tạo ID duy nhất cho session
* re: Xử lý biểu thức chính quy

## **Luồng hoạt động**

# **CHƯƠNG 2: THÀNH PHẦN ỨNG DỤNG**

## **Speech-to-Text (STT) – speech\_recognition**

Speech-to-Text (STT) là công nghệ giúp máy tính hoặc thiết bị có thể nghe được ngôn ngữ nói của con người bằng cách chuyển đổi âm thanh giọng nói thành chuỗi văn bản. STT đóng vai trò là bước đầu tiên để có thể tiếp tục xử lý ở các thành phần sau.

speech\_recognition là một thư viện Python kết hợp với PyAudio để có thể chuyển đổi giọng nói thành văn bản (Speech-to-Text). Hỗ trợ ở nhiều nền tảng, nhiều nguồn âm thanh, tích hợp nhiều engine nhận dạng giọng nói, có thể điều chỉnh xử lý tiếng ồn môi trường (adjust\_for\_ambient\_noise).

import speech\_recognition as sr

def listen(update\_status=None):

    r = sr.Recognizer() # PyAudio hỗ trợ

    with sr.Microphone() as mic: #sử dụng mic thu âm

        if update\_status: update\_status("Đang lắng nghe...")

# Điều chỉnh với tiếng ồn xung quanh (1 giây)

        r.adjust\_for\_ambient\_noise(mic, duration=1)

        try:

            # thời gian chờ 3 giây, nói 5 giây

            audio = r.listen(mic, timeout=3, phrase\_time\_limit=5)

            # Nhận dạng giọng nói tiếng Việt bằng google Speech Recognition

# đồi thành tiếng anh thì en-EN

            text = r.recognize\_google(audio, language='vi-VI')

            return text

        except:

            # không nhận thấy âm thanh hãy lỗi trả về none

            return None

* Tự động điều chỉnh độ nhạy microphone dựa trên tiếng ồn môi trường
* Timeout 3 giây để tránh treo ứng dụng, tức là không có âm thanh trong 3 giây thì sẽ tự động dừng.
* Giới hạn thời gian nói 5 giây cho mỗi lượt.
* Sử dụng Google Speech Recognition API cho độ chính xác cao.
* Hỗ trợ tiếng Việt (vi-VI) và cả tiếng anh (en-EN).

## **Mô hình ngôn ngữ lớn (LLM) – Ollama**

Ollama là một nền tảng mã nguồn mở giúp dễ dàng chạy các mô hình ngôn ngữ lớn (LLM) trên máy cục bộ hoặc server riêng. Trong dự án này, mô hình sử dụng là LLama3:8b

### **2.1 Bộ điều phối (wrapper)**

bộ điều phối trung gian (wrapper) cho toàn bộ hệ thống trợ lý ảo, giúp giữ tương thích với các đoạn mã cũ và đơn giản hóa cách sử dụng đối với các thành phần khác trong dự án.

def ask\_llm(message: str) -> str:

    try:

        assistant = get\_assistant()

        return assistant.ask\_llm(message)

    except Exception as e:

        return "Xin lỗi, có lỗi xảy ra. Vui lòng thử lại."

def cleanup():

    global \_assistant

    if \_assistant:

        \_assistant.close()

        \_assistant = None

### **2.2** **Bộ xử lý chính**

Bộ xử lý chính là thành phần trung tâm của hệ thống trợ lý ảo, chịu trách nhiệm quản lý toàn bộ quá trình tương tác giữa người dùng và mô hình ngôn ngữ lớn (LLM). Các thư viện được sử dụng:

* **Datetime:** lấy thời gian hiện tại
* **Locale:** cài đặt ngôn ngữ hiển thị ngày giờ
* **Re:** Làm sạch câu trả lời

#### **2.2.1 Khởi tạo**

Khởi tạo mô hình OllamaLLM với tham số model (mặc định "llama3:8b"). Kết nối bộ nhớ hội thoại qua lớp ConversationMemory sử dụng MongoDB để lưu trữ. Thiết lập locale Tiếng Việt để format ngày tháng phù hợp. Định nghĩa tập từ khóa liên quan đến các câu hỏi về thời gian.

 def \_\_init\_\_(self, model\_name: str = "llama3:8b", mongo\_uri: str = "mongodb://localhost:27017/"):

        """

        Khởi tạo LLM handler với memory

        """

        self.model = OllamaLLM(model=model\_name)

        self.memory = ConversationMemory(mongo\_uri)

        self.current\_session = None

        # Thiết lập locale tiếng Việt (fallback nếu không có)

        try:

            locale.setlocale(locale.LC\_TIME, 'vi\_VN.UTF-8')

        except:

            try:

                locale.setlocale(locale.LC\_TIME, 'Vietnamese\_Vietnam.1258')

            except:

                pass  # Sử dụng default locale

        # Từ khóa để nhận diện câu hỏi về thời gian

        self.time\_keywords = [

            'mấy giờ', 'bây giờ', 'hiện tại', 'thời gian',

            'ngày', 'tháng', 'năm', 'hôm nay', 'bữa nay',

            'thứ mấy', 'thứ', 'chủ nhật', 'chú nhật',

            'giờ', 'phút', 'giây', 'sáng', 'chiều', 'tối', 'đêm'

        ]

        # System prompt được tối ưu cho câu trả lời ngắn gọn

        self.system\_prompt = """Bạn là trợ lý AI người Việt Nam thông minh và thân thiện. 1. LUÔN trả lời bằng tiếng Việt 2. Trả lời NGẮN GỌN, tối đa 1-2 câu 3. Đi thẳng vào vấn đề, không lan man

"""

#### **2.2.2 Quản lý phiên cuộc trò chuyện**

Tạo session mới theo user\_id và thiết lập session hiện tại:

    def start\_new\_session(self, user\_id: str = "default\_user") -> str:

        self.current\_session = self.memory.create\_session(user\_id)

        return self.current\_session

    def set\_session(self, session\_id: str):

        self.current\_session = session\_id

#### **2.2.3 Xử lý câu hỏi thời gian**

Hàm \_is\_time\_question():này sẽ kiểm tra xem câu hỏi có liên quan tới thời gian.

    def \_is\_time\_question(self, message: str) -> bool:

        message\_lower = message.lower()

        return any(keyword in message\_lower for keyword in self.time\_keywords)

Hàm \_get\_current\_time\_info: Lấy thông tin thời gian hiện tại, gồm ngày tháng năm, thứ trong tuần, buổi trong ngày (sáng, chiều, tối, đêm).

    def \_get\_current\_time\_info(self) -> dict:

        """

        Lấy thông tin thời gian hiện tại

        """

        now = datetime.now()

        # Tên các thứ trong tuần

        weekdays = [

            'Thứ Hai', 'Thứ Ba', 'Thứ Tư', 'Thứ Năm',

            'Thứ Sáu', 'Thứ Bảy', 'Chủ Nhật'

        ]

        # Tên các tháng

        months = [

            '', 'Tháng Một', 'Tháng Hai', 'Tháng Ba', 'Tháng Tư',

            'Tháng Năm', 'Tháng Sáu', 'Tháng Bảy', 'Tháng Tám',

            'Tháng Chín', 'Tháng Mười', 'Tháng Mười Một', 'Tháng Mười Hai'

        ]

        # Xác định buổi trong ngày

        hour = now.hour

        if 5 <= hour < 12:

            period = "sáng"

        elif 12 <= hour < 18:

            period = "chiều"

        elif 18 <= hour < 22:

            period = "tối"

        else:

            period = "đêm"

        return {

            'datetime': now,

            'time': now.strftime("%H:%M:%S"),

            'time\_12h': now.strftime("%I:%M %p"),

            'date': now.strftime("%d/%m/%Y"),

            'weekday': weekdays[now.weekday()],

            'day': now.day,

            'month': months[now.month],

            'year': now.year,

            'period': period,

            'full\_date': f"{weekdays[now.weekday()]}, ngày {now.day} {months[now.month]} năm {now.year}"

        }

Hàm \_handle\_time\_question: Trả lời trực tiếp câu hỏi thời gian mà không gọi LLM, ví dụ trả lời giờ hiện tại, ngày hôm nay, tháng, năm.

   def \_handle\_time\_question(self, message: str) -> str:

        """

        Xử lý câu hỏi về thời gian

        """

        message\_lower = message.lower()

        time\_info = self.\_get\_current\_time\_info()

        # Các pattern câu hỏi phổ biến

        if any(word in message\_lower for word in ['mấy giờ', 'bây giờ', 'hiện tại', 'giờ']):

            return f"Bây giờ là {time\_info['time']} ({time\_info['period']})."

        elif any(word in message\_lower for word in ['hôm nay', 'bữa nay', 'ngày']):

            if 'thứ' in message\_lower:

                return f"Hôm nay là {time\_info['full\_date']}."

            else:

                return f"Hôm nay là ngày {time\_info['date']}."

        elif any(word in message\_lower for word in ['thứ mấy', 'thứ']):

            return f"Hôm nay là {time\_info['weekday']}."

        elif 'tháng' in message\_lower:

            return f"Hiện tại là {time\_info['month']} năm {time\_info['year']}."

        elif 'năm' in message\_lower:

            return f"Năm nay là năm {time\_info['year']}."

        # Trả về thông tin tổng quát

        return f"Hiện tại là {time\_info['time']}, {time\_info['full\_date']}."

#### **2.2.4 Xử lý câu hỏi chung**

   def ask\_llm(self, message: str) -> str:

        if not self.current\_session:

            self.start\_new\_session()

        try:

            # Lưu câu hỏi của user

            self.memory.add\_message(self.current\_session, "user", message)

            # Kiểm tra xem có phải câu hỏi về thời gian không

            if self.\_is\_time\_question(message):

                response = self.\_handle\_time\_question(message)

                # Lưu câu trả lời

                self.memory.add\_message(self.current\_session, "assistant", response)

                return response

            # Lấy ngữ cảnh gần đây

            context = self.memory.get\_context\_string(self.current\_session, limit=4)

            # Thêm thông tin thời gian vào system prompt nếu cần

            current\_time = self.\_get\_current\_time\_info()

            time\_context = f"\nThông tin thời gian hiện tại: {current\_time['time']}, {current\_time['full\_date']}"

            # Tạo prompt với ngữ cảnh

            if context:

                full\_prompt = f"""{self.system\_prompt}{time\_context}

Ngữ cảnh cuộc trò chuyện gần đây: {context}

Câu hỏi mới: {message}

Trả lời (nhớ: NGẮN GỌN, 1-2 câu):"""

            else:

                full\_prompt = f"""{self.system\_prompt}{time\_context}

Câu hỏi: {message}

Trả lời (nhớ: NGẮN GỌN, 1-2 câu):"""

            # Gọi model

            response = self.model.invoke(full\_prompt)

            # Làm sạch response và giới hạn độ dài

            cleaned\_response = self.\_clean\_response(response)

            # Lưu câu trả lời

            self.memory.add\_message(self.current\_session, "assistant", cleaned\_response)

            return cleaned\_response

        except Exception as e:

            error\_msg = "Xin lỗi, tôi gặp sự cố. Bạn có thể thử lại không?"

            if self.current\_session:

                self.memory.add\_message(self.current\_session, "assistant", error\_msg)

            return error\_msg

Làm sạch câu trả lời và rút gọn:

   def \_clean\_response(self, response: str) -> str:

        # Xóa các phần thừa

        response = response.strip()

        # Loại bỏ các pattern không mong muốn

        patterns\_to\_remove = [

            r"^(Trả lời:|Câu trả lời:|Response:)",

            r"^(Xin chào|Hello)",

            r"\n\n+",  # Nhiều dòng trống

        ]

        for pattern in patterns\_to\_remove:

            response = re.sub(pattern, "", response, flags=re.IGNORECASE | re.MULTILINE)

        response = response.strip()

        # Giới hạn độ dài (tối đa 2 câu)

        sentences = re.split(r'[.!?]+', response)

        if len(sentences) > 2:

            response = '. '.join(sentences[:2]) + '.'

        # Giới hạn ký tự (tối đa 200 ký tự)

        if len(response) > 200:

            response = response[:197] + "..."

        return response

#### **2.2.5 Quản lý bộ nhớ hội thoại**

Lưu câu hỏi và câu trả lời từng bước vào bộ nhớ:

    def get\_conversation\_history(self, limit: int = 10) -> list:

        if not self.current\_session:

            return []

        return self.memory.get\_recent\_context(self.current\_session, limit)

    Lấy lịch sử hội thoại gần nhất:

    def get\_conversation\_history(self, limit: int = 10) -> list:

        if not self.current\_session:

            return []

        return self.memory.get\_recent\_context(self.current\_session, limit)

Xóa toàn bộ hội thoại hiện tại khi cần và đóng kết nối

    def clear\_conversation(self):

        if self.current\_session:

            self.memory.clear\_session(self.current\_session)

            self.current\_session = None

    def close(self):

        self.memory.close()

## **Text-to-Speech (TTS) – gTTS**

Text-to-Speech (TTS) là công nghệ ngược lại với STT, chuyển đổi văn bản thành giọng nói, cho phép hệ thống máy tính phát âm một cách tự nhiên từ dữ liệu dạng chữ. Trong ứng dụng trợ lý giọng nói, TTS đóng vai trò là bước cuối cùng để hệ thống giao tiếp lại với người dùng bằng giọng nói.

gTTS (Google Text-to-Speech) là một thư viện Python mã nguồn mở, sử dụng dịch vụ TTS của Google để chuyển văn bản thành file âm thanh .mp3

from gtts import gTTS

import playsound

import os

def speak(text):

    tts = gTTS(text=text, lang='vi')

    filename = "voice.mp3"

    tts.save(filename)

    playsound.playsound(filename)

    os.remove(filename)

Trong hàm speak() xây dựng với 3 thư viện: thư viện Google Text-to-Speech hỗ trợ tiếng Việt với giọng đọc tự nhiên, playsound dùng để phát âm thanh (đơn giản, hỗ trợ MP3), os dùng để xóa file sau khi phát. Sau khi LLM xử lý đầu vào từ STT thì sẽ trả ra 1 text, TTS sẽ nhận text từ LLM rồi chuyển đổi thành giọng nói, tức là đầu ra của ứng dụng.

## **Lưu trữ cuộc trò chuyện – MongoDB**

Trong hệ thống trợ lý giọng nói, chức năng lưu trữ cuộc trò chuyện này nhằm ghi lại toàn bộ lịch sử hội thoại giữa người dùng và trợ lý ảo, giúp hệ thống tái sử dụng ngữ cảnh trong các lần tương tác tiếp theo, đồng thời phục vụ mục đích phân tích, cải tiến trải nghiệm người dùng. Các thư viện sử dụng:

* **PyMongo**: Thư viện Python chính thức để tương tác với MongoDB.
* **uuid**: Tạo định danh duy nhất cho mỗi session.
* **datetime**: Lưu thời gian tin nhắn và cuộc hội thoại.
* **typing** (List, Dict, Optional): Gợi ý kiểu dữ liệu trong hàm, giúp mã dễ hiểu và dễ kiểm tra.

import pymongo

from datetime import datetime

from typing import List, Dict, Optional

import uuid

### **4.1 Khởi tạo**

Khởi tạo kết nối đến MongoDB với "mongodb://localhost:27017/" và chỉ định cơ sở dữ liệu voice\_assistant với collection là conversations.

class ConversationMemory:

    def \_\_init\_\_(self, mongo\_uri: str = "mongodb://localhost:27017/", db\_name: str = "voice\_assistant"):

        """

        Khởi tạo kết nối MongoDB để lưu trữ lịch sử hội thoại

        """

        self.client = pymongo.MongoClient(mongo\_uri)

        self.db = self.client[db\_name]

        self.conversations = self.db.conversations

### **4.2 Hàm tạo cuộc hội thoại**

Tạo một seesion với \_id duy nhất bằng thư viện uuid, khi tạo một session sẽ được ghi nhận bằng thời gian tạo "created\_at": datetime.now() và tin nhắn rỗng "messages": []

    def create\_session(self, user\_id: str = None) -> str:

        """

        Tạo session mới cho cuộc hội thoại

        """

        session\_id = str(uuid.uuid4())

        session\_data = {

            "\_id": session\_id,

            "user\_id": user\_id or "default\_user",

            "created\_at": datetime.now(),

            "messages": [],

            "active": True

        }

        self.conversations.insert\_one(session\_data)

        return session\_id

### **4.3 Hàm tạo tin nhắn vào cuộc hội thoại**

Có hàm tạo session thì phải có hàm thêm tin nhắn để có thể thêm tin nhắn vào session, hàm này sẽ nhận tin nhắn (từ người dùng và máy), nội dung tin nhắn và thời gian của tin. Sau đó sẽ thêm vào phiên hội thoại.

    def add\_message(self, session\_id: str, role: str, content: str, timestamp: datetime = None):

        message = {

            "role": role,

            "content": content,

            "timestamp": timestamp or datetime.now()

        }

        self.conversations.update\_one(

            {"\_id": session\_id},

            {

                "$push": {"messages": message},

                "$set": {"last\_updated": datetime.now()}

            }

        )

### **4.4 Hàm xóa phiên cuộc trò chuyện**

Xóa hoàn toàn phiên hội thoại khỏi cơ sở dữ liệu theo session\_id:

def clear\_session(self, session\_id: str):

        self.conversations.delete\_one({"\_id": session\_id})

### **4.5 Hàm đóng kết nối**

    def close(self):

        self.client.close()

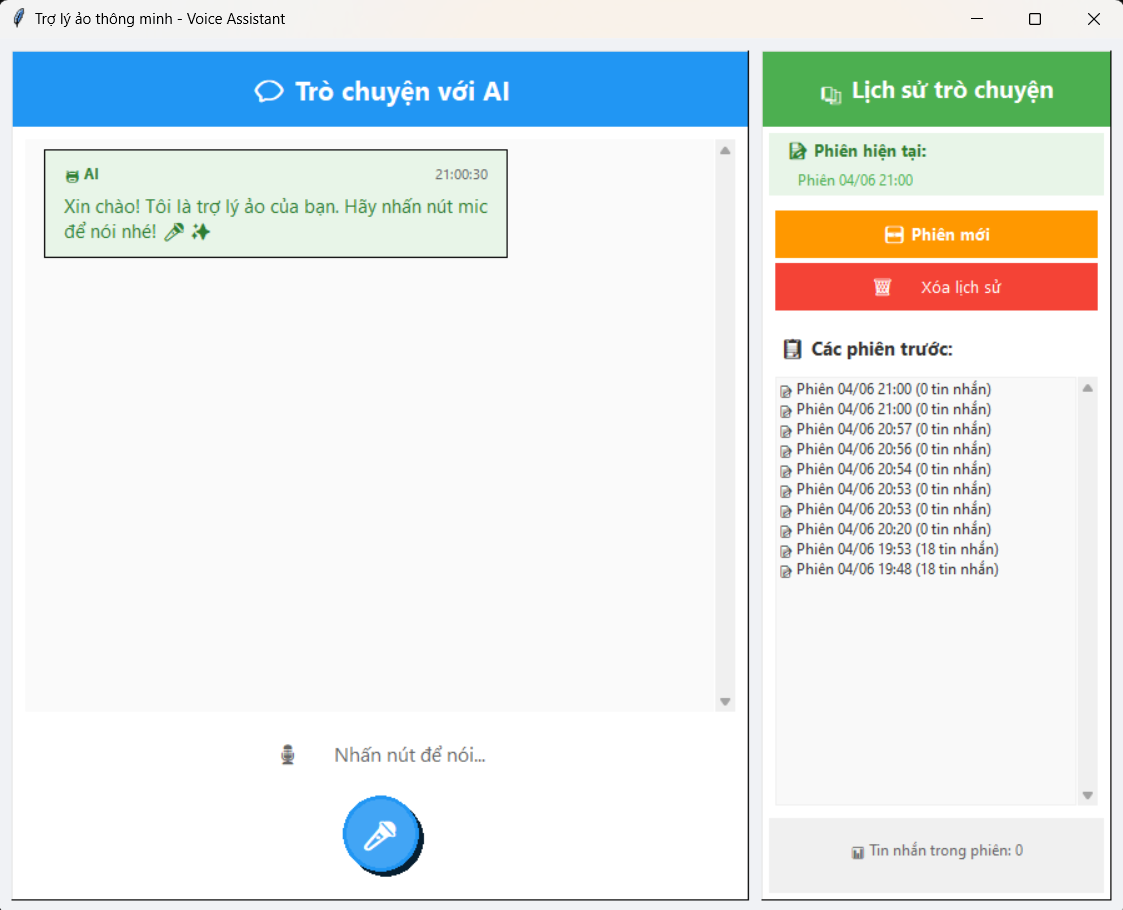
## **Giao diện người dùng (GUI)**

Giao diện của ứng dụng được thiết kế đơn giản, thân thiện với người dùng, sử dụng thư viện Tkinter. Người dùng có thể dễ dàng thao tác chỉ với một nút bấm để bắt đầu nói chuyện với trợ lý ảo.

Các thư viện sử dụng xây dựng GUI: Tkinter, PIL (Pillow), Threading

Các thành phần chính:

* Khung hội thoại: là nơi hiển thị đoạn hội thoại giữa người dùng và trợ lý ảo, Mỗi dòng hội thoại được phân biệt rõ ràng bởi nhãn “Bạn” (user) và “AI” (assistant), sử dụng thanh kéo lên xuống (Scrollbar) để xem lại hội thoại nếu quá dài.
* Khung lịch sử trò chuyện: hiển thị các cuộc trò chuyện trước, ngoài ra còn có các chức năng như tạo phiên mới và xóa lịch sử trò chuyện.
* Phần nút Micro: hỗ trợ ghi lại giọng nói bằng STT.



# **CHƯƠNG 3: ĐÁNH GIÁ & KẾT QUẢ THỬ NGHIỆM**

## **Thử nghiệm với câu hỏi thời gian**

A screenshot of a chat

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.

Độ chính xác STT và NLP  
Trải nghiệm người dùng

# **CHƯƠNG 4: HƯỚNG PHÁT TRIỂN TƯƠNG LAI**

Hỗ trợ đa ngôn ngữ

Thêm chức năng cá nhân hóa

Tích hợp thiết bị IoT

**CHƯƠNG 5: KẾT LUẬN**

A screenshot of a chat

AI-generated content may be incorrect.