一、项目背景

随着数字化技术的快速发展，主流语言（如汉语、英语）的智能工具和资源已经非常丰富，但少数民族语言的数字化资源和智能工具却明显不足。这种不平衡使得少数民族语言数字化学习资源稀缺，难以满足学生学习需求，而市面上也十分缺乏针对性少数民族的只能翻译学习工具，因此设计一款少数民族多语言翻译器是有必要且有需求的。

项目针对人群包括少数民族学生(更加便捷的用自己的语言进行大模型交流并得到反馈)及对少数民族语言感兴趣的汉族学生。我们希望通过本工具，学生能够使用自己的少数民族语言进行学习交流并获得智能反馈，同时因为少数民族语言兴趣者提供便捷的语言学习工具。

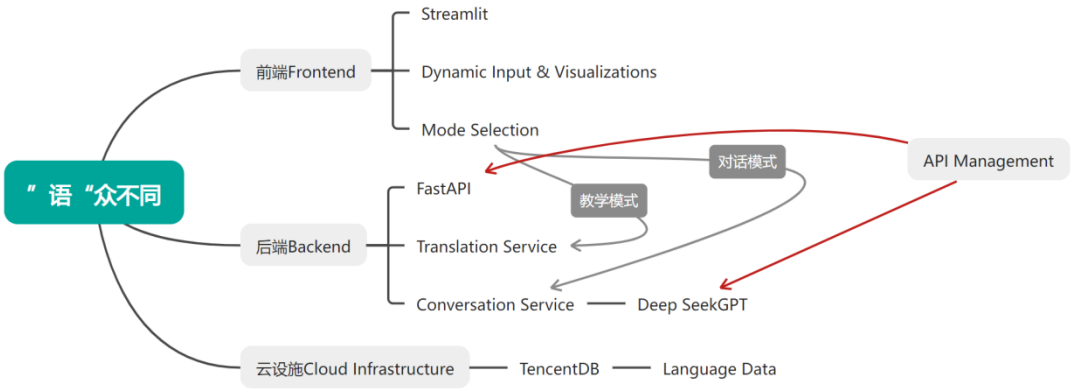
项目开发了一款结合人工智能与 LangChain 框架的双语教育工具，同时兼顾教学模式与对话模式两大核心功能，并利用语言模型增强工具的学习能力和生成能力，接入更多语言模块，覆盖更多少数民族语言，并添加了自适应学习功能，根据用户的学习行为和反馈动态调整学习内容，提供更具个性化的学习体验。

功能/工具	现有工具	本项目	创新点
多语言支持	主要支持主流语言（如汉语、英语）， 少数民族语言支持有限。	支持多种少数民族语言（如哈萨克语、朝鲜语、 维吾尔语、藏语、蒙古语），并持续扩展。	专注于少数民族语言，填补市场空白。
教学模式	提供基础翻译功能， 但缺乏教材生成和语音合成。	支持双语教材生成、语音合成和翻译功能， 提供全面的学习资源。	结合 LangChain 框架，生成高质量的双语教材， 并支持语音合成，提升学习体验。
对话模式	对话功能局限于主流语言， 无法支持少数民族语言。	提供基于少数民族语言的智能对话功能， 支持多轮对话和历史记录。	利用 LangChain 的对话链功能， 实现更自然的少数民族语言对话。
自适应学习	缺乏个性化学习功能， 无法根据用户行为调整内容。	根据用户的学习行为和反馈，动态调整学习内容， 提供个性化学习体验。	引入自适应学习算法， 提升用户的学习效率和满意度。
语音合成	语音合成功能局限于主流语言， 少数民族语言支持不足。	支持多种少数民族语言的语音合成， 帮助学生练习发音。	结合 gTTS 和自定义语音模型， 提供高质量的少数民族语言语音合成。
开源与扩展性	大多数工具为封闭系统， 无法扩展或定制。	基于开源框架（如 LangChain、FastAPI）， 支持功能扩展和定制。	开放性强，便于开发者根据需求扩展功能。

二、设计思路

设计系统架构分为前端 Frontend、后端 Backend 及云基础设施 Cloud Infrastructure

部分。



·前端部分：①通过 streamlit 搭建轻量化用户界面②Dynamic Input & Visualizations 节点处方便用户输入问题，并实时查看系统的输出结果。③Mode Selection 节点选择模式（对话模式或教学模式）。

·后端部分：根据用户选择的模式，分别调用教学功能或对话功能。①Translation Service 调用教学模式翻译②Conversation Service 对话服务调用 DeepSeek 模型实现对话功能，为用户提供智能化回答，将少数民族语言和汉语之间进行高效转换。③以 FastAPI 构建后端框架。

·云设施部分：使用 TencentDB 存储用户数据，包括语言数据等，将所有的用户数据、学习记录以及翻译内容文件存储在云端。方便语言模型形成知识库来自行学习。

·通过 API Management 管理和调度各种接口，包括 DeepSeek 模型接口、FastAPI 等。

三、前端部分技术实现（代码实现于 App.py）

前端界面基于 Streamlit 框架，实现了多功能的交互界面，主要包含以下几个功能模块：

①页面布局

·使用 Streamlit 中的 st.set_page_config 设置页面标题、图标和布局（宽屏）

```
st.set_page_config(page_title="语众不同", page_icon="🌎", layout="wide")
```

②模式选择

用户可以通过页面左侧的侧边栏选择两种工作模式

·教学模式：为用户提供少数民族语言翻译、双语教材生成、语音合成等功能。

1 翻译

- 允许用户输入少数民族语言的文本，并选择将其翻译为主流语言（汉语）。

- 用户输入文本后，点击 "翻译为主流语言" 按钮，程序会向后端发送请求并返回翻译结果，显示在页面上。

2 双语教材生成

- 允许用户上传包含主流语言内容的教材文件（.txt 文件），并生成该教材的少数民族语言版本在文本框中。

- 添加语言选择下拉框，允许用户定向选择生成的目标少数民族语言（如藏语、朝鲜语、维吾尔语等）。用户选择语言后，点击 "生成少数民族语言版本" 按钮，程序会向后端发送请求，并返回生成的少数民族语言版本教材内容。

3 语音合成

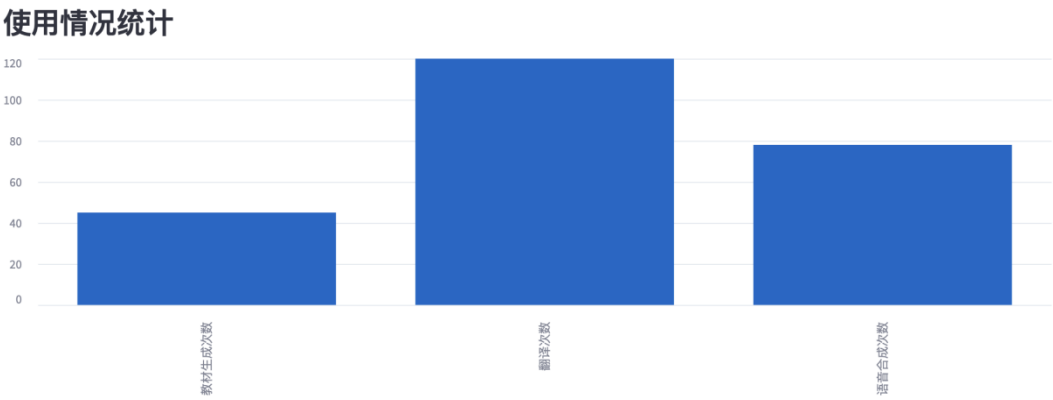
- 将用户输入的少数民族语言文本转化为语音。

- 用户输入文本后，点击 "生成语音" 按钮，程序会向后端发送请求生成语音文件并播放。

4 使用情况统计

- 动态显示各功能的使用次数统计。

- 使用 pandas 库展示功能的使用情况，并通过条形图（st.bar_chart）展示统计数据。



·对话模式：通过 deepseek 接口调用语言模型进行少数民族语言对话。

用户可以输入少数民族语言的问题，点击"发送消息"后，程序会模拟向后端 API 发起请求，获取语言模型的回答，并在页面上显示出来。经过使用 langchain 优化后，页面可以存储用户的问题与模型的回答（*conversation_history*），方便用户进一步对先前的问答进行回顾。

```python
if mode == "对话模式":
    st.subheader("对话模式")
    st.markdown("在这里，您可以使用少数民族语言进行对话。")

    # 初始化对话历史记录
    if "conversation_history" not in st.session_state:
        st.session_state.conversation_history = []
```

③健壮性与用户友好

1 错误处理：对 API 请求进行了错误处理，包括检查后端服务是否正常、提示用户上传文件、选择目标语言等。

2 界面交互性：根据用户的输入或选择动态更新界面内容，使得用户体验更加流畅。

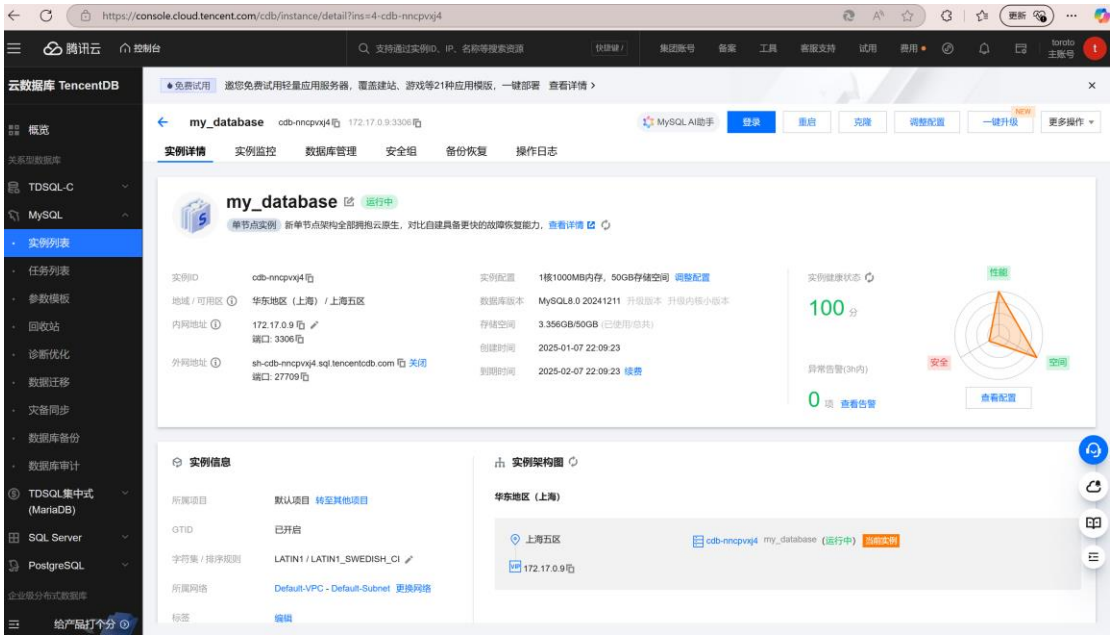
3 后端交互：通过 requests.post 向后端 API 发起请求，并处理响应，显示翻译结果、生成的内容或语音。

四、后端部分技术实现

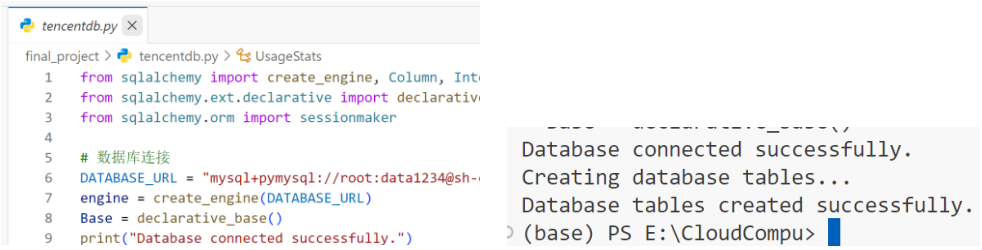
1．云数据库**（代码实现于 tencentdb.py 和 backend.py 数据库交互部分）**

该部分是本项目云服务的主要体现，使用腾讯云 MySQL 数据库，方便小组成员访问及修改，并且相比本地数据库具有更多优点：高可用性、弹性扩展、安全性、性能优化、运维简化、成本效益、全球部署和专业支持等，能够更好地满足本次实验项目数据库需求。

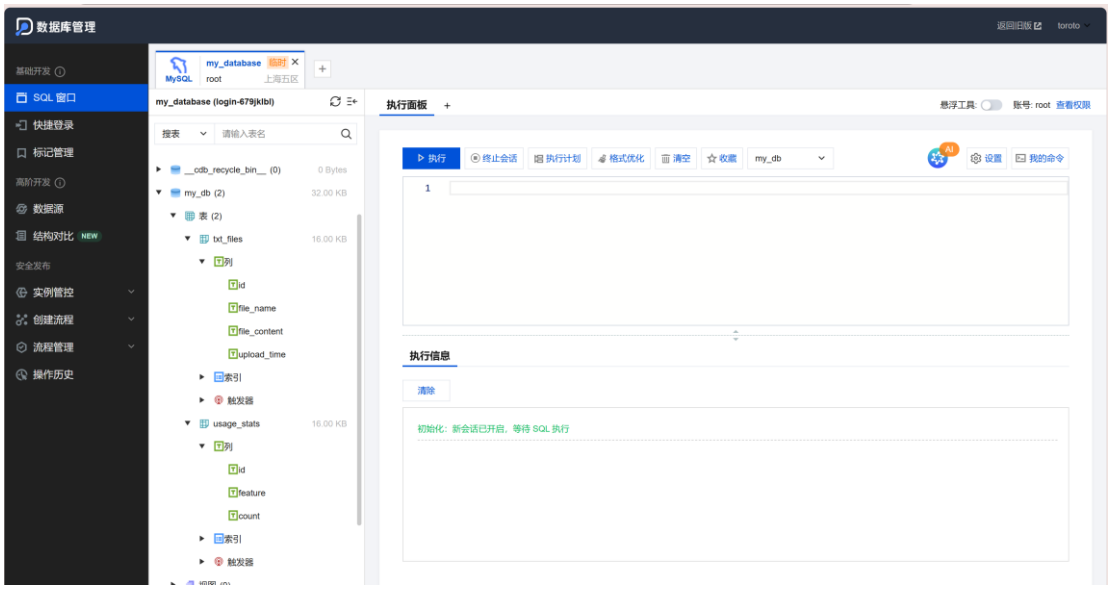
如下是购买的云数据库 MySQL 实例：



python 代码使用 SQLAlchemy 框架连接腾讯云数据库，并通过 ORM 模式定义和创建数据库表。通过创建数据库会话，可以方便地与数据库进行交互，包括插入、查询、更新和删除数据等操作。



网页可视化数据库页面如下（为了方便统筹管理，使用 python 代码，没有采用网页执行面版）：



2．**功能实现（代码实现于 ds_moremode.py 与 gtts_sound.py）**

a. ds_moremode.py

该部分主要封装了一个 DeepSeekAPI 类，用于与 DeepSeek API 进行交互。它通过 OpenAI 客户端与 DeepSeek 的 API 进行通信，支持对话、生成、翻译和语音翻译等功能。

核心：call_api

backend.py 通过调用该函数，向功能传递用户的问答参数 config，根据不同的需求调用不同 mode，根据不同 mode 构建不同的 promp_text,。可以通过最大 token 数调整生成内容，且提供流式输出 stream，启用后用户无需等待整个内容生成完毕，能够更快看到结果。

```
if stream:
    output = ""
    for chunk in response:
        content = chunk.choices[0].delta.get("content", "")
        print(content, end="", flush=True)
        output += content
    return output.strip()
else:
    return response.choices[0].message.content.strip()
```

b. gtts_sound.py

使用 gTTS（Google Text-to-Speech）库将输入的文本转换为语音，并保存为音频文件（此处为 MP3），该函数支持指定语言。

核心：gTTS 调用

gTTS 没有翻译功能，但指定语言和文本内容后能够直接生成语音文件。因此传入内容后，需要调用 DeepSeekAPI 中 call.api 的 synthesize 类将文本翻译为指定语言，才能够进一步生成语音，该功能可选择发音速度。

```
tts = gTTS(text=text, lang=lang, slow=False)  # slow=False 表示快速发音
```

3．后端接口**（代码实现于 backend.py）**

主要使用 FastAPI 框架来构建 Web API 服务，搭配 SQLAlchemy 作为 ORM 工具来与 MySQL 数据库进行交互。

通过定义请求数据模型、数据库表结构和 API 路由，应用能够处理聊天翻译、生成、语音合成和获取使用统计的请求。

```
# 接口：根目录
@app.get("/")
async def read_root():
    return {"message": "Welcome to the DeepSeek

# 接口：对话
@app.post("/chat")
async def chat(request: ChatRequest):
    response = deepseek_api.call_api(request.me
    session = SessionLocal()

# 接口：翻译
@app.post("/translate")
async def translate(request: TranslationReques
    translation_result = deepseek_api.call_api
    session = SessionLocal()

# 接口：上传文本文件
@app.post("/upload_txt")
async def upload_txt(file: UploadFile = File(.
    content = await file.read()

# 接口：获取用量统计
@app.get("/usage_stats")
async def get_usage_stats():
    session = SessionLocal()
```

最后，通过 Uvicorn 运行 ASGI 服务器来提供服务。

```python
# 启动服务
if __name__ == '__main__':
    uvicorn.run(app, host="127.0.0.1", port=8000)
```

五、团队分工

李芳：后端框架与接口实现，云数据库部署，PPT 及项目说明负责部分补充，代码统筹完善，答辩汇报，讲解视频录制。

寇璟奕：前端页面基础框架及模块设计，PPT 制作，讲稿、项目说明前端部分撰写，最终系统架构图绘制及说明。

闫研：后端模型调用及训练，langchain 前端优化，PPT 及项目说明负责部分补充，讲稿后端部分撰写，功能视频录制。

邵程叶：前端页面代码具体实现与完善，用户交互功能实现，中期系统架构图绘制，答辩汇报。