设计

功能设计

- 前端+后端
  - 注册登录
  - 信息填写+录入数据库
  - 查看数据
  - 表单提交
- ai
  - 人脸识别登录
  - 用户意图输入+意图解析
  - 任务规划生成
  - 提取信息+表格填写生成
  - 语音输入
  - 差分隐私保护
  - 语义校对

平台对接API

前端，后端，py三端通讯

工具模块（开放在port=3304）（无需握手的，无需提供任何信息的，快速的ai-socket接口）

人脸识别(path=/face_predict)

- **socket-send**

```
直接传一张图片二进制
```

- **socket-recieve**

  未检测完成，继续读取

```
{
    "user_id":"",
    "type":"face_predict",
    "hash":"a1dadafgdas3asd4s2sfad",

    "message":"continue"
}
```

  为获取成功

```
{
    "user_id":"asdasd",
    "type":"face_predict",
    "hash":"a1dadafgdas3asd4s2sfad",

    "message":"success"
}
```

请求次数过多，进行完此次回复，服务器将会断开socket

```
{
    "user_id":"asdasd",
    "type":"face_predict",
    "hash":"a1dadafgdas3asd4s2sfad",

    "message":"fail"
}
```

语音转文字(path=/stt)

- **socket-send**

```
直接传入音频二进制
```

- **socket-recieve**

```
"音频的文本"
```

文字转语音(path=/tts)

- **socket-send**

```
"要转换的文本"
```

- **socket-recieve**

```
音频二进制
```

流程（socket开放在port=444，http在port=10925）

首次连接（握手）(此握手在所有流程请求前是必要的)

- **post（后端此处要查询数据库，把所有用户信息（必要+基本）放到info，没有的填None/null）**

  文本形式

```
{
    "user_id":"asdasd",
    "type":"handshake",

    //必要信息+基本信息
    "info":{
        "name":"a",
        "age":18,
        "birth":20250721,
        ".....":"....."
    }
}
```

- **response**

```
{
    "user_id":"asdasd",
    "type":"handshake",
    "hash":"a1dadafgdas3asd4s2sfad",

    "message":"success"
}
```

首次发送信息-用户意图

- **post**

```
{
    "user_id":"asdasd",
    "type":"classify",
    "hash":"a1dadafgdas3asd4s2sfad",


    "input":{
        "type":"text",
        "text":"我要办身份证"
    }
}
```

- **response**

```
{
    "user_id":"asdasd",
    "type":"classify",
    "hash":"a1dadafgdas3asd4s2sfad",

    "classify":"身份证",
    "flow":"办理身份证呢，需要先去寻找最近的政务大厅blablabla，然后带一寸照片blblbl"
}
```

人脸录入

- **socket-send（handshake）握手**

```
{
    user_id:"xiaofeng",     //用户id
    hash:"",                //预留，可以不实现
    type:"face_train",
}
```

- **socket-send**

```
直接传图片二进制
```

- **socket-recieve**

  未检测完成，继续读取

```
{
    "user_id":"xiaofeng",
    "type":"face_train",
    "hash":"a1dadafgdas3asd4s2sfad",

    "message":"continue"
}
```

  为录入成功

```
{
    "user_id":"xiaofeng",
    "type":"face_train",
    "hash":"a1dadafgdas3asd4s2sfad",

    "message":"success"
}
```

对话

- **socket-send（handshake）**

```
{
    "user_id":"xiaofeng",
    "type":"qna",
    "hash":"a1dadafgdas3asd4s2sfad",
}
```

- **socket-recieve（我们提问的环节）**

```
文本
```

无内容，即为提问结束

```

```

- **socket-send（用户回答的环节）**

```
文本
```

- **socket-revieve（总结表格）（注意有多张表）**

```
{
    "tables":[
        {
            "header":[
                "name",
                "age",
                "gender",
                "phone"
            ],
            "row":[
                "xfgg",
                1234,
                "men",
                "18929293939"
            ]
        },
        {
            "header":[
                "name",
                "age",
                "gender",
                "nation"
            ],
            "row":[
                "xfgg",
                1234,
                "men",
                "中国"
            ]
        },
        {},
    ],
}
```

录入（询问完成时）（总结必要信息+基本信息，将output录入数据库）

- **post**

```
{
    "user_id":"asdasd",
    "type":"storage",
    "hash":"a1dadafgdas3asd4s2sfad"
}
```

- **response**

```
{
    "user_id":"asdasd",
    "type":"storage",
    "hash":"a1dadafgdas3asd4s2sfad",


    "output":{
        "name":"xfgg",
        "age":1234,
        "gender":"men",
        ".....":"....."
    }
}
```

总结呈现（仅前端）

- **post**

```
{
    "user_id":"asdasd",
    "type":"summary",
    "hash":"a1dadafgdas3asd4s2sfad"
}
```

- **response**

```
{
    "user_id":"asdasd",
    "type":"summary",
    "hash":"a1dadafgdas3asd4s2sfad",


    "output":{
        //对于这个tables，写成header和row你们更方便吗？如果方便我就改
        "tables":[
            {
                "header":[
                    "name",
                    "age",
```

```
                    "gender",
                    "phone"
                ],
                "row":[
                    "xfgg",
                    1234,
                    "men",
                    "18929293939"
                ]
            },
            {
                "header":[
                    "name",
                    "age",
                    "gender",
                    "nation"
                ],
                "row":[
                    "xfgg",
                    1234,
                    "men",
                    "中国"
                ]
            },
            {},
        ],

        "classify":"身份证/户口本",
        "flow":"办理身份证呢，需要先去寻找最近的政务大厅blablabla，然后带一寸照片blblbl"
    }


}
```

特别的响应！！！（仅http）

当请求正在处理中，请过几秒再请求

- **response**

```
{
    "user_id":"asdasd",
    "type":"processing",
    "hash":"a1dadafgdas3asd4s2sfad",
}
```

服务器繁忙

- **response**

```
{
    "user_id":"asdasd",
    "type":"busy",
    "hash":"a1dadafgdas3asd4s2sfad",
}
```

数据有误/用户尚未握手/流程有误

- **response**

```
{
    "user_id":"asdasd",
    "type":"error",
    "hash":"a1dadafgdas3asd4s2sfad",
}
```

数据库

- **必要信息**

```
{
    "name":"英锐gg",
    "gender":"男",
    "nationality":"汉",
    "phone":"12345678901",
    "id_card":"440209200601018080"
}
```

- **基本信息(初始留空)**

```
{
    "native_place":"广州",
    "birth":"2025年8月13日",
    "email":"10925508@qq.com",
    "career":"炼丹师",
    "address":"广东省广州市番禺区广东工业大学西二-803",
    "hukou":"广州市天河区",
    "political_status":"党员",
    "marital_status":true,
    "religion":"伊斯兰教/基督教/无",
    "education":"本科"
}
```

python环境配置

```
conda install pytorch torchvision torchaudio torchtext pytorch-cuda=11.8 -c pytorch -c nvidia

conda install numpy matplotlib requests regex

conda install -c conda-forge librosa

conda install -c conda-forge ffmpeg-python

conda install dotenv=0.9.9 python-dotenv=1.1.1 websockets

pip install openai-whisper


pip install TTS
```

Python封装

- **main+平台对接**

```
if name == '__main__':
    ThreadingHTTPServer(("127.0.0.1", 10925), httpserverHandler)



class Handler(http.server.BaseHTTPRequestHandler):
    def do_GET(self):
        content_length = int(self.headers.get("Content-Length", 0))
        content_dic = json.loads(self.rfile.read(content_length).decode("utf8"))



        if content_dic["type"] == "handshake":
            a

        if content_dic["type"] == "question":
            user = flow_dic[content_dic["user_id"]]

            # 获取问题

            # 响应问题

        if content_dic["type"] == "answer":
            user = flow_dic[content_dic["user_id"]]
```

```
            user.waiter.set()

        if content_dic["type"] == "storage":
            user = flow_dic[content_dic["user_id"]]
```

- **人脸识别**

```
#人脸识别模型训练
#imgs_bin: 许多png/jpg图像的二进制格式的数组
#min_acc: 最小准确率

#return: 一个经过测试的结果，如果小于预期的准确率，则返回False
def cv2_train(imgs_bin:list(list),user_id,min_acc=0.95)->bool
```

```
#人脸识别模型预测
#imgs_bin: 许多png/jpg图像的二进制格式的数组
#min_acc: 最小准确率

#return: 识别出的用户id，如果小于最小准确率，则返回None
def cv2_predict(imgs_bin:list(list),min_acc=0.8)->user_id:str
```

```
#图像二进制格式人脸CV图像
#image_binary:

#return: 若没有在图中发现人脸，则返回None，发现则返回一个新的二进制
def face_fetcher(image_binary):
```

- **用户封装**

```
class User:
    def __init__(self,name,birth,):
        return

    @property
    def info()->dict


    #一个流程的主程序
    def activate():

    def
```

- **语音转换**

```
#语音转文字

#voice: 音频的二进制数据格式
#return: 识别出的文本
def voice2text(voice:list(int))->str
```

```
#文字转语音

#text: 文本内容
#return: 音频的二进制数据格式
def text2voice(text:str)->list(int)
```

- **输入分类**

```
#用户输入业务类别分类

#text: 用户输入的文本
#return:
def classify(text:str)->int
```

- **询问校对**

```
#提问函数
#info: 目前的信息列表，包含一些None的键值
#return: 提问的键值和提问的自然语言
def inquire(info:dict)->key:str,sentence:str
```

```
#通过用户的回答，提取关键字，转为可以录入数据库的键值
#answer: 用户问题的自然语言
#return: 返回用户提取的关键字
def get_answer(answer:str,key:str)->str
```

```
#检测校对用户的回答
#answer: 用户问题的自然语言
#key: 用户此问题的键
#value: 用户此问题回答的关键词（即为get_answer的值）
#return: 若返回None，则说明用户回答无问题，若有问题，则返回问题
def check_answer(answer:str,key:str,value:str)->str|None
```

- **获取数据**

```
#类别的对应实现，这里其实有两种实现方式，一种是分大类（比如证件办理，民事纠纷等等，再分成身份证户口本），另一种就是如下一类一个直接分，具体
type_dic={
    "身份证":0,
    "户口本":1,
    ........
}

#表格中的特殊标记，比如图片，签名，印章
table_mark={
    "<IMG>":0,
    "<SIG>":1,
    "<SEL>":2,
    "<PAD>":3,
}

class Data:

    #从文件读取数据，至于文件怎么存你们随便
    #data_path: 数据路径，或者你们看着怎么写好点
    def __init__(self,data_path)


    #用type_dic的字典，获取一个办理类别的流程和表单字典

    #index: 类别编号
    #return: str为办理的流程，dict为所填申请表单的字典，空值设键值为None，考虑到不止一个申请表，这里先用list

    '''
    空值为None
    预留图片<IMG>
    签名<SIG>
    签章<SEL>
    预留<PAD>
    '''
    def get_tables(self,index:int)->tables:list(dict)

    def get_flow(self,index:int,user_info:dict)->flow:str


    #将用户信息表字典（即getitem中返回的tables）中的键key，翻译
    def translate(self,text:str,lang="zh-cn")->str

```

- ## zh-cn

```
{
    "name":"名字",
    "gender":"性别"
}
```