

**《程序设计实验》详细设计文档**



**题目: 药品说明书信息提取与语音播报系统**

**班 级 2022219107**

**学 号 2021211907 2022211824**

**2022211762 2022211758**

**姓 名 陈科儒 任思宇 吕炫霖 冯路馨**

**指导教师 刘瑞芳**

**2024 年 5 月**

# 1、系统概述

**1.1、系统简介**

简要概述系统的基本情况和背景。

该系统致力于服务老年群体以及其他具有文字、视力障碍的群体，通过OCR文字识别技术识别药品说明书，再通过大语言模型提取关键信息（用药方法、注意事项、用药禁忌），生成精简的用药事项说明，再采用语音合成技术播放介绍上述说明，帮助用户以视、听两种方式快速了解药物说明。考虑到方便老年群体的使用，为了最大化系统的便携易用性，用户界面设计（UI）将简约直观。

**1.2、术语表**

定义系统或产品中涉及的重要术语，为读者在阅读文档时提供必要的参考信息。

|  |  |  |
| --- | --- | --- |
| **序号** | **术语或缩略语** | **说明性定义** |
| 1 | OCR | Optical Character Recognition,光学字符识别 |
| 2 | NLP | Natural Language Processing，自然语言处理 |
| 3 | PyQt5 | PyQt5 Python GUI库：用于创建图形用户界面的Python绑定库，基于Qt框架，允许开发者用Python编写具有原生外观的跨平台桌面应用程序。 |
| 4 | PaddleOCR | PaddleOCR 百度飞桨OCR：百度开发的OCR工具库，基于深度学习技术，提供文本检测和识别的端到端解决方案，常用于文档和图像中的文字信息提取。 |
| 5 | PyTorch/TensorFlow | 机器学习框架：均为流行的开源机器学习库，支持深度学习模型的构建、训练和部署，广泛应用于自然语言处理、计算机视觉等领域。 |
| 6 | DashScope | 阿里云服务：阿里云提供的一项服务，包含多种AI和大数据处理功能，本系统中可能用于高级自然语言处理或模型调用。 |
| 7 | PyTTS3 | Text-to-Speech Engine：Python文本转语音库，允许程序将文本数据转换为语音输出，增强系统与用户的交互性。 |
| 8 | QThread | 多线程模块：Python Qt 绑定中的一个类，用于实现多线程编程，以在执行耗时任务时不阻塞用户界面。 |
| 9 | Git/GitHub | 版本控制与代码托管：Git是一种分布式版本控制系统，用于追踪代码变更；GitHub是一个基于Git的代码托管平台，支持代码管理和团队协作。 |
| 10 | PyDev | Python调试器：Eclipse下的Python IDE插件，提供强大的调试功能，支持Python应用程序的开发、调试和测试。 |
| 11 | CPU/GPU | 中央处理器/图形处理器：CPU负责通用计算任务，GPU擅长并行处理，特别是在深度学习等大规模计算密集型任务中加速计算。 |
| 12 | VUE前端框架 | 一种用于构建用户界面的渐进式JavaScript框架，易于上手，可构建高效且灵活的Web应用程序。 |
| 13 | HTTP | 超文本传输协议：互联网上应用最为广泛的一种网络协议，用于传输超文本信息，确保客户端和服务器之间的数据交换安全可靠。 |
| 14 |  |  |

**1.3、系统运行环境**

包括对硬件平台、操作系统、数据库系统、编程平台、网络协议等的描述。

1.硬件平台

服务器端：选用高性能的可以支持GPU加速的服务器，并具备足够的内存资源以支持OCR处理、跨模态阅读理解等任务，以及足够的存储空间来存储用户上传的药品说明书图片、OCR识别结果与重要性排序打分等数据。

客户端：支持网页浏览器的智能手机、平板电脑、个人电脑等设备。一般来说，现代智能手机和个人电脑都具备足够的计算和显示能力来支持系统的使用。

2.操作系统

服务器端：Windows Server

客户端：支持主流操作系统，如Android、IOS、Windows、macOS等

3.数据库系统：

无数据库系统。

4.编程平台：

后端：Python。

前端：Python中的PyQt5模块。使用HTML、VUE、CSS前端框架构建所有用户交互界面。

4.网络协议：

采用HTTP作为通信协议，确保客户端与服务器之间的数据传输安全可靠。

**1.4、开发环境**

列举进行系统分析、程序设计和程序开发时要使用的工程工具和开发语言。应描述每一工具软件的名称、版本等。

1、开发环境和框架：

**后端开发语言：**Python 3.x，支持TensorFlow或PyTorch等机器学习库。

**前端开发语言和框架：**使用PyQt5跨平台图形用户界面（GUI）开发库进行UI界面的开放，它是Qt库的Python绑定。Qt本身是一个强大的C++库，用于创建具有现代化用户界面的应用程序。通过PyQt5，开发者可以利用Python的高效率和易读性来构建具有原生外观和感受的桌面应用程序，同时利用Qt框架的强大功能，包括窗口、按钮、菜单、对话框等各种UI元素，以及网络、数据库、多媒体等功能。

2、工程工具

**集成开发环境（IDE）：**使用PyCharm和Visual Studio Code。

**版本控制系统：**使用Git进行版本控制，结合GitHub进行团队写作和代码管理

**测试工具：**使用PyDev调试器进行单元测试、集成测试和端到端测试，以确保代码的指令和功能的稳定性。

**1.5、关键知识点**

列举本项目涉及到的知识点及其关系，比如粗粒度知识点“包含”细粒度知识点，可以是对比关系、顺序关系等等。

知识点范围包括：计算机应用、人工智能、信息与通信等领域。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 知识点1 | 关系 | 知识点2 |
| 1 | Web开发技术 | Web开发技术用于构建用户界面和实现网页交互功能，而数据库技术则用于存储和管理系统所需的数据。二者之间存在着紧密的关联关系，Web应用通过与数据库的交互，实现了用户数据的存储、查询和更新等功能，从而为用户提供了丰富的交互体验。 | 数据库技术 |
| 2 | 图像处理技术 | 图像处理和识别技术用于对药品说明书图片进行文字识别和布局分析，而自然语言处理技术则用于理解用户提出的问题和抽取药品说明书中的答案。二者之间存在着对应关系，即图像处理技术负责处理图片中的信息，而自然语言处理技术则负责处理文字信息，二者共同实现了对用户问题和药品信息的理解和提取。 | 自然语言处理 |
| 3 | 机器学习与深度学习技术 | 机器学习和深度学习技术用于模型训练和推理，主要应用于图像和文本处理任务，而跨模态理解技术则用于从图片中提取文本信息并理解用户的问题。二者之间存在着依赖关系，即跨模态理解技术在机器学习和深度学习技术的基础上进行了进一步的应用和拓展，实现了从图片到文本的跨模态信息理解。 | 跨模态阅读理解技术 |
| 4 | 信息检索 | 信息检索和展示模块负责将用户提出的问题与药品说明书中的相关信息进行匹配和展示，而语音播报模块则负责将系统输出的答案通过语音播报的形式传递给用户。二者之间存在着顺序关系，即信息检索和展示模块先将相关信息展示给用户，然后语音播报模块再将信息以语音形式传递给用户，共同实现了系统信息的呈现和传递。 | 语音播报 |
| 5 | GUI页面设计与美化 | 通过精心制作的GUI并不断与后端功能进行联系运行来达到通过简明扼要的要求实现各个后端功能板块的目的 | 后端功能 |

**1.6、知识链条**

Web开发技术与数据库技术（1，包含）图像处理技术与自然语言处理技术（2，包含）机器学习与深度学习技术与跨模态阅读理解（3，包含）信息检索与语音播报（4，顺序）GUI页面设计与美化与后端功能链接（顺序）

# 2、数据结构说明

本章说明本程序系统中使用的全局数据常量、变量和数据结构。

2.1、常量

包括数据文件名称及其所在目录，功能说明，具体常量说明等。

各文件一些导入的库以及部分常量：

1. **main.py文件：**

**PyQt5**：PyQt5库以及其相关组件：用于UI界面的构建。

**display\_page、option\_page、waiting\_page**：自定义页面类，用于自定义UI的构建。

Qwen1：信息获取以及处理文件（模块）

1. **OCR.py文件：**

**paddleocr：**PaddleOCR是百度飞桨（PaddlePaddle）团队研发的光学字符识别（Optical Character Recognition，OCR）工具库，它基于深度学习技术，提供了端到端的文本检测（Text Detection）和文本识别（Text Recognition）解决方案。

1. **Qwen1.py文件：**

**dashscope：**导入阿里云DashScope服务相关库的语句加载DashScope提供的功能和类。

**dashscope.api\_key：**阿里云apikey

**def get\_response(messages):** 定义调用大模型的函数

1. **display\_page.py文件：**

**PyQt5：**PyQt5库以及其相关组件：用于UI界面的构建。

**pyttsx3：**一个Python库，全称为Python Text-to-Speech Engine version 3，用于将文本转换为语音。

**threading：**Python标准库中的threading模块。这个模块提供了创建和管理线程的功能，使得开发者能够轻松地在程序中实现多线程并发执行。

2.2、变量

本章说明本程序系统中使用的全局数据常量、变量和数据结构。

由于程序文件中变量较多，下面举出一些重要变量：

1. **main.py文件：**

class Worker(QThread)：定义一个工作线程类Worker，用于在后台执行耗时的药品信息提取任务，完成后通过信号finished发送提取到的文本和语音文本。

class MainWindow(QMainWindow)：主窗口类，其中包含UI界面，信息获取及处理模块等的调用。

1. **OCR.py文件：**

ocr：ocr 变量是一个PaddleOCR对象，用于处理光学字符识别（OCR）即图像文字识别任务。

1. **Qwen1.py文件：**

prompt：prompt是一个列表，它接受两个参数 drug\_info 和 Info，并根据这些输入构建一个结构化的提示（prompt）用于后续的处理，与AI模型交互以获取特定格式的药品信息摘要。

1. **display\_page.py文件：**

class Mainwindow4(QMainWindow): 定义主窗口类，继承自QMainWindow，用于构建主UI界面

1. **option\_page.py文件：**

class Mainwindow2(QMainWindow):定义主窗口2类，用于构建选项UI界面

1. **waiting\_page.py文件：**

class Mainwindow3(QMainWindow):定义主窗口3类，用于构建信息提取后的UI界面，即展示界面。

2.3、数据结构

包括数据结构名称，功能说明，具体数据结构说明（定义、注释、取值）等。

下面主要展示各文件中重要函数：

1. **main.py文件：**

def run(self): run函数设计为在Python的多线程环境下工作的，从而在使用 PyQt 或类似的 GUI 库时，可以避免耗时的操作阻塞主线程，从而保持用户界面的响应性。并同步进行图片和prompt传入。

def upload(self): 上传图片功能函数。

def uploadFileFolder(self):从文件夹选择图片并显示函数。

def showcamera(self): 显示摄像头实时画面并捕捉图片函数。

def change\_option\_page(self): 切换到选项配置页面函数。

def getInfo(self): 获取用户配置信息并启动信息提取流程函数。

1. **OCR.py文件：**

def extract\_text\_from\_images(img\_path):图片识别文字函数。

1. **Qwen1.py文件：**

def get\_response(messages):调用大模型的函数。

def build\_prompt(drug\_info, Info): 构建提取关键信息的prompt的函数。

def extract\_drug\_info(drug\_info\_text, Info)**:** 从药品说明书文本中提取关键信息并格式化输出。

def split\_response(content): 分割模型返回的文本，提取用于UI和语音播报的部分。

def format\_ui\_text(ui\_text): 对UI展示的文本进行格式化处理，确保每项信息清晰且易于阅读。

def get\_drug\_info\_from\_images(img\_path, Info): 直接返回提取的信息,总发送函数（即将图片传输给大模型并识别文字，后总结的重要函数）。

1. **display\_page.py文件：**

def display\_text(self, text):设置字体大小与显示模式的函数。

def speak\_text(self):语音合成函数。

1. **option\_page.py文件：**

def setupInfo(self,MainWindow):创建窗口函数。

def optionalInfoButton(self):选项信息处理函数，将用户所选的功能整理成集成信息并传给大模型。

1. **waiting\_page.py文件：**

class Mainwindow3(QMainWindow):定义主窗口3类，用于构建信息提取后的UI界面，即展示界面。

# 3、模块设计

**3.1、软件结构**

以图形方式给出软件系统的子系统（或软件包）划分，模块划分，子系统间、模块间关系等，并用接口来描述各模块之间的调用关系，给出各模块之间的松散耦合关系。



**3.2、功能设计说明**

结合上图阐述软件的基本设计思想和理念。

**接口描述：**

* 应用阿里云DashScope服务，把OCR文字识别后的图片传输给通义千问大模型进行文字信息的提取和总结。
* OCR文件中运用本地的PaddleOCR模型进行图片文字的识别。

**3.3、模块1**

详细描述各功能模块的功能、数据结构、具体算法和流程等。

3.3.1、设计图

3.3.2、功能描述

简要描述模块1的业务功能。

3.3.3、输入数据

详细描述用户输入的数据(包括任何输入设备)以及这些数据的有效性检验规则。

详细描述从物理模型中的哪些表获取数据以及获取这些数据的条件。

3.3.4、输出数据

详细描述模块1所产生的数据以及这些数据的表现形式。

3.3.5、数据设计

给出本程序中的局部数据结构说明，包括数据结构名称，功能说明，具体数据结构说明（定义、注释设计、取值）等。相关数据库表，数据存储设计（具体说明需要以文件方式保存的数据文件名、数据存储格式、数据项及属性等。）

3.3.6、算法和流程

详细描述根据输入数据产生输出数据的算法和流程。

3.3.7、函数说明

具体说明模块中的各个函数，包括函数名称及其所在文件，功能，格式，参数，全局变量，局部变量，返回值，算法说明，使用约束等。

3.3.8 全局数据结构与该模块的关系

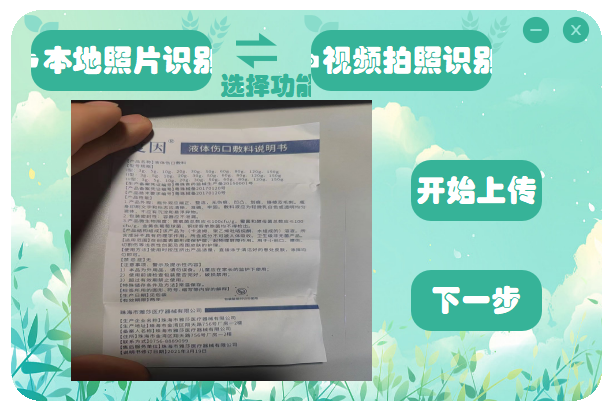
说明该模块访问了哪些全局数据结构。

**3.3、图形用户界面（GUI）设计模块**

3.3.1、功能描述

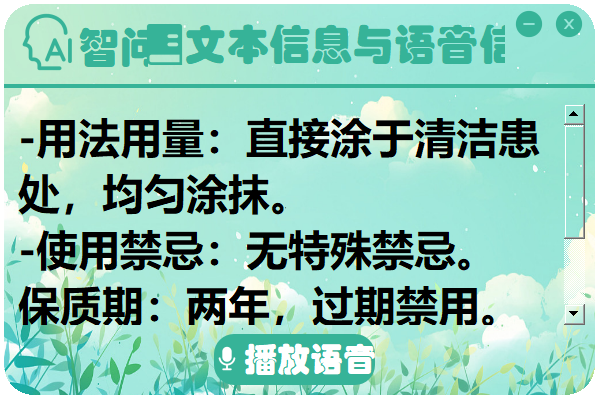












为项目用户展示了绿色精美护眼治愈系的符合老年人需求的GUI操作界面，简单可操作，按钮便捷易于接受，共分为五大模块：

首页：用于启动信息提取流程，有两款功能（本地照片+拍照）可供点击直接跳转至相关功能页面；

上传照片以及摄影拍照的具体功能页面：

供用户根据需求定制查询内容：

等待页面：展示加载动画提升用户体验；

结果展示页面：以图文并茂的形式展示抽取信息并提供语音播报功能，并且可以通过滑动滚动条浏览全文本。

3.3.2、接口描述

在Mainwindow4中，没有显式的接口定义，但其构造函数接受参数（text\_ui, speech\_ui），可以被视为一种内部接口，用于接收外部传入的数据（如要显示的文本信息和语音文本）。

而在Mainwindow2中，通过按钮点击事件（如OK和others）和复选框状态改变的响应，这些是GUI层面的交互，它们也构成了用户与程序交互的接口。

3.3.3、数据结构描述

* **UploadPage**中，通过def uploadFileFolder(self):和def showcamera(self):等函数实现可选择性地上传本地文件以及实时摄影拍照的方式来选择功能。
* **MainAPP中，通过跳转函数实现到选择功能页面。**
* **Mainwindow4**中，通过self.Info字典在OK方法内的使用，隐含了一个数据结构设计。这个字典用于存储用户选择的信息，例如年龄(age)和其他输入(others)，展现了键值对的形式来组织数据。
* **Mainwindow2**中，同样使用了字典self.Info作为主要的数据结构来收集用户的选择信息，包括年龄、其他特定输入的文本以及一系列勾选的选项。这种结构清晰地反映了应用状态和用户偏好。
* **Mainwindow3**中并没有直接涉及复杂的数据结构，主要操作是一个QMovie对象的初始化和播放，用来展示加载动画。

3.3.4、实现思路

* **Mainwindow4**的实现围绕着展示文本信息和播放语音的功能展开。通过加载预先设计的UI文件，动态添加文本内容和播放按钮，采用多线程技术保证语音播放不会阻塞UI界面，提升了用户体验。
* **UploadPage**中，围绕可选择性地上传本地文件以及实时摄影拍照的方式来选择人性化功能，大大增加便利性。
* **Mainwindow2**的设计注重于用户配置选项的管理。通过UI界面收集用户对查询选项的选择，包括基本信息、特定年龄段以及其他自定义输入。实现中利用事件监听机制响应用户操作，确保至少有一项被选中，并且能够灵活处理“其他”选项的输入情况。
* **Mainwindow3**专注于展示等待界面的实现，通过加载并播放GIF动画(loading.gif)，为用户提供视觉反馈，表明系统正在处理请求或加载数据，提高了用户等待过程中的体验。

3.3.5、其他

* **模块化设计**：每个Mainwindow类代表了一个独立的功能模块，通过继承QMainWindow实现了高度的可复用性和扩展性。这种设计使得后续增加新功能或调整现有界面变得相对容易。
* **资源管理**：各UI文件既有通过uic.loadUi动态加载，也有通过import UI文件导入，这意味着UI设计与逻辑代码分离，便于UI设计师和开发人员协同工作，同时也简化了资源管理和维护。
* **用户体验**：各模块均考虑了用户体验，如通过动态调整布局、控制组件可见性、即时反馈用户操作等，体现了良好的人机交互设计理念。

**3.4、图像识别技术（OCR）模块**

3.4.1、功能描述

图像识别模块主要实现了从图像中提取文字信息的功能，采用了PaddleOCR库，这是一个基于深度学习的光学字符识别工具，适用于中文字符的识别。模块通过调用PaddleOCR的接口，能够对指定路径下的图片进行分析，准确地识别并提取图片中的文字内容，同时支持文字方向检测，即使图片中的文字存在倾斜也能有效识别。

3.4.2、接口描述

* **extract\_text\_from\_images(img\_path: str) -> str**

输入参数：

img\_path: 字符串类型，表示待识别图片的文件路径。

返回值：

识别结果字符串，包含图片中所有识别到的文字信息，每行文字之间用换行符\n分隔。

功能描述： 该接口负责接收图片路径，利用PaddleOCR对图片进行文字识别处理，整合所有识别到的文字输出为一个字符串。

3.4.3、数据结构描述

* **result**: 列表类型，PaddleOCR识别结果的集合。每个元素是一个元组，包含文字的位置信息和文字内容及其置信度。
* **text\_result**: 字符串类型，最终返回的文本结果。由识别出的所有文本行组成，每行文本通过换行符分隔。

3.4.4、实现思路

1. **环境配置**: 首先通过os.environ['KMP\_DUPLICATE\_LIB\_OK']='True'解决潜在的多线程冲突问题，保证程序运行稳定。
2. **模型初始化**: 利用PaddleOCR类初始化OCR模型，开启角度分类（use\_angle\_cls=True）以便更准确地识别倾斜文字，并指定使用GPU加速（use\_gpu=True）和中文识别（lang="ch"）。
3. **文字识别**: 定义extract\_text\_from\_images函数，接收图片路径作为输入。调用PaddleOCR的ocr方法对图片进行识别，并通过循环遍历识别结果，提取每一行文字内容，合并成单一字符串返回。
4. **结果整合**: 在循环中，每个识别结果中的文字和其置信度被解包，仅保留文本内容并通过换行符连接，形成连贯的文本输出。

3.4.5、其他

* **性能与扩展性**: 通过配置使用GPU，显著提升了图像识别的速度，适合处理大量或高分辨率图像。此外，由于PaddleOCR支持多种语言识别，只需调整初始化参数，即可轻松适应多语言环境。
* **错误处理与日志**: 虽然当前代码未直接体现，但在实际应用中，应考虑添加异常处理逻辑，比如处理图片读取失败、PaddleOCR执行错误等情况，并记录日志以供调试和监控。
* **资源管理**: 注意，使用GPU时需确保系统已安装并正确配置PaddlePaddle GPU版本，且考虑GPU资源的有效管理和分配，避免与其他GPU密集型任务冲突。

**3.5、信息处理与大模型交互模块**

3.5.1、功能描述

本模块旨在从药品说明书图片中提取关键信息，并利用大型语言模型（如Qwen-Turbo）对提取的文本进行理解和加工，生成既适合图形用户界面展示又适合语音播报的定制化内容。通过与图像识别模块的集成，它首先识别图片中的文字信息，然后构建特定的提示（prompt）来引导大模型提取并格式化信息，最后将处理后的信息分发给GUI模块以实现信息的高效、友好展示和语音传达。

3.5.2、接口描述

* **get\_drug\_info\_from\_images(img\_path, Info)**

**输入参数**:

img\_path: 字符串，表示药品说明书图片的路径。

Info: 字典，包含用户配置信息，如年龄范围、特定查询选项等。

**输出**:

两个字符串，分别为UI展示的文本和语音播报的文本。

**功能**: 整合图像识别与信息处理流程，从图片中提取信息并经大模型处理后返回适配GUI和语音的文本内容。

3.5.3、数据结构描述

**Info**: 字典结构，用于存储用户查询的配置信息

**Response Structure**: 大模型返回的响应结构，包含一个或多个选择，每个选择含有内容信息

3.5.4、实现思路

1. **图像识别**: 利用PaddleOCR从图片中提取文字信息，作为后续处理的基础数据源。
2. **构建Prompt**: 根据用户配置信息Info，构建一个细致的提示文本，引导大模型理解和聚焦于特定需求，如年龄段特化的解释要求。
3. **大模型交互**: 调用DashScope API与Qwen-Turbo模型交互，传入构建的prompt，获取模型的智能处理结果。
4. **信息分割与格式化**: 分离大模型返回的综合文本为UI展示文本和语音播报文本，并对UI文本进行进一步格式化处理，确保信息清晰且易于用户理解。
5. **结果输出**: 将处理后的信息返回至调用者，用于界面展示和语音合成。

3.5.5、其他

* **API Key管理**: 使用环境变量dashscope.api\_key存储API密钥，确保安全性和可配置性，但需注意在生产环境中应避免硬编码敏感信息。
* **错误处理**: 当前代码中，如果大模型响应中未找到预期的标签，程序会直接终止。在实际应用中，应增加更详细的错误处理逻辑，比如尝试重新请求、记录日志或返回默认提示信息给用户。
* **性能与成本考量**: 大模型的调用可能会带来一定的延时和成本开销，应考虑缓存机制减少重复调用，尤其是在处理相同或相似药品说明书的情况下。
* **安全性与合规性**: 处理医疗信息时，确保遵守相关法律法规，保护用户隐私和数据安全。

**3.6、语音合成与播报模块**

3.6.1、功能描述

此模块旨在将从药品说明书中提取的文字信息转化为语音，实现语音播报功能，以辅助视觉受限的用户理解药品使用方法、注意事项等重要信息。用户在界面上选择或上传图片后，应用会自动提取信息并转换成语音播放，提升用户体验和信息获取的便捷性。

3.6.2、接口描述

* **接口名称**: synthesize\_speech
* **输入参数**:
  + text: 待转换的文本字符串（必填），即从图片中提取的药品信息文本。
  + language: 语言代码（可选），默认为中文（如："zh-CN"），用于指定合成语音的语言。
  + voice\_type: 声音类型（可选），比如"female"或"male"，用于选择不同的发音人声。
* **输出**:
  + audio\_file\_path: 合成的语音文件的存储路径。
* **异常处理**:
  + 如果合成失败，应抛出异常并提供错误信息。

3.6.3、数据结构描述

* **SpeechRequest**: 用于封装合成语音请求的类。
  + text: 文本字符串。
  + language: 语言代码。
  + voice\_type: 声音类型。
* **SpeechResponse**: 表示语音合成响应的类。
  + file\_path: 语音文件的路径。
  + status: 成功或失败的状态码。
  + message: 可选的错误或成功信息。

3.6.4、实现思路

1. **集成语音合成库**: 首先，需要集成一个语音合成服务，例如Google Text-to-Speech、Microsoft Azure TTS或阿里云TTS等。这些服务通常提供了丰富的API来支持不同语言和声音类型的语音合成。
2. **设计语音合成逻辑**: 在MainWindow类中添加一个方法，如generate\_and\_play\_speech，用于调用语音合成接口，并处理合成后的音频文件。这个方法首先利用上文提到的synthesize\_speech接口，传入从图片提取的文本信息以及预设的语音参数（如语言和声音类型）。
3. **音频播放**: 合成语音后，使用PyQt5的多媒体组件（如QMediaPlayer）来播放生成的语音文件。确保在播放完毕后正确地清理资源，如关闭播放器实例。
4. **错误处理**: 在语音合成和播放过程中，加入异常处理逻辑，确保任何错误都能被妥善捕获并通知用户。

3.6.5、其他

* **性能与异步处理**: 考虑到语音合成可能需要一定时间且不应阻塞UI线程，可以考虑在单独的线程中执行语音合成任务，类似于已有的药品信息提取的后台处理逻辑。
* **用户反馈**: 提供简单的用户控制，如暂停、继续和重复播放语音的功能，增加应用的交互性和实用性。
* **音频文件管理**: 设计机制定期清理临时生成的语音文件，避免长时间运行导致存储空间占用过多。
* **国际化支持**: 根据应用的定位，语音合成模块应支持多语言，以便全球用户使用。这要求接口设计时考虑语言参数的灵活性。

# 4、 接口设计

**4.1、用户接口**

**GUI模块**:

* **功能展示**: 提供以草绿色为主的护眼精美治愈系并且清晰的界面图片及各控件布局，展示从图像识别和信息处理得到的多种药品信息，包括基本用法、禁忌等。
* **交互按钮**: 包括“上传图片”按钮用于选择或拍摄药品说明书图片，“打开摄像头”可以随时打开本地摄像头达到实时反复摄影，“拍照”实现摄影保存。“播放语音”按钮用于听取药品信息的语音播报，“配置查询”按钮跳转到选项配置页面。
* **进度指示**: 显示等待界面和加载动画，当系统处理图像或与大模型交互时，让用户了解进程状态。
* **反馈提示**: 通过弹窗提示用户操作结果，如上传成功、配置选项未选或处理完成等。

**语音合成与播报模块**:

* **播放控制**: 用户可以通过界面控制语音播放，包括播放、暂停、停止。并且如果文本信息过多，可以使用简单明了的滚动条来达到观看所有文本信息的目的。
* **语音设置**: 提供选项让用户选择语音速度、性别等个性化设置

**4.2、外部接口**

**图像识别与OCR模块**:

* **外部API接入:** PaddleOCR库可视为外部接口，通过它调用图像识别服务。
* **文件系统:** 读取本地图片文件，作为图像识别的输入，可看作与文件系统的接口交互。

**信息处理与大模型交互模块:**

* **DashScope API:** 通过DashScope API与Qwen-Turbo大模型交互，提交处理请求和接收响应。
* **文件读写:** 读取图片处理后的文本信息，以及保存临时或处理后的数据到文件系统，如提取的药品信息文本。

**4.3、内部接口**

**GUI与后端逻辑交互:**

* **上传图片:** GUI调用图像识别模块的接口，传递图片路径，获取识别后的文本信息。
* **配置信息:** 选项配置页面收集用户设置后，通过内部接口传递给信息处理模块，作为构建prompt的基础。
* **结果显示:** 信息处理模块处理后，通过内部接口将UI展示文本和语音文本传回GUI模块，用于展示和播报。

**信息处理模块与语音合成**:

* **文本到语音**: 信息处理模块处理完成后，通过内部接口将语音文本传递给语音合成模块，触发语音合成流程。

**线程管理**:

* **后台任务**: 通过Worker类（继承自QThread）实现的内部接口，管理耗时任务（如信息提取和语音合成）在后台线程执行，完成后通过信号（如finished信号）通知主线程更新UI。

5、数据库设计

描述所使用的数据库系统,及数据库和数据表设计。如果系统不以数据库方式存储数据则可省略。

无数据库设计。

# 6、系统出错处理

**6.1、 出错信息**

用一览表的方式说明每种可能的错误和故障，以及系统输出信息的形式、含义和处理方式。

系统应详细列出所有可能出现的错误场景，包括但不限于：

* **序号 | 错误码 | 描述 | 原因 | 处理建议** --- | --- | --- | --- | --- | --- | --- E00001 | 图片上传失败 | 格式不支持 | 请上传JPG、JPEG或PNG格式。 E02 | 识别失败 | 图片清晰度低 | 请提供更清晰图片。 E3 | 问题 | 未识别 | 检有相关问题 | 提供问句式E4 | 数据库连接 | 数据库错误 | 检误 | 重试或联系管理员

**6.2、 补救措施**

说明故障出现后可能采取的补救措施，如恢复、再启动技术等。

* **日志记录**：所有错误信息应被详细记录在日志中，便于追踪分析。
* **提示用户反馈**：友好\*\*：错误信息需转换为用户友好的提示，告知问题及解决方案。
* **重试机制**：对于临时错误，自动重试机制，如重连接数据库。
* **监控与报警**：实时系统监控，一旦错误发生，立即通知开发或运维团队。

# 7、其他设计

如系统安全设计、性能设计等。

**7.1、系统安全设计**

* **数据加密**：上传的图片和敏感数据应使用SSL/TLS加密传输，数据库存储加密。
* **隐私保护**：遵循法规：处理用户数据时遵守GDPRIV、HIPAA等隐私法规。
* **权限控制**：严格访问权限管理，仅授权人员访问特定模块。

**7.2、性能设计**

* **优化**：优化算法与模型，减少处理时间，提高OCR、检索速度。
* **缓存**：使用，减少数据库查询，如索引，提高响应速度。
* **并发**处理：设计，利用并行处理多线程或分布式系统，处理请求。

**7.3、可扩展设计**

* **模块化**设计：确保易于添加新功能模块，不影响现有系统稳定。
* **配置**动态：资源，可根据负载动态调整资源，应对高流量。
* **兼容**：技术栈，技术栈选择保持技术栈的前瞻性，便于未来升级。

**7.4、维护设计**

* **文档**：详尽维护更新文档，方便交接和理解。
* **测试**自动化：集成测试套件，确保每次修改后系统稳定性。
* **日志**：维护日志，记录变更，便于追溯历史操作。