1、系统概述

1.1、系统简介

(1)项目背景及基本情况介绍

1. 时间，这位无声无息却又无处不在的存在，拥有着无穷无尽的寿命。自宇宙诞生之初，它便悄然存在，目睹了无数沧海桑田的变迁，一切都在它的流转中消逝，仿佛只是弹指一挥间。相对于时间而言，我们人类的生命显得如此短暂，宛如宇宙中的一粒尘埃。然而，即使在这微小的存在里，我们仍有着对自己而言无比珍贵的东西。
2. 时间无情地带走了我们的体力，消磨了我们的精力，甚至模糊了我们的记忆。但正是在这无尽的流逝中，它逐渐加深了我们对于故人的思念。人们常说睹物思情，当我们凝视那些承载着过往记忆的物品，心中便会涌起无尽的思念。然而，仅仅凭借一张模糊的照片，又怎能真正缓解我们内心深处的思念之情呢？
3. 因此，我们应该更加珍惜与亲人朋友相处的每一刻，让那些美好的瞬间在心中永恒。同时，也要学会释放过去的遗憾和痛苦，让心灵在时间的洗礼中得到净化和升华。
4. 修复老照片系统无疑是人们保存珍贵记忆的重要工具，它在岁月的长河中起到了无可替代的作用。随着时间的流转，老照片所承载的过往记忆逐渐模糊，照片本身也难免受到损害。而修复老照片系统则通过先进的数字图像处理技术，为这些珍贵的记忆提供了修复、保护和传承的可能。

5. 在这个系统的帮助下，那些因岁月侵蚀而变得模糊、破损甚至老化的照片，得以重获新生。通过精确的图像处理算法，系统能够去除照片上的污渍、划痕和褪色，修复破损的边缘和细节，让照片重新展现出原本的光彩。这不仅是对过往美好瞬间的延续，更是对家族文化传统的传承。

6. 修复老照片系统的重要性不仅在于其技术价值，更在于它所承载的人文意义。通过修复这些老照片，我们能够更好地缅怀过去，回忆那些与亲人共度的美好时光。这些照片见证了我们的成长、家庭的变迁以及时代的演进，它们是我们情感连接的纽带，也是我们文化认同的基石。

7. 因此，修复老照片系统不仅是对技术的运用，更是对人文精神的传承和发扬。它让我们得以在时间的长河中，留住那些我们所珍视的过往与情感，让记忆得以永恒。通过这个系统，我们可以更好地纪念过去，陶冶心灵，为未来的生活留下更多宝贵的回忆。

(2) 相关术语介绍

经过调研，我们的系统中可能会使用包括但不限于以下技术，术语介绍如下：

1. **Flask 框架**:

Flask 是一个轻量级 Web 应用框架，采用 Werkzeug 和 Jinja2，自由灵活且可扩展性强。适用于小型网站和 API 开发，快速上手且功能齐全。通过 pip 安装，提供通用接口避免重复工作。路由通过装饰器实现，支持蓝图规划。适用于各种 Web 应用开发。

例如:

- **@app.route()**：Flask 框架中的装饰器，用于定义路由规则，将 URL 映射到特定的视图函数。
- **request**：Flask 中的一个全局对象，表示客户端发送的 HTTP 请求。它包含了请求的各种信息，如方法（GET、POST 等）、参数等。
- **make_response()**：Flask 中用于创建响应对象的函数。
- **app.run(debug=True)**：启动 Flask 应用的命令，debug=True 表示在开发模式下运行，可以自动重新加载代码。

2. **文件操作**:Python 文件操作非常便捷，可通过内置函数实现读写。使用 open()函数打开文件，指定模式（如'r'读取、'w'写入等）。read()、write()方法分别用于读取和写入内容。操作完成后，应调用 close()关闭文件。同时，可利用上下文管理器自动管理文件资源。

例如:

- **open()**：Python 内置函数，用于打开文件。在这里，它用于读取图片文件。

- **read()**：文件对象的方法，用于读取文件内容。
- **os.path.join()**：Python 的 os 模块中的函数，用于连接路径。在这里，它用于构造图片文件的路径.

3. **HTTP 响应**:HTTP 响应是 Web 服务器对客户端请求的回复，包含状态码、响应头和响应体。状态码指示请求处理结果，如 200 表示成功，404 表示未找到。响应头包含元信息，如内容类型、长度等。响应体则是实际返回的数据内容。

   例如:

- **response**：通常指的是服务器对客户端请求的回应。在这里，它包含图片数据以及相应的 HTTP 头信息。
- **response.headers**：HTTP 响应的头部信息。在这里，它被设置为包含图片的类型（Content-Type）。
- **Content-Type**：HTTP 头部的一个字段，表示响应体的媒体类型。在这里，它被设置为'image/jpg'，表示响应体包含 JPEG 格式的图片。

4. **Python 异常处理:**Python 异常处理通过 try-except 语句实现，当 try 块中的代码引发异常时，程序会跳转到相应的 except 块处理异常，避免程序崩溃。同时，还可以使用 finally 块来执行无论是否发生异常都需要执行的代码。

   例如:

- **try...except**：Python 中的异常处理结构。try 块中的代码可能会抛出异常，except 块则捕获并处理这些异常。
- **subprocess.CalledProcessError**：Python 的 subprocess 模块抛出的异常，当子进程返回非零退出状态时触发。在这里，它用于处理执行命令行命令时可能出现的错误。
- **jsonify()**：Flask 中的函数，用于将 Python 字典转换为 JSON 格式的响应体。

5. **Vue 框架**：Vue 框架是一款轻量且强大的 JavaScript 框架，专注于构建用户界面。它采用声明式编程和组件化开发，使开发者能高效构建复杂的前端应用。Vue 的响应式系统能自动跟踪数据变化并更新视图，简化开发流程。

   例如:

   - **v-bind**：用于动态绑定一个或多个属性，或一个组件 prop 到表达式。
   - **v-if**：根据表达式的真假条件渲染元素。
   - **v-on**：用于监听 DOM 事件，并在触发时执行一些 JavaScript 代码。
   - **v-else**：和 v-if 或 v-else-if 一起使用，表示“否则”的条件。
   - **v-for**：基于源数据多次渲染一个元素或模板块。
   - **v-model**：创建在输入元素和 Vue 实例的数据之间的双向绑定。
   - **v-show**：根据条件来切换元素的 CSS 属性 display.
   - **v-pre**：跳过这个元素和它的子元素的编译过程。可以用来显示原始 Mustache 标签。
   - **v-cloak**：保持在元素上直到关联实例结束编译。和 CSS 规则一起用时，这个指令可以隐藏未编

译的 Mustache 标签直到实例准备完毕。

6. **Mysql:**基于 SQL 查询的开源跨平台数据库管理系统)，在我们的系统中，用于存储用户账号中的各种信息，并计划将其部署到云端服务器上便于远程访问。

7. **GFP-GAN**:腾讯在人像复原、超分等方面的开源模型，我们的系统准备利用它的照片超分功能实现老照片的修复功能。详细介绍如下:

GFP-GAN（Generative Face Perfector GAN）是一个基于生成对抗网络（GAN）的模型，专门设计用于人脸图像的生成和优化。其主要功能包括高分辨率生成和人脸美化。通过生成器和判别器之间的对抗训练，GFP-GAN 能够生成高分辨率的人脸图像，同时保持细节和真实感。此外，它还可以对输入的人脸图像进行美化和优化，去除皱纹、瑕疵，增强肤色等，使人物看起来更加年轻和美丽。

在架构上，GFP-GAN 利用了一个退化清除模块，引入了面部恢复损失和一个预训练的 GAN 作为先验，二者通过 Channel-Split SFT 进行桥接。训练过程由三类 Loss 混合控制，包括 Adversarial Loss 以消除复杂退化，Facial Component Loss 以增强面部细节，以及 Identity Loss 以保留面部身份。

在应用中，GFP-GAN 不仅可以生成高质量的人脸照片，例如将低分辨率的照片升级为高分辨率，同时使人脸特征更加清晰和逼真，还可以在视频处理中发挥作用，如在 wav2lip 中作为视频质量判别器，对嘴唇修复后的图像帧进行质量修复，提供更高质量的视频效果。

8. **PaddleGAN**：百度生成对抗网络开发的模型套件，包括多种功能，包括但不限于超分、插帧、上色、换妆、图像动画生成等功能，我们打算利用其中的动作迁移模型（first-order-demo）实现照片动起来的功能.详细介绍如下:

> PaddleGAN 是一个基于 PaddlePaddle 深度学习框架的生成对抗网络（GAN）工具包。它能够用于图像、视频和音频等多种领域的应用，如图像修复、图像合成、视频超分辨率和语音转换等。该模型利用生成器和判别器两个模块的相互博弈学习，生成高质量的伪造样本。
>
> PaddleGAN 图像生成模型库覆盖当前主流的 GAN 算法，用户可简单上手各类 GAN 任务，并方便扩展自己的研究。例如，Pix2Pix 和 CycleGAN 可以采用 cityscapes 数据集进行风格转换，而 StarGAN、AttGAN 和 STGAN 则可以采用 celeba 数据集。
>
> 除了基本的图像生成功能，PaddleGAN 还提供了多种应用场景。它不仅可以修复历史影像，还可以进行 AI 换脸、动作迁移、图像风格迁移等功能。在医学领域，PaddleGAN 可用于生成医学图像，帮助医生进行诊断和治疗。在游戏领域，它可以用于生成游戏场景和角色。
>
> PaddleGAN 的优点在于能够快速地训练和生成高质量的伪造样本。此外，它还内置了一些预训练模型和示例代码，使用户可以快速上手并进行自定义训练。
>
> 总之，PaddleGAN 是一个功能强大、灵活易用的 GAN 工具包，适用于多种应用场景，并为用户提供了丰富的资源和支持。如需更多关于 PaddleGAN 模型的信息，建议查阅 PaddleGAN 的官方文档或相关学术论文。

9. **Json**(JavaScript Object Notation):即 JavaScript 对象表示法，是一种轻量级的数据交换格式。它基于 ECMAScript 的一个子集，采用完全独立于语言的文本格式来存储和表示数据。Json 具有简洁、清晰的层次结构，使得数据交换变得容易和迅速。

1.2、系统目标

（1）功能目标

1. 客户端新用户注册功能描述：

允许新用户通过客户端界面注册账号，提供必要的注册信息。

2. 客户端用户登录功能描述：

为已注册用户提供登录服务，验证用户身份，并授权访问系统。

3. 老照片清晰度修复功能描述：

利用图像处理技术，提升老照片的清晰度，使其焕发新生，还原当时的记忆。

4. 表情动作迁移功能描述：

通过面部识别和图像生成技术，为老照片中的人物添加微笑、眨眼等表情动作，使其更加生动。

（2）性能目标

1. 清晰度提升：显著提高照片的整体清晰度，包括细节和纹理的恢复。

2. 脸型保持：在修复过程中，保持原照片的脸型不变，尽量减少脸型的扭曲或变形。

3. 色彩还原：恢复照片的真实色彩，使其看起来更加自然和生动。

4. 噪声去除：减少照片中的噪声和杂点，提高整体视觉效果。

5. 实时处理：实现快速处理，以满足用户实时修复的需求。

6. 用户体验流畅:提供简洁明了的用户界面，确保操作流畅，响应迅速。同时，为用户提供详细的操作指南和常见问题解答，以提升用户满意度。

说明:由于我们自己的电脑算力有限，并且未购置服务器，目前前端网页只能在本地运行 localhost，因此只能用我们的电脑同时模拟后端和用户前端界面操作,未来可能会实现服务器功能，甚至面向大众进行测试和使用.

1.3、系统运行环境

操作系统	Windows 64 位操作系统
数据库系统	MySQL 数据库
编程平台	本地使用 Pycharm WebStorm Vscode 进行编程
网络协议	TCP 传输控制协议
硬件平台	笔记本电脑

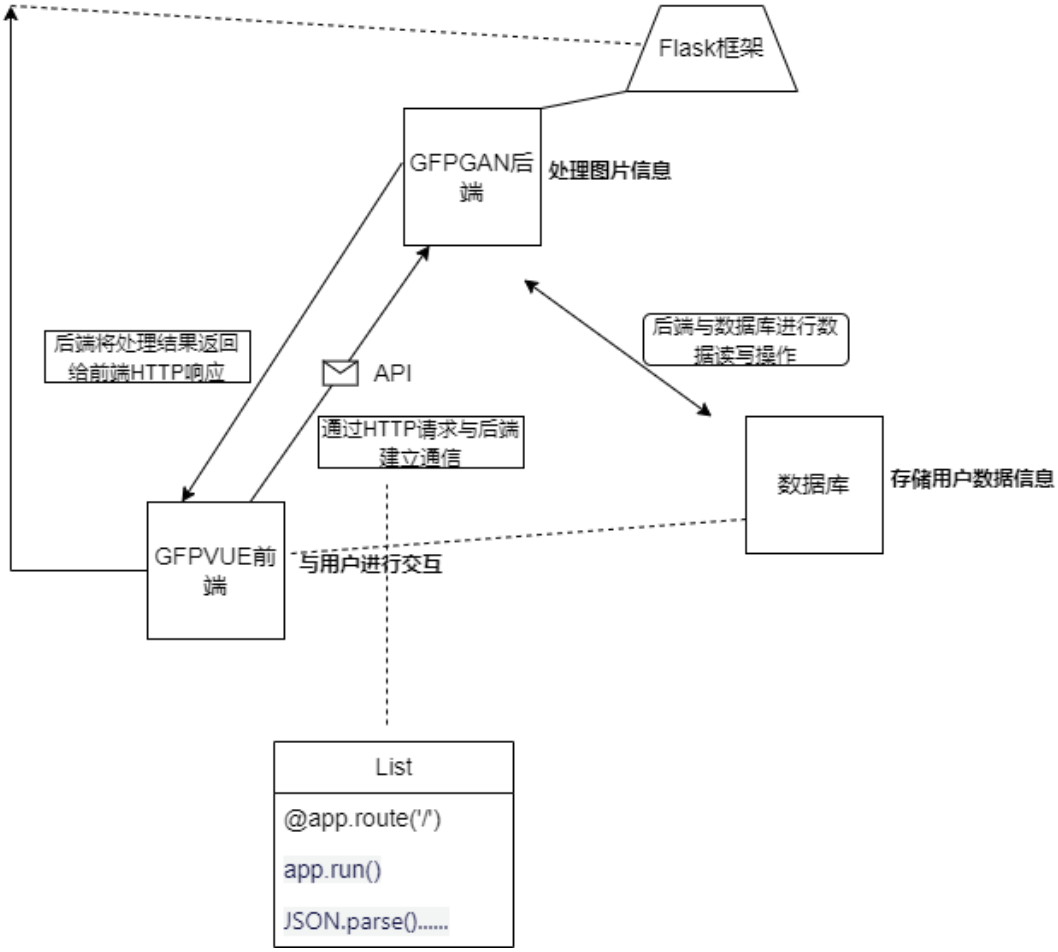
1.4、开发环境

工具软件	PyCharm 2019.3.3 x64 WebStorm 2023.1 Vscode1.88.0
开发语言	Python3.7 .Vue3.0 .Web 开发(HTML5 CSS3)
模块(库)版本	tqdm~=4.66.2 PyYAML==5.3 scikit-image~=0.20.0 scipy~=1.9.1 opencv-python==4.9.0.80 imageio==2.6.1 imageio-ffmpeg==0.4.9 librosa==0.8.1 numba==0.48.0 easydict==1.12 munch==4.0.0 natsort==8.4.0 matplotlib==3.1.3 basicsr==1.4.2

	facexlib==0.3.0 lmdb==1.4.1 numpy~=1.24.4 tb-nightly==2.12.0a20230113 torch==1.13.1 torchvision==0.14.1 yapf==0.28.0

2、总体结构设计

2.1、软件结构



2.2、设计思想

结合上图阐述软件的基本设计思想和理念。

1. 首先我们将 GFPGAN 后端 数据库部分和 GFPVUE 前端互相解耦合，这样使代码具有更高的可维护性和可拓展性，便于开发。对于后端，

2. 该软件基于 Flask 框架构建，通过采用这个轻量级且易于扩展的 web 框架，可以实现高效的 web 应用开发。Flask 框架的灵活性使得开发者能够根据需要定制和扩展软件功能，满足特定需求。
3. 其次，该软件在设计中强调数据的处理和存储。后端服务器与数据库进行数据读写操作，确保数据的准确性和一致性。在处理 GFPGAN 生成的图像信息时，后端服务器进行必要的后处理，并将处理结果返回给前端。这种设计思想使得软件能够高效地处理大量图像数据，并为用户提供高质量的图像输出。
4. 此外，该软件还注重前后端的交互和通信。前端通过 HTTP 请求与后端建立通信，获取处理后的结果。同时，前端还使用 GFPVUE 与用户进行交互，接收用户的输入或输出数据。这种设计使得软件能够实时响应用户的操作，提供流畅的用户体验。
5. 最后，软件在整体架构上采用模块化的设计思路。通过使用@app.route(...)装饰器定义路由和 URL 模式，使得不同的功能模块可以独立开发和维护。同时，app.run()命令用于启动应用程序，简化了软件的部署和启动过程。

3、模块设计

表 3-1 各模块功能、接口信息

名称	GFPVUE(前端)	GFPGAN(后端)	MySQL 数据库
功能	实现与用户之间的交互,例如上传照片,注册账号,登录,用户信息存储等等功能,后端处理后的图片返回到前端，以返回给用户结果.	处理图片信息利用大模型和算法对老照片进行修复，同时能够根据驱动视频实现动态效果.	完成相应的数据库存储与增删改查操作
接口	应用 API（应用程序编程接口）与后端进行连接，向后端发送 http 请求.	通过 Flask 框架定义/process_image 路由请求,建立 app.py 文件实现接口	通过数据库管理系统（DBMS）和 SQL（结构化查询语言）来实现

3.1、模块 1——GFPVUE(前端)

3.1.1、功能描述

403.vue 404.vue	Vue 框架中的自定义 403 和 404 错误页面组件，包括错误代码、描述和处理方式，提供返回首页和返回上一页功能。
home.vue icon.vue	home.vueVue 框架中的布局组件，包含头部、侧边栏和内容区域，支持侧边栏折叠，内容区域可动态切换页面并缓存部分页面。icon.vue 实现了图标使用说明的展示，并提供了搜索图标的功能。用户可查看图标示例，通过关键词搜索图标，并直接通过

	类名使用图标。
login.vue	实现了图像处理系统的登录界面，包含用户名和密码的输入验证，点击登录后执行表单验证，验证成功则显示成功提示并设置权限。
upload.vue	展示了 Element Plus 的上传组件，支持拖拽上传文件，并提供了 vue-cropperjs 组件的链接和使用示例。
user.vue	展示用户基础信息和账户编辑界面，支持裁剪并上传头像，修改用户名和密码，编辑个人简介，保存修改。
uploadFile_Dynamic.vue	实现了图片上传功能，并附带视频展示。用户可上传图片，上传成功后展示相关视频，支持拖动上传和点击上传两种方式。
uploadFile_static_state.vue	实现图片数据上传功能，上传成功后显示检测结果，并可通过 Element Plus 组件进行图片预览和缩放。
components 模块	设计网页页面基本样式例如，侧边栏，顶边栏，标签栏等
assets 模块	css 样式和图片为网页提供一致的视觉样式和布局。它确保了页面元素在视觉上的一致性和美观性，同时提供了交互效果（如过渡动画）。这些样式也增强了用户体验，使页面内容更加易于阅读和浏览。特别是 Element UI 的图标和组件样式，为页面提供了丰富的功能和交互性。
router 模块	定义路由关系,用于配置 Vue.js 应用程序的路由（routes）

3.1.2、接口描述

1. 登陆页面(login.vue):Element Plus 组件库接口

Vue Router 接口.Vue Composition API 接口

2.上传页面(upload.vue):Element Plus 组件库接口

3.用户界面(user.vue):Element Plus 组件库接口.Vue-Cropper 插件接口.自定义接口（@click="showDialog"：点击头像时触发的事件，用于显示图片裁剪对话框。@click="onSubmit"：点击保存按钮时触发的事件，用于提交表单数据。@change="setImage"：文件输入框选择图片后触发的事件，用于设置裁剪组件的图片源。@click="saveAvatar"：点击上传并保存按钮时触发的事件，用于处理图片裁剪后的上传和保存逻辑。）

4.动态上传处理(uploadFile_Dynamic.vue)：

http://localhost:5000/process_image（链接后端代码）

5.静态上传处理(uploadFile_static_state.vue):

文件上传接口：http://localhost:5000/upload

模型调用接口:/gfpgan

3.1.3、数据结构描述

本功能区域使用的数据结构主要为数组和对象。数组用于存储列表数据，对象则用于描述单个实体的属性和方法，实现数据的层次化组织，另外包括包括文件上传状态、图片 URL 列表、分页查询参数及响应结果.

3.1.4、实现思路

我们采用将前端一整个大的模块分为小模块进行编程，以实现代码的可调性，下面给出关键部分实现思路.

1. uploadFile_Dynamic.vue

一、Vue 组件的基本结构

首先，代码使用了 Vue 的单文件组件结构，包含了模板（template）和脚本（script）部分。模板部分主要定义了组件的 HTML 结构，脚本部分则包含了组件的逻辑和数据。

二、组件的功能模块

1. 图片上传模块

组件的上方部分是一个图片上传模块，使用了 Element Plus 的 el-upload 组件。用户可以通过拖拽或点击上传图片，上传的图片会发送到后端服务器进行处理。在上传过程中，使用了全屏加载的提示效果，通过 v-loading 指令实现。

在脚本部分，定义了相关的响应式数据（如 query、fullscreenLoading 等）和方法（如 openFullScreen1、handleUploadChanged 等）。当图片上传成功或失败时，会触发相应的回调函数（dealimage 和 uploadFileError）。

2. 视频展示模块

组件的下方部分是一个视频展示模块，使用了 HTML 的 video 标签。当视频源（videoValue）发生变化时，会重新加载视频。视频的源可以通过 Base64 字符串指定。

在脚本部分，定义了视频源（videoValue）的响应式数据，并通过 watch 监听其变化，当视频源发生变化时，调用 videoPlayer 的 load 方法重新加载视频。

三、组件的交互逻辑

1. 图片上传与后端交互

当用户选择图片进行上传时，组件会将图片文件发送到后端服务器进行处理。这个过程通过 axios 库实现 HTTP 请求。在请求发送前，可以通过设置请求头、请求体等参数来控制上传行为。上传成功后，后端会返回处理结果，组件可以通过回调函数来处理这个结果。

2. 视频展示与数据绑定

视频展示模块通过绑定视频源（videoValue）来展示视频。视频源可以是一个 Base64 字符串，也可以是一个 URL。当视频源发生变化时，组件会自动重新加载视频。这种数据绑定的方式使得组件更加灵活和可维护。

四、组件的样式与图标

组件的样式通过 CSS 类名进行定义，如 container、content-title、plugins-tips 等。这些样式类可以在组件的外部样式文件中定义，也可以在组件内部通过 style 标签定义。

此外，组件还使用了 Element Plus 的图标组件（如 Delete、Edit、Search、Plus）以及自定义的上传图标（upload-filled）。这些图标增强了组件的视觉效果和用户体验。

五、组件的响应式与监听

组件使用了 Vue 的响应式系统和监听机制来管理数据和状态。通过 reactive 和 ref 函数创建响应式数据，通过 computed 计算属性和 watch 监听器来观察和响应数据的变化。这种机制使得组件能够根据数据的变化自动更新视图，实现了数据与视图的双向绑定。

2. uploadFile_static_state.vue

1. **Vue 组件构建**:

在<template>标签中，通过 Vue 的模板语法，定义了一个带有图片上传和展示功能的组件。该组件包含两个部分，一部分是图片上传区域，另一部分是图片展示区域。

2. **图片上传区域**:

   a. 使用<el-upload>组件作为图片上传的容器，并设置了 drag、limit、show-file-list 等属性，以支持拖拽上传、限制上传文件数量以及显示文件列表。

   b. action 属性指定了上传图片的服务器地址。

   c. :on-success 和:on-error 属性分别绑定了处理上传成功和上传失败的回调函数。

   d. <el-icon>和 提供了上传区域的视觉提示。

3. **图片展示区域**:

a. 使用<el-image>组件展示上传后的图片，通过:src 绑定图片的 URL，同时设置了一些缩放和预览相关的属性。

4. **响应式数据和方法**：

a. 使用 reactive 和 ref 创建了响应式的数据对象，如 query、pageTotal、fullscreenLoading 等，用于存储组件的状态。

b. 定义了 openFullScreen1 方法，用于控制全屏加载状态的显示。

c. dealimage 方法作为上传成功的回调函数，当图片上传成功后，它会将图片的路径作为参数发送到/gfpgan 接口，并处理返回的数据。

5. **请求处理**：

a. 使用 axios 库发送 HTTP 请求，当图片上传成功后，通过 axios.get 方法发送一个带有图片路径的请求到/gfpgan 接口，处理返回的响应数据。

6. **生命周期钩子**：

a. 在 onMounted 生命周期钩子中，可以执行一些组件挂载后的初始化操作，比如获取图片数量等。

7. **样式和图标**：

a. 通过<style>标签或者外部 CSS 文件定义了组件的样式。

b. 使用了 Element Plus 的图标组件，如<upload-filled />等，提供了上传区域的图标。

3.2、模块 2 ——GFPGAN(后端)

3.2.1、功能描述

1. 收到前端网站传来的信号（HTTP 请求），对用户上传的图像做不同的处理。根据用户需求对照片进行超分，动态化等处理。为保证效果最好，我们分开进行处理。对照片处理完毕后，将结果返回给网页，网页交互界面显示结果。同时，将用户上传图片的存储地址和结果地址储存到数据库中，记录用户操作记录。

3.2.2、接口描述

1. 与前端网站链接，建立 flask 框架，通过 app.py 与前端网站进行链接。

```
上传文件接口：@app.route('/upload', methods=['POST'])
显示图片接口：@app.route('/img/<string:name>')
获取上传文件接口：@app.route('/getupload/<string:name>')
提供面部修复和增强功能:@app.route('/gfpgan', methods=['GET'])
```

2.

3. 与数据库进行连接，数据库名 gan，端口 3306，user：root，域名是本地域名：127.0.0.1。通过导入 import pymysql，使用数据库语句进行数据库的操作
4. 连接数据库 db = pymysql.connect(host='localhost', user='root', passwd=' ', port=3306)

```
# 使用 execute()  方法执行 SQL 语句
cursor.execute("SELECT VERSION()")
# 使用 fetchone() 方法获取单条数据.
data = cursor.fetchone()
```

5. 通过创建 sql 语句并通过 execute()方法执行，实现数据的插入，删除，和更新；例如：INSERT INTO+表名

3.2.3、数据结构描述

1. 本功能区域使用的数据结构。

数据结构为字符串类型，后端代码从前端接口中获取用户上传照片的存储地址，在修复或动态化之后，将结果的存储地址，都以字符串的形式存储到数据库 gan 的 user 表中的对应字段中。为用户的操作进行记录。

3.2.4、实现思路

我们的整体后端设计是建立在两个开源大模型的基础上实现的。我们利用 GFPGAN 在 git 上的开源项目中已经训练好的超分模型，根据使用文档搭建超分处理模块，调整超分参数，在对照片尽可能还原的基础上，实现对老照片像素的修复。PaddleGAN 是一个多功能的集成模型，我们利用其中的动作迁移算法实现人像照片的动态处理。具体就是把相关模块的算法提取出来，自己配置好路径接口，调整动作迁移算法中的参数，形成我们自己的照片动态处理模块。照片动态处理需要用户提供照片，得到照片后根据我们提供的驱动视频，将驱动视频中人物的表情动作迁移到用户提供的照片上，生成一个 MP4 的视频，这就是照片动态化的整个过程。完成这两个模块之后，将其融合在一起，形成我们的后端项目 GFPGAN，最后我们再写与前端，数据库的连接接口，将前端，后端，数据库全部连接起来。

3.3、模块 3——数据库

3.3.1、功能描述

数据层 db，这里我们使用 mysql 数据库，通过 cpp 文件中函数的编写，实现用户的注册输入，登陆判定，头像上传，在线用户的登录信息更改，用户操作记录等众多数据信息的存储与增删改查操作。

3.3.2、接口描述

该模块更多是定义并编写数据库操作函数，由业务层 cpp 文件中各业务处理函数进行调用，对指定数据库中的数据进行增删改查操作。

数据库名 gan，端口 3306，user：root，域名是本地域名：127.0.0.1。

sql 语句方法：

如果数据表 EMPLOYEE 已经存在，删除表 DROP TABLE IF EXISTS EMPLOYEE

创建表：CREATE TABLE EMPLOYEE (字段一 数据类型，字段二 数据类型，...)

3.3.3、数据结构描述

存储过程中利用 string 或 int 结构输入数据，具体数据类型根据更改记录的字段选择

用 bool 作函数返回值，可以反映数据库的连接情况

3.3.4、实现思路

后端：导入 MySQL 数据库操作模块 import MySQLdb as MySQL

有 MySQL.connect()函数连接数据库，用获取操作游标 cursor = db.cursor()，MySQL.update（）函数进行数据更新，使用 cursor.execute()方法执行 SQL 语句,将 sql 操作语句传入 cursor.execute(sql)，从而实现数据的增删改查。

前端：

有 MySQL::connect()函数判断是否连接成功，MySQL::update（）函数进行数据更新，以及还有 MySQL::query（）查询函数，由业务层相应的 model 调用这些函数，并将 sql 操作语句传入，从而实现数据的增删改查。

4、数据库与数据结构设计

4.1、 数据库及数据表

因为还没有完成数据库部分的代码完成，数据库和数据表的设计为暂定设计。使用 MySQL 数据库，里面包含记录用户信息，包括用户的用户名，昵称，密码，上传记录，头像等五个字段，共一个表。主要结构如图所示

```
DESC user
```

Field	Type	Null	Key	Default	Extra
id	int	YES		NULL	
nick_name	varchar(100)	YES		NULL	
mima	int	YES		NULL	
touxiang	varchar(100)	YES		NULL	

OK, 4 records retrieved in 2.48ms

4.2、 数据结构设计

给出本系统内所使用的每个数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系。

我们存储的数据主要为 varchar，int 两种类型，varchar 所设定的最大长度大多为 100.

表格命名为 user 表，user 表下有 id，nick_name，mima，touxiang，jilu 五个字段，分别对应用户 id，用户名昵称 nick_name，密码，头像与上传记录。字段类型分别是 int，varchar，int，varchar，varchar。昵称，上传记录，头像都设置为字符串类型，长度限制 100，满足用户需求的同时，也满足存储需求，因为上传记录，头像存储的都是图像路径，字符串占用长度很长。

4.3、 数据存储设计

给出本系统内所使用的每个数据结构中的每个数据项的存储要求,访问方法、存取单位、存取的物理关系(索引、设备、存储区域)、设计考虑和保密条件。

user 表 里的 id 我们设为 AUTO_INCREMENT 即自动延续，nick_name 考虑到不可重名设定为 UNIQUE，所有字段都默认为 NULL 。

在使用时，我们利用 mysql 数据库的 connect（）函数判断是否连接成功，连接成功后，利用 mysql_query（）进行 sql 语句的更新或者查询操作，这是基本的更新与查询数据库操作，具体实现则是在业务层各种 model 里面中。

并且我们起初将数据库配置在本地，待整体测试完成后，再将其部署在云服务器上，实现更加大范围的访问使用。

5、接口设计

5.1、外部接口

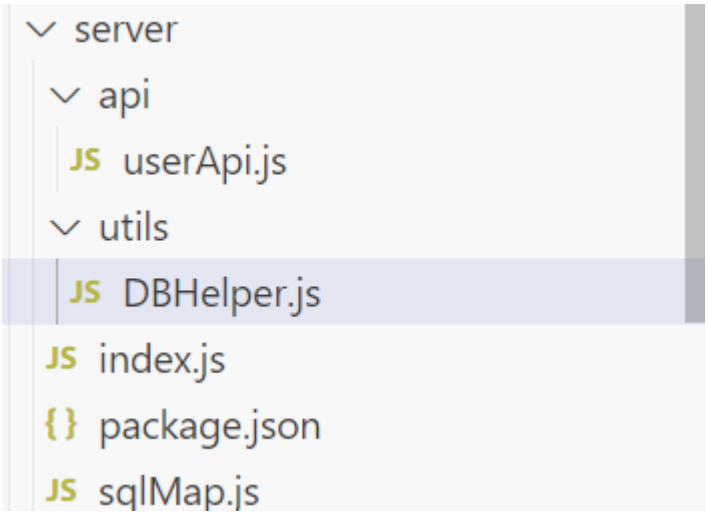
我们的系统是一个相对独立的前端和后端一体的系统，没有调用其他的 API 接口，目前还是在本地调试代码，进行设计，还没有将项目部署到云端，故没有设计外部接口供外部访问与调用。

目前我们考虑未来将我们代码上传到的服务器：按照本机 ip 为 127.0.0.1

服务器 IP 地址：10.103.239.100　端口号：38629

5.2、内部接口

前端通过相应代码，与 MySQL 数据库相连，实现用户账户注册，登陆判定，以及头像显示等操作



前端代码通过 app.py 与后端代码相连，传输上传图片的储存路径，后端代码根据储存路径实现图片的修复或动态处理

上传文件接口：@app.route('/upload', methods=['POST'])

显示图片接口：@app.route('/img/<string:name>')

获取上传文件接口：@app.route('/getupload/<string:name>')

提供面部修复和增强功能:@app.route('/gfpgan', methods=['GET'])

后端代码通过引入 MySQLdb 库的引入，实现对数据库的操作

```
# 使用 execute()  方法执行 SQL 语句
cursor.execute("SELECT VERSION()")
# 使用 fetchone() 方法获取单条数据.
data = cursor.fetchone()
```

Mysql 端口地址：数据库名 gan，端口 3306，user：root，域名是本地域名：127.0.0.1。