修复老照片系统

Repair old photo system

期末汇报

撰稿人：薛皓林 苏珈磊　汇报人：薛皓林 苏珈磊

汇报时间：2024年5月30日

目录 | CONTENTS

需求分析

Requirement Analysis

需求分析—选题背景

选题背景

照片图片是承载记忆和信息的一种很重要的方式，它们的重要性是不言而喻的，身处在如火如荼的互联网科技的新时代，我们有时会缅怀过去的记忆，过去的人物，但有些图片确是损坏模糊的。因此，功能完善,性能优越,安全可靠的照片修复系统受到越来越多人的青睐，同时也弘扬了我们的人文精神。

需求分析—目标内容

动态效果选择

动态效果随心选，表情丰富最生动

登录注册以及游客登录

无需注册烦恼多，账号有无任君游

动态修复效果

动态修复展魔力，图像流转韵律齐

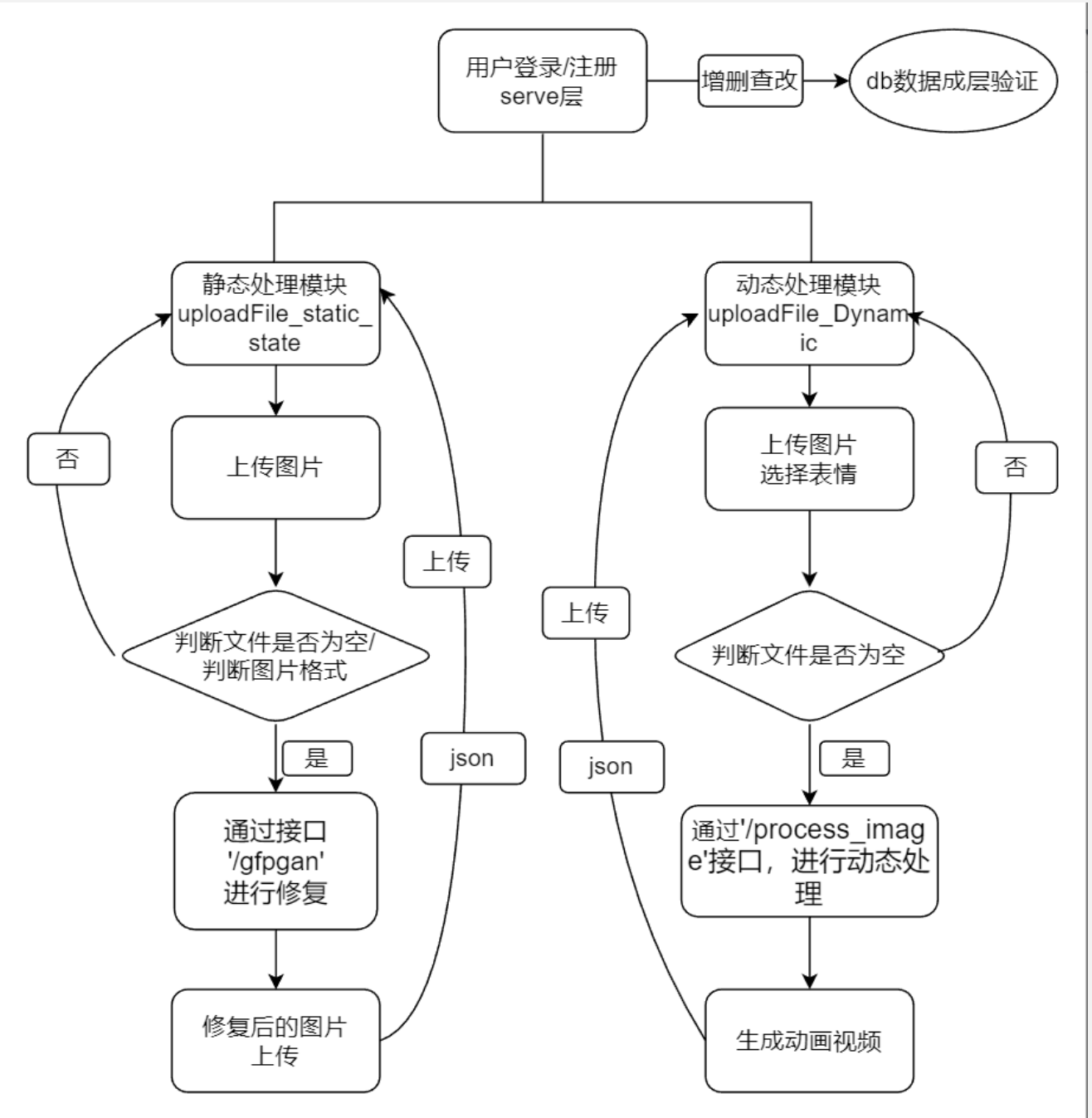
静态修复效果

静态修复显神通，旧照新生韵无穷

技术方案

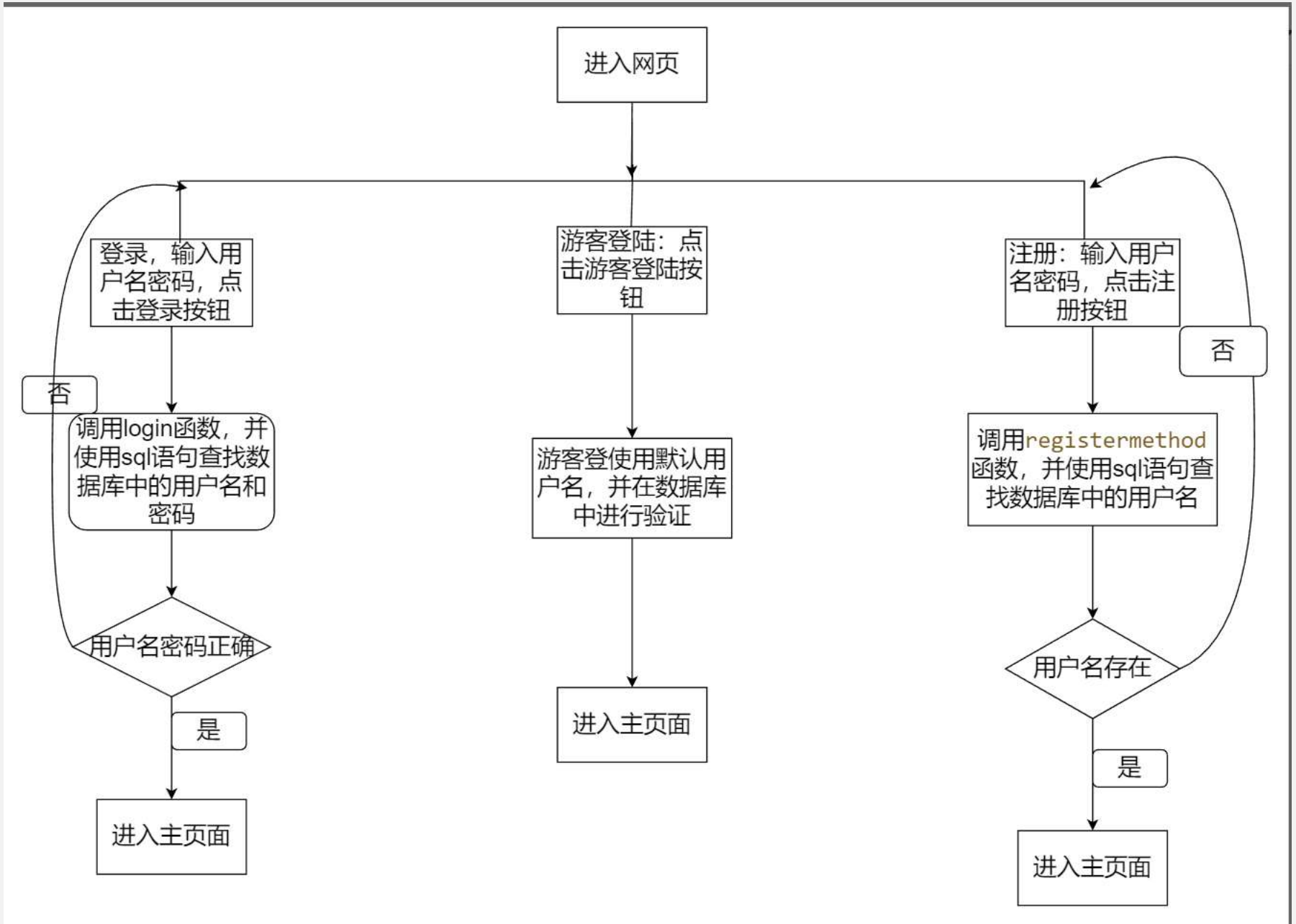
Technical Proposal

总体方案架构图


用户登录/注册
serve层
增删查改
db数据成层验证
静态处理模块
uploadFile_static_
state
动态处理模块
uploadFile_Dynam
ic
否
上传图片
上传图片
选择表情
否
上传
上传
判断文件是否为空/
判断图片格式
判断文件是否为空
是
是
json
json
通过接口
'/gfpgan'
进行修复
通过'/process_imag
e'接口，进行动态处
理
修复后的图片
上传
生成动画视频

技术方案—客户端

技术方案—数据库模块流程图



技术方案—数据库模块关键代码功能

```
exports.register = (req, res) => {
  const sql1 = 'SELECT * FROM suer WHERE username = ?;'
  const sql2 = 'insert into suer (username, password) value (?, ?)'
  db.query(sql1, [req.body.username], (err, result) => {
    console.log("result:",result)
    if(err) {
      return res.send({
        status: 500,
        message: "操作失败"
      })
    }
    if(result.length > 0) {
      return res.send({
        status: 202,
        message: '用户名已存在'
      })
    }else{
      // return res.send({
      //   status: 200,
      //   message: '注册成功'
      // })
      db.query(sql2, [req.body.username, req.body.password], (err, result) => {
        if(err) {
          return res.send({
            status: 400,
            message: "注册失败"
          })
        }
        res.send({
          status: 200,
          message: "注册成功"
        })
      })
    }
  })
}
```

实现登录(login)效果

- 通过查询数据库验证用户名和密码，返回登录成功或者登录失败的响应。

实现注册(register)效果

- 实现了用户注册的功能，首先检查用户名是否已经存在，若不存在则将用户信息插入数据库，并且返回注册成功或者失败的响应。

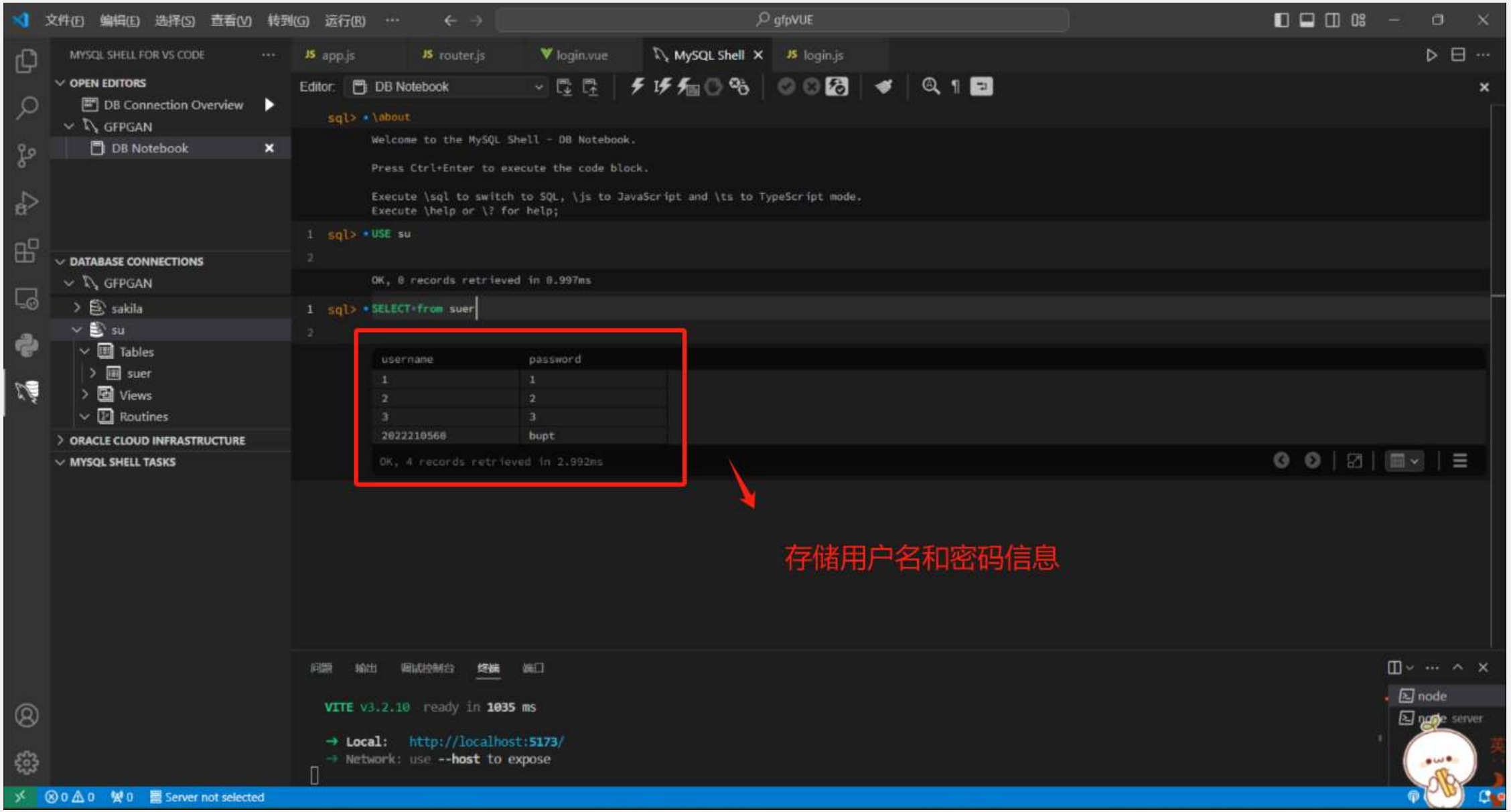
技术方案—数据库模块关键代码功能

```
227   async youke() {
228       try {
229           // 发送登录请求
230           axios.defaults.baseURL = "http://localhost:3000"
231           const response = await axios.post('/login', {
232           username: '1',
233           password: '1'
234           });
235           console.log("response.data.status",response.data.status)
236           // 根据响应状态处理结果
237           if (response.data.status === 200) {
238           alert('登录成功')
239           axios.defaults.baseURL = "http://localhost:5000"
240           const permiss = usePermissStore();
241           localStorage.setItem('ms_username', this.Loginform.username);
242           const keys = permiss.defaultList[this.Loginform.username];
243           permiss.handleSet(keys);
244           this.toPage();// 跳转到登录页面
245           } else if (response.data.status === 202) {
246               this.Loginform.password='',
247               this.Loginform.username=''
248           alert('用户名密码错误')
249           } else if (response.data.status === 400) {
250           this.$message.error(response.data.message || '登录失败，请检查输入信息');
251           } else {
252           this.$message.error('未知错误，请稍后再试');
253           }
254       } catch (error) {
255           // 处理错误
256       if (error.response) {
257       // 请求已发出，服务器也响应了状态码，但是状态码不在 2xx 范围内
258       console.error('服务器返回错误', error.response.data);
259       this.$message.error('注册失败，请稍后再试');
260       } else if (error.request) {
261       // 请求已发出，但是没有收到任何响应
```

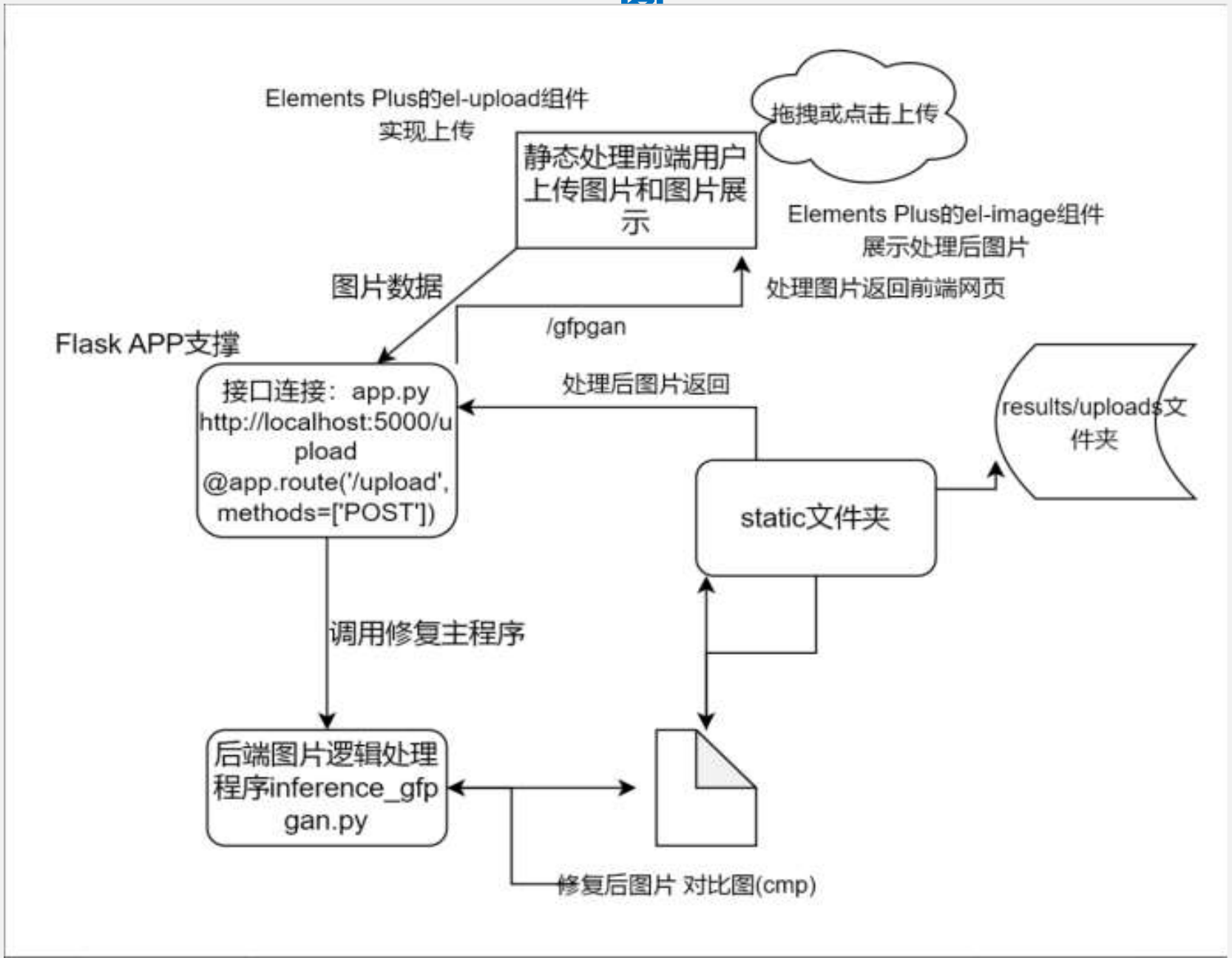
实现游客登录(youke)效果

- 通过发送硬编码的用户名和密码到服务器，根据响应状态提示登录结果，并且处理异常。

技术方案—数据库模块后端



技术方案—静态处理模块流程图



技术方案—静态生成模块


图像处理平台
图像处理
图像动态处理
图像修复处理
图片数据上传
建议上传的图片格式为JPG & PNG
将图片拖到此处，或 点击上传
检测结果

技术方案—静态模块

```python
@app.route('/upload', methods=['POST'])
def upload_pic():
    print("进行上传")
    if 'file' not in request.files:
        return jsonify({'error': '没有文件部分'}), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': '没有选择文件'}), 400
    if file:
```

```python
@app.route('/img/<string:name>', methods=['GET'])
def show_img(name):
    img_url = os.path.join(app.config['current_selectimage'], name)
    if name:
        image_data = open(img_url, "rb").read()
        response = make_response(image_data)
        response.headers['Content-Type'] = 'image/jpg'
        return response
```

```
e(file.filename).split('.')[1]
```

1.实现图片上传的接口，通过检查请求中的图片部分，对上传的文件进行安全重命名，并保存到指定目录，最后返回上传成功的消息和文件路径。
2.实现图像显示的接口，通过文件名从指定的路径读取图像数据，设置响应内容类型为JPG，并返回图像内容给客户端。

技术方案—静态模块

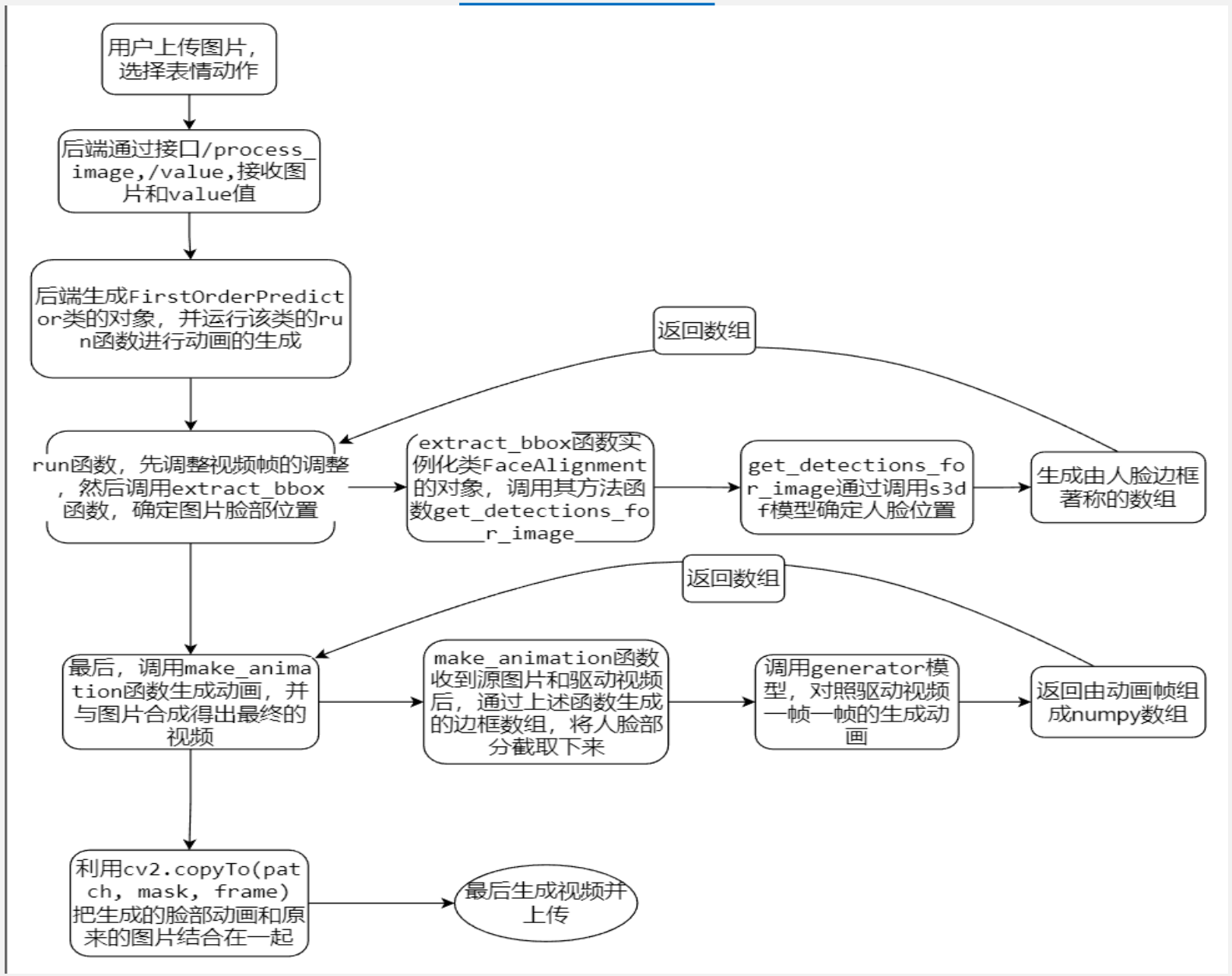
```python
@app.route('/getupload/<string:name>', methods=['GET'])
def get_upload(name):
    img_url = os.path.join(app.config['current_uploadimage'], name)
    if name:
        image_data = open(img_url, "rb").read()
        response = make_response(image_data)
        response.headers['Content-Type'] = 'image/jpg'
```

1.实现获取上传图像的接口，通过文件名读取指定目录中的图像数据并且以JPG格式返回给客户端。

```python
@app.route('/gfpgan', methods=['GET'])
def gfpgan():
    input_path = request.args.get('input')
    input_path_full = os.path.join(project_root_path, input_path)
    if not input_path_full:
        return jsonify({'error': 'Input path is required'}), 400

    getupload = f"http://localhost:5000/getupload/{os.path.basename(input_path)}"
    handle_res = image_handle({
        'input': input_path_full,
        'output': 'static/images/results',
    })
    result_image = f"http://127.0.0.1:5000/{handle_res[0]}"

    return jsonify({'message': 'Face restoration completed', 'input': getupload, 'output': result_image}), 200

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

2.实现图像修复的接口，通过接收输入图像的路径，调用函数，调用执行修复的主文件，返回处理后的图像URL和成功消息。

技术方案—动态生成模块流程图



技术方案—动态生成模块


图像处理平台
图像处理
图像动态处理
图像修复处理
图像动态处理
图片数据上传
建议上传的图片格式为JPG & PNG
将图片拖到此处，或 点击上传
微笑
眨眼
点头
摇头
提交

技术方案—动态生成模块

```python
@app.route('/value', methods=['POST'])
def value():
    data = request.json
    global value
    value = data.get('selectedValue')
    print("value:", value)
    return jsonify({'message': 'Received selected value: {}'.format(value)})
```

```python
@app.route('/process_image', methods=["GET",'POST'])
def process_image():
    print("Success!")
    global value
    if (value == 3):
        os.environ['DRIVING_VIDEO'] = r'D:\python\project-finally\GFPGAN\GFPGAN\applicati
    elif (value == 6):
        print(value)
        os.environ['DRIVING_VIDEO'] = r'D:\python\project-finally\GFPGAN\GFPGAN\applicati
    elif (value == 9):
        print(value)
        os.environ['DRIVING_VIDEO'] = r'D:\python\project-finally\GFPGAN\GFPGAN\applicati
    elif (value == 12):
        print(value)
        os.environ['DRIVING_VIDEO'] = r'D:\python\project-finally\GFPGAN\GFPGAN\applicati
    else:
        os.environ['DRIVING_VIDEO'] = r'D:\python\project-finally\GFPGAN\GFPGAN\applicati
```

`'/value'`用于接收用户选择的value值

`'/process_image'`用于接收用户上传的图片，根据value值选则驱动视频

技术方案—动态生成模块

```
    paddle.set_device('xpu')

predictor = FirstOrderPredictor(output=args.output,
                                filename=args.filename,
                                weight_path=args.weight_path,
                                config=args.config,


predictor.run(args.source_image, args.driving_video)
    # 读取 result.mp4 文件
    result_file_path = r'D:\python\project-finally\GFPGAN\GFPGAN\applicati
    with open(result_file_path, 'rb') as file:
        result_data = file.read()

    # 将文件内容转换为 base64 编码
    result_base64 = 'data:video/mp4;base64,' + base64.b64encode(result_da

    # 输出 base64 编码后的结果

    result_json = json.dumps({"result_base64": result_base64})

    return result_json
```

实例化类的对象，传递相应参数

调用类中run方法生成动态视频

将生成的视频以接送形式上传到前端

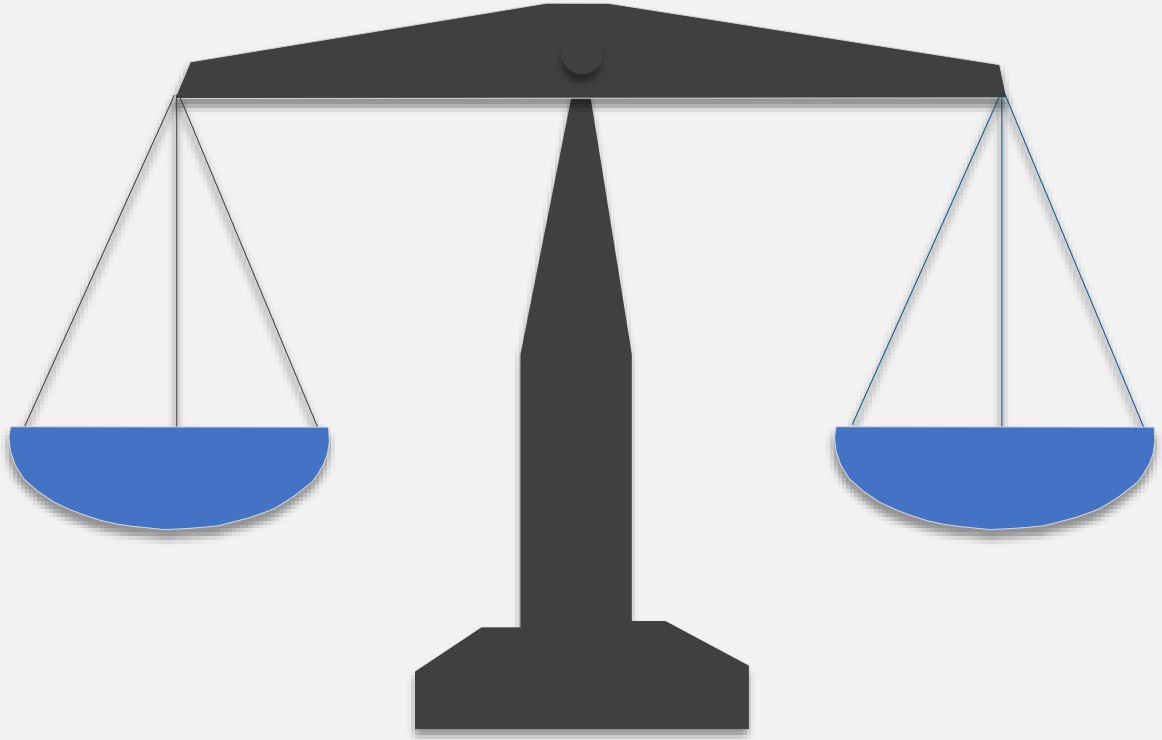
团队分工

Team Division

团队分工

苏珈磊

前端网页搭建
静态修复功能的实现
前后端接口连接



薛皓林

登陆注册模块的实现
动态功能的实现
数据库连接

亮点自评

Highlights Self-Assessment

亮点自评

3.巨人肩膀，团队增量

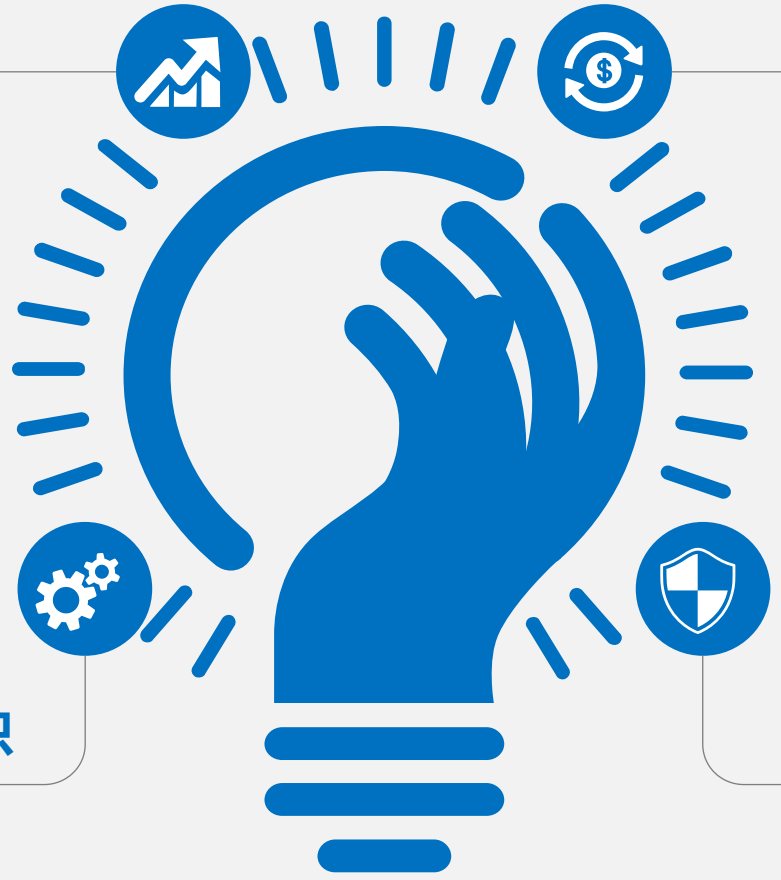
- 提高并发性能

1.与ChatGPT人机结队编程

- 借助大模型探索可行方案
- 帮助理解不解其意的代码

5.设计模式：分模块设计

- 提高代码可维护性
- 提高代码可扩展性



2.运用《项目管理》课程知识

- 及时有效的团队沟通
- 使用看板记录项目进度

4.面向用户编程

- 注重用户体验

亮点自评1—与ChatGPT结队编程

提供设计思路

用户注册登录选用什么数据库比较好

在选择用于用户注册和登录的数据库时，需要
下是一些常见的数据库选择及其特点：

1. **关系型数据库（RDBMS）**：
   - **MySQL**：这是一个广泛使用的开源
     接，并且有许多成熟的插件和工具可
   - **PostgreSQL**：PostgreSQL也是一个
     别适合需要复杂查询和高级功能的应
   - **SQL Server**：这是微软提供的一个
     Windows平台集成。
   - **Oracle Database**：Oracle是一个领
     性能、可靠性和安全性。
2. **NoSQL数据库**：
   - **MongoDB**：MongoDB是一个流行
     它非常适合存储和查询大量非结构化
   - **Redis**：Redis是一个开源的内存数据
     数据结构，如字符串、哈希、列表、

在选择数据库时，您需要考虑以下因素：

辅助理解代码

这些代码做了些什么，请详细分析这段代码

这段代码是一个使用PaddlePaddle框架的Python脚本，它实现
类，用于生成基于第一人称视角的视频动画。这个类主要利用了
（GAN）来实现视频帧之间的转换，使得源视频中的人脸可以
代码的详细分析：

1. **导入依赖**：代码开始部分导入了所需的库，包括操作系统
   cv2、数学库math、YAML和Pickle文件解析库、图像IO库
   库tqdm、科学计算工具库scipy，以及PaddlePaddle框架
2. **FirstOrderPredictor类**：这个类继承自BasePredictor，
3. **构造函数**：__init__方法初始化了FirstOrderPredictor
   模型的行为，例如输出目录、权重文件路径、配置文件、
   例、是否寻找最佳帧等。
4. **读取图像**：read_img方法用于读取图像文件，并将其转换
5. **运行模型**：run方法是类的主要入口点，它接受源图像和驱
   首先提取源图像和驱动视频中的人脸区域，然后使用这些
6. **加载检查点**：load_checkpoints方法用于加载预训练的模
7. **生成动画**：make_animation方法使用加载的模型和关键点
8. **寻找最佳帧**：find_best_frame_func是一个辅助函数，用
   的帧。
9. **提取边界框**：extract_bbox方法使用人脸检测器来提取图
10. **计算交并比**：IOU方法用于计算两个边界框的交并比，这有
    果集中。
11. **辅助功能**：代码中还包含了一些辅助功能，如normalize_
    get_prediction用于获取单个帧的预测结果。

整体来看，这段代码是一个完整的视频动画生成流程，它涉及
取、模型加载、动画生成和后处理等多个步骤。这个类可以用
务。

提示修改报错

login.vue:170
Uncaught (in promise) TypeError: Cannot read properties of undefined (reading 'error')
at Proxy.registermethod (login.vue:170:1)
registermethod @ login.vue:170
Promise.catch(异步)
handleClick @ use-button.ts:57
显示 6 更多框架

这个错误信息表示在 `login.vue` 文件的 `registermethod` 方法中，你尝试访问一个未定义对象的 `error` 属性，并且这个错误是在一个异步操作（Promise）中发生的。这意味着在尝试读取属性时，你期望的对象是 `undefined`。

这里有几个步骤可以帮助你解决这个问题：

1. **检查返回的对象**：

   确保你的异步操作（可能是使用 `axios` 或其他 HTTP 库发出的请求）确实返回了一个对象，并且这个对象在你尝试访问其 `error` 属性之前已经被定义。

2. **检查响应结构**：

   如果你使用的是 `axios`，通常错误处理不是在 `response.error` 中进行的，而是在 `catch` 块中处理 `reject` 的错误对象。确保你理解你正在使用的库的响应和错误处理机制。

3. **更新异步错误处理**：

   你需要确保在 `registermethod` 方法中正确处理异步操作可能抛出的错误。如果使用的是 `axios`，你的代码可

亮点自评1—与ChatGPT结队编程

ChatGPT在系统中贡献:

连接数据库(插入和查询)

```typescript
import db from '../db/index';

// 现在你可以在你的代码中使用 db 对象了
```

```
// 定义 login 函数
export const login = (req: Request, res: Response) => {
  const sql = 'SELECT * FROM user WHERE name = ? AND password = ?'
  db.query(sql, [req.query.name, req.query.password], (err, data) => {
    if (err) {
      return res.status(400).send({
        status: 400,
        message: "登录失败"
      });
    }

    if (data.length > 0) {
      res.status(200).send({
        status: 200,
        message: "登录成功"
      });
    } else {
      res.status(202).send({
        status: 202,
        message: '用户名或密码错误'
      });
    }
  });
};
```

设计value前后端的接口

```
<el-radio-group v-model="radio">
<el-radio :value="3">微笑</el-radio>
<el-radio :value="6">眨眼</el-radio>
<el-radio :value="9">点头</el-radio>
<el-radio :value="12">摇头</el-radio>
</el-radio-group>
怎么把选项的value值传给后端python文件
```

要将选项的值传递给后端的 Python 文件，您需要使用 Vue 来捕获选择的值，然后将其发送到后端。这可以通过发送 HTTP 请求来实现。通常使用 axios 或 fetch 库来执行此操作。以下是一个简单的示例代码，说明了如何实现这一点：

```vue
<template>
  <div>
    <el-radio-group v-model="selectedValue">
      <el-radio :label="3">微笑</el-radio>
      <el-radio :label="6">眨眼</el-radio>
      <el-radio :label="9">点头</el-radio>
      <el-radio :label="12">摇头</el-radio>
    </el-radio-group>

    <el-button @click="sendToBackend">发送到后端</el-button>
  </div>
</template>
```

```
      console.log(response.data); // 处理后端响应
    })
    .catch(error => {
      console.error(error); // 处理错误情况
    });
  }
};
</script>
```

登陆注册的实现(前端登陆)

例如，当用户名已存在时，您应该返回 409 Conflict 状态码，而不是 202 Accepted。

6. **代码组织**：您的代码嵌套较深，这可能会使得代码难以理解和维护。您可以通过使用 async/await 来简化异步操作，并使用 try/catch 来处理错误。

针对以上问题，下面是一个改进后的示例：

```javascript
exports.register = async (req, res) => {
  try {
    // 假设您已经设置了一个验证中间件来确保 req.body 包含用户名和密码
    const { username, password } = req.body;

    // 使用参数化查询来避免 SQL 注入
    const userExists = await db.query('SELECT * FROM user WHERE username = ?', [usern

    if (userExists.length > 0) {
      return res.status(409).json({ status: 409, message: '用户名已存在' });
    }

    // 在插入数据库之前，先对密码进行哈希处理
    const hashedPassword = await hashPassword(password); // 假设您有一个 hashPassword

    await db.query('INSERT INTO user (username, password) VALUES (?, ?)', [username,

    return res.status(201).json({ status: 201, message: '注册成功' });
  } catch (err) {
    // 统一的错误处理
    console.error('注册过程中发生错误:', err);
    return res.status(500).json({ status: 500, message: '注册失败' });
  }
};
```

请注意，您需要根据您的具体需求和环境调整上述代码。例如，您可能需要使用不同的数据库库或密码哈希库。

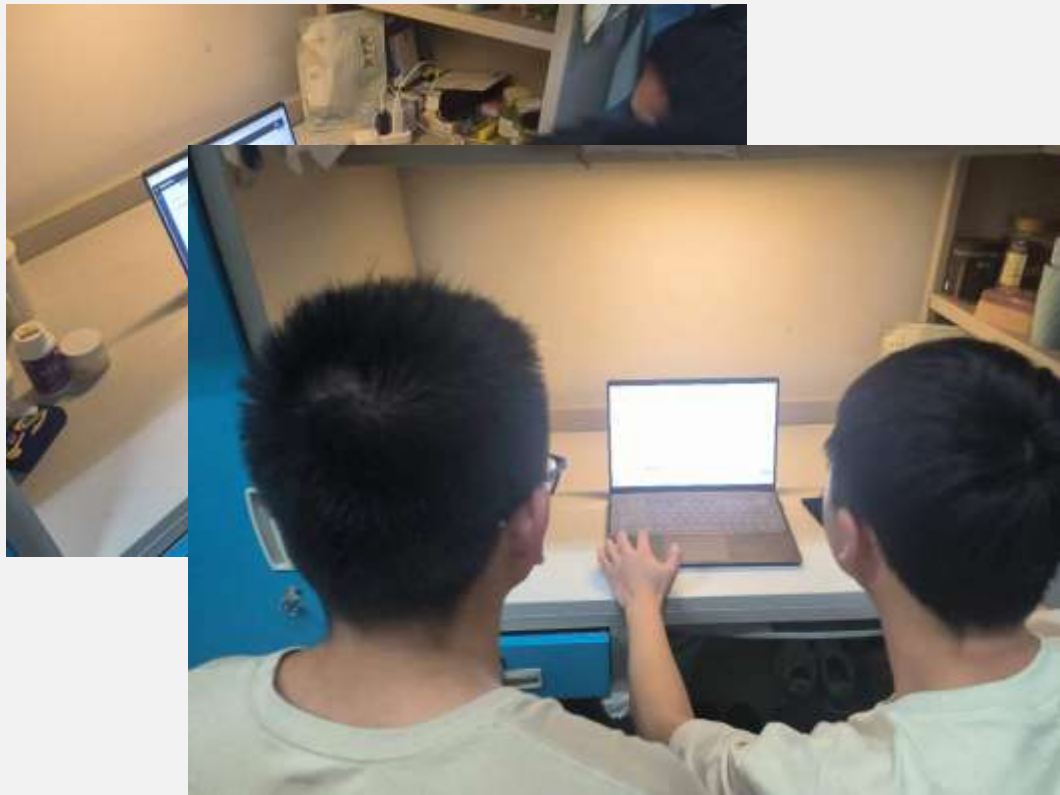
```
    // 请求设置错误，或发生了其他错误
    console.error('请求设置错误', error.message);
    this.$message.error('未知错误，请稍后再试');
  }
}
```

亮点自评2—运用项目管理知识

线上沟通

项目相关人员
紧密沟通协作

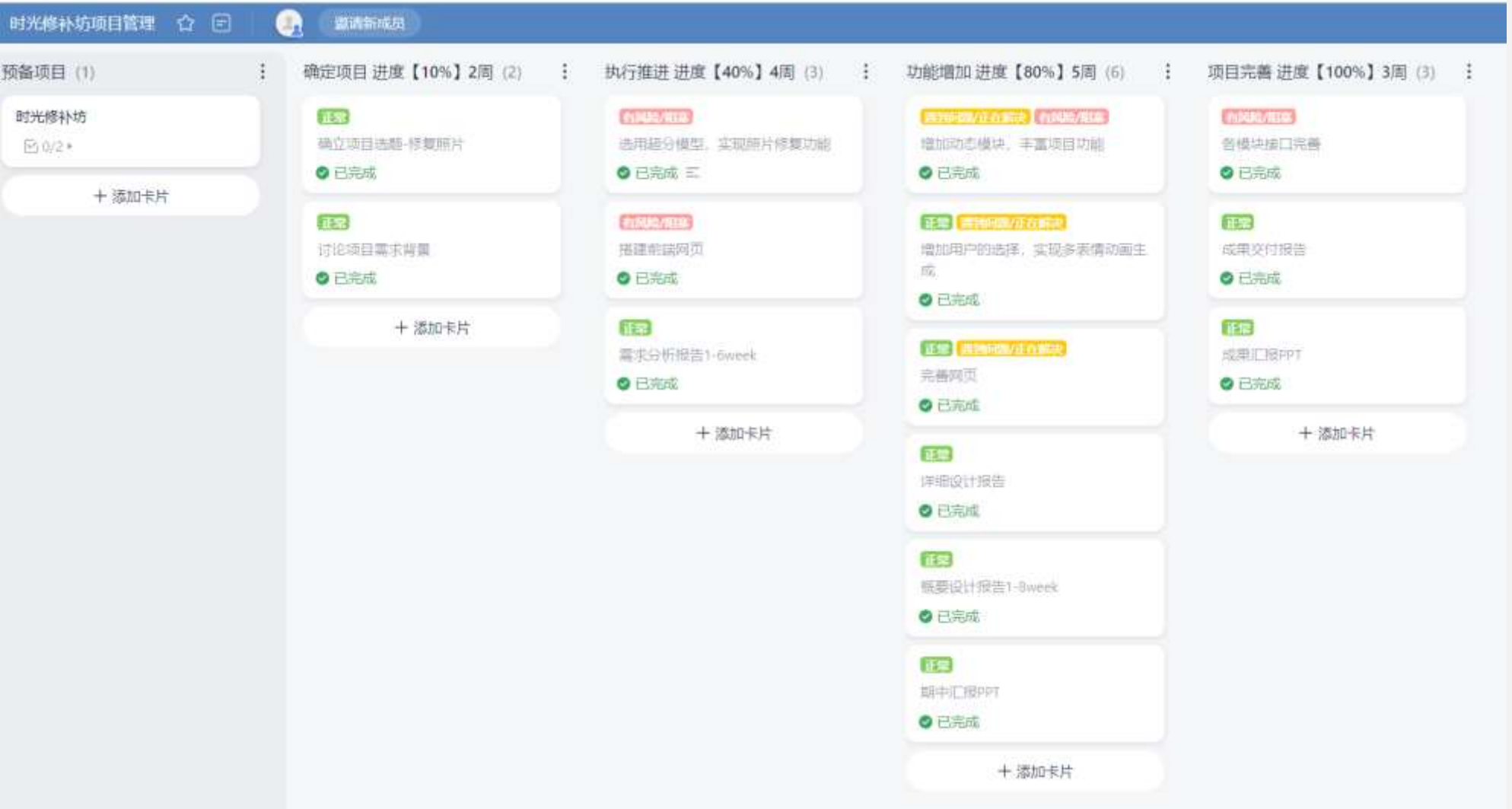
线下研讨



亮点自评2—运用项目管理知识

Teambition

使用项目看板
记录**项目进度**



制定项目计划
合理分工协作

亮点自评3—巨人肩膀，团队增量

巨人肩膀

GFPGAN是腾讯开源的人脸修复算法，它利用预先训练好的面部修复模型，并且封装了各种丰富多样的先验因素进行盲脸(blind face)修复，可以对照片进行很好的修复。我们利用由它提供的模型，搭建框架，自定义参数，实现照片的修复处理。

团队增量

- 添加动态人像修复的效果 ← 满足用户功能性需求
- 多人动态人像效果 ← 满足用户功能性需求
- 用户可选具体动态效果 ← 满足用户多样性,个人性需求

亮点自评4—面向用户编程

游客登录更方便，快速体验

界面清晰直观，不花里胡哨，上手简单。

亮点自评5—设计模式：各模块间解耦合，分模块设计

数据库server模块

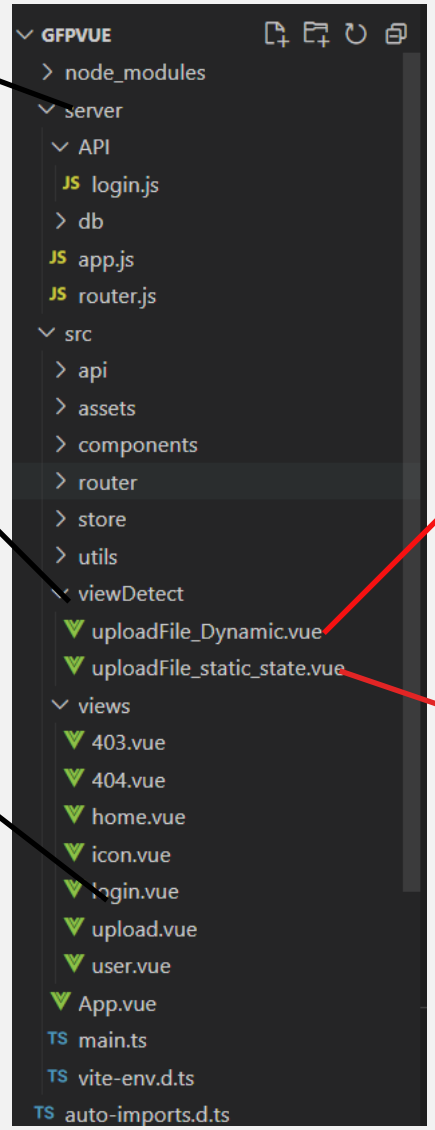
客户端与业务后端连接

viewDectect模块实现

客户端与数据库连接

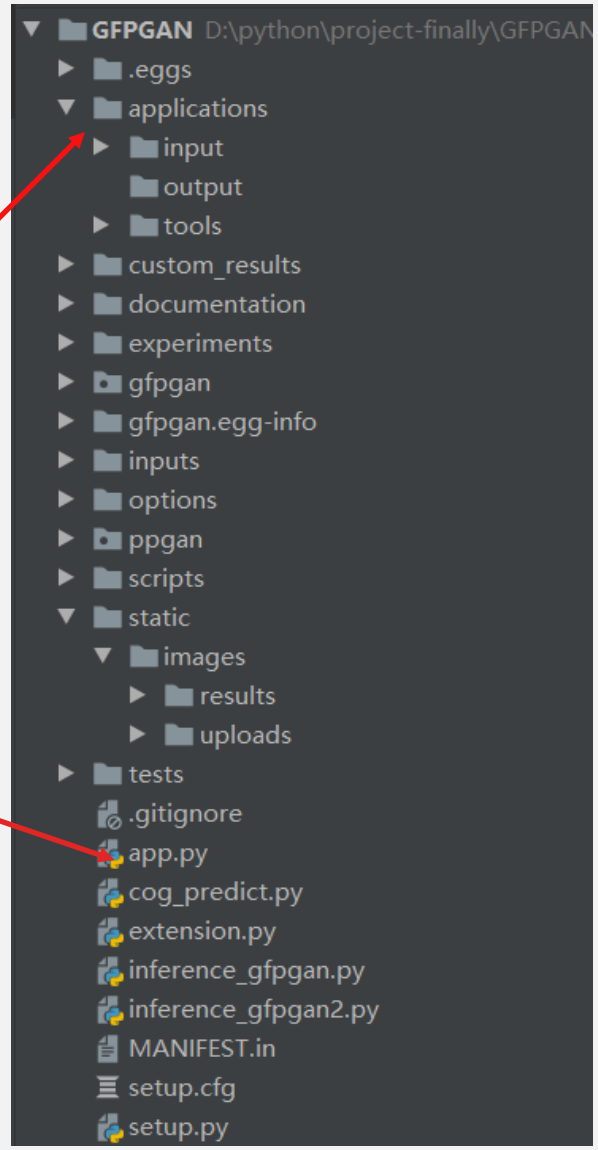
login模块

前后端项目目录



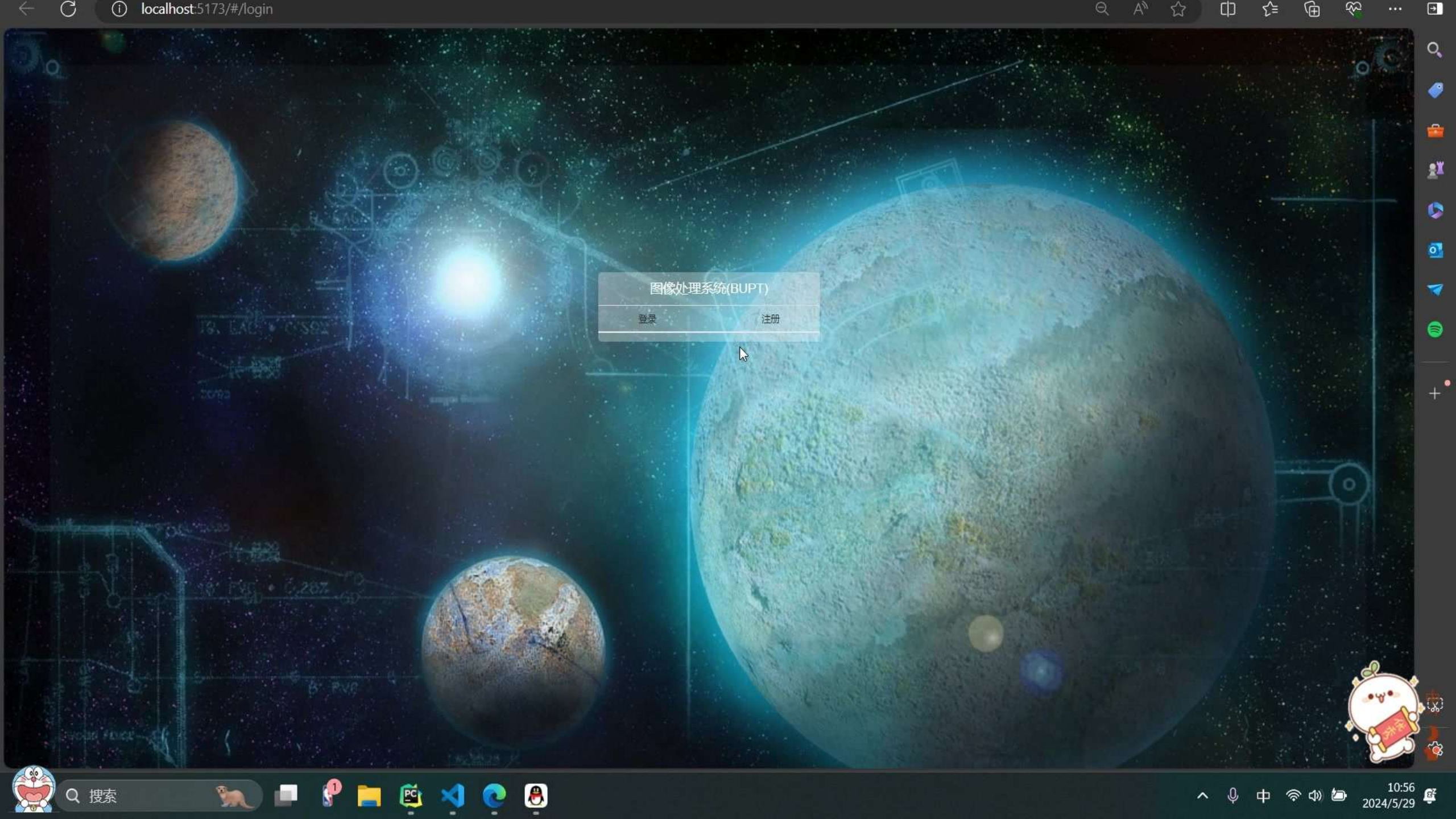
动态

静态



成果展示

Achievement Display


localhost:5173/#/login
图像处理系统(BUPT)
登录
注册
搜索
中
10:56
2024/5/29

感谢聆听

Appreciate For Your Listening

汇报完毕

撰稿人：苏珈磊 薛皓林　汇报人：苏珈磊 薛皓林

汇报时间：2024年5月30日