****

**义眼盯真 —— 图像篡改检测系统**

**项目测试文档**

目录

[1测试计划 3](#_Tocbw5qn5)

[1.1测试策略与目标 3](#_Toct13fn2)

[1.1.1测试策略 3](#_Toce0lp6q)

[1.1.2目标 5](#_Tocw7s6ub)

[1.2测试环境 5](#_Tocleug9e)

[2功能测试 7](#_Tocu47jjx)

[2.1图像篡改检测功能 7](#_Toc2xqh86)

[2.1.1测试用例与结果分析 7](#_Tocn1ng78)

[2.1.2测试分析与结论 8](#_Toc35u4xa)

[3性能测试 10](#_Toc3ohl76)

[3.1测试用例与结果分析 10](#_Tocycfhio)

[3.2测试分析与结论 12](#_Toc9w371b)

[4接口测试 13](#_Tocjouozy)

[4.1测试用例与执行结果 13](#_Tockkqg7z)

[4.2测试分析与结论 16](#_Tocfnah9l)

[5文档与介质一致性测试 18](#_Toc08vr6i)

[5.1安装说明一致性 18](#_Toc85jzd6)

[5.2使用说明一致性 18](#_Tocwmen8l)

[5.3测试结论 18](#_Toc74jxso)

[6自由操作测试 19](#_Toc5iksut)

[6.1易用性测试 19](#_Toccvdfsb)

[6.2兼容性测试 19](#_Tocet9538)

[6.3测试结论 19](#_Tochu8x1u)

# 测试计划

## 测试策略与目标

### 测试策略

针对 “义眼盯真 —— 图像篡改检测系统”，测试重点在于系统核心功能，如图像真实性检测、视频真实性分析、可视化解释的准确性与稳定性。测试难点在于深度学习模型对各类复杂伪造图像和视频的检测准确性，以及系统在不同设备和网络环境下的性能表现。采用分层测试策略，进行集成测试，验证各模块间的交互和协同工作是否正常；最后进行系统测试，对整个系统的功能、性能、兼容性等方面进行全面测试。

表1.1.1.1 测试策略表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试分层** | **测试对象** | **测试重点** | **测试难点** | **测试方法** |
| 数据层 | 数据存储 | 数据存储格式是否符合规范，数据存储的完整性和准确性，数据访问的效率和安全性 | 模拟大规模数据存储和高并发访问场景进行测试 | 静态测试、动态测试、黑盒测试 |
| 数据传输 | 数据传输的准确性、及时性和稳定性，传输过程中的数据加密和完整性校验 | 模拟不同网络环境（如低带宽、高延迟、不稳定网络）下的数据传输 | 动态测试 |
| 模型数据 | 模型数据的准确性和完整性，模型加载和更新的正确性 | 确保模型数据在不同环境下的兼容性和可移植性 | 静态测试、动态测试 |
| 逻辑层 | 图像检测功能 | 对各类图像（正常、伪造、复杂伪造）的检测准确性，检测结果的一致性和可靠性 | 构造具有代表性的复杂伪造图像进行测试，评估模型对新型伪造技术的检测能力 | 静态测试、动态测试、黑盒测试 |
| 视频分析功能 | 视频逐帧检测的准确性，视频整体分析结果的合理性，视频篡改痕迹的识别精度 | 处理长时长、高分辨率视频的性能问题，模拟不同类型的视频篡改场景 | 动态测试、黑盒测试 |
| 可视化解释 | 热图和标注的准确性和清晰度，可视化效果与检测结果的一致性 | 评估不同图像和视频内容下可视化效果的适用性 | 动态测试、黑盒测试 |
| 模型推理逻辑 | 深度学习模型推理过程的正确性和效率，模型参数的合理性 | 理解和验证复杂的深度学习模型算法和逻辑 | 静态测试、动态测试 |
| 业务逻辑处理 | 系统业务流程的正确性和完整性，不同功能模块之间的协同工作 | 梳理复杂的业务逻辑，设计全面的测试用例覆盖各种业务场景 | 静态测试、动态测试、黑盒测试 |
| 用户层 | 前端界面 | 界面布局的合理性和美观性，操作的便捷性和流畅性，界面响应的及时性 | 确保界面在不同设备和屏幕分辨率下的兼容性和显示效果 | 动态测试、黑盒测试 |
| 用户交互功能 | 图像和视频上传、检测结果查看、报告导出等交互操作的正确性和易用性 | 模拟不同用户操作习惯和错误输入情况进行测试 | 动态测试、黑盒测试 |
| 信息提示与反馈 | 系统提示信息的准确性和清晰性，错误提示和异常处理的友好性 | 设计全面的异常情况进行测试，确保信息提示能有效引导用户 | 动态测试、黑盒测试 |
| 设备兼容性 | 系统在不同类型设备（PC、手机、平板）和操作系统（Windows、Mac、iOS、Android）上的正常运行 | 缺乏足够的设备进行测试，需要使用模拟器和虚拟机进行部分替代 | 动态测试、黑盒测试 |
| 安全层 | 数据安全 | 数据在传输和存储过程中的加密和保护，防止数据泄露和篡改 | 模拟黑客攻击场景进行数据安全测试 | 动态测试、静态测试 |
| 系统安全漏洞 | 检测系统是否存在常见的安全漏洞（如 SQL 注入、XSS 攻击等） | 及时发现和修复新出现的安全漏洞 | 静态测试、动态测试、安全扫描工具辅助测试 |

### 目标

通过全面、系统的测试，明确项目发布时的质量等级。确保系统在功能上能够准确检测图像和视频的真实性，提供可靠的检测结果；在性能上满足不同用户场景的使用需求，具备高效的处理速度和良好的稳定性；在兼容性上能够在多种操作系统和设备上正常运行，为用户提供一致的使用体验。

## 测试环境

表1.3.1 测试环境表

|  |  |
| --- | --- |
| **类别** | **内容** |
| **软件环境** | Windows 11, Ubuntu 24.04.2 LTS, Android 14, iOS 17 |
| **硬件环境** | 手机：Redmi Note 12T Pro, Vivo X200 Pro |
| 平板：iPad Pro 2022 |
| 测试终端电脑：  CPU: intel i7-12700H  RAM: 32GB |
| **AI 服务器:**  CPU：Intel(R) Xeon(R) Platinum  8272CL CPU @ 2.60 GHz  RAM：128GB  GPU：GeForce RTX 4090D \* 2  **文件/业务服务器：**  CPU：Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz  RAM：8G |
| **网络环境** | 内部： 1Gbps  外部： 100Mbps |
| **测试工具** | 代码开发：PyCharm 2024.2.2, Visual Studio Code  接口测试：Postman |

# 功能测试

## 图像篡改检测功能

### 测试用例与结果分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | F001 | | | |
| 测试单元描述 | 测试图像篡改检测模块的拼接移动篡改检测功能 | | | |
| 用例目的 | 验证系统能正确检测通过传统图像编辑工具（如Photoshop）进行的拼接、移动、裁剪等篡改操作 | | | |
| 前提条件 | 1. 系统正常运行   1. 用户已上传待检测图片 2. 用户选择“拼接移动篡改检测”模式 | | | |
| 特殊的规程说明 | 测试图片需包含明显的拼接、移动或裁剪痕迹 | | | |
| 用例间的依赖关系 | 无 | | | |
| 具体步骤 | 输入 | 期望输出 | 实际输出 | 备注 |
| 1 | 点击上传按钮，上传一张经Photoshop拼接的JPEG图片 | 系统接收图片并返回“上传成功”提示 | 后端成功接受图片前端出现上传成功提示 | 图片尺寸：800x600 |
| 2 | 选择“拼接移动篡改检测”模式 | 系统切换至传统篡改检测模式，显示“检测中”状态 | 后端开始进行检测，前端出现“检测中”提示 |  |
| 3 | 服务器处理图片 | 服务器在5秒内完成分析，返回篡改区域坐标及置信度 | 服务器在4.2秒完成分析并返回了标记后的图片和置信度 | 超时阈值为10秒 |
| 4 | 前端可视化展示结果 | 图片中篡改区域被框标记，并显示“检测到拼接篡改”提示 | 原始图片和标记后图片及其置信度同时展示在界面上，出现提示 |  |

测试结果分析：

实际输出与期望输出一致，系统成功检测到拼接篡改区域，并在前端框选标记置信度达到95%。

服务器响应时间为4.2秒，满足性能要求。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | F002 | | | |
| 测试单元描述 | 测试图像篡改检测模块的 AI深度篡改检测 功能 | | | |
| 用例目的 | 验证系统能正确检测通过AI工具（如FaceApp）进行的换脸、物体消除等深度篡改操作 | | | |
| 前提条件 | 1. 系统正常运行  2. 用户已上传待检测图片  3. 用户选择“AI深度篡改检测”模式 | | | |
| 特殊的规程说明 | 测试图片需包含AI生成的换脸或物体消除痕迹 | | | |
| 用例间的依赖关系 | 无 | | | |
| 具体步骤 | 输入 | 期望输出 | 实际输出 | 备注 |
| 1 | 上传一张经FaceApp换脸的PNG图片 | 系统接收图片并返回“上传成功”提示 | 后端成功接受图片前端出现上传成功提示 | 图片尺寸：1024x768 |
| 2 | 选择“AI深度篡改检测”模式 | 系统切换至AI篡改检测模式，显示“检测中”状态 | 后端开始进行检测，前端出现“检测中”提示 |  |
| 3 | 服务器处理图片 | 服务器在8秒内完成分析，返回篡改区域坐标及置信度 | 服务器在7.1秒完成分析并返回了标记后的图片和置信度 | 超时阈值为15秒 |
| 4 | 前端可视化展示结果 | 图片中换脸区域被蓝色框标记，并显示“检测到AI深度篡改”提示 | 原始图片和标记后图片及其置信度同时展示在界面上，出现提示 |  |

测试结果分析:

实际输出与期望输出一致，系统正确识别换脸区域，但置信度为86%（略低于拼接篡改模式置信度）。

服务器响应时间为7.1秒，耗时高于拼接篡改模式，且接近期望输出时间上限。

### 测试分析与结论

1.综合分析：

传统篡改检测：功能稳定，准确率高（95%），但对小范围裁剪的敏感度不足。

AI深度篡改检测：对换脸检测效果良好，但对复杂背景下的物体消除存在漏报（测试漏报率10%）。

性能问题：AI检测模式的平均响应时间较高（6-8秒），需优化模型推理速度。

2.改进建议：

增加对低分辨率图片的适配性，提升小区域篡改的检测能力。

优化AI模型的轻量化设计，降低计算资源消耗。

引入多模态检测（如元数据分析）以提高AI篡改检测的置信度。

## 视频篡改检测功能

### 测试用例与结果分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | F003 | | | |
| 测试单元描述 | 测试视频篡改检测模块的AI篡改检测功能 | | | |
| 用例目的 | 验证系统能正确检测通过AI换脸工具（如Deepfake）进行的视频篡改操作 | | | |
| 前提条件 | 1. 系统正常运行   1. 用户已准备待检测视频 | | | |
| 特殊的规程说明 | 测试视频时长需少于5分钟，大小在1GB以内 | | | |
| 用例间的依赖关系 | 无 | | | |
| 具体步骤 | 输入 | 期望输出 | 实际输出 | 备注 |
| 1 | 点击上传按钮，上传一段经Deepfake换脸的MP4视频 | 系统接收视频并进行预处理，之后并返回“上传成功”提示 | 后端成功接受视频，前端出现上传成功提示 |  |
| 2 | 点击“开始检测”按钮 | 系统开始检测视频，显示“检测中”状态 | 后端开始进行检测，前端出现“检测中”提示 |  |
| 3 | 服务器处理视频 | 服务器在200秒内完成分析，返回处理后的视频 | 服务器在49秒完成分析并返回了标记后的视频 | 超时阈值为240秒 |
| 4 | 前端可视化展示结果 | 视频中篡改区域被框标记 | 原始视频和标记后视频同时展示在界面上，出现视频篡改区域标记 |  |

测试结果分析：

实际输出与期望输出基本一致，系统检测到视频中约50%的AI换脸区域，并在前端框选标记置信度达到80%。

服务器响应时间为49秒，满足性能要求。

### 测试分析与结论

1.综合分析：

AI视频篡改检测：能够检测到视频中部分AI换脸的篡改帧，但检测准确率不高（50%)。

性能问题：检测的平均响应时间较高（1分钟视频40-60秒），需优化模型推理速度。

2.改进建议：

继续优化模型，选择合适数据集进行进一步训练，提高模型检测性能

开发视频连续变化检测算法，用于取代原来的抽帧检测方法

# 性能测试

## 测试用例与结果分析

预期性能指标测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | P001 | | | |
| 性能描述 | 移动端CPU利用率 | | | |
| 用例目的 | 验证移动端模型运行时CPU占用 | | | |
| 前提条件 | 测试机网络稳定，后台无其他耗时进程 | | | |
| 特殊的规程说明 | 测试前清理缓存 | | | |
| 用例间的依赖关系 | 无 | | | |
| 具体步骤 | 输入/动作 | 期望的性能  （平均值） | 实际的性能  （平均值） | 备注 |
| 1 | 登录网页 | - | - |  |
| 2 | 连续执行50次“图片监测”功能 | CPU≤30% | 22.3% |  |
| 3 | 测试记录cpu利用率平均值 | ≤30% | 22.3% | 通过测试 |
| … | … | … | … | … |

测试结果分析：

实测均值在 22.3%，较阈值 30% 有 7.7 个百分点的富余空间，峰值为28.7%，也没有超过阈值，模型在测试机型上运行表现良好。有较大冗余空间。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | P002 | | | |
| 性能描述 | 云端高并发 CPU 利用率 | | | |
| 用例目的 | 验证云端服务在 500 用户并发时， CPU 占用是否≤70% | | | |
| 前提条件 | 1. 部署单节点服务  2. 模拟并发用户脚本准备就绪 | | | |
| 特殊的规程说明 | 并发脚本使用 JMeter  测试前确保无其他干扰进程 | | | |
| 用例间的依赖关系 | 无 | | | |
| 具体步骤 | 输入/动作 | 期望的性能  （平均值） | 实际的性能  （平均值） | 备注 |
| 1 | 启动 JMeter，模拟 500 个并发图像检测请求 | - | - |  |
| 2 | 持续发送请求 1 分钟 | CPU 利用率≤70% | CPU利用率68.3% | 峰值超过了70%到达72% |
| 3 | 监控并记录单节点 CPU 利用率 | ≤70% |  | 通过 |
| … | … | … | … | … |

测试结果分析：

实测均值在 68.3%，较阈值 70% 有 1.7 个百分点的富余空间，峰值为72%，略微超过阈值，云端处理处理500并发请求能力达标

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | P003 | | | |
| 性能描述 | 并发在线用户数 | | | |
| 用例目的 | 验证系统支持 ≥500 用户在线并发 | | | |
| 前提条件 | WebSocket 模拟工具准备就绪 | | | |
| 特殊的规程说明 | 连接保持至少 30 秒 | | | |
| 用例间的依赖关系 | I002 | | | |
| 具体步骤 | 输入/动作 | 期望的性能  （平均值） | 实际的性能  （平均值） | 备注 |
| 1 | 使用 LoadRunner 模拟 500 个 WebSocket 连接 | - | - |  |
| 2 | 每个连接发送心跳和检测指令 | ≥500 连接成功；队列并发线程 ≥50 | 500 连接成功；队列并发线程 = 52 |  |
| 3 | 记录连接成功率、响应时延 | ≥99% 成功率；时延 ≤200ms | 成功率 100%；平均时延 183ms | 通过测试 |
| … | … | … | … | … |

测试结果分析：

完成 500 个 WebSocket 连接，连接成功率100%，平均时延 180ms，满足 ≥500 连接和 ≤200ms 延时要求。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | P004 | | | |
| 性能描述 | 单图检测时延 | | | |
| 用例目的 | 验证单图检测响应时长是否≤3秒 | | | |
| 前提条件 | 同用例001 环境；网络抖动±10ms | | | |
| 特殊的规程说明 | 上传图片大小固定为 200KB | | | |
| 用例间的依赖关系 | I001 | | | |
| 具体步骤 | 输入/动作 | 期望的性能  （平均值） | 实际的性能  （平均值） | 备注 |
| 1 | 登录网页 并上传 200KB 测试图 | - | - |  |
| 2 | 触发“单图检测” | 平均时延 ≤3秒 | 平均时延 2.8秒 |  |
| 3 | 记录多次检测的平均响应时长 | ≤3秒 | 平均时延 2.8秒 | 通过测试 |
| … | … | … | … | … |

测试结果分析：

平均响应时长 2.8 秒，优于 3 秒阈值，用户体验流畅。

## 测试分析与结论

本次测试通过，总体表现中，所有项均达到或优于预期指标，系统在移动端与云端的资源利用、并发承载和时延响应上都表现出色。

优化建议：

网络抖动容错：在客户端增加“重试”与“降级”逻辑（例如网络超时后自动重试一次），提升弱网环境下的成功率。

持续监控：结合 Prometheus + Alertmanager，对关键指标（CPU、时延、连接数）配置实时告警，确保异常时能快速定位并扩容或降级处理。

多机型覆盖：移动端测试建议扩展到低端机型（如 1GB RAM）进行回归，验证在资源更紧张环境中的表现。

# 接口测试

## 测试用例与执行结果

接口功能测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | I001 | | | |
| 接口描述 | 数据库crud接口 | | | |
| 用例目的 | 测试数据库crud接口是否正常使用 | | | |
| 前提条件 | 代码运行无出错 | | | |
| 特殊的规程说明 | 无 | | | |
| 用例间的依赖关系 | 无直接依赖 | | | |
| 具体步骤 | 输入/动作 | 期望的输出  （平均值） | 实际的输出  （平均值） | 备注 |
| 1 | 对数据库写入数据 | 状态码200  数据库服务器接受数据 | 状态码200  数据库服务器接受数据 | 正常 |
| 2 | 查看数据库是否保存了数据 | 数据库保存了数据 | 数据库保存了数据 | 正常 |
| 3 | 验证数据一致性 | 数据与写入文件内容一致 | 数据与写入文件内容一致 | 通过 |
| 4 | 对数据库删除 | 数据库删除正常 | 数据库删除正常 | 正常 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | I002 | | | |
| 接口描述 | 图像上传接口 /api/upload | | | |
| 用例目的 | 验证图像上传接口在不同文件大小下的响应性能与正确返回值 | | | |
| 前提条件 | 系统后端已启动，网络连接正常，前端正常可访问 | | | |
| 特殊的规程说明 | 使用Postman或自动化脚本进行文件上传，模拟不同图像大小 | | | |
| 用例间的依赖关系 | 无直接依赖 | | | |
| 具体步骤 | 输入/动作 | 期望的输出  （平均值） | 实际的输出  （平均值） | 备注 |
| 1 | 上传50KB图像 | 状态码200，耗时<300ms | 状态码200，耗时210ms | 正常 |
| 2 | 上传2MB图像数据 | 状态码200，耗时<800ms | 状态码200，耗时732ms | 正常 |
| 3 | 上传超过5MB图像 | 状态码400（超出大小限制） | 状态码400 | 正常处理错误 |
| 4 | 上传空文件 | 状态码400（非法输入） | 状态码400 | 错误处理正确 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | I003 | | | |
| 接口描述 | 篡改检测接口 /api/detect | | | |
| 用例目的 | 验证YOLO检测接口在不同图像下的响应性能与输出格式正确性 | | | |
| 前提条件 | 已上传图像并获得图像ID | | | |
| 特殊的规程说明 | 使用模拟图像调用检测接口 | | | |
| 用例间的依赖关系 | 依赖上传接口（I002）执行成功返回图像ID | | | |
| 具体步骤 | 输入/动作 | 期望的输出  （平均值） | 实际的输出  （平均值） | 备注 |
| 1 | 传入图像ID进行检测 | 返回JSON标注结果，耗时<1s | 返回JSON，耗时850ms | YOLO模型检测成功 |
| 2 | 传入非法ID | 状态码404，错误提示 | 状态码404 | 错误提示准确 |
| 3 | 并发调用10次 | 响应全部成功，平均耗时<1.2s | 响应正常，平均耗时1.05s | 服务器承压正常 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | I004 | | | |
| 接口描述 | 分析报告生成 /api/report | | | |
| 用例目的 | 验证根据检测结果生成报告的准确性与稳定性 | | | |
| 前提条件 | 已完成检测，具备检测结果JSON数据 | | | |
| 特殊的规程说明 | 检测数据传入后自动调用分析模块 | | | |
| 用例间的依赖关系 | 依赖I003执行完毕获得检测数据 | | | |
| 具体步骤 | 输入/动作 | 期望的输出  （平均值） | 实际的输出  （平均值） | 备注 |
| 1 | 提交标准检测JSON | 返回PDF/HTML报告 | 成功生成PDF报告 | 格式规范，内容完整 |
| 2 | 提交空JSON | 状态码400，提示缺失数据 | 状态码400 | 报错明确 |

1.数据库CRUD接口（I001）：

功能验证：数据库写入、查询、验证和删除都成功完成，状态码均为200，说明基本的读写和数据一致性得到了保障。

数据一致性：验证步骤确认，数据存入文件内容与数据库中的内容保持一致，保证了数据的准确性。

潜在问题：没有明显缺陷，但应注意在高并发场景下，可能存在短暂的响应延迟或锁竞争，建议后续增加压力测试。

2.图像上传接口（I002）：

性能表现：

50KB 文件上传耗时210ms，远优于预期<300ms，表明系统上传响应性能良好。2MB 文件上传耗时732ms，同样满足<800ms的性能要求，表现出稳定的上传速度。

错误处理：

超出5MB尺寸限制返回400，响应正确，说明文件大小限制机制正常。

空文件上传时返回400，提示系统能正确识别非法输入。

潜在问题：

从测试结果看，超出大小限制和空文件的错误响应较为正确，但还可以进一步优化用户体验，比如返回详细错误信息。

大文件上传时，耗时明显增加，建议关注网络环境下的极端情况，未来可考虑实现分片上传以提升体验。

3.篡改检测接口（I003）：

正常检测：在获得图像ID后，检测接口对于不同大小的图像，都能快速返回检测结果，响应时间符合预期，表现良好。

错误处理：超出限制和非法输入也被系统正确捕获，返回400，无异常。

依赖关系：依赖上传接口（I002）成功，说明接口的前置条件明确，流程逻辑合理。

4.分析报告生成接口（I004）

功能验证：成功提交标准检测JSON数据后，系统能自动生成符合规范的PDF/HTML 报告，且内容完整、格式规范，验证报告的正确性和稳定性。在提交空JSON时，接口能正确返回状态码400，并提示缺失必要数据，表明错误处理机制健全。

性能表现：在正常检测数据情况下，报告生成速度令人满意，生成时间符合预期，满足实际应用需求。报告格式规范，内容完整，满足用户对报告质量的基本要求。

依赖关系：依赖上传接口（I003）成功，说明接口的前置条件明确，流程逻辑合理。

## 测试分析与结论

通过此次系列功能性测试，系统在数据库操作、图像上传和篡改检测等关键接口上表现出较强的稳定性和可靠性，整体功能满足预期。

各接口均能按预期完成对应的操作，数据库CRUD操作顺利，数据一致性得到验证。图像上传接口在正常文件大小下响应迅速且符合性能指标，错误处理机制也得到了验证，超出限制或非法文件能被正确拦截。篡改检测接口能够在成功上传图像ID基础上，快速返回检测结果，且能正确处理异常输入。测试中的响应时间都优于设定的阈值，显示出系统在常规负载下具有良好的响应能力。大文件上传过程中，响应时间逐渐增加，但仍处于合理范围内，说明系统对于大数据量的处理能力较强。

测试中存在的不足与风险包括以下部分：性能测试仅覆盖较为理想的环境，未来需要进一步进行并发和压力测试，确保系统在高负载、多用户同时操作下仍能保持稳定。错误信息提示相对简洁，建议增强用户体验，提供更详细的错误原因说明，如“文件超出最大尺寸限制”或“非法文件格式”。高风险点在于依赖系统外部（如数据库或文件存储），如网络不稳定或存储故障可能引发数据丢失或接口响应异常，应加强监控与容错设计。

此次测试已通过，表明系统核心功能表现良好，满足基本业务需求。系统在性能、稳定性和安全性方面已展示出一定的成熟度，但仍需持续优化，特别是面对高并发、极端条件时的表现。此外，应结合用户实际应用场景，持续完善异常处理和用户提示机制，以提升整体用户体验。具体优化部分包括以下部分：

部分接口在高并发场景下响应时间略有波动，建议进行压力测试以优化性能。

上传超过限制文件大小时，仅返回400错误，可考虑加入更友好的提示信息或自动压缩机制。

数据库连接偶尔会出现延迟，建议优化数据库配置或增加连接池。

# 文档与介质一致性测试

## 安装说明一致性

Web 端安装与使用：文档说明 Web 端无需安装，通过主流浏览器访问服务器提供的网址即可进入系统。实际操作中，在 Chrome 浏览器地址栏输入服务器网址后，顺利进入系统，与文档描述一致。

服务器端硬件环境准备：文档推荐使用 Intel Xeon 处理器、16GB 以上内存、大容量硬盘，涉及模型训练需安装 NVIDIA GPU。实际准备的服务器符合上述配置要求，硬件环境准备与文档一致。

服务器端软件环境准备操作系统选择：文档提到服务器操作系统可选 Linux 或 Windows Server。本次测试选用 Windows Server 2019，安装过程顺利，与文档描述相符。

后端依赖安装：按照文档使用 Anaconda 进行虚拟化隔离，通过 pip install -r requirements.txt 命令安装 PyTorch、OpenCV 等依赖包，安装过程中无报错，依赖包安装成功，与文档一致。

数据库设置：使用 MySQL 8.0，执行 image\_db.sql 文件创建数据库和数据表结构，配置数据库连接信息后，后端服务能正确访问数据库，与文档说明一致。

网络环境搭建：文档要求服务器网络稳定、带宽充足，移动端用户使用 4G 及以上移动网络或稳定 Wi-Fi。测试中服务器网络稳定，移动端使用 Wi-Fi 连接系统进行操作，数据传输正常，与文档相符。

## 使用说明一致性

图像检测：进入 “图像检测” 页面，可选择 “拼接篡改检测” 或 “AI 深度篡改检测”，点击 “上传图片” 按钮选择本地图片，格式不支持时会提示，上传成功后系统自动检测并生成包含多种信息的报告，支持 PDF、HTML 格式导出，实际操作与文档描述完全一致。

视频检测：在 “视频检测” 页面选择并上传 MP4 格式视频，上传过程有进度条显示，点击 “开始检测” 后检测过程有动态进度展示，检测完成生成标注篡改部分的视频并展示在页面右方，与文档描述一致。

系统交互：系统界面简洁，检测时进度展示区和结果展示区功能正常，“AI 分析助手” 可与 OpenAI 大语言模型交互，支持自然语言、批量提问和多轮对话，提问和回答加密处理，实际体验与文档描述相符。

## **测试结论**

经测试，“义眼盯真 —— 图像篡改检测系统” 的用户文档说明书与系统实际安装、使用情况一致，用户可依据文档顺利完成系统的安装、部署和使用操作，文档能有效指导用户使用该系统。

# 自由操作测试

## 易用性测试

系统界面设计简洁直观，操作流程较为清晰，新用户经过简单摸索即可上手。但在移动端设备上，部分按钮（如 “报告设置”“导出报告”）尺寸较小，点击操作不太方便，容易误触。

系统在检测过程中，进度展示区能够实时反馈检测进度，但进度条的变化有时不够平滑，出现跳跃式增长，给用户造成检测进度不稳定的感觉。

“AI 分析助手” 功能较为实用，支持自然语言交互、批量提问和多轮对话。但在回答一些复杂专业问题时，回答内容不够准确和深入，无法满足专业用户的需求。

## 兼容性测试

在不同桌面操作系统和浏览器上，系统功能和界面显示均正常，未出现兼容性问题。但在移动端的某些小众浏览器（如三星浏览器旧版本）上，系统部分页面布局混乱，部分功能按钮无法正常点击。

测试过程中，在检测任务进行时切换网络（如从 Wi-Fi 切换到移动数据），系统能够自动尝试重新连接，但部分检测任务出现中断情况，需要用户重新上传文件进行检测。

## **测试结论**

本次自由操作测试通过，“义眼盯真 —— 图像篡改检测系统” 在基本功能实现上较为可靠，但仍存在一些问题需要改进。易用性方面，移动端按钮尺寸和进度条显示有待优化，“AI 分析助手” 回答准确性需提高；兼容性方面，需进一步优化对小众浏览器的支持以及网络切换时的稳定性。建议开发团队针对这些问题进行修复和优化，以提升系统的整体质量和用户体验。