**义眼盯真 —— 图像篡改检测系统**

软件设计文档

成员：刘信方，王劲东，张芳博，袁崇昕，李康

日期：2025/04/09

目录

[1. 引言 4](#_Toc197500410)

[1.1. 编写目的 4](#_Toc197500411)

[1.2. 项目范围 4](#_Toc197500412)

[1.3. 文档概述 4](#_Toc197500413)

[1.4. 参考文献 5](#_Toc197500414)

[1.5. 术语表 5](#_Toc197500415)

[2. 系统概述 8](#_Toc197500416)

[2.1. 系统功能 8](#_Toc197500417)

[2.2. 系统架构 8](#_Toc197500418)

[3. 系统架构 9](#_Toc197500419)

[3.1. 架构设计 9](#_Toc197500420)

[3.2. 分解描述 10](#_Toc197500421)

[3.3. 接口描述 12](#_Toc197500422)

[3.4. 设计依据 13](#_Toc197500423)

[4. 数据设计 14](#_Toc197500424)

[4.1. 数据描述 14](#_Toc197500425)

[4.2. 数据字典 14](#_Toc197500426)

[5. 组件设计 17](#_Toc197500427)

[5.1. 前端交互 17](#_Toc197500428)

[5.2. 后端处理 19](#_Toc197500429)

[5.3. 检测模型 21](#_Toc197500430)

[5.4. 数据存储与管理 23](#_Toc197500431)

[5.5. 性能设计 24](#_Toc197500432)

[5.6. 部署环境 24](#_Toc197500433)

[6. 人机界面设计 25](#_Toc197500434)

[6.1. 用户界面概述 25](#_Toc197500435)

[6.2. 屏幕图像 26](#_Toc197500436)

[6.3. 屏幕对象和操作 27](#_Toc197500437)

[7. 需求矩阵 29](#_Toc197500438)

# 引言

## 编写目的

本软件设计文档旨在为开发团队提供全面且详细的技术蓝图，明确系统架构、模块设计、数据结构以及接口规范等关键要素，为代码开发提供直接依据，确保开发过程的准确性与高效性；同时，为项目后续的测试、维护以及升级工作提供坚实的参考基础，助力项目顺利推进并达成预期目标。

## 项目范围

“义眼盯真 ——AI 图像真实性检测系统” 借助深度学习技术，专注于图像真实性检测。功能涵盖图像和视频的篡改检测、可视化解释、报告生成、集成式 API 服务以及微信小程序服务等，支持单张和批量上传检测。应用于新闻媒体审查、司法取证、社交平台审核以及教育与公众宣传等多个领域，为各行业和公众提供图像真实性检测服务。系统采用 B - S 架构与前后端分离模式，前端基于 HTML5、CSS3、JavaScript 构建，后端选用 Python 的 Flask 框架，集成 PyTorch 深度学习框架，使用 MySQL 数据库存储数据，云存储存放图像和视频文件 。

## 文档概述

本文档首先介绍项目背景、目的及范围，使读者对系统有初步认识。接着深入阐述系统架构设计，包括整体架构、各模块功能及相互关系。随后详细说明数据设计，涵盖数据描述和数据字典。组件设计部分对各功能组件进行拆解分析，明确其内部逻辑。人机界面设计聚焦用户交互层面，提升用户体验。需求矩阵展示组件、数据结构与需求的对应关系，便于跟踪和管理。最后附录部分提供补充信息，增强文档的完整性和可参考性。

## 参考文献

[1] 《义眼盯真 —— 图像篡改检测系统软件需求规格说明书》

[2] 《义眼盯真 —— 图像篡改检测系统项目计划书》

[3] 李明，张华. AI 图像篡改检测技术进展与挑战 [J]. 计算机科学前沿，2024, 18 (5): 123 - 135.

[4] SMITH J, JOHNSON A. Deep Learning for Image Forgery Detection: A Comprehensive Review [M]. London: Academic Press, 2023.

[5] CHEN Y, WANG Z. Secure Data Transmission in AI - based Image Detection Systems [J]. IEEE Transactions on Information Security and Forensics, 2023, 40 (3): 456 - 468.

[6] Microsoft. ONNX Runtime Optimization Guide [EB/OL]. (2024 - 06 - 15)[2025 - 04 - 02]. [https://onnxruntime.ai/docs/optimization](https://onnxruntime.ai/docs/optimization#tdsub).

[7] International Organization for Standardization. ISO/IEC 27001:2022 Information security management systems - Requirements [S]. Geneva: ISO, 2022.

[8] 张悦，李强。基于区块链的 AI 图像数据可信存储方案 [J]. 信息安全学报，2024, 9 (2): 56 - 68.

[9] APPLE INC. Core ML in Image Analysis Applications [EB/OL]. (2023 - 12 - 01)[2025 - 04 - 02]. [https://developer.apple.com/documentation/coreml](https://developer.apple.com/documentation/coreml#tdsub).

## 术语表

|  |  |
| --- | --- |
| **术语 / 缩写** | **定义** |
| AI | 人工智能（Artificial Intelligence），系统利用 AI 技术进行图像真实性检测，如基于深度学习模型识别图像篡改 |
| CNN | 卷积神经网络（Convolutional Neural Network），系统采用基于 CNN 的模型进行图像特征提取和分析 |
| RESTful API | 一种基于 HTTP 协议的应用程序编程接口设计风格，系统通过提供标准化的 RESTful API 接口，便于与第三方应用集成 |
| YOLO | You Only Look Once，一种高效的目标检测算法，系统基于 YOLO - 11n - seg 优化模型进行图像篡改检测 |
| LLM | 大语言模型（Large Language Model），系统集成 LLM 接口辅助用户理解检测结果、提供相关知识和建议 |
| HTML | 超文本标记语言（HyperText Markup Language），前端开发用于构建页面结构的语言 |
| CSS | 层叠样式表（Cascading Style Sheets），用于描述 HTML 页面的样式和布局 |
| JavaScript | 一种高级的、多范式的编程语言，在前端实现交互功能 |
| PyTorch | 深度学习框架，系统使用 PyTorch 进行模型训练和推理，搭配 Python 3.12 作为主要编程语言 |
| MySQL | 一种开源的关系型数据库管理系统，系统选用其存储用户信息、检测记录等数据，遵循第三范式设计数据表结构 |
| JSON | JavaScript 对象表示法（JavaScript Object Notation），一种轻量级的数据交换格式，系统客户端与服务器之间的数据交换采用 JSON 格式，因其简洁、易解析 |
| ONNX Runtime | 系统采用的统一推理引擎，用于实现模型格式标准化，确保 Web/PC/ 移动端检测结果一致性误差≤1% |
| OpenCV | 开源计算机视觉库（Open Source Computer Vision Library），用于实现图像的预处理和后处理 |

# 系统概述

## 系统功能

系统提供图像和视频的真实性检测功能。对于图像检测，可精准识别如 Deepfake 深度伪造、Photoshop 局部篡改等主流伪造手段；视频检测则通过逐帧检测并综合分析，生成帧间篡改痕迹变化轨迹。具备可视化解释功能，利用热图和标注展示伪造区域及其特征。还能生成详细的检测报告，支持多种格式导出，并可根据用户需求定制报告内容。提供集成式 API 服务，方便第三方平台和应用接入图像真实性检测功能。

## 系统架构

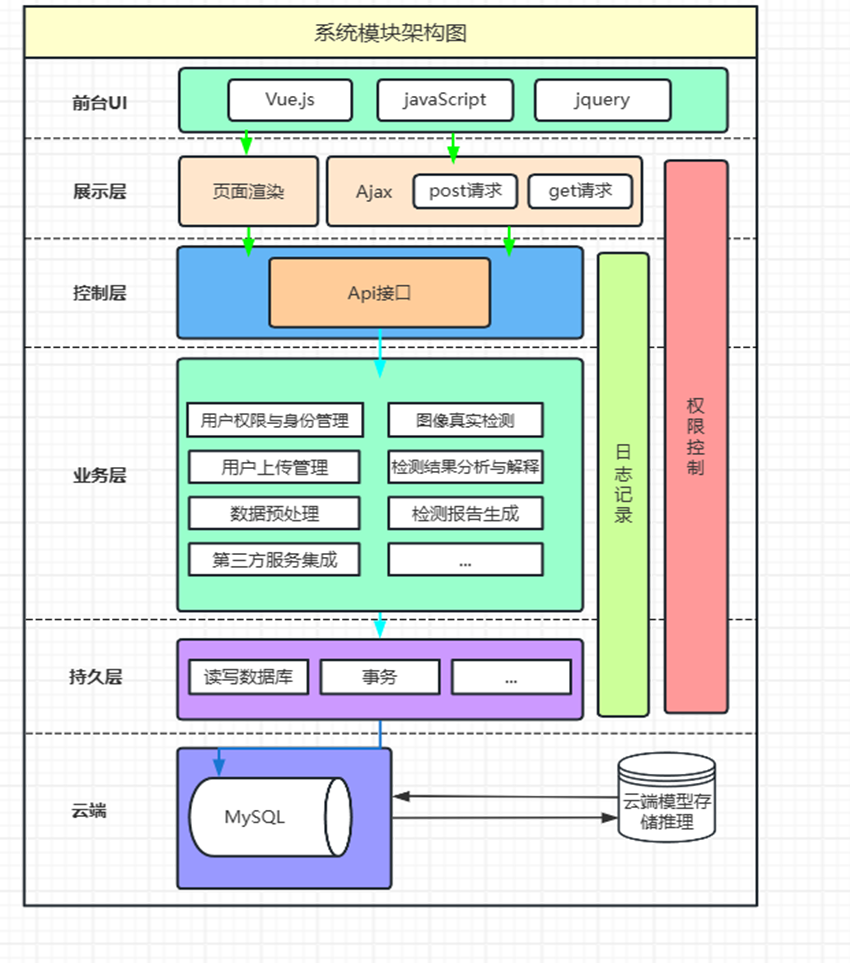
采用 B - S 架构与前后端分离模式。前端以 HTML5、CSS3、JavaScript 为基础构建交互界面，负责用户操作的接收和展示，实现图像和视频上传、检测进度展示、结果可视化及报告导出等功能。后端选用 Python 的 Flask 框架处理业务逻辑，集成 PyTorch 深度学习框架执行图像检测模型推理任务。利用 MySQL 数据库存储用户、图像元数据与检测结果，云存储存放图像和视频文件，保障数据安全与高可用。前后端通过 JSON 格式进行数据交互，接口支持多种请求方法，满足前端多样化交互需求。

# 系统架构

## 架构设计

“义眼盯真 ——AI 图像真实性检测系统” 采用分层架构设计，主要分为前台 UI、展示层、控制层、业务层、持久层以及云端，各层紧密协作，实现系统功能，同时引入权限控制和日志记录机制保障系统安全与可追溯性。

3.1.1 整体架构图



3.1.2 各层功能概述

前台 UI：作为用户交互界面，负责呈现直观友好界面，接收用户操作指令，如上传图像、发起检测请求等，并展示检测结果和报告，提供良好用户体验。

展示层：通过页面渲染模块，将业务数据转化为可视化页面元素展示给用户。

控制层：核心为 Api 接口，接收展示层请求，依据请求类型和参数调用业务层对应服务，起到请求分发与协调作用，是前后端交互桥梁。

业务层：涵盖多个关键模块。用户权限与身份管理模块负责验证用户身份、保障系统安全；图像真实检测模块利用深度学习等技术对图像进行真实性分析；用户上传管理模块处理用户图像上传相关事宜；检测结果分析与解释模块剖析检测结果；数据预处理模块对上传图像进行前期处理；检测报告生成模块依据检测结果生成规范报告；第三方服务集成模块负责对接服务，增强系统功能。

持久层：通过读写数据库模块实现对数据的持久化存储与读取，事务模块保障数据操作的原子性、一致性、隔离性和持久性，确保数据可靠。

云端：MySQL 数据库部署于此，存储用户信息、图像元数据、检测记录等结构化数据。云端模型存储推理模块存放图像检测模型，为检测任务提供推理支持。

权限控制：贯穿系统各层，依据用户角色和权限配置，限制对敏感功能和数据的访问，保障系统安全性。

日志记录：对系统关键操作、错误信息等进行记录，便于问题排查、审计追踪和系统性能分析。

## 分解描述

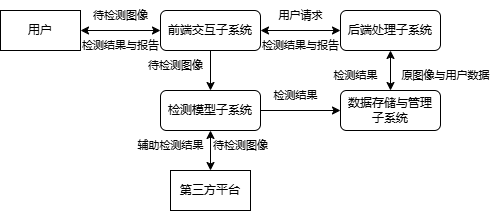
“义眼盯真 ——AI 图像真实性检测系统” 在架构上清晰地划分为前端交互、后端处理、检测模型、数据存储与管理这四个子系统，各子系统紧密协作，共同达成图像真实性检测及相关功能。

图3.2.1 顶层数据流图

前端交互子系统：此系统聚焦于用户交互体验，运用 HTML5、CSS3、JavaScript 以及现代前端框架搭建交互界面。在功能实现上，它为用户提供便捷的操作流程。用户可通过直观的上传区，以点击或拖拽的方式上传图像和视频；进度展示区实时反馈检测进度，消除用户等待的不确定性；结果展示区借助热图和标注，精准呈现图像的篡改区域，并配合详细文字报告，让用户清晰了解图像的真伪、篡改方式及置信度。为适配不同设备，界面采用响应式设计，在 PC、平板、手机端均能提供良好的显示效果与操作体验。

后端处理子系统：该子系统以 Python 的 Flask 框架为依托，承担着业务逻辑处理与数据调度的关键任务。它接收前端发送的请求，对用户进行身份验证和权限检查，确保系统的安全性。在数据流转管理方面，它与数据库和云存储交互，实现用户信息、图像元数据、检测结果等数据的存储、读取和更新。在图像和视频检测过程中，后端处理子系统调用深度学习模型（如 YOLO 模型）进行分析，并将检测结果返回给前端。它还提供集成式 API 服务，支持第三方平台和应用接入，方便企业或开发者进行二次开发，扩展图像真实性检测功能的应用范围。

检测模型子系统：作为系统核心，检测模型子系统基于卷积神经网络（CNN）架构，采用如 YOLO11n 等先进模型，并结合注意力机制等技术进行高效的特征提取和分析。在图像检测方面，它能够精准识别通过传统图像编辑工具（如 Photoshop）进行的拼接、移动、裁剪等篡改，以及通过 AI 图像编辑工具（如 FaceApp）进行的换脸、物体消除等篡改行为。对于视频检测，它采用逐帧检测与结果合并策略，对视频的每一帧进行细致检测，精准识别篡改痕迹，并整合帧级检测结果，为用户生成帧间篡改痕迹变化轨迹，全面呈现视频的真实性变化情况，为用户提供准确的检测结果。

数据存储与管理子系统：该子系统采用关系型数据库（如 MySQL）和云存储（如阿里云 OSS）相结合的方式，实现数据的可靠存储与高效管理。MySQL 数据库用于存储用户信息、图像元数据、检测结果等结构化数据，通过合理设计数据表结构，遵循第三范式，确保数据的规范化，减少数据冗余，保持数据完整性。云存储则用于存放用户上传的图像和视频文件，提供高可用性、可扩展性和数据安全性保障。数据存储与管理子系统还负责数据的备份、清理和优化，保障系统性能。

## 接口描述

3.3.1 内部接口

系统内部各组件通过接口进行高效交互。前端交互子系统将用户操作数据（如登录信息、图像上传数据）以 JSON 格式经 HTTP 协议发往后端处理子系统，后端处理子系统处理后将结果（检测结果、报告数据）回传。后端处理子系统调用检测模型子系统接口时，传递图像数据和检测参数，检测模型子系统返回检测结果。同时，后端处理子系统与数据存储与管理子系统交互，进行数据的存储与读取 。

3.3.2 外部接口

OpenAI LLM 接口：集成 OpenAI 大语言模型接口辅助用户理解检测结果。用户查看报告时可提问，支持自然语言交互、批量提问和多轮对话。选用市场主流稳定模型及官方最新 API 版本，加密用户提问和模型回答，过滤审核提问内容 。

RESTful API 接口：提供标准化 RESTful API 接口，支持第三方应用接入图像篡改检测功能。对请求进行权限验证和参数校验，确保数据安全与请求合法，保证高可用性和稳定性，记录调用日志 。

## 设计依据

3.4.1 分层架构的优势

分层架构设计使得系统的各个部分职责清晰，便于开发团队进行分工协作，提高开发效率。同时，各层之间的松耦合关系使得系统的维护和扩展更加容易，例如可以在不影响其他层的情况下对某一层进行升级或替换。

3.4.2 采用 RESTful API 的原因

RESTful API 具有良好的通用性和易用性，便于第三方系统集成。通过标准化的接口，不同的系统可以方便地调用本系统的图像和视频检测功能，促进了系统的开放性和扩展性。

3.4.3 选择 MySQL的考虑

MySQL 是一种成熟的关系型数据库，适合存储结构化数据，如用户信息、检测记录等。它具有高可靠性、高并发处理能力和丰富的功能，能够满足系统的数据存储需求。

# 数据设计

## 数据描述

本系统采用 MySQL 关系型数据库进行结构化数据存储，并结合云存储方案存放图像、视频等大文件。数据库设计遵循第三范式，以确保数据的规范性、完整性和可扩展性。

数据库主要包括以下核心数据实体：

用户信息（users）：存储用户账号、权限、个人信息等。

上传文件（uploads）：存储用户上传的图像或视频的元数据。

检测任务（tasks）：记录检测请求及其状态。

检测结果（results）：存储检测任务的分析结果。

日志记录（logs）：记录系统关键操作，便于审计和问题排查。

## 数据字典

#### 4.2.1 用户信息表（users）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **约束** | **说明** |
| user\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | 用户唯一标识 |
| username | VARCHAR(50) | UNIQUE, NOT NULL | 用户名 |
| password\_hash | VARCHAR(255) | NOT NULL | 密码哈希值 |
| email | VARCHAR(100) | UNIQUE, NOT NULL | 邮箱 |
| role | ENUM('admin', 'user') | DEFAULT 'user' | 用户角色 |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | 账户创建时间 |

#### 4.2.2 上传文件表（uploads）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 约束 | 说明 |
| upload\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | 上传文件唯一标识 |
| user\_id | INT | FOREIGN KEY -> users(user\_id) | 上传用户 |
| file\_name | VARCHAR(255) | NOT NULL | 文件名称 |
| file\_type | ENUM('image', 'video') | NOT NULL | 文件类型 |
| file\_size | INT | NOT NULL | 文件大小（字节） |
| storage\_path | VARCHAR(255) | NOT NULL | 云存储路径 |
| uploaded\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | 上传时间 |

#### 4.2.3 检测任务表（tasks）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 约束 | 说明 |
| task\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | 检测任务唯一标识 |
| upload\_id | INT | FOREIGN KEY -> uploads(upload\_id) | 关联的上传文件 |
| status | ENUM('pending', 'processing', 'completed', 'failed') | DEFAULT 'pending' | 任务状态 |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | 任务创建时间 |
| completed\_at | DATETIME | NULL | 任务完成时间 |

#### 4.2.4 检测结果表（results）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 约束 | 说明 |
| result\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | 检测结果唯一标识 |
| task\_id | INT | FOREIGN KEY -> tasks(task\_id) | 关联的检测任务 |
| detected\_forgery | BOOLEAN | NOT NULL | 是否检测到篡改 |
| confidence\_score | FLOAT | NOT NULL | 置信度得分（0-1） |
| result\_details | TEXT | NULL | 详细分析结果 |
| generated\_report | VARCHAR(255) | NULL | 生成的报告存储路径 |

#### 4.2.5 日志记录表（logs）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 约束 | 说明 |
| log\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | 日志唯一标识 |
| user\_id | INT | FOREIGN KEY -> users(user\_id) | 操作用户 |
| action | VARCHAR(255) | NOT NULL | 操作描述 |
| ip\_address | VARCHAR(50) | NOT NULL | 用户 IP 地址 |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | 操作时间 |

#### 

# 组件设计

## 前端交互

5.1.1 上传图像和视频功能

PROCEDURE UploadMedia()

// 初始化上传区域

INITIALIZE upload\_area

// 监听点击和拖拽事件

WHILE TRUE

IF upload\_area 被点击 THEN

OPEN file\_selector

SELECTED\_FILES = GET\_SELECTED\_FILES(file\_selector)

IF SELECTED\_FILES 不为空 THEN

SEND\_REQUEST(后端处理子系统, SELECTED\_FILES)

END IF

END IF

IF 有文件被拖拽到 upload\_area THEN

DRAGGED\_FILES = GET\_DRAGGED\_FILES(upload\_area)

SEND\_REQUEST(后端处理子系统, DRAGGED\_FILES)

END IF

END WHILE

END PROCEDURE

5.1.2 进度展示功能

PROCEDURE ShowProgress()

// 初始化进度展示区域

INITIALIZE progress\_area

WHILE TRUE

// 从后端处理子系统获取进度信息

PROGRESS\_INFO = GET\_PROGRESS(后端处理子系统)

// 更新进度展示区域

UPDATE\_PROGRESS\_DISPLAY(progress\_area, PROGRESS\_INFO)

// 检查是否完成

IF PROGRESS\_INFO 为完成状态 THEN

BREAK

END IF

// 等待一段时间后再次检查

WAIT(1 秒)

END WHILE

END PROCEDURE

5.1.3结果展示功能

PROCEDURE ShowResult()

// 初始化结果展示区域

INITIALIZE result\_area

// 从后端处理子系统获取检测结果

DETECTION\_RESULT = GET\_RESULT(后端处理子系统)

// 生成热图和标注

HEATMAP = GENERATE\_HEATMAP(DETECTION\_RESULT)

ANNOTATIONS = GENERATE\_ANNOTATIONS(DETECTION\_RESULT)

// 显示热图和标注

DISPLAY\_HEATMAP(result\_area, HEATMAP)

DISPLAY\_ANNOTATIONS(result\_area, ANNOTATIONS)

// 生成文字报告

TEXT\_REPORT = GENERATE\_TEXT\_REPORT(DETECTION\_RESULT)

// 显示文字报告

DISPLAY\_TEXT\_REPORT(result\_area, TEXT\_REPORT)

END PROCEDURE

## 后端处理

5.2.1 用户身份验证和权限检查功能

PROCEDURE AuthenticateAndAuthorize()

// 从前端请求中获取用户信息

USER\_INFO = GET\_USER\_INFO(前端请求)

// 从数据库中查询用户信息

DB\_USER\_INFO = QUERY\_DB(数据库, USER\_INFO)

IF DB\_USER\_INFO 不为空 THEN

// 验证密码

IF VERIFY\_PASSWORD(USER\_INFO.password, DB\_USER\_INFO.password) THEN

// 检查权限

IF CHECK\_PERMISSION(DB\_USER\_INFO.permission) THEN

RETURN 验证通过

ELSE

RETURN 权限不足

END IF

ELSE

RETURN 密码错误

END IF

ELSE

RETURN 用户不存在

END IF

END PROCEDURE

5.2.2 图像和视频检测功能

PROCEDURE DetectMedia()

// 从前端请求中获取图像和视频

MEDIA = GET\_MEDIA(前端请求)

// 调用检测模型子系统进行检测

DETECTION\_RESULT = CALL\_DETECTION\_MODEL(检测模型子系统, MEDIA)

// 将检测结果返回给前端

SEND\_RESULT(前端交互子系统, DETECTION\_RESULT)

END PROCEDURE

5.2.3 API 服务功能

PROCEDURE APIService()

// 监听 API 请求

WHILE TRUE

API\_REQUEST = GET\_API\_REQUEST()

// 处理 API 请求

API\_RESPONSE = PROCESS\_API\_REQUEST(API\_REQUEST)

// 返回 API 响应

SEND\_API\_RESPONSE(API\_REQUEST.sender, API\_RESPONSE)

END WHILE

END PROCEDURE

## 检测模型

5.3.1 图像检测功能

PROCEDURE ImageDetection()

// 接收图像

IMAGE = RECEIVE\_IMAGE(后端处理子系统)

// 特征提取

FEATURES = EXTRACT\_FEATURES(IMAGE)

// 模型推理

PREDICTION = INFERENCE(FEATURES)

// 识别篡改行为

TAMPERING\_BEHAVIORS = IDENTIFY\_TAMPERING(PREDICTION)

// 返回检测结果

RETURN TAMPERING\_BEHAVIORS

END PROCEDURE

5.3.2 视频检测功能

PROCEDURE VideoDetection()

// 接收视频

VIDEO = RECEIVE\_VIDEO(后端处理子系统)

// 逐帧检测

FRAME\_RESULTS = []

FOR EACH FRAME IN VIDEO

// 特征提取

FEATURES = EXTRACT\_FEATURES(FRAME)

// 模型推理

PREDICTION = INFERENCE(FEATURES)

// 识别篡改痕迹

TAMPERING\_TRACES = IDENTIFY\_TAMPERING(PREDICTION)

FRAME\_RESULTS.append(TAMPERING\_TRACES)

END FOR

// 合并帧级检测结果

VIDEO\_RESULT = MERGE\_FRAME\_RESULTS(FRAME\_RESULTS)

// 生成帧间篡改痕迹变化轨迹

TRACK = GENERATE\_TRACK(VIDEO\_RESULT)

// 返回检测结果

RETURN VIDEO\_RESULT, TRACK

END PROCEDURE

## 数据存储与管理

5.4.1 数据存储功能

PROCEDURE DataStorage()

// 接收数据

DATA = RECEIVE\_DATA(后端处理子系统)

// 存储结构化数据到数据库

STORE\_STRUCTURED\_DATA(数据库, DATA.user\_info, DATA.image\_metadata, DATA.detection\_result)

// 存储图像和视频文件到云存储

STORE\_FILES(云存储, DATA.images, DATA.videos)

END PROCEDURE

5.4.2 数据备份、清理和优化功能

PROCEDURE DataMaintenance()

// 备份数据

BACKUP\_DATA(数据库, 云存储)

// 清理过期数据

CLEAN\_EXPIRED\_DATA(数据库, 云存储)

// 优化数据库性能

OPTIMIZE\_DATABASE(数据库)

END PROCEDURE

## 性能设计

资源利用：移动端轻量化模型运行时，CPU 利用率≤30%，内存占用≤50MB；云端服务高并发时，单节点 CPU 利用率≤70%，内存利用率≤75%。检测数据定期清理，存储利用率≤60%，模型训练数据存储利用率≤70%。

扩展能力：云端服务可动态扩缩容，峰值流量下能够扩展至5个GPU计算节点。采用模块化、插件化设计，新增伪造检测类型功能扩展耗时≤2 人周。

并发性能：支持≥500用户并发在线，视频检测任务队列≥50个并发线程。单节点图像检测≥50请求/秒，视频检测≥10请求/秒。

用户体验：单图检测≤3秒，云端1分钟视频检测≤5分钟。Web 端主界面加载≤2秒，检测结果页加载≤1.5秒，移动端交互流畅度 FPS≥60。

## 部署环境

硬件环境：服务器推荐 Intel Xeon 处理器、16GB 以上内存、大容量硬盘，涉及模型训练时需 NVIDIA GPU。移动端要求3GB 以上内存、64GB 以上存储。

软件环境：服务器操作系统可选 Linux 或 Windows Server，后端 Python 3.9+（建议 3.12）搭配 Flask 框架、PyTorch 深度学习框架，数据库 MySQL 8.0+。前端支持主流桌面和移动操作系统，兼容常见浏览器新版本。

网络环境：服务器网络稳定、带宽充足，移动端推荐 4G 及以上移动网络或稳定 Wi-Fi。

部署架构：前端静态文件部署在 Nginx 或 Apache 服务器，后端采用 Docker 容器化部署，MySQL 配置主从复制。模型推理层用 PyTorch Serve 部署并 GPU 加速。部署时设置防火墙、加密数据、实现身份认证和授权。

# 人机界面设计

## 用户界面概述

从用户角度看，本系统提供图像和视频篡改检测功能。在图像检测界面，用户可通过导航栏切换功能。点击 “上传图片” 上传图像，随后选择 “拼接移动检测” 或 “AI 深度篡改检测” 进行分析。检测后，“原始图片” 区展示原图，“检测结果” 区呈现检测结论。用户能利用 “报告设置” 定制报告，通过 “导出报告” 保存。还可在 “AI 分析助手” 输入问题获取检测结果解释。

在视频检测界面，导航栏功能一致。用户先 “选择视频” 并 “上传视频”，再选 “逐帧检测” 或 “连续检测”。“原始视频” 区支持播放操作，“检测结果” 区显示检测结果。同样可进行报告设置与导出，借助 “AI 分析助手” 了解检测详情。系统通过界面状态变化和提示信息为用户提供反馈，助力用户完成操作。

## 屏幕图像



图6.2.1 图像篡改检测界面原型



图6.2.2 视频篡改检测界面原型

## 屏幕对象和操作

6.3.1 图像篡改检测界面

导航栏：

首页：点击可返回系统首页，方便用户快速切换到起始界面。

图片检测：当前所在的功能页面，表明用户可在此进行图像相关的检测操作。

视频检测：点击可切换到视频检测功能页面，为用户提供视频篡改检测的入口。

功能按钮区：

上传图片：用户点击该按钮可从本地设备选择图片上传到系统，进行后续的篡改检测。

拼接移动检测：选择此功能，系统会针对图片中可能存在的拼接或移动元素进行检测，分析图片是否存在这类篡改情况。

AI 深度篡改检测：对图片进行深度分析，检测更为复杂和隐蔽的篡改痕迹。

报告设置：点击后可对检测报告的格式、内容等进行个性化设置，满足不同用户对报告的需求。

导出报告：将检测结果生成报告并导出，用户可保存到本地设备，方便查看和分享。

显示区域：

原始图片：展示用户上传的原始图片，便于用户核对检测对象。

检测结果：呈现系统对图片的检测结果，可能包括是否篡改、篡改位置标注等信息。

AI 分析助手区：

AI 分析助手：提供一个交互区域，系统可在此输出对检测结果的分析和解释。

输入框：提示文字为 “有什么想告诉 AI 助手的”，用户可在此输入问题或指令，与 AI 进行互动，获取更多关于检测结果的信息。

发送：点击该按钮，可将用户在输入框中的内容发送给 AI 助手，以获取相应回复 。

6.3.2 视频篡改检测界面

导航栏及AI分析助手与图片篡改检测页面相同，故不再赘述。

功能按钮区：

选择视频：点击后可从本地设备中选取要检测的视频文件。

上传视频：用于将选好的视频上传至系统，以便进行后续的篡改检测操作。

逐帧检测：系统会对视频的每一帧画面分别进行篡改检测，能精准发现每帧中的篡改情况。

连续检测：对视频进行连续的、整体性的检测分析，可检测出视频中连续画面的篡改特征。

报告设置：可对检测报告的格式、内容等进行个性化设置，满足不同用户的报告需求。

导出报告：将视频检测结果生成报告并导出，方便用户保存和分享。

显示区域：

原始视频：展示用户上传的原始视频，用户可在此播放、暂停、快进或后退视频，还能查看视频的播放进度和分辨率等信息。

检测结果：用于呈现视频篡改检测的结果，如哪些部分存在篡改、篡改类型等信息。

# 需求矩阵

需求矩阵用于清晰展示系统组件和数据结构与软件需求规格说明书（SRS）中功能需求的对应关系，确保每个功能需求都能在系统设计中找到明确的实现载体，便于开发、测试和维护过程中的需求追溯与管理。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **需求编号** | **需求描述** | **所属功能模块** | **相关系统组件** | **对应数据结构** |
| REQ - 1 | 支持邮箱、手机号和常见第三方账号（如微信、QQ）登录，集成对应 API 进行身份验证 | 用户登录注册 | 前端交互子系统：提供登录界面  后端处理子系统：处理登录验证逻辑 | 用户信息表（users）：存储用户账号、权限、邮箱、手机号、第三方账号关联等信息 |
| REQ - 2 | 提供登录状态错误提示功能，在用户连续多次输入错误时锁定账户 | 用户登录注册 | 前端交互子系统：显示错误提示  后端处理子系统：记录错误次数并锁定账户 | 用户信息表（users）：包含登录错误次数、账户锁定状态字段 |
| REQ - 3 | 支持密码找回，通过邮箱或手机号验证完成密码重置 | 用户登录注册 | 前端交互子系统：提供密码找回界面  后端处理子系统：验证并重置密码 | 用户信息表（users）：用于更新用户密码 |
| REQ - 4 | 支持单图和批量检测功能，批量检测支持一次上传 50 张图像 | 图像检测 | 前端交互子系统：提供图像上传界面  后端处理子系统：接收上传文件并调度检测 | 上传文件表（uploads）：存储上传图像或视频的元数据，如文件名、上传时间等  检测任务表（tasks）：记录检测请求及状态 |
| REQ - 5 | 系统自动识别图像格式，对不支持的格式进行提示 | 图像检测 | 后端处理子系统：识别图像格式并返回提示  前端交互子系统：显示提示信息 | 上传文件表（uploads）：包含文件格式字段 |
| REQ - 6 | 利用优化的深度学习模型，实现对主流伪造手段的精准检测，检测准确率≥85% | 图像检测 | 检测模型子系统：运行检测模型  后端处理子系统：调用模型并处理结果 | 检测结果表（results）：存储检测分析结果 |
| REQ - 7 | 生成详细的定制化检测报告，支持多种格式导出 | 图像检测 | 后端处理子系统：生成报告内容和格式  前端交互子系统：提供报告展示和导出界面 | 检测结果表（results）：提供报告所需数据 |
| REQ - 8 | 支持用户查看、下载和分享检测报告 | 报告管理 | 前端交互子系统：提供操作界面  后端处理子系统：处理查看、下载和分享请求 | 检测结果表（results）：存储报告数据 |
| REQ - 9 | 对分享的报告提供加密选项，保障报告安全性 | 报告管理 | 后端处理子系统：加密报告 | 无新增特定表，可在检测结果表（results）中标记加密状态 |
| REQ - 10 | 系统记录报告的查看、下载和分享记录，方便用户追溯 | 报告管理 | 后端处理子系统：记录操作信息  前端交互子系统：提供记录查看界面 | 日志记录表（logs）：记录报告操作信息 |
| REQ - 11 | 在报告过期时，及时清理过期报告，释放存储空间 | 报告管理 | 后端处理子系统：检查并清理过期报告 | 检测结果表（results）：包含报告过期时间字段 |
| REQ - 12 | 支持系统管理员对用户信息进行全面管理 | 系统管理 | 前端交互子系统：提供管理界面  后端处理子系统：处理管理请求 | 用户信息表（users）：存储用户相关信息 |
| REQ - 13 | 对深度学习模型进行管理，确保模型的准确性和时效性 | 系统管理 | 后端处理子系统：处理模型上传和更新  检测模型子系统：加载新模型 | 无新增特定表，可在系统配置中记录模型版本信息 |
| REQ - 14 | 管理系统数据，进行数据备份、清理和优化 | 系统管理 | 后端处理子系统：执行数据管理操作 | 日志记录表（logs）：记录数据管理操作信息 |
| REQ - 15 | 实时监控系统运行状态，出现异常时及时发出警报 | 系统管理 | 后端处理子系统：监控状态并发出警报  前端交互子系统：显示警报信息 | 日志记录表（logs）：记录系统运行状态信息 |
| REQ - 16 | 提供标准化的 RESTful API 接口，支持第三方应用接入 | API 集成与对接 | 后端处理子系统：实现 API 接口 | 日志记录表（logs）：记录 API 调用信息 |
| REQ - 17 | 对 API 请求进行严格的权限验证和参数校验 | API 集成与对接 | 后端处理子系统：验证和校验请求 | 用户信息表（users）：提供权限信息 |
| REQ - 18 | 保证 API 的高可用性和稳定性，在高并发情况下也能及时响应 | API 集成与对接 | 后端处理子系统：采用技术保障可用性和稳定性 | 无新增特定表，可在系统配置中记录相关参数 |
| REQ - 19 | 记录 API 的调用日志，便于追踪和审计 | API 集成与对接 | 后端处理子系统：记录调用日志 | 日志记录表（logs）：记录 API 调用详细信息 |