系统需求规格说明书

知识图谱构建子系统需求规格说明书

一、引言

1.1 目的

本文档旨在对知识图谱构建子系统的功能需求与非功能需求进行全面、系统的阐述和定义，确保项目各方在需求理解、设计实现、测试验证和后期运维等环节保持一致。

1.2 背景分析

该项目的目标是构建一个关于海外文物的知识图谱。通过爬取大英博物馆，普林斯顿大学博物馆，哈佛艺术博物馆网站上的中国文物信息，将这些数据加工成三元组形式，并最终存储到图数据库中，从而实现知识图谱的构建和可视化。项目的主要功能模块包括数据的爬取、建模、及存储。为了实现这些功能，首先需要研究网站结构爬取文物信息，然后对爬取到的文物信息进行提取、整理和翻译处理（为了适配中文的可视化系统）。最终整理好的数据将被存储到图数据库，构建知识图谱，为其他小组的问答系统开发提供支持。

项目同时需要另外收集一些补充的博物馆图片与补充信息，以帮助其他小组获得更丰富的展示内容。

1.3 系统目标

1.系统的完整性与一致性：

正确理解不同的网站结构，确保从博物馆网站抓取的文物信息准确完整，且在后续的数据处理和存储过程中始终保持一致，避免出现数据丢失或冗余的情况。

2.数据的准确性和可靠性：

确保爬取到的文物信息来源准确无误，避免爬取到错误的数据，并配置正确的数据清洗，整理，翻译系统，对爬取出的数据进行去重等处理，保证数据高质量。

2.系统的性能和响应速度：

系统应具备高效的数据处理能力，能够在大规模文物数据爬取、处理和存储的过程中，以并行脚本处理等方式保持高效的响应速度，确保用户能够流畅使用系统，快速获取所需的文物信息。

4.为其他小组提供支撑：

根据问答系统需求开发合适的知识图谱系统，并保证数据量和数据的多样化。

二、运行环境与系统架构

2.1 系统架构

本子系统为其他系统提供数据支持，包含简单的数据爬取（包括并行化），数据清洗翻译，图数据库构建等过程架构

2.2 运行环境

操作系统：支持 Python 开发的操作系统，如 Windows、Linux 或 macOS。

Python ：建议使用 Python 3.x 版本。确保安装了需要的 Python 库，如 selenium 库保证完成相应的网站访问操作。

网络连接：确保系统能够正常连接互联网，以便爬取海外博物馆网站的数据。

数据库：MySQL，用于存储用户数据和其他非图数据。数据库应与使用我们爬取的数据的小组共享使用。

图数据库：Neo4j，用于存储和查询文物信息的三元组数据。

存储空间：足够的存储空间用于保存爬取到的文物信息和图数据库的数据。

三、功能需求

3.1 数据爬取

该部分是获取数据的第一步工作，要求正确分析网站结构，尽可能高效的爬取文物的各字段信息。应正确处理伪分页，防爬虫机制等问题。

3.2 数据清洗

将数据清洗为与数据库中格式匹配的数据，并且各属性列应尽量直观易懂以满足其他系统的显示。

对于无法从网站中爬取但较为重要的信息，应考虑综合其他信息，使用大模型等方式生成。

3.3 数据翻译与补充

- 将原始数据中英文的部分翻译成中文
- 为部分名称繁杂的文物进行重新命名

3.4 数据存储与建模

- 将原始数据中英文的部分翻译成中文
- 为部分名称繁杂的文物进行重新命名
- 将原始数据中英文的部分翻译成中文
- 为部分名称繁杂的文物进行重新命名
- 将清洗完的数据导入 MySQL 数据库
- 连接 MySQL 数据库根据数据库中的表建立实体和实体间关系（三元组）

- 将三元组导入图数据库

四、非功能需求

4.1 并行加速

考虑到本项目对爬取的数据量有一定的硬性要求，我们应考虑使用并行化方法加速。可考虑使用脚本并行运作多个 python 爬虫程序。

对于数量庞大的清洗和翻译任务，同样可以考虑并行处理。

五、 数据需求

知识图谱数据：系统需要整合知识图谱中的数据，包括文物的相关信息。主要包括文物图片，名称等基本信息和各异的详细信息。

互联网百科数据：系统需要获取互联网百科等数据源中的信息，丰富系统知识库。如其他博物馆的介绍，图片等。

知识服务子系统需求规格说明书

一、引言

1.1 目的

本文档旨在详细说明海外藏中国文物知识管理与服务平台中海外文物知识服务子系统的软件需求。本系统将提供数据浏览、查询和可视化服务，帮助用户有效管理和探索中国文物信息。

1.2 预期阅读人员及阅读建议

本文旨在说明海外知识服务子系统的功能需求和非功能需求。本文的预期阅读对象如下：

- 项目开发者
- 测试人员
- 维护人员
- 用户

1.3 参考资料

- 查询：克利夫兰博物馆网站
- 时间轴：时间轴工具、全历史时间轴
- 知识图谱可视化：全历史关系图谱
- 关系图谱：历史人文大数据平台
- Neo4j 图数据库
- 相关项目文档和 API 文档

二、项目总体描述

2.1 项目应用分析

本项目为海外藏中国文物知识管理与服务平台的一个组成部分，它与其他子系统共同构成了一个综合的服务平台，目的在于通过相关文物图片及其信息的展示以及时间轴和知识图谱的引入，使使用者更加直观、清晰地了解文物

2.2 产品功能

- **数据浏览**：显示文物基本信息，如名称、参数、年代、材料、类型、描述、作者、图片等，同时支持图片的下载、分享、评论、点赞、收藏等交互性较强的功能。
- **数据查询**：支持关键字查询和高级多条件查询
- **数据可视化**：通过知识图谱展示文物间的关系，包括力导向图和时间轴
- **用户信息管理**：支持用户注册、登录，同时记录评论、点赞、收藏、浏览历史、个人信息，以及信息编辑

2.3 用户类别和特征

- **一般访客**：无需登录，浏览和查询文物信息
- **注册用户**：保存浏览历史、收藏、评论等功能

2.4 运行环境

- 部署平台：Linux 服务器
- 支持浏览器：Chrome、Firefox、Safari 等

2.5 设计和实施限制

- 使用技术：HTML5、CSS3、vue3、flask

- 后端语言：python

三、功能需求

3.1 数据浏览

3.1.1 描述与优先级

- 功能：允许用户查看和筛选文物信息
- 优先级：高

3.1.2 刺激/响应序列

1. 系统展示默认文物列表
2. 用户点击某一文物，系统展示详细信息

3.1.3 功能需求

- REQ-1：系统应支持按文物类型、年代、博物馆等信息筛选
- REQ-2：系统应支持按时间或大小写字母升序或降序排序
- REQ-3：点击文物应显示详细信息（包括图片、描述、历史）
- REQ-4：详细视图中应支持图片放大缩小功能
- REQ-5：文物详情页中应支持点赞、收藏、下载、转发等提升用户使用体验的功能

3.2 数据查询

3.2.1 描述与优先级

- 功能：关键词搜索和高级搜索
- 优先级：中

3.2.2 刺激/响应序列

1. 用户输入关键词，提交查询
2. 系统返回匹配结果
3. 用户进入高级搜索，选择字段
4. 系统返回多条件匹配结果

3.2.3 功能需求

- REQ-6：支持按名称、博物馆、年代关键词搜索
- REQ-7：高级搜索支持组合多个字段
- REQ-8：未找到结果时，系统应提示用户

3.3 数据可视化

3.3.1 知识图谱展示

3.3.1.1 描述与优先级

- 功能：基于知识图谱展示文物关系
- 优先级：中

3.3.1.2 刺激/响应序列

1. 用户选择查看知识图谱
2. 系统展示图谱
3. 用户操作图谱（缩放、拖动、点击）
4. 系统响应操作并更新视图
5. 用户输入名称

6. 系统展示对应图谱

3.3.1.3 功能需求

- **REQ-9**：提供基于 Web 的交互式知识图谱
- **REQ-10**：节点代表文物，边表示其关系（同一时期、地点等）
- **REQ-11**：用户点击节点查看详细信息
- **REQ-12**：支持图谱缩放和拖动操作
- **REQ-13**：支持按名称或属性选择图谱

3.3.2 时间轴展示

3.3.1.1 描述与优先级

- 功能：基于时间轴展示文物关系
- 优先级：中

3.3.1.2 刺激/响应序列

1. 用户选择查看时间轴
2. 系统展示时间轴
3. 用户操作时间轴（拖动、放大、缩小、点击）
4. 系统响应操作并显示相应时间节点对应的文物

3.3.1.3 功能需求

- **REQ-14**：展示时间轴，并支持拖动和放大、缩小时间轴
- **REQ-15**：时间节点对应文物
- **REQ-16**：用户拖到相应节点显示当前节点文物的详细信息

3.4 用户个人信息管理

3.4.1 描述与优先级

- 功能：用户账户管理与个性化功能
- 优先级：中

3.4.2 刺激/响应序列

1. 用户注册或登录账户
2. 用户浏览展品，进行评论、点赞或收藏
3. 系统保存并展示用户互动数据
4. 用户修改个人信息或密码

3.4.3 功能需求

- **REQ-17**：允许用户注册，填写用户名、密码、手机号码等
- **REQ-18**：提供用户登录功能
- **REQ-19**：用户可对展品评论
- **REQ-20**：用户可收藏展品，在个人中心查看
- **REQ-21**：用户可修改个人资料和密码

四、非功能需求

4.1 性能需求

- **响应时间**：页面加载时间 ≤ 5 秒
- **并发处理**：支持多用户并发访问，系统稳定

- **数据库查询**：常用查询响应时间 ≤ 2 秒

4.2 安全性需求

- **数据完整性**：防止未经授权的数据修改
- **数据加密**：个人数据需加密存储与传输

4.3 软件质量属性

- **可用性**：界面直观，提供用户手册与在线帮助
- **可维护性**：代码结构清晰，符合规范
- **可扩展性**：设计支持功能拓展
- **互操作性**：支持与其他系统（如数据库/API）集成

知识问答子系统需求规格说明书

一、 引言

1.1 文档目的及系统概述

本文档旨在详细描述知识问答系统的功能需求、非功能需求以及技术实现方案，为系统开发、测试和维护提供指导和依据。文档面向项目管理人员、开发团队和测试团队，明确系统功能边界和实现标准，确保开发与需求保持一致。

知识问答系统是一个基于知识图谱、向量检索和大语言模型的智能问答平台，能够为用户提供准确、丰富的知识服务。系统支持多种问答模式。同时，系统采用前后端分离和微服务架构，具有良好的扩展性和可靠性，以提升用户交互体验。

1.2 开发背景

本系统旨在构建一个面向用户提问的智能知识问答平台，支持单实体单属性的精确问答，同时能够处理具有上下文依赖的复杂问题。系统将覆盖至少十类常见问答主题，如文物的收藏地、时代、材质、类型、背景介绍、书画作者等，满足多样化的知识查询需求。平台通过构建和利用知识图谱，将子系统中提取的数据与互联网百科类数据融合形成高质量的知识库，确保问答结果的准确性和权威性。同时，系统具备与其他子系统协同工作的能力，如数据库查询接口、上下文信息共享等，增强整体问答的连贯性与实用性。开发过程中充分结合知识图谱、语义向量检索和大语言模型技术，为用户提供详实、可信的答案服务。

二、 系统架构

系统采用前后端分离架构，整体划分为前端展示层、后端服务层和数据存储层，基于 FastAPI、Vue3 和 Python 技术栈构建，确保系统具备良好的可维护性和扩展性：前端使用 Vue3 构建现代化 Web 界面，实现良好的用户交互体验与响应式布局，负责问题输入、答案展示及知识库管理等功能。后端基于 FastAPI 开发 RESTful 接口，承担问题解析、知识检索、答案生成、用户管理等核心业务逻辑，具备高性能和高并发处理能力。数据层系统采用多种数据存储方案协同工作，包括：MySQL：用于存储用户信息、会话记录、系统配置等结构化数据；

Neo4j：作为图数据库，用于构建和查询知识图谱，实现结构化知识的推理与问答；Milvus：用于存储文本和文档的向量表示，支持高效的语义检索。

三、 功能需求

3.1 普通问答

普通问答功能基于大语言模型（如 DeepSeek、doubao、gemini 等 6 个大模型）的 API 能力，为用户提供结构清晰、内容详实且包含背景知识的问答服务。系统通过自然语言处理技术理解用户输入，调用大模型生成回答，并对结果进行结构化处理后返回给用户。基于大模型的回答结构清晰、内容全面，并且可以提供背景知识，基本解决用户的普通需求。

3.2 知识库问答

知识库问答功能通过引入用户预先上传的文档，实现对特定领域内容的精准回答。当用户提出问题时，系统会基于文档内容进行智能检索和匹配，从中提取相关信息并生成回答。例如，当用户询问“故宫的历史是怎样的”时，系统将不依赖外部通用知识，而是优先根据用户提供的资料进行分析和概括，确保回答内容准确反映文档所述的信息，如提到东方艺术馆藏有“来自中国等国的数万件文物”等内容。该功能有效提升了问答系统对特定语境和专业内容的适应性与准确性。

3.3 知识图谱查询问答

知识图谱问答功能通过直接对接知识图谱，支持基于结构化数据的精准查询与回答，能够满足用户在特定语义下的复杂问答需求。当用户提出如“介绍一下有哪些清朝文物”这类问题时，系统可通过语义解析将自然语言转化为对知识图谱的查询请求，并从图谱中提取相关实体、属性和关系，返回清晰、准确的结构化结果。对检索到的多条图谱数据进行聚合和筛选，生成符合用户问题的答案。这一功能不仅提高了回答的准确性和可解释性，也适用于对数据一致性和逻辑关联有较高要求的场景。

3.4 Milvus 向量数据库语义检索

系统使用向量数据库存储文本与语料的向量表示，实现语义级的文本检索。

- 对外部文档、知识库内容和用户提问进行文本向量化。
- 用户问题向量化后，在向量数据库中检索相似文本或文档。
- 基于检索到的相关内容，结合大语言模型生成回答，提高问答覆盖率和答案准确度。
- 支持全文检索引擎结合向量检索，提供混合检索策略。

四、 非功能需求

4.1 实现流式输出

AI 回答的流式输出功能通过后端异步生成与前端实时渲染相结合的方式，实现了用户界面中 AI 回答内容的逐步呈现，显著提升了交互体验。该功能在后端由 AiHubMixLLM 类中的三个核心方法实现：get_streaming_response（基础问答）、get_kb_streaming_response（基于知识库）、get_kg_streaming_response（基于知识图谱）。它们统一采用 async_client.chat.completions.create 方法并设置 stream=True 来启动流式响应，通过 async for 循环异步获取模型返回的内容块，使用 yield 语句将每段内容实时返回前端。前端则通过监听事件流，逐条追加新内容并动态更新界面，直到回答结束。这种机制避免了长时间等待完整回答生成，提高了系统响应的即时性与用户的沉浸感，尤其适用于长文本生成或复杂问答场景。

4.2 接入多个大模型

系统支持接入多个主流大模型 API，包括 GPT-4o、Gemini 2.5 Flash、DeepSeek R1、Qwen 3.0（通义千问）、MoonshotAI、豆包 1.5 Thinking Pro 共 6 个大模型，充分发挥各模型在不同任务中的优势。通过统一接口调度机制，系统可根据问题类型、时效需求和回答质量动态选择最适合的模型进行响应。例如，GPT-4o 支持联网搜索，适合处理热点资讯和精准问答；Gemini 2.5 Flash 响应速度极快，适合用于快速反馈和简答场景；DeepSeek R1 在中文技术问答中表现出色，结构清晰；MoonshotAI 擅长自然流畅的长文本生成；Qwen 3.0 在文化、通识类内容中逻辑严谨，表达完整；豆包 1.5 则以平实稳定的风格满足一般通用场景需求。多模型协同接入不仅提升了系统的响应灵活性和内容质量，也增强了在不同业务场景下的适应能力和可用性。

4.3 身份验证

系统的身份验证功能基于 JWT（JSON Web Token）机制实现，具备无状态、灵活、安全的特点。核心逻辑集中在 backend/core/auth/jwt.py 中，采用 FastAPI 内置的 OAuth2 认证机制，通过 OAuth2PasswordBearer 定义统一的令牌获取入口（如 v1/api/account/token）。在生成令牌时，系统使用配置中的 SECRET_KEY 和 HS256 加密算法，结合自定义的过期时间生成签名 JWT，确保令牌的完整性与时效性。验证过程中，FastAPI 通过依赖注入自动调用 get_current_user 方法，从请求中提取并解码 JWT 令牌，校验有效性后返回包含用户 ID、手机号和角色类型的 TokenData 对象，供后续 API 调用安全地识别用户身份。该机制不仅提升了接口访问的安全性，也支持细粒度的权限控制，是构建可靠后端服务的基础保障之一。

4.4 结合对话历史记录

系统支持多轮对话和会话管理，提升交互体验。会话功能：用户可以创建新会话或恢复历史会话，系统记录会话上下文，实现上下文连续问答。会话管理：支持查询历史会话列表、更新或删除旧会话，用户可清理无用数据。消息处理：用户消息和系统回复均被记录，保持完整的对话记录。

4.5 md5 加密密码

项目中的密码加密功能通过 Python 标准库中的 hashlib 模块实现，采用 MD5 算法对用户密码进行加密处理。定义了 hash_password 函数，接收原始密码字符串，将其编码为 UTF-8 格式后生成对应的 MD5 哈希值，并返回十六进制表示结果。该函数被用于用户注册时的密码加密存储，以及登录验证过程中的密码比对，从而实现基本的密码保护机制。不过需要注意，MD5 算法存在安全性不足的问题，无法防止彩虹表攻击，实际应用中建议结合盐值或使用更安全的加密算法（如 bcrypt 或 argon2）以增强安全性。

4.6 异步性

项目在处理请求（尤其是流式输出）时充分利用了 Python 的异步编程能力，通过 async/await 语法、异步生成器、事件循环让步机制以及线程池等多种技术手段，实现了高效、非阻塞的响应流程。核心逻辑中，流式响应通过异步生成器逐步 yield 输出内容，配合 await asyncio.sleep(0.001) 的短暂暂停，让出事件循环控制权，从而保障其他请求也能被

及时处理。同时，对于可能阻塞的 IO 操作，如数据库连接或模型加载，通过 run_in_executor 将其移至线程池中异步执行，避免阻塞主事件循环。在 API 层，结合 FastAPI 的 StreamingResponse，将异步生成器绑定为 HTTP 响应流，使得 AI 生成内容可以边处理边返回，有效提升用户体验的实时性与系统的并发处理能力。这种异步架构设计确保了系统在面对大量并发请求和复杂流式任务时，仍具备良好的扩展性与响应性能。

4.7 可扩展性

架构可扩展：系统采用模块化、微服务架构设计，各功能模块解耦，可通过增加服务实例水平扩展系统容量。

业务扩展：系统设计支持后续新知识源（如新的数据库、API）接入，支持接入更多大语言模型或问答引擎，新增功能模块时对现有系统影响最小。

数据扩展：支持知识图谱的动态更新与扩充，能够平滑加载新实体关系；向量库可增量添加新语料向量，保证检索效率。

性能扩展：通过集群和负载均衡机制，可根据访问量自动增加计算和存储资源，满足高负载环境下的需求。

可维护性：代码和配置管理规范化，提供清晰的接口文档，便于后续维护和升级。

五、 数据需求

知识图谱数据：系统需要整合知识图谱中的数据，包括文物的相关信息。

互联网百科数据：系统需要获取互联网百科等数据源中的信息，丰富系统知识库。

六、 部署方案

本地开发部署：开发人员可在本地环境使用 Docker Compose 等工具部署各个微服务（前端、后端、数据库、模型服务等）进行开发和调试。配置开发/测试环境与生产环境的差异（如连接不同的数据源）。

测试环境部署：在独立的测试环境服务器上部署系统，用于集成测试与用户验收测试。应与生产环境尽量一致，但可使用小规模集群。

掌上博物馆 APP 子系统需求规格说明书

一、引言

1.1 文档目的及系统概述

本文档旨在详细定义掌上博物馆 APP 子系统的功能需求、非功能需求以及技术实现方案，为系统的开发、测试和维护提供明确的指导和依据。掌上博物馆模块是海外藏中国文物知识管理与服务平台的重要组成部分，通过手机应用为用户提供便捷的文物展示、浏览、交互等功能，让用户随时随地了解海外藏中国文物的相关信息。

1.2 开发背景

随着移动互联网的飞速发展，人们对于随时随地获取信息的需求日益增长。在文化领域，越来越多的人希望通过手机等移动设备便捷地了解各类文化遗产和文物知识。海外藏中国文物作为中国文化遗产的重要组成部分，具有极高的历史、艺术和科学价值，但由于其分散在世界各地，普通大众获取相关信息的渠道相对有限。为了更好地传播和弘扬中国传统文化，让公众能够更加便捷地了解海外藏中国文物，开发一款掌上博物馆 APP 显得尤为重要。

掌上博物馆 APP 旨在打破时间和空间的限制，将海外藏中国文物的信息整合到一个手机应用中，为用户提供一个随时随地浏览、学习和互动的平台。通过该 APP，用户不仅可以查看文物的基本信息、高清图片和讲解内容，还可以与其他用户进行互动交流，分享自己的见解和体验。此外，APP 还提供了以图搜图等功能，进一步提升了用户的使用体验和探索乐趣。开发这款 APP 不仅有助于满足公众对文化遗产的认知需求，还能促进文化交流与传播，增强民族自豪感和文化自信。

1.3 范围

本系统为海外藏中国文物知识管理与服务平台的掌上博物馆 APP 子系统，主要包含文物浏览、用户交互、以图搜图、用户个人信息管理等功能模块。

1.4 读者对象

· 系统开发人员

· 系统测试人员

· 项目管理人员

· 最终用户代表

二、系统架构

2.1 技术路线/方案

2.1.1 开发平台与工具

· **操作系统**：基于华为 HarmonyOS 进行开发，充分利用其分布式特性、跨设备协同能力以及对多种硬件设备的支持，为用户提供流畅、高效的使用体验。

· **开发工具**：使用华为 DevEco Studio 作为主要的开发工具，它为 HarmonyOS 应用开发提供了全面的开发、调试和性能优化功能，能够有效提高开发效率。

· **代码管理**：采用 Git 进行代码版本管理，通过 GitHub 平台建立项目代码仓库，方便团队成员协作开发、代码提交、版本控制以及代码审查等操作，确保代码的稳定性和可追溯性。

· **测试工具**：利用华为云测试工具进行应用的测试工作，包括功能测试、性能测试、兼容性测试等，及时发现并修复问题，保证应用的质量和稳定性。同时，结合 DevEco Studio 自带的测试工具进行单元测试和集成测试，确保代码的正确性和可靠性。

2.1.2 技术架构

· **前端**：采用 HarmonyOS 的 ArkUI 框架进行界面开发，结合 ArkUI 的组件化开发方式，快速构建出美观、易用的用户界面。同时，利用 HarmonyOS 的分布式 UI 能力，实现多设备间的界面协同显示和交互，提升用户体验。

· **后端**：后端采用 Flask 和 Node.js 进行开发。Flask 用于提供上传图片和以图搜图的接口服务，能够处理图片上传的请求，将图片存储到服务器，并为以图搜图功能提供支持。Node.js 用于处理基本的数据库信息请求和评论图片请求，负责从数据库中获取数据和在服务器获取图片。

· **网络通信**：使用 HTTPS 协议进行网络通信，保证数据传输的安全性和可靠性。同时，采用适当的网络优化技术，如数据压缩、缓存策略等，提高网络请求的效率和响应速度。

三、功能需求

3.1 博物馆浏览模块

· **公告查看**：用户可以查看 APP 和博物馆发布的公告。

· **博物馆详细信息查看**：用户可以查看博物馆的详细信息，包括博物馆地址、博物馆名称、博物馆图片、博物馆拥有的文物、博物馆官网链接、博物馆预约链接等。

· **博物馆排行查看**：用户可以查看博物馆排行（按照博物馆拥有文物数量排行）。

3.2 文物浏览模块

· **文物基本信息展示**：显示文物的基本信息，如名称、年代、来源、尺寸、材料、类型、介绍等。

· **文物图片展示**：展示文物的高清图片。

· **简单搜索**：用户可以按照关键字进行搜索。用户可以按照文物的名称、类型、年代等信息进行搜索，会返回上述信息中存在某一项与搜索关键字模糊匹配的文物。

3.3 用户交互模块

· **点赞功能**：用户可以对单个文物点赞。

· **留言评论**：用户可以对单个文物发表留言评论。评论当中可以包含图片。

· **收藏功能**：用户可以对单个文物进行收藏，并在收藏记录中查看已经收藏的文物。

· **浏览历史功能**：用户浏览某一文物后，回记录浏览历史，并在浏览记录中按时间顺序查看浏览过的文物。

3.4 以图搜图模块

用户可以上传图片或直接拍摄一个照片，根据上传图片特征搜索相关文物。

3.5 用户个人信息管理模块

· **注册登录**：用户可以注册登录该系统，设置用户名密码等个人信息。用户信息可以和海外文物知识服务子系统共用或单独使用一套用户系统。

· **信息修改**：用户可以修改自己的个人信息。

3.6 用户个人动态模块

· **动态发布**：用户可以发表动态，上传文字和图片，分享文物或博物馆等。

· **点赞评论**：其他用户可以对动态进行点赞、评论等。

四、非功能需求

4.1 性能需求

· 系统应支持同时处理至少 100 个并发用户请求。

· 页面响应时间不超过 3 秒。

· 数据库查询响应时间不超过 1 秒。

4.2 安全性需求

· **敏感数据加密**：用户密码采用 MD5 算法进行加密存储，确保用户密码在存储和传输过程中的安全性。

· **数据传输安全**：使用 HTTPS 协议进行网络通信，确保数据传输过程中的加密和完整性，防止数据泄露和中间人攻击。

4.3 可靠性需求

· **系统可用性**：系统应保证 99.9%的可用性，确保用户可以持续稳定地访问和使用应用。

· **灾难恢复能力**：系统应具备灾难恢复能力，确保在发生故障或数据丢失等意外情况时，能够快速恢复数据和服务，减少对用户的影响。

4.4 可维护性需求

· **代码规范**：后端代码应符合 Python 和 JavaScript 的最佳实践，确保代码的可读性、可维护性和一致性。

· **API 文档**：提供完整的 API 文档，包括接口定义、请求参数、返回值等详细信息，便于开发人员理解和使用。

· **版本管理**：采用 Git 进行代码版本管理，通过 GitHub 平台建立项目代码仓库，方便团队成员协作开发、代码提交、版本控制以及代码审查等操作，确保代码的稳定性和可追溯性。

· **单元测试**：编写单元测试，覆盖核心功能和关键模块，确保代码的正确性和可靠性，便于后续的代码修改和维护。

4.5 兼容性需求

· **设备兼容性**：支持多种屏幕尺寸和分辨率的移动设备，确保应用在不同设备上都能正常显示和运行。

· **网络环境兼容性**：支持不同的网络环境，包括 Wi-Fi 和移动数据网络，确保用户在各种网络条件下都能流畅使用应用。

五、数据需求

5.1 文物数据

· 文物基本信息：包括文物名称、年代、来源、尺寸、材质、类型、收藏地等。

· 文物图片：包括文物的高清图片资源，支持多角度展示。

· 用户互动数据：包括用户对文物的点赞数、收藏数、浏览次数等。

5.2 用户数据

· 用户基本信息：包括用户名、密码、昵称、注册时间等。

· 用户行为数据：包括浏览历史、点赞记录、收藏记录、搜索记录、评论内容等。

· 用户动态：包括用户发布的动态信息，如文字、图片等。

· 用户反馈数据：包括用户提交的反馈、建议、问题等信息。

5.3 博物馆数据

· 博物馆基本信息：包括博物馆名称、地址、介绍、官网链接、预约链接等。

· 博物馆公告：包括博物馆发布的公告、展览信息等。

· 博物馆图片：包括博物馆的高清图片资源。

5.4 其他数据

· 以图搜图数据：主要存储每张文物图片的特征向量。

六、部署要求

6.1 服务器配置

· CPU：4 核以上。

· **内存**：8GB 以上。

· **存储**：100GB 以上（根据文物数据量调整）。

6.2 软件环境

· **MySQL**：8.0+。

· **Node.js**：14+。

· **Python**:3.8

6.3 网络要求

· 稳定的互联网连接。

· 支持多种网络环境，包括 Wi-Fi 和移动数据网络。

· 确保低延迟和高吞吐量的网络性能，以提供流畅的用户体验。

后台管理子系统需求规格说明书

一、 引言

1.1 文档目的及系统概述

本文档旨在定义海外藏中国文物知识与服务平台的后台管理子系统的功能和非功能需求，为系统开发提供明确的技术规范和指导。

1.2 范围

本系统为海外藏中国文物知识与服务平台的后台管理子系统，主要包含用户管理、审核管理、数据管理、管理员日志和数据备份等功能模块。

1.3 读者对象

- 系统开发人员
- 系统测试人员
- 项目管理人员
- 最终用户代表

二、 系统架构

系统采用前后端分离架构：

- 前端：Vue.js 框架
- 后端：Java Spring Boot 框架
- 数据库：MySQL
- ORM 框架：MyBatis

三、 功能需求

3.1 用户管理模块

3.1.1 用户列表

用户列表查询

用户添加

用户编辑

用户删除

用户状态修改

3.2 评论审核模块

3.2.1 评论审核列表

评论列表查询

评论审核

评论审核

评论删除

3.2.2 图片审核列表

图片列表查询

图片审核

图片审核

图片删除

3.2.3 审核通过

3.2.4 审核拒绝

3.3 数据管理模块

3.3.1 博物馆数据列表

博物馆列表查询

博物馆添加

博物馆编辑

博物馆删除

3.3.2 文物数据列表

文物列表查询

文物添加

文物编辑

文物删除

3.3.3 公告数据列表

公告列表查询

公告添加

公告编辑

公告删除

3.4 管理员日志模块

3.4.1 日志查询

3.4.2 日志导出

3.5 数据备份模块

3.5.1 备份列表

3.5.2 创建备份

3.5.3 恢复备份

3.5.4 删除备份

四、 非功能需求

4.1 性能需求

1. 系统应支持同时处理至少 100 个并发用户请求
2. 页面响应时间不超过 3 秒
3. 数据库查询响应时间不超过 1 秒

4.2 安全性需求

1. 所有敏感数据必须加密存储
2. 管理员操作必须记录详细日志
3. 实现基于角色的访问控制(RBAC)
4. 所有 API 请求必须进行身份验证

4.3 可靠性需求

1. 系统应保证 99.9%的可用性

2. 数据备份应每天自动执行

3. 系统应具备灾难恢复能力

4.4 可维护性需求

1. 代码应符合 Java 和 Vue.js 的最佳实践

2. 提供完整的 API 文档

3. 系统应具备良好的日志记录能力

4.5 兼容性需求

1. 支持主流浏览器(Chrome, Firefox, Safari, Edge)

2. 支持移动端访问

3. 支持 Windows 和 Linux 服务器环境

五、部署要求

1. 服务器配置：

- CPU: 4 核以上

- 内存: 8GB 以上

- 存储: 100GB 以上(根据文物数据量调整)

2. 软件环境：

- JDK 11+

- MySQL 8.0+

- Node.js 14+

- Nginx(用于前端部署)

3. 网络要求：

- 稳定的互联网连接

- 建议配置 CDN 加速文物图片加载