古瓷器智能检索与知识科普平台设计与实现

Ancient Porcelain Intelligent Retrieval and Knowledge Section Design and Implementation of Common Platform

专　　业：计算机科学与技术

姓　　名：周露玲

指导教师姓名：王媛

申请学位级别：学士

论文提交日期：2023 年 05 月　日

学位授予单位：天津科技大学

摘　要

随着人们对文化遗产的重视和文化旅游的兴起，古瓷器文化逐渐成为人们关注的热点。然而，由于古瓷器数量众多、种类繁多，一般人很难对其进行准确的识别和分类。为了提升人们对古瓷器文化的认知和理解，古瓷器的智能检索与知识科普平台变得尤为重要。

本平台主要包括两个部分：智能检索系统和知识科普平台。智能检索系统采用图像识别技术，能够对用户上传的古瓷器图片进行快速识别和匹配，从而提供相关的古瓷器信息和历史背景。用户还可以通过关键词搜索和分类浏览的方式获取更多的古瓷器相关信息。该平台的实现主要依赖于深度学习和图像处理技术，使用百度云 AI 实现对古瓷器图像的自动识别。知识科普平台提供了古瓷器的相关知识和文化背景，包括古瓷器的历史、特点、文化传承等方面的内容。同时，该平台还提供了在线学习和交流的功能，用户可以通过论坛、问答等方式互动交流，进一步增进对古瓷器文化的认知和理解。该平台的实现主要使用 Springboot + uniapp 主流框架进行开发，使用常见数据库 MySQL、Redis 进行数据的持久化和缓存，提高了平台的响应速度和用户体验，通过构建一个可扩展、易维护的平台，实现古瓷器文化知识的传播和分享。

本文还对平台的应用前景进行了探讨。通过对古瓷器的智能识别和知识科普，本平台能够为广大爱好者和文化爱好者提供一个全面、便捷、深入的古瓷器文化认知平台。同时，该平台也有望促进古瓷器文化的传承和发展，推动相关产业的发展和壮大。为广大古瓷器爱好者和文化爱好者提供了一个全面、便捷、深入的古瓷器文化认知平台。同时该研究成果也可为文化遗产保护和文化旅游发展提供有益的参考。

关键词： Uni-app； Springboot； Redis； 百度云 AI

ABSTRACT

With the increasing emphasis on cultural heritage and the rise of cultural tourism, ancient porcelain culture has gradually become a hot topic of concern. However, due to the large number and variety of ancient porcelain, it is difficult for ordinary people to accurately identify and classify them. In order to enhance people's understanding and understanding of ancient porcelain culture, intelligent retrieval and knowledge popularization platforms for ancient porcelain have become particularly important.

This platform mainly consists of two parts: an intelligent retrieval system and a knowledge popularization platform. The intelligent retrieval system adopts image recognition technology, which can quickly identify and match the ancient porcelain images uploaded by users, thereby providing relevant ancient porcelain information and historical background. Users can also obtain more information related to ancient porcelain through keyword search and classified browsing. The implementation of this platform mainly relies on deep learning and image processing technology, using Baidu Cloud AI to achieve automatic recognition of ancient porcelain images. The knowledge popularization platform provides relevant knowledge and cultural background of ancient porcelain, including the history, characteristics, cultural inheritance, and other aspects of ancient porcelain. At the same time, the platform also provides online learning and communication functions, allowing users to interact and exchange ideas through forums, Q&A, and other means, further enhancing their understanding and understanding of ancient porcelain culture. The implementation of this platform is mainly developed using the mainstream framework of Springboot+uniapp. Common databases such as MySQL and Redis are used for persistence and caching of data, which improves the response speed and user experience of the platform. By building an extensible and easy to maintain platform, the dissemination and sharing of cultural knowledge of ancient porcelain can be realized.

This article also explores the application prospects of the platform. Through intelligent recognition and knowledge popularization of ancient porcelain, this platform can provide a comprehensive, convenient, and in-depth platform for enthusiasts and cultural enthusiasts to understand ancient porcelain culture. At the same time, the platform is also expected to promote the inheritance and development of ancient porcelain culture, and promote the development and growth of related industries. It provides a comprehensive, convenient, and in-depth platform for ancient porcelain enthusiasts and cultural enthusiasts to understand ancient porcelain culture. At the same time, the research results can also provide useful references for the protection of cultural heritage and the development of cultural tourism.

Keywords: Uni-app; Springboot; Redis; Baidu Cloud AI

目　录

第一章 引言

第一节 选题背景及研究意义

随着经济的发展和文化的繁荣，人们对于历史文化的关注度也越来越高。作为中国传统文化的重要代表，古瓷器具有丰富的历史和文化内涵，其独特的艺术价值、历史价值和收藏价值备受关注。然而，由于古瓷器数量众多、种类繁多，且古瓷器鉴定需要专业知识，因此很多人对古瓷器并不了解，甚至存在一些误解和偏见。

为了更好地推广古瓷器文化，提高古瓷器的认知度，促进文化传承和保护，建立一个古瓷器智能检索与知识科普平台具有重要的意义。该平台可以通过智能检索功能[1,2]，让用户快速准确地了解古瓷器的种类、特点、历史背景等相关知识，从而提高对古瓷器的认知度。同时，平台的知识科普功能可以让用户深入了解古瓷器的历史和文化内涵，从而增强对古瓷器的保护意识，促进古瓷器文化的传承。此外，该平台还可以为古瓷器爱好者提供交流平台，方便大家互相探讨，更多地更深刻地了解到中国的古瓷器文化。

最后，该平台的建立可以将古瓷器文化和现代科技相结合，实现文化与科技的融合，为推动文化创新和发展提供了新的思路和方法。因此，建立一个古瓷器智能检索与知识科普平台对于推广古瓷器文化、提高古瓷器的认知度、促进文化传承和保护、为古瓷器爱好者提供便捷的信息服务、推动文化和科技的融合等方面具有重要的意义。

第二节 相关研究综述

古瓷器是中国传统文化的重要组成部分，具有深厚的历史和文化价值。为了更好地保护和传承古瓷器文化，近年来，国内外学者和机构开展了一系列关于古瓷器智能检索与知识科普平台的研究。

国内研究方面，中国国家文物局与清华大学合作开发了“中国文物古瓷器数字化资源库”，该库集成了大量古瓷器的数字化图像、文字介绍和相关文献资料[3]，实现了对古瓷器的智能检索和文化科普。此外，各大博物馆也纷纷开发了自己的古瓷器数字化资源库，并通过网络和移动应用程序向公众提供了古瓷器的在线展示和科普服务。

国外研究方面，美国麻省理工学院的“文物智库”项目致力于将全球文化遗产数字化，并通过人工智能技术实现对文物的智能检索和分析。该项目还开发了移动应用程序，向公众提供了对文物的科普和互动体验[4]。此外，英国大英博物

馆也开发了自己的“智能博物馆”系统[5]，通过移动应用程序向公众提供了对文物的智能检索和文化科普服务。

总体而言，古瓷器智能检索与知识科普平台的研究在国内外均得到了广泛关注和支持。未来，随着人工智能和数字技术的不断发展，古瓷器智能检索与知识科普平台的应用前景将会更加广阔[6,7]。

第三节 论文结构说明

本文共分六个章节，各章节安排如下：

第一章：引言部分。在该部分讲述了在当代经济发展文化的繁荣展的背景下，随着人们对历史文化的关注度日益增加，但获取相关信息的渠道却很少。借此说明了推出一款集瓷器识别和信息科普系统的重要性[8]。同时根据阅读的国内外相关研究，并结合当今信息化时代的发展，以及现在小程序渐渐受大众所喜爱，提出了做一款基于 Java 的智能检索与科普平台小程序的必要性。

第二章：系统开发相关技术和工具部分。首先通过说明该项目的整体架构，前后端分离开发，然后介绍了本系统开发过程中需要用到的技术，前端的 Uni-app，后端的主流框架 Springboot，以及为了简化而生的半自动化持久层框架 MyBatis-Plus，轻量级权限框架 Shiro， 用极受欢迎高性能数据库的 MySQL 做数据的持久化，使用 Redis 做数据缓存，并简单介绍他们在各自领域内所体现出来的优点和缺点。

第三章：系统需求分析。主要分析本系统需要做什么，完成了系统可行性分析，技术可行性分析，经济可行性分析，操作可行性分析，系统性能需求分析等内容，重点完成了系统功能的需求分析，介绍了本系统的四个模块，最终确定了本系 统在上述方面都是可行的。

1. 古瓷器信息识别模块：使用百度云 AI [9,10]以及构建的图形库进行瓷器的拍照识别。
2. 古瓷器科普模块：对瓷器信息图形库进行完善，对瓷器图形库的模糊查询。
3. 用户模块：用户登陆，注册系统，以及获取用户的基本信息。
4. 讨论区模块：

(1) 帖子：用户可以发布相关的帖子，也可以对帖子进行收藏，评论，点赞，分享，同时也借助 Redis 对帖子的浏览量进行了统计。

(2) 评论区：用户发布评论，点赞评论，以及查询时父子评论的处理。

第四章：系统设计与实现部分。从第三章阐述的各种需求出发，结合实际设计并建立了本系统所需要的数据库及各种表结构，根据设计需求设计系统的各个实现功能细节，并通过编写代码来实现各功能模块的具体功能。

第五章：系统测试部分。测试第四章系统所实现的功能。

第六章：总结了本系统的目前现状，并结合实际情况分析提出了系统的完善规划。

第二章 系统开发相关技术和工具

第一节 前后端分离开发概述

前后端分离开发是指将前端和后端的开发分离，让前端和后端各司其职，通过接口进行数据交换，从而提高开发效率和系统的可维护性。在传统的开发模式中，前端和后端是耦合在一起的，前端需要依赖后端的开发进度和接口，后端也需要考虑前端的需求和展示效果，这样会导致开发效率低下，维护困难。而前后端分离开发则能够有效解决这些问题，提高开发效率和系统的可维护性。

前后端分离开发的主要优势包括：

(1) 前后端开发并行：前后端分离后，前端和后端可以并行开发，无需等待对方的进度，从而提高开发效率。

(2) 前端技术更新快：前端技术更新非常快，而后端技术相对稳定。前后端分离后，前端可以根据需求灵活选择最新的技术，而无需考虑后端是否支持。

(3) 更好的可维护性：前后端分离后，前端和后端各司其职，代码耦合度降低，代码结构更加清晰，便于维护和升级。

(4) 更好的性能和体验：前后端分离后，前端可以通过异步请求和缓存等技术提高系统的性能和用户体验。

(5) 更好的安全性：前后端分离后，前端只负责展示数据和处理用户交互，不涉及敏感数据和业务逻辑，从而提高系统的安全性。

同样前后端分离开发也多了很多必须要考虑的问题：

(1) 接口设计和文档维护：前后端分离后，接口设计变得非常重要，需要定义清晰的接口规范和文档，以便前后端开发人员进行沟通和协作。

(2) 跨域问题：由于前后端分离后，前端和后端运行在不同的域名或端口上，会出现跨域问题，需要通过跨域解决方案来解决。

(3) 部署和维护：前后端分离后，需要分别部署前端和后端代码，并进行协同工作，从而增加了部署和维护的复杂度。

第二节 Uni-app 简介

Uni-app 是一款基于 Vue.js 开发的非常优秀的跨平台应用开发框架，可以通过一套代码实现多端发布，包括 iOS、Android、H5、小程序等多个平台。Uni-app

具有开发效率高、开发成本低、维护成本低等优势，被广泛应用于各种应用程序的开发[11]。

Uni-app 主要包括以下几个部分：

(1) 基础组件库：Uni-app 提供了丰富的基础组件库，包括按钮、输入框、列表、表单等常用组件，可以大大提高开发效率。

(2) 高级组件库：Uni-app 还提供了一些高级组件库，包括图表、地图、视频等组件，可以实现更加复杂的功能。

(3) 开发工具：Uni-app 提供了一些开发工具，包括调试器、模拟器、代码编辑器等，可以大大提高开发效率。

(4) 打包工具：Uni-app 提供了自动化构建和部署工具，可以大大简化打包和部署流程。

Uni-app 的优势主要有以下几点：

(1) 跨平台：Uni-app 可以通过一套代码实现多端发布，包括 iOS、Android、H5、小程序等多个平台，大大降低了开发成本和维护成本。

(2) 开发效率高：Uni-app 采用了 Vue.js 框架，开发者可以使用熟悉的 Vue.js 语法进行开发，同时也提供了丰富的组件库和开发工具，可以大大提高开发效率。

(3) 维护成本低：Uni-app 使用了统一的开发规范和组件库，可以大大降低维护成本，同时也提供了自动化构建和部署工具，可以大大简化维护流程。

(4) 性能优良：Uni-app 采用了高性能的渲染引擎和数据处理机制，可以实现流畅的用户体验和高效的数据处理。

Uni-app 的应用场景非常广泛，可以用于各种应用程序的开发，包括电商、社交、游戏、教育、医疗等多个领域。Uni-app 的优势在于可以通过一套代码实现多端发布，同时也可以实现类似原生应用的用户体验，可以满足各种应用程序的需求[12]。

第三节 Springboot 简介

Springboot 是一个基于 Spring 框架的快速开发框架，提供了一系列的开箱即用的特性，如自动配置、快速构建、无需 XML 配置文件、内嵌 Web 服务器等，这些特性可以让开发者快速地搭建一个可运行的、独立的、生产级别的应用程序，可以帮助开发者快速地构建 Spring 应用程序，减少开发者在配置和部署方面的工作量，从而让开发人员更加专注于业务逻辑的实现。

(1) 自动配置：Springboot 可以根据应用程序的依赖关系自动配置应用程序的环境，无需手动配置。

(2) 快速构建：Springboot 提供了一系列的快速构建工具，如 Spring Initializr，可以帮助开发者快速搭建应用程序的基本框架。

(3) 无需 XML 配置文件：Springboot 可以通过注解和 Java 配置类 application.yml / application.properties 来代替 XML 配置文件，使得配置更加简单、直观。

(4) 内嵌 Web 服务器：Springboot 内置了多种 Web 服务器，如 Tomcat、Jetty 和 Undertow 等，可以快速地构建 Web 应用程序。

(5) 健康检查：Springboot 提供了健康检查的功能，可以检查应用程序的状态和运行情况。

(6) 优化性能：Springboot 可以通过自动配置和优化来提高应用程序的性能。

第四节 MyBatis-Plus 简介

MyBatis-Plus 是一个基于 MyBatis 的增强工具，它提供了许多实用的功能和工具，使得开发人员可以更加高效地进行数据库操作[13]。

1. 代码生成器：MyBatis-Plus 提供了一个强大的代码生成器，可以根据数据库表自动生成 Model、Mapper、Service 和 Controller 等代码，极大地简化了开发人员的工作量。生成的代码符合 MyBatis 的规范，可以直接使用或者进行二次开发。
2. CRUD 操作：MyBatis-Plus 提供了许多实用的 CRUD 操作，包括单表查询、多表查询、分页查询、插入、更新、删除等操作。这些操作都是基于 MyBatis 的基本操作进行封装的，使用起来非常方便。
3. 条件构造器：MyBatis-Plus 提供了一个方便的条件构造器，可以根据需要动态生成 SQL 条件语句，支持各种复杂的查询条件，包括等于、不等于、大于、小于、Like、In 等操作。使用条件构造器可以大大简化 SQL 语句的编写，提高代码的可读性和可维护性。
4. 分页插件：MyBatis-Plus 提供了一个快速的分页插件，可以自动进行分页查询，支持各种数据库类型，包括 MySQL、Oracle、SQL Server 等。使用分页插件可以方便地进行分页查询，提高查询效率和用户体验。
5. 注解支持：MyBatis-Plus 提供了许多实用的注解支持，包括@TableId、@TableField、@TableName 等注解，可以方便地进行实体类和数据库表的映射。使用注解可以大大简化代码的编写，提高代码的可读性和可维护性。
6. 自动填充：MyBatis-Plus 提供了一个自动填充功能，可以在插入或更新操作时自动填充一些字段，比如创建时间、更新时间等。使用自动填充可以减少重复代码的编写，提高代码的可读性和可维护性。

7. 性能分析：MyBatis-Plus 提供了一个性能分析功能，可以方便地查看 SQL 执行的时间和执行计划，帮助开发人员优化 SQL 语句和数据库性能。

第五节 Shiro 简介

Shiro 是一个开源的 Java 安全框架，提供了身份认证、授权、加密和会话管理等功能，可以帮助 Java 应用程序快速地实现安全性。Shiro 的设计目标是简单、易用、灵活、高效和可扩展，使开发人员能够专注于业务逻辑的实现，而不是安全细节的处理。Shiro 的核心组件包括 Subject、SecurityManager、Realm 和 SessionManager 等。Subject 是 Shiro 的核心概念，代表了一个用户或系统的安全主体，可以进行身份认证和授权等操作。SecurityManager 是 Shiro 的安全管理器，负责协调各个组件的工作，处理安全相关的操作。Realm 是 Shiro 的数据源，负责提供用户和权限等数据，可以从数据库、LDAP、文件、缓存等数据源中获取数据。SessionManager 则负责管理用户会话，可以实现会话的创建、销毁、过期等操作。Shiro 提供了多种身份认证和授权的方式，包括基于用户名密码的认证、基于角色的授权、基于资源的授权等。Shiro 还支持多种加密算法，包括 MD5、SHA1、AES 等，可以对密码、数据等进行加密处理。Shiro 还提供了与 Spring、Struts2 等框架的集成支持，可以方便地与其他框架进行整合。

第六节 MySQL 简介

MySQL 的发展始于 1995 年，最初由瑞典人 Michael Widenius 和 David Axmark 开发，后来被 Sun 公司收购，现在属于 Oracle 公司，是一种非常优秀的开源关系型数据库管理系统，具有很高的性能和可靠性，可以满足各种应用场景的需求，是开发人员和企业用户的首选数据库之一。MySQL 支持多种数据类型，包括整数型、浮点型、字符型、日期型等，还支持多种数据操作，如增加、删除、修改、查询等。MySQL 还提供了多种安全机制，如密码保护、访问控制、数据加密等，可以保证数据库的安全性和可靠性[14,15]。MySQL 的主要特点如下：

(1) 开源：MySQL 是一个开源的数据库管理系统，可以免费下载、使用、修改和分发，用户可以自由地查看和修改源代码，也可以参与到 MySQL 的开发和维护中来。

(2) 高性能：MySQL 的性能非常优秀，具有高速的查询和处理能力，支持多种优化技术，如索引、缓存、分区等，可以提高数据库的性能和效率。

(3) 可靠性高：MySQL 具有很高的可靠性和稳定性，支持多种数据备份和恢复技术，可以避免数据丢失和损坏，保证数据的完整性和安全性。

(4) 易于使用和管理：MySQL 的安装、配置和使用非常简单，用户可以通过图形界面或命令行界面来管理和操作数据库，也可以通过 API 来进行编程和开发。

(5) 支持多种编程语言和操作系统：MySQL 支持多种编程语言和操作系统，如 Java、PHP、Python、C++、GO 等，可以方便地与其他应用程序进行集成和交互，也可以在多种操作系统上运行，如 Windows、Linux、Unix 等。

MySQL 的架构主要包括三个部分：客户端、服务端和存储引擎。

(1) 客户端：客户端是指连接 MySQL 服务器的应用程序，可以通过命令行界面、图形界面或 API 来访问和操作 MySQL 数据库。

(2) 服务端：服务端是指运行 MySQL 服务器的计算机，负责处理客户端的请求，管理 MySQL 数据库，提供数据存储和检索的服务。

(3) 存储引擎：存储引擎是 MySQL 的核心组件，负责存储和管理数据，可以选择不同的存储引擎来满足不同的需求，如 MyISAM、InnoDB、Memory 等[16]。

第七节 Redis 简介

Redis 是一个高性能、高可用性、可扩展性和灵活性的内存数据存储系统。它支持多种数据结构和高级功能，能够满足不同的应用需求[17,18]。Redis 的优点主要有以下几点：

(1) 高性能：Redis 是一个基于内存的数据存储系统，因此它的读写速度非常快。Redis 的所有操作都是原子性的，能够保证数据的一致性。

(2) 高可用性：Redis 支持主从复制和 Sentinel 集群模式，能够实现高可用性。当主节点宕机时，从节点会自动接管主节点的工作，保证系统的正常运行。

(3) 可扩展性：Redis 支持集群模式，能够实现数据的水平扩展。集群模式下，数据会被分散到多个节点上进行存储，能够提高系统的性能和可扩展性。

(4) 灵活性：Redis 支持多种数据结构，包括字符串、哈希、列表、集合、有序集合等。它还支持事务、发布/订阅、Lua 脚本等高级功能，能够满足不同的应用需求。

Redis 主要的应用场景：

(1) 缓存：Redis 能够快速地读取和写入数据，因此它非常适合作为缓存系统。将热点数据存储在 Redis 中，能够提高系统的性能和响应速度。

(2) 计数器：Redis 支持原子性操作，因此它非常适合作为计数器。将计数器存储在 Redis 中，能够实现高并发的计数功能。

(3) 分布式锁：Redis 支持分布式锁，能够实现多个节点之间的同步。将锁存储在 Redis 中，能够保证数据的一致性和可靠性。

(4) 消息队列：Redis 支持发布/订阅机制，能够实现消息队列功能。将消息存储在 Redis 中，能够实现高可靠性和高并发的消息传递。
(5) 地理位置：Redis 支持地理位置计算，能够实现附近人和地点搜索功能。将地理位置信息存储在 Redis 中，能够实现高性能和高可扩展性的位置搜索。

第三章 系统需求分析

第一节 系统可行性分析

可行性分析是系统开发中必要的一环，是指在系统开发前进行的必要的可行性评估，具体应包括经济、技术、操作环境等多方面的分析与评估，如果在评估过程中发现不满足某一环节，比如经济支出超出了预期，就应该重新进行方案评估并及时调整实施方案，所以说，可行性分析是十分必要的。系统开发前通过对各方面的预测分析，为今后的系统建设制定了指导方针，明确了系统的建设目标、功能范围，也间接地提高了软件的开发效率，减少了试错成本。在系统可行性评价标准中，技术可行性可以说是最核心的一项，因为即便在确定了具体需求的情况下，没有技术能力的支撑，即使有成本预算，想要开发出满足需要的系统也只能是一句空谈，所以本次可行性分析的关键就是能否应用本人熟悉的软件技术和理论知识开发出满足用户需求的功能。综上，本小节将从技术可行性、经济可行性和操作可行性三个方面对本系统进行可行性分析。

第二节 技术可行性分析

技术可行性分析是对系统的技术可行性进行评估，包括系统的设计、开发、测试和实施。评估系统的技术可行性可以确定系统是否能够实现预期的功能和性能要求。在技术可行性分析中，需要考虑以下几个方面：① 系统设计方案的可行性：对于系统的设计方案，需要评估其是否符合技术标准和规范，是否能够满足用户需求，是否具有可扩展性和可维护性等。② 系统开发技术的可行性：对于系统的开发技术，需要评估其是否能够满足系统设计方案的要求，是否具有高效性和可靠性等。③ 系统测试方案的可行性：对于系统的测试方案，需要评估其是否能够全面覆盖系统的功能和性能要求，是否能够发现系统中存在的缺陷和问题。④ 系统实施方案的可行性：对于系统的实施方案，需要评估其是否能够满足系统的上线需求，是否能够保证系统的稳定性和安全性等。通过对系统的技术可行性进行评估，可以确定系统是否具有实现的可能性，以及需要投入的技术资源和成本。本文基于 Java 所开发的古瓷器智能检索与知识科普平台，主要是为了方便人们更方便地了解到中国的古瓷器文化，同时也会遇到同样志同道合的人们，一块去深入了解中国文化。

在系统的设计和开发方面，主要面向广大用户为主，设计了需要有的很多功能，在技术层面，使用主流的 Springboot + Uni-app 进行开发，首先对于后端而言，Java 已经问世很久了，而且 Java 的生态也相当不错，再加上自从几年前 Spring 家族的出现，它就向它的名字一样，也是 Java 的春天，更加让 Java 变得广为人知，也使得后端开发变得相当容易，而本文所使用的 Springboot 就是 Spring 家族的一员，也是现在主流的后端框架之一，对于前端框架 Uni-app，虽然现在前端技术更新迭代都很快，但它也被大家广泛应用去做小程序端的界面，作为前后端的两个主流框架，开发相对而言会容易很多，同时它良好的生态，也让开发容易了很多，遇到问题也会有很多的解决方式。而且该系统不需要太多的算法和技术，因此在开发过程中更容易、更方便。总之，从技术上讲，该系统也是完全可行的。

第三节 经济可行性分析

经济可行性是指一个项目或计划在经济条件下是否可行。它通常包括一个全面的财务分析，以确定项目或计划的成本、收益、现金流量和利润潜力。经济可行性分析是企业在考虑投资或开展新项目之前必须进行的一项关键性工作，它可以帮助企业确定是否值得投资，以及投资的规模和时间表。在经济可行性分析中，还应考虑市场需求、竞争情况、技术可行性、政策法规等因素。而在本项目中，目前只需考虑研发成本即可，对于新用户，后端：一个 2 核 2G 的阿里云轻量应用服务器足 99 元/年，腾讯云 COS 存储桶 9 元/年，MySQL，Redis 数据库均使用云服务器上的即可，百度云 AI 新用户可以免费申请相似图片检索的图形库容量；前端：微信公众平台可以免费申请一个小程序。

综上所述，在经济成本方面来看，本系统的建设完全可行。

第四节 操作可行性分析

操作可行性分析是对系统的操作可行性进行评估，包括系统的易用性、可维护性、可扩展性等。评估系统的操作可行性可以确定系统是否能够被用户接受和维护。在操作可行性分析中，需要考虑以下几个方面：① 系统的易用性：对于系统的易用性，需要评估其是否能够满足用户的操作需求，是否具有友好的用户界面和操作流程等。② 系统的可维护性：对于系统的可维护性，需要评估其是否能够方便地进行系统的。

随着经济技术的快速发展，网络化，信息化已经成为人们生活中的一部分，现在，人们可以通过互联网轻松地获取各种信息，进行在线购物、支付、社交等活动。同时，各种数字化技术也使得人们的生活更加便利和智能化，例如智能家

居、智能手机等等。网络化和信息化的发展也促进了各个领域的数字化转型和升级，使得社会经济的发展更加快速和高效。本系统采用的是目前常见的小程序模式，用户只需通过搜索或扫码进入小程序，系统就 会以小程序的形式进行展示，在信息化的背景下，可以说，只要是了解基础的手机操作知识，就能很快地适应本系统的操作。并且本系统界面简洁美观，同时具有简单的导航栏，各功能模块都展现在导航栏上，只需点击即可跳转到所需要使用的模块中去，不涉及复杂的操作或使用流程，并且出现问题时会有弹窗提示来告诉用户，用户使用系统将会更加地方便快捷，从操作可行性来看，本系统使用简单方便，是完全可行的。

第五节 系统功能需求分析

古瓷器信息识别模块：是本系统一个重要的模块，这儿为了方便，使用百度云 AI 的构建的图形库以及相似图片识别进行瓷器检索功能的实现，用户可以通过手机拍照或者上传文件进行检索，进而可以得知该古瓷器的外观特征，生产朝代及其相关民间流传的美誉等。拍照的功能使用腾讯云 COS 存储桶进行上传图片到云端。

古瓷器科普模块：该模块对古瓷器按照出现朝代进行了分类：夏，商，周，春秋战国，唐，宋，元，明，清，同时用户也可以按照特征、朝代等信息作为关键词进行模糊查询搜索，从而了解中国悠久的古瓷器文化。

用户模块：现在几乎每个系统都会有登录注册的功能，同时这样的标识也可以作为用户的个性化，也更符合当今社会用户所喜爱的独特，也会记录用户对该系统的使用情况，与此同时也会给用户带来更好的体验感，用户通过邮箱去注册平台，也可以保证每个人账号的唯一性，也更方便用户记忆自己的账号，减少了账号丢失已经被盗的风险，当然了，为了更加满足大众的需求，并非所以的功能都只有登录后才可进行，比如拍照检索，知识科普平台，以及查看别人发布的帖子等就可以使用游客进行访问。

讨论区模块：帖子部分，用户可以发布属于自己的帖子，同时也可以对帖子进行收藏，评论，点赞和分享，浏览过后，也会有相关的记录，同时也会带动更多的人参与讨论。评论区部分，用户发布评论，点赞评论，以及通过递归查询获取贴子对应的父评论及其子评论列表。

第六节 系统性能需求分析

确定了系统功能模块需求之后，系统的建设方案也有了大致的目标，但是要达到系统正式运行的目的，还需要进行系统的性能需求分析，如果设计完成的系统不能满足市场实际需求，那么这个系统的开发便是毫无意义的，所以系统的性能需求分析也是十分重要的，系统良好的性能不仅是系统稳定运行的保障，也是

提高用户体验和系统竞争力的重要因素。系统的性能是与系统的安全性、可靠性、稳定性、实用性紧密关联的，以下是对这些指标的详细评价。系统的安全性：完整的系统运行需要数据的支撑，这些数据通常涉及用户的隐私信息，数据的安全是系统潜在用户能否转变为真正用户的主要决定因素之一，因此系统的开发者在设计之初就要考虑到系统中数据的安全性，这是系统能否成功的关键。在本系统中，用户的日常操作是满足安全需要的，首先 Java 作为本系统的开发语言，可以使系统的功能可行性更高。另外数据存储方面，MySQL 数据库作为一种成熟的数据库类型，拥有较高的数据安全性，Java 和 MySQL 技术的结合，是本系统安全性的重要保障。

系统的可靠性：正常使用的系统，对于用户提交的请求，反应应该是快速又准确的，不能出现不明原因的卡顿或者 bug，使用户能对需要的信息进行正确的查看和更改，要给予用户良好的交互感。本系统通过运用成熟的开发技术，保证了系统的可靠性要求，并且在系统完成后，对系统进行全面而系统的测试工作，即时发现系统存在的问题，保证系统的正常运行。

系统的健壮性和稳定性：健壮性和稳定性是用来判断该系统能否正常运转的关键性指标，当系统的某个功能模块出现问题，系统便会出现卡顿甚至无法运行的现象，所以各个功能模块都要保证是完好的，保障前台良好的交互和后台紧密的逻辑需要高质量的代码，本系统前端采用 Uni-app + uView 框架进行界面设计，提高了前端开发速度的同时也能保证界面的美观度。后端使用成熟的 Springboot + MyBatis-Plus + Shiro + MySQL + Redis 技术使后台的逻辑分工明确的同时，利用缓存提高了系统的性能，也数据流向更加的简洁明了。

第四章 系统设计及实现

第一节 数据库设计

数据库设计是系统设计的基础，系统产生的所有需要持久化存储的信息都要存储在数据库中。数据库设计是系统开发环节中极其重要的一环，数据库设计的目的是建立系统所需的数据库，数据库设计就是对数据库本身的设计，从系统的需求环境和实际出发，建立出符合系统所需的数据库，以满足系统和用户的需求。要根据用户的需求，在某一具体的数据库管理系统上，设计数据库的结构和建立数据库的过程。数据库系统需要操作系统的支持。数据库设计是建立数据库及其应用系统的技术，是信息系统开发和建设中的核心技术。由于数据库应用系统的复杂性，为了支持相关程序运行，数据库设计就变得异常复杂，因此最佳设计不可能一蹴而就，而只能是一种“反复探寻，逐步求精”的过程，也就是规划和结构化数据库中的数据对象以及这些数据对象之间关系的过程，它不仅涉及到业务规则和需求，还涉及到数据完整性、可靠性、性能、可维护性和可用性等诸多方面。比如，应该符合业务规则和需求：数据库设计应该能够支持业务规则和需求，以便能够有效地存储和检索数据，在进行数据库设计之前，需要对业务规则和需求进行详细的分析和了解，以便能够设计出符合业务规则和需求的数据库；应该具有可扩展性：数据库设计应该能够支持未来的扩展和变化，以便能够适应不断变化的业务需求；应该具有数据完整性：采用合适的数据类型、约束和规则，确保数据的准确性和一致性；应该具有可靠性：采用合适的备份和恢复策略，以确保数据的安全性和可靠性；应该具有高性能：考虑到数据的访问频率和数据量，以采用合适的索引和优化策略，以提高数据库的性能；应该具有可维护性：需要采用合适的命名规范和注释，以便能够方便地进行维护和管理；应该具有可用性：需要考虑到用户的使用习惯和需求，以设计出易于使用和访问的数据库[19]。

数据库的三大范式是指数据库设计中的三个规范化等级。这三个规范化等级为第一范式、第二范式和第三范式，分别对应着不同的数据结构和数据关系的要求：

(1) 第一范式（1NF）：第一范式要求关系型数据库中的每个属性都应该是原子性的，即不可再分的。也就是说，每个属性都应该是不可再分的最小数据单元。例如，一个人的姓名、性别、出生日期等属性都应该是原子性的，而不应该拆分成多个属性。

(2) 第二范式（2NF）：第二范式要求关系型数据库中的每个非主属性都应该完全依赖于主键，而不是依赖于主键的一部分。也就是说，每个非主

属性都应该与主键有一个完整的依赖关系。例如，一个订单的商品名称、商品数量、商品单价等属性都应该与订单号有一个完整的依赖关系，而不应该仅仅与订单号的一部分有关。

(3) 第三范式（3NF）：第三范式要求关系型数据库中的每个非主属性都不能依赖于其他非主属性，即不存在传递依赖关系。也就是说，每个非主属性都应该与主键有一个直接的依赖关系。例如，一个订单的收货地址和收货人姓名等属性应该与订单号直接相关，而不应该通过商品名称等属性间接关联。

该项目的数据库设计如下表所示：

列名	数据类型	长度	类型	NULL	说明
id	bigint		主键	×	帖子 id
title	varchar	512		√	帖子标题
type	varchar	64		√	类型
content	text			√	内容
user_id	bigint			√	帖子的用户 id
gmt_create	datetime			√	创建日期
gmt_modified	datetime			√	修改日期
is_deleted	tinyint			√	逻辑删除

表 4-1 article（帖子表）

列名	数据类型	长度	类型	NULL	说明
id	bigint		主键	×	评论 id
article_id	bigint			√	帖子 id
parent_id	bigint			√	父评论 id
user_id	bigint			√	用户 id
content	text			√	评论内容
gmt_create	datetime			√	创建日期
gmt_modified	datetime			√	修改日期
is_deleted	tinyint			√	逻辑删除

表 4-2 comment（评论表）

列名	数据类型	长度	类型	NULL	说明
id	bigint		主键	×	瓷器 id
title	varchar	512		√	瓷器标题
type	varchar	64		√	类型
dynasty	varchar	32		√	朝代
url	varchar	1024		√	相关图片
introduction	text			√	介绍
detail	text			√	细节描述
gmt_create	datetime			√	创建日期
gmt_modified	datetime			√	修改日期
is_deleted	tinyint			√	逻辑删除

表 4-3 porcelain（瓷器表）

列名	数据类型	长度	类型	NULL	说明
id	bigint		主键	×	用户 id
avatar	varchar	512		√	头像
username	varchar	256		√	用户名
password	varchar	256		√	密码
email	varchar	256		√	邮箱
signature	varchar	1024		√	个性签名
gmt_create	datetime			√	创建日期
gmt_modified	datetime			√	修改日期
is_deleted	tinyint			√	逻辑删除

表 4-4 user（用户表）

列名	数据类型	长度	类型	NULL	说明
user_id	bigint		主键	×	用户 id
token	varchar	100		×	token
identity	int			√	用户身份 1：user
expire_time	datetime			√	过期时间
update_time	datetime			√	更新时间

表 4-5 sys_user_token（token 表）

第二节 用户模块的设计与实现

一、 注册

本系统注册为了方便用户记忆平台账号，同时也为了账号的安全性进行考虑，最终决定采用邮箱的方式进行注册，平台账号即为邮箱号，主要流程为：

1. 用户通过平台发送瓷器验证码到自己邮箱。此过程中 Redis 通过 k-v 键值对将验证码存入缓存，k 即为邮箱账号，v 为具体的验证码，设定五分钟后自动删除。

```
public R sendCode(UserEntity user) {
    if (redisUtil.get(RedisKeys.getEmailKey(user.getEmail())) != null) {
        return R.error("验证码已发送，请注意查收...");
    }
    String code = CharUtil.getRandomString(6);
    redisUtil.setEx(RedisKeys.getEmailKey(user.getEmail()), code, 5,
TimeUnit.MINUTES);
    emailUtil.send(user.getEmail(), "瓷器系统邮箱验证码", "您收到的邮箱验证码
为: " + code + "，有效时常五分钟...");
    return R.ok();
}
```

2. 用户给小程序界面输入注册的具体信息，用户名，邮箱，密码，验证码。

3. 用户点击登陆，前端像后端发起注册请求，后端进行一系列逻辑处理，先对验证码是否有误，如果正确则判断该邮箱之前是否有注册过账号，注册过则给出提示，未注册过则注册成功，并清空缓存，设置账号的初始化数据。

```
public R register(UserPO po) {
    if (!po.getCode().equals(redisUtil.get(RedisKeys.getEmailKey(po.getEmail())))) {
        return R.error("验证码输入有误...");
    }
    if (getOne(new LambdaQueryWrapper<UserEntity>().eq(UserEntity::getEmail,
po.getEmail())) != null) {
        return R.error("该邮箱已注册...");
    }
```

```
    redisUtil.delete(RedisKeys.getEmailKey(po.getEmail()));
    UserEntity user = new UserEntity().setEmail(po.getEmail()).setPassword(new
Sha256Hash(po.getPassword()).toHex()).setUsername(po.getUsername());
    save(user);

    redisUtil.set(RedisKeys.getUserVisitCntKey(user.getId()), 0);
    redisUtil.set(RedisKeys.getUserLoveCntKey(user.getId()), 0);
    return R.ok();
}
```

二、 登陆

对于前后端分离的项目而言，默认会让 session 会话失效，这儿有两种解决办法，一种是配置跨域的时候配置允许 session ，另一种则是直接改用 token，那么 session 和 token 分别是什么呢？session 和 token 都是常见的身份认证和授权机制，session 存储在服务器端，而 token 存储在客户端(因此，从安全性角度来看，session 比 token 更安全，因为 Session ID 只存储在服务器端，不会被攻击者获取。而 token 存储在客户端，如果被攻击者获取，就可能被篡改。)，通常是在浏览器的 cookie 中,session 是有状态的，需要在服务器端存储相关信息，而 token 是无状态的，不需要在服务器端存储任何信息。针对跨域问题，session 在跨域时会遇到问题，需要额外处理；而 token 可以在不同域名下进行身份认证和授权。但相比 session，token 的实现复杂度更高，需要考虑加密、签名、刷新等问题[20,21]。

综上所述，session 和 token 都有自己的优缺点，选择哪种方式取决于具体的应用场景和需求。如果安全性要求较高，可以选择 session，如果需要跨域使用或无状态认证，可以选择 token。

本系统我们选用第二种方法，使用 token 进行用户的身份认证，主要流程为：

1. 用户在登陆界面输入邮箱和密码。

2. 用户点击登陆。前端向后端发起登陆请求，后端通过查询 MySQL 数据库对其进行用户账号密码的校验，如果校验未通过，即数据库不存在这个账号或者密码输入有误，则告诉前端 403 状态码（登陆失败），如果登陆成功，则生成 token 签证，并将其签证和过期时间存入数据库，然后再将 token 及其相关信息返回给前端，前端之后每次请求后端接口都在请求头携带 token，用于之后每次请求的认证校验。

```
// 登陆
public R login(UserPO po) {
    UserEntity one = getOne(new LambdaQueryWrapper<UserEntity>()
            .eq(UserEntity::getEmail, po.getEmail())
            .eq(UserEntity::getPassword, new
Sha256Hash(po.getPassword()).toHex())
    );
    if (one == null) {
        return R.error(403, "邮箱或密码输入有误，请重新输入...");
    }
    return sysUserTokenService.createToken(one.getId(), 1);
}
// 生成 token 签证
public R createToken(long userId, int identity) {
    // 生成一个 token
    String token = TokenGenerator.generateValue();
    //当前时间
    Date now = new Date();
    // 过期时间
    Date expireTime = new Date(now.getTime() + EXPIRE * 1000);
    //判断是否生成过 token
    SysUserTokenEntity tokenEntity = getOne(new
LambdaQueryWrapper<SysUserTokenEntity>().eq(SysUserTokenEntity::getUserId,
userId).eq(SysUserTokenEntity:
    if(tokenEntity == null){
        tokenEntity = new SysUserTokenEntity();
        tokenEntity.setUserId(userId);
        tokenEntity.setToken(token);
        tokenEntity.setUpdateTime(now);
        tokenEntity.setExpireTime(expireTime);
        tokenEntity.setIdentity(identity);
        //保存 token
        this.save(tokenEntity);
    }else{
        //更新 token
```

```
            update(new LambdaUpdateWrapper<SysUserTokenEntity>()
                    .eq(SysUserTokenEntity::getUserId, userId)
                    .eq(SysUserTokenEntity::getIdentity, identity)
                    .set(SysUserTokenEntity::getToken, token)
                    .set(SysUserTokenEntity::getUpdateTime, now)
                    .set(SysUserTokenEntity::getExpireTime, expireTime)
            );
        }
        R r = R.ok().put("token", token).put("expire", EXPIRE);
        return r;
    }
```

三、 用户个人信息

这儿为了保持系统的可扩展性，方便以后给系统加入新功能，这儿除了使用了 Shiro 的认证过滤器链，还通过 ThreadLocal + 拦截器进行登陆的用户信息的拦截，方便后续接口直接获取，而无需通过过滤器链一层一层地传递[22]。

ThreadLocal 相关介绍：ThreadLocal 是 Java 中的一个线程级别的变量，它允许你在每个线程中存储和访问数据，而不必担心线程之间的数据共享问题。在多线程编程中，线程安全是一个非常重要的问题。在某些情况下，为了保证线程安全，我们需要使用锁或同步机制，但这会导致性能问题和代码复杂度增加。ThreadLocal 可以帮助我们避免这些问题，提高程序的性能和可读性。ThreadLocal 的使用方法很简单，只需要在每个线程中创建一个 ThreadLocal 对象，然后调用它的 set() 方法存储数据，调用 get() 方法获取数据即可。每个线程都有自己的 ThreadLocal 对象，数据存储在该对象中，不会被其他线程访问到。因此，不需要考虑线程安全问题，也不需要使用锁或同步机制。ThreadLocal 的实现原理是使用一个 Map 来存储每个线程的数据，其中 Map 的 key 是线程对象，value 是数据对象。当调用 get() 方法时，ThreadLocal 会获取当前线程对象，并使用它从 Map 中获取数据对象。当调用 set() 方法时，ThreadLocal 会获取当前线程对象，并将数据对象存储到 Map 中。由于每个线程都有自己的 Map 对象，在多线程并发访问时不会出现线程安全问题。ThreadLocal 的优点有很多，比如，使用方法简单明了，不需要复杂的同步机制，代码可读性高；每个线程都有自己的数据存储区域，不会出现线程安全问题；不需要使用锁或同步机制，因此性能比使用锁要高。毕竟有得必有失，它也总不是完美无缺的，比如，容易导致内存泄漏（由于 ThreadLocal 存储的数据只有在当前线程中使用时才有意义，因此需要及时清理不再使用的数据。如果不注意清理，可能会导致内存泄

漏。）；不支持继承（ThreadLocal 存储的数据只在当前线程中有效，不支持线程之间的数据传递。如果需要在多个线程之间共享数据，需要使用其他机制，例如共享变量或消息队列等。）

```
// ThreadLocal 定义
@Component
public class UserThreadLocal {
    private static final ThreadLocal<Long> threadLocal = new ThreadLocal<>();
    public void set(Long userId) {
        threadLocal.set(userId);
    }
    public Long get() {
        return threadLocal.get();
    }
    public void remove() {
        threadLocal.remove();
    }
}
```

拦截器简单介绍：拦截器（Interceptor）是一种 AOP（面向切面编程）技术，可以在程序执行过程中对方法进行拦截和处理。在 Java Web 开发中，拦截器通常用于实现权限控制、日志记录、性能监控等功能。拦截器可以在方法执行前、执行后或异常抛出时执行一些额外的逻辑处理。在 Spring 框架中，拦截器是通过实现 HandlerInterceptor 接口来实现的。该接口定义了三个方法：preHandle()、postHandle() 和 afterCompletion()。其中，preHandle() 方法在方法执行前被调用，可以进行一些前置处理；postHandle() 方法在方法执行后被调用，可以进行一些后置处理；afterCompletion() 方法在视图渲染完成后被调用，可以进行一些资源清理等操作。拦截器的使用方法比较简单，只需要在 Spring 配置文件中配置拦截器，并指定需要拦截的 URL 或 Controller 即可[23]。

```
// 用户拦截器定义
@Component
@Slf4j
public class UserInterceptor implements HandlerInterceptor {

    @Autowired
    private UserThreadLocal userThreadLocal;
```

```
    @Autowired
    private SysUserTokenService sysUserTokenService;

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
        //从 header 中获取 token
        String token = request.getHeader("token");
        if (StrUtil.isNotBlank(token)) {
            SysUserTokenEntity tokenEntity = sysUserTokenService.getOne(
                    new LambdaQueryWrapper<SysUserTokenEntity>()
                            .eq(SysUserTokenEntity::getToken, token)
                            .eq(SysUserTokenEntity::getIdentity, 1)
            );
            if (!ObjectUtil.isEmpty(tokenEntity)) {
                userThreadLocal.set(tokenEntity.getUserId());
            }
        }
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {
        HandlerInterceptor.super.postHandle(request, response, handler, modelAndView);
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {
        userThreadLocal.remove();
        HandlerInterceptor.super.afterCompletion(request, response, handler, ex);
    }
```

```
}

// 获取当前登陆的用户信息
// 通过重写 getById() 方法，判断当前前端需要展示的关注状态及其各种缓存信息
public UserEntity getById(Serializable id) {
    return super.getById(id)
            .setFollowCnt(redisUtil.sSize(RedisKeys.getUserFollowKey(id)))
            .setFanCnt(redisUtil.sSize(RedisKeys.getUserFanKey(id)))
            .setVisitCnt(Long.parseLong(redisUtil.get(RedisKeys.getUserVisitCnt
Key(id)).toString()))
            .setLoveCnt(Long.parseLong(redisUtil.get(RedisKeys.getUserLoveCnt
Key(id)).toString()))
            .setFollowStatus(userThreadLocal.get() == null ? 0 :
(userThreadLocal.get().equals(id) ||
redisUtil.sIsMember(RedisKeys.getUserFollowKey(userThreadLocal.get()), id) ? 1 :
0))
            .setArticleList(articleService.list(new HashMap<String, Object>() {{
                put("userId", id);
            }}));
}
@Override
public R info() {
    return R.ok().put("user", getById(userThreadLocal.get()));
}
```

第三节 瓷器智能检索的设计与实现

百度云AI是百度云基于人工智能技术推出的一系列AI服务，涵盖语音识别、自然语言处理、图像识别、机器翻译、人脸识别等多个领域。百度云AI提供了API接口和SDK开发包，方便开发者将AI技术集成到自己的应用中，并提供了自己的AI应用，如人脸识别门禁、智能客服等，为用户提供更加智能的服务。该模块主要借用了百度云 AI 的相似图片检索功能，对用户提供的目标图片与现

有的图形库进行比对，该平台通过底层机器学习深度学习计算后会得到相似的最高的瓷器 id，然后通过查询 MySQL 数据库，进而获得相似的最高的完整的瓷器信息。

```
// 1. 获取凭证 token
private String getToken() {
    // 获取 token 地址
    String authHost = "https://aip.baidubce.com/oauth/2.0/token?";
    String getAccessTokenUrl = authHost
            // 1. grant_type 为固定参数
            + "grant_type=client_credentials"
            // 2. 官网获取的 API Key
            + "&client_id=" + imageConfig.getApiKey()
            // 3. 官网获取的 Secret Key
            + "&client_secret=" + imageConfig.getSecretKey();
    try {
        URL realUrl = new URL(getAccessTokenUrl);
        // 打开和 URL 之间的连接
        HttpURLConnection connection = (HttpURLConnection) realUrl.openConnection();
        connection.setRequestMethod("GET");
        connection.connect();
        // 获取所有响应头字段
        // 定义 BufferedReader 输入流来读取 URL 的响应
        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String result = "";
        String line;
        while ((line = in.readLine()) != null) {
            result += line;
        }
        JSONObject jsonObject = JSONObject.parseObject(result);
        return jsonObject.getString("access_token");
    } catch (Exception e) {
        log.error("获取图形库 token 失败！");
```

```
        }
        return null;
    }
    // 2. 形似度检索
    public R check(UrlPO po) {
        String url =
    "https://aip.baidubce.com/rest/2.0/image-classify/v1/realtime_search/similar/search";
        try {
            String data = "url=" + po.getUrl();
            String resultStr = HttpUtil.post(url, getToken(), data);
            AIEntity aiEntity = JSONObject.parseObject(resultStr, AIEntity.class);
            AIEntity.AI ai = aiEntity.getResult().get(0);
            return R.ok().put("res", porcelainService
                    .getById(Long.parseLong(ai.getBrief().toString()))
            );
        } catch (Exception e) {
            log.error(e.getMessage());
        }
        return R.error();
    }
```

腾讯云 COS 存储简介：腾讯云 COS（Cloud Object Storage）是腾讯云推出的一种对象存储服务，可以存储多种类型的数据，包括文本、图片、音视频等。COS 存储采用分布式架构，可以实现高可用性、高可靠性、高扩展性、高并发性等特点，同时支持多种接口和协议，如 HTTP、HTTPS、COS SDK 等，方便开发者进行数据的上传、下载和管理。COS 存储提供了多种存储类型，包括标准存储、低频存储、归档存储等。标准存储适用于频繁访问的数据，低频存储适用于不经常访问但需要长期保存的数据，归档存储适用于长期不需要访问但需要长期保存的数据。不同的存储类型有不同的数据访问费用和存储费用，用户可以根据自己的需求选择合适的存储类型。COS 存储还提供了多种安全性措施，如数据加密、访问控制、防盗链等，保障数据的安全性和隐私性。用户可以对数据进行加密存储，只有授权的用户才能访问数据，可以通过访问控制策略来控制用户的访问权限，可以通过防盗链来防止数据被非法盗链。COS 存储还支持数据的自动化管理，包括数据迁移、数据备份、数据恢复等。用户可以将数据从其他

存储系统迁移到 COS 存储，也可以将 COS 存储中的数据备份到其他存储系统，还可以通过快速恢复功能来快速恢复数据。

```
// 上传多文件到 COS 存储桶
public R upload(MultipartFile[] fileArr) {
    JSONArray fileList = new JSONArray();
    Arrays.stream(fileArr).forEach(file -> {
        // 实例化 json 数据
        JSONObject jo = new JSONObject();
        String originalFilename = file.getOriginalFilename();
        String bucketName = cosConfig.getBucketName();
        String key = cosConfig.getPrefix()
                + UUID.randomUUID().toString().replaceAll("-", "")
                + originalFilename.substring(originalFilename.lastIndexOf('.'));
        try {
            //将 MultipartFile 类型 转为 File 类型
            File localFile = File.createTempFile("temp", null);
            file.transferTo(localFile);
            cosClient.putObject(new PutObjectRequest(bucketName, key,
localFile));
            jo.put("fileUrl", cosConfig.getUrl() + key);
            jo.put("originalFilename", originalFilename);
            fileList.add(jo);
        } catch (Exception e) {
            throw new GlobalException("文件上传失败...");
        }
    });
    // 返回 json
    return R.ok("文件上传成功...").put("fileList" ,fileList);
}
```

第四节 瓷器科普平台的设计与实现

该模块主要用于瓷器信息的分类展示以及模糊搜索，从请求中拿到前端传入的参数，然后使用 MyBatis-Plus 做分页+ 模糊查询即可，其实这儿主要涉及了怎么用 mp 写 SQL 句子的子查询[24]。

```
public R list(Map<String, Object> params) {
    return R.ok().put("list", list(new LambdaQueryWrapper<PorcelainEntity>()
                    .eq(ObjectUtil.isNotEmpty(params.get("name"))
&& !params.get("name").toString().equals("全部"), PorcelainEntity::getDynasty,
params.get("name"))
                    .and(
                            ObjectUtil.isNotEmpty(params.get("key")),
                            i -> i.like(PorcelainEntity::getTitle,
params.get("key"))
                                    .or()
                                    .like(PorcelainEntity::getDynasty,
params.get("key"))
                                    .or()
                                    .like(PorcelainEntity::getIntroduction,
params.get("key"))
                                    .or()
                                    .like(PorcelainEntity::getDetail,
params.get("key"))
                    ).orderByDesc(PorcelainEntity::getGmtCreate)
            )
    );
}
```

第五节 讨论区模块的设计与实现

该模块主要就是用户可以对帖子进行点赞、评论、分享。点赞这里使用 Redis 中的 set 这一数据结构来进行存储，用于存储点赞过该篇帖子的用户集，在展示该帖子的时候，可以根据 sSize()方法来获取该篇帖子的点赞数，同时如果当前是登陆状态（即请求头中的 token 不等于空且没过期）时，可以通过 isMember() 来判断当前用户是否对该帖子进行了点赞，否则为游客账号，一律显示未点赞的样式即可。当用户点赞的时候，如果未登陆则直接跳转登录页（前端通过封装全局 axios 来实现页面的跳转，后端通过全局拦截器来进行拦截，并告诉前端当前操作必须要先登陆认证）。如果是登陆状态，则可以按着前端传递给后端当前的点赞状态来判断当前操作是点赞还是取消点赞。评论部分，若直接对帖子进行评论，则该评论为父评论，该条记录的 parent_id 为 null 或者‘’，如果是对评论进行评论就会出现父、子评论查询的问题，这儿选择先用 article_id 查询获取该篇帖子所有的评论，然后使用递归将父子评论进行组合，将最终结果返回给前端。 帖子分享部分，使用微信小程序提供的 onShareAppMessage 事件，分享按钮时触发，返回一个对象，用于自定义分享内容[25]。

1. 前端封装 axios 部分：

```
export const baseURL = "http://182.92.3.1:9979/porcelain"
```

```
export const request = (options) => {
    return new Promise((resolve, reject) => {
        uni.request({
            url: baseURL + options.url,
            method: options.method || 'GET',
            data: options.data || {},
            header: {
                token: uni.getStorageSync('token')
            },
            success: (res) => {
                if (res == '') {
                    return uni.showToast({
                        icon: 'loading',
                        title: '获取数据失败'
                    })
                }
                if (res.data.code == 401 || res.data.code == 403) {
                    uni.navigateTo({
                        url: '/pages/login/login'
                    })
                }
                resolve(res)
            },
            fail: (err) => {
                return uni.showToast({
                    icon: 'loading',
                    title: '请求失败'
                })
                reject(err)
            }
        })
    })
}
```

2. 后端拦截器部分参照本章第二节用户模块的设计与实现。

3. 父、子评论的递归查询

```
private void getChildCommentList(Map<Long, CommentEntity> all, List<CommentEntity> childCommentList, Long parentCommentId) {
    if (parentCommentId == null) {
        return;
    }
    List<CommentEntity> list = all.values().stream().filter(comment -> parentCommentId.equals(comment.getParentId())).collect(Collectors.toList());
    childCommentList.addAll(list);
    list.forEach(comment -> getChildCommentList(all, childCommentList, comment.getId()));
}
```

4. 帖子分享

```
<button :data-id='article.id' open-type="share" style="
margin: 0;
padding: 0;
border: none !important;
outline: none;
border-radius: 0;
background-color: transparent;
line-height: inherit;
">
        <u-icon name="share-square" size="40rpx"></u-icon>
</button>

onShareAppMessage: function (options) {
    if (options.from === 'button') {
        return {
          title: '讨论',
          path: '/pages/talk-in/talk-in?id=' + options.target.dataset.id,
          // imageUrl:goods_img //不设置则默认为当前页面的截图
        }
    }
},
```

第五章 系统测试

测试是指在软件开发过程中，通过一系列的测试活动来发现和解决软件中存在的问题和缺陷，以提高软件的质量和可靠性。测试是软件开发过程中非常重要和关键的环节。

测试可以分为多种类型，包括单元测试、集成测试、系统测试、验收测试等。不同类型的测试针对不同的测试对象和测试目标，采用不同的测试方法和技术，以全面、有效地发现和解决软件中存在的问题和缺陷。

测试的过程包括测试计划、测试设计、测试执行和测试报告等步骤。在测试计划阶段，我们需要制定测试计划和测试策略，确定测试的范围、目标和方法等。同时也需要根据测试计划和需求文档等，设计测试用例和测试场景，以此来检测软件的各项功能和性能等方面的问题。在测试执行阶段，测试人员需要按照测试用例和测试场景，对软件进行全面、有效的测试，以发现和解决可能存在的问题和缺陷。在测试报告阶段，测试人员需要整理测试结果，编写测试报告，向开发人员和用户反馈测试结果，以此来改进软件的质量和可靠性。

图 5-1 系统封面图

第一节 登录注册模块测试

在本模块的测试中，主要验证系统能否确定用户输入的信息与数据库是否相匹配。

一、登陆模块

用户进入登陆界面后，如果输入正确的用户名密码，用户登录成功，告诉用户登陆成功并跳转到小程序首页。如果输入错误的用户名密码，给用户提示输入有误，停留在登陆界面。

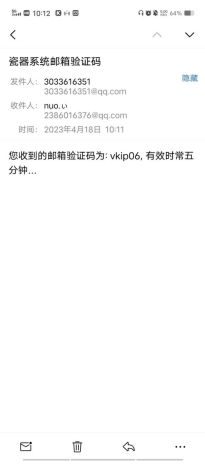
图 5-2-1 登陆成功

图 5-2-2 登陆失败

二、注册模块

用户进入注册界面后，需要先输入注册信息，然后通过输入的邮箱获取邮箱验证码，填写验证码后即可点击注册，完成用户注册功能，如果验证码有误或者邮箱已注册或者密码等格式有误，则停留在注册界面，如果一切都满足，则注册成功，跳转到登陆界面。

图 5-3-1 发送邮箱验证码



5-3-2 用户接收到验证码

图 5-3-3 注册成功跳转登陆界面

图 5-3-4 注册失败界面

第二节 瓷器检索模块的测试

用户通过拍照或者上传图片，通过检索瓷器信息科获取最匹配的瓷器信息。

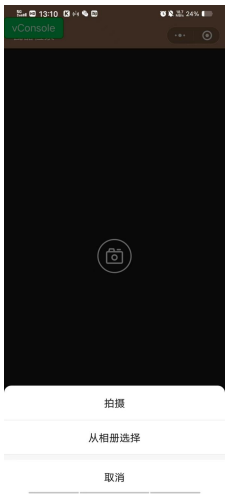


图 5-4-1 点击图片准谱拍照

图 5-4-2 拍照

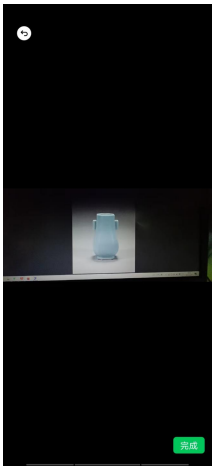


图 5-4-3 上传

图 5-4-4 获取信息

第三节 瓷器科普平台的测试

进入系统后，首先获取到所有瓷器分类及其列表，然后，用户可以通过模糊查询或者分类去获取相关的瓷器信息。

图 5-5-1 瓷器列表

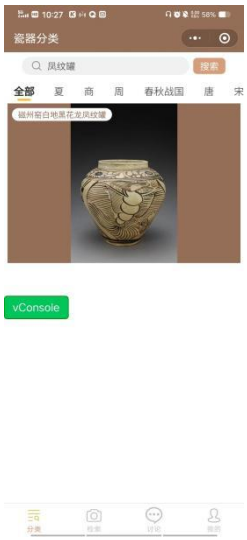


图 5-5-2 瓷器查询

图 5-5-3 通过瓷器分类检索

图 5-5-4 瓷器详细信息

第四节 帖子讨论区的测试

获取全部帖子列表，点击点赞图标，即可对帖子进行点赞，如果重复点击，则会取消点赞。点击分享按钮，即可完成帖子的分享。点击帖子，进入帖子详情

页，点击即回复即可对帖子或评论进行评论，点击点赞，即可对帖子的评论进行点赞。

图 5-6-1 对帖子进行点赞

图 5-6-2 点赞成功

图 5-6-3 对帖子取消点赞

图 5-6-4 取消点赞成功

图 5-6-5 分享帖子

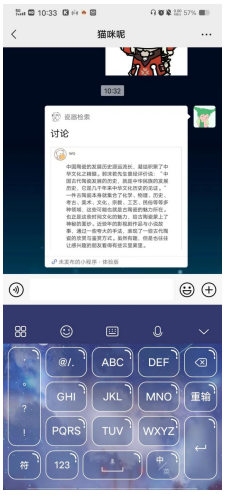


图 5-6-6 帖子分享成功

图 5-6-7 帖子详情界面

图 5-6-8 评论

5-6-6 帖子评论成功

图 5-6-10 评论点赞成功

第六章 总结和展望

古瓷器是中国传统文化的重要组成部分，具有极高的历史价值、文化价值和收藏价值。然而，由于其数量众多、种类繁多，对于广大古瓷器爱好者来说，要想全面了解古瓷器的相关知识，是一项非常艰巨的任务。本文所研究的古瓷器智能检索与知识科普平台就可以使人们更加容易地了解到古瓷器文化。

对于本系统的几个核心模块，设计之初，斟酌了很久点赞收藏等相关信息存储，了解过 Redis 后，比较了 MySQL 和 Redis 的异同，最终选取了缓存数据库 Redis，高效而且不需要去担心并发的问题，串行执行，避免了加锁的操作，从而提高了系统的性能，瓷器的智能检索这儿为了方便调用第三方接口来实现。

开发初期对于很多业务场景和都不是很熟悉，也遇见了很多问题，但发现问题并解决问题也是自我提升的过程，随着不断的摸索自己解决问题的能力也不断提高，也能够灵活运用所有知识，开发过程也变得越来越得心应手，这个成长过程对我日后的学习和工作都是十分有益的。

经过真机启动运行本文设计的古瓷器智能检索与知识科普平台，且已经可以达到预期的设计目标，但由于本人目前学习方向是 Java 后端开发，对于前端的一些交互不是很熟悉，且开发时间也比较紧迫，系统功能还有许多需要完善的地方：

(1) 本系统的前端交互做的不够完美，尚有部分功能页还需完善。

(2) 管理系统尚不完善，目前只可以通过接口文档手动调用平台接口实现对系统的管理。

(3) 平台现有的瓷器信息和智能识别的图形库也尚不完整。

(4) 未来利用推荐算法改善瓷器的搜索结果。

(5) 瓷器智能检索的完善。

上述问题既是系统的不足，也是系统未来要改善的方向，对于未来，本系统将会实现更加高效完善的功能模块，提供更加具有互动性的使用体验，系统运行能更加稳定，成为一个成熟的产品。

参考文献

[1] 靳晶晶,王佩. 基于卷积神经网络的图像识别算法研究[J]. 通信与信息技术,2022,(02):76-81.

[2] 翁政魁,管业鹏,罗宏杰.基于机器视觉古陶瓷无损分类识别[J].硅酸盐学报,2017,45(12):1833-1842.

[3] Jan Steinbrener,Konstantin Posch,Raimund Leitner. Hyperspectral fruit and vegetable classification using convolutional neural networks[J]. Computers and Electronics in Agriculture,2019,162(C):364-375.

[4] Zhou Lijie,Yu Weihai. Improved Convolutional Neural Image Recognition Algorithm based on LeNet-5[J]. Journal of Computer Networks and Communications,2022,2022(1630203).

[5] 杨礼坤. 基于机器视觉的瓷器检测分类系统研究与实现[D].杭州:杭州电子科技大学,2014.

[6] 邱元鑫. 基于图像匹配和区块链存储的陶瓷认证方法的研究[D].闽南：闽南师范大学,2022.

[7] 罗宏杰,杨云,王芬,苗鸿雁,梁宝鎏. 不同历史时期耀州窑碗器型结构特征之研究[J].中国陶瓷工业,2003,(06):1-4.

[8] 穆天红.基于人工智能的古陶瓷器型和纹饰图像特征识别研究[D].陕西:陕西科技大学,2020.

[9] Paul Krill.Oracle Java popularity sliding, New Relic reports.InfoWorld.com,2022.

[10] 杨云,郭立文.中国古陶瓷器型结构数据库的建立[J].陶瓷学报,2005,(01):53-56.

[11] 周建辉.基于 uni-app 的场馆预约微信小程序的设计与开发[J].江苏工程职业技术学院学报,2022,22(04):7-11.

[12] 谢志妮.基于uni-app的微信小程序关键技术运用[J].电子技术与软件工程,2021,(12):32-33.

[13] 欧阳宏基,葛萌,程海波.MyBatis 框架在数据持久层中的应用研究[J].微型电脑应用,2023,39(01):73-75.

[14] Ghalem Belalem,Houcine Matallah,Karim Bouamrane.Evaluation of NoSQL Databases: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB[J]. International Journal of Software Science and Computational Intelligence (IJSSCI),2020,4(12):71-91.

[15] Jian Chen,Chen Jian,Pan Hailan.Design of Man Hour Management Information System on SpringBoot Framework[J].Journal of Physics: Conference Series,2020,1(1646):012136-.

[16] Wu Daiwen.The Application and Management System of Scientific Research Projects Based on PHP and MySQL[J].Journal of Interconnection Networks,2022,22(02).

[17] Taufik,Bramantyo Adhilaksono,Faried Effendy.Performance Comparison of Web Backend and Database: A Case Study of Node.JS, Golang and MySQL, Mongo DB[J].Recent Advances in Computer Science and Communications,2021,14(6):195-1961.

[18] Guy Harrison.Redis and Amazon's MemoryDB[J].Database Trends and Applications,2021, 5(35):30-30.

[19] Benymol Jose,Sajimon Abraham.Performance analysis of NoSQL and relational databases with MongoDB and MySQL[J].Materials Today: Proceedings.2020,3(24):2036-2043.

[20] 王志亮, 纪松波. 基于 SpringBoot 的 Web 前端与数据库的接口设计[J]. 工业控制计算机, 2023, 36(03):51-53.

[21] 朱志慧, 蔡洁. 基于 SpringBoot+Vue+Uni-app 框架的校园失物招领系统[J]. 电子技术与软件工程, 2022(17):62-65.

[22] 齐善鲁, 马德俊, 梁雪. 基于SpringBoot的开放式软件开发案例教学平台设计[J]. 电脑知识与技术, 2021, 17(28):71-73.

[23] 喻佳, 吴丹新. 基于 SpringBoot 的 Web 快速开发框架[J]. 电脑编程技巧与维护, 2021(09):31-33.

[24] Yang Nan,Wang Zhiyi,Wang Shihao. Computer Image Recognition Technology and Application Analysis[J]. IOP Conference Series: Earth and Environmental Science, 2021, 769(032065).

[25] Lei Liu.Design and Implementation of a High Performance Red Packet Rushing System[J].Advances in Computer and Communication,2023,(3)

致谢

随着这纸论文的完成，在这篇论文的结尾写下“致谢”二字，也意味着我四年的大学生活也即将告一段落了，我也即将踏上新的征程，很高兴在这段时光遇见了许许多多的人，也见证了许许多多的事，或美好愉悦，亦或难过焦虑，同时，我也在慢慢的成长。

首先，我要感谢我的导师王嫄老师。在我大学四年的学习生涯中，我的导师一直是我学术生涯上的指导者和引路人。在我的毕业论文写作过程中，我的导师给予了我无微不至的指导和帮助，从选题、研究方法到论文撰写，她都给了我宝贵的建议和指导。她的耐心和细心让我受益终身。在此，我要对导师的教诲和关心表示最真挚的感谢。

其次，我要感谢我的学校。在我四年的大学生涯中，学校为我提供了一个优良的学习环境和广阔的发展空间。在学校的指导下，我不仅学到了专业知识，还培养了批判性思维和创新精神。在学校的帮助下，我参加了各种社团和活动，丰富了自己的课余生活，结交了许多志同道合的朋友。在此，我要对学校的培养和关爱表示最真挚的感谢。

最后，我要感谢我的家人和朋友。感谢他们在我学习和生活中的支持和鼓励，他们的陪伴和关爱让我能够克服各种困难和挑战，迎接未来的挑战。

在大学四年的学习生涯中，我收获了许多宝贵的经验和知识。我将永远铭记导师和学校的培养和关爱，感恩家人和朋友的支持和鼓励。在未来的工作和学习中，我将继续努力，不断前进，相信守得云开见月明！祝大家前程似锦，万事顺遂！