个人博客系统

软件需求规格说明（SRS）

教师：余仲星

队员：戴梦琪 张政航 许嘉诺

郑春英 姚佳悦

日期：2025/5/15

目录

1 范围

1.1 标识

名称：新迹（NexTecht）

标识号：BT 1.0

版本号：Version 1.0

发行号：XX-XX-XX-XX

适用系统：具有 web 的系统

1.2 系统概述

（1）适用系统：具有 web 的系统。

（2）用途：本个人博客系统旨在为用户提供一个便捷、高效的博客创作与管理平台。系统面向普通用户和管理员，支持用户进行内容发布、评论互动、个人信息维护等操作，同时为管理员提供后台管理工具，用于内容审核、用户管理及系统维护等。该系统具有如下主要功能和特点：

① 支持富文本博客创作与发布；

② 提供评论、点赞、收藏等互动功能，提升用户参与度；

③ 具备用户注册、登录与个人资料管理模块；

④ 后台管理系统支持文章管理、用户管理与数据监控；

⑤ 系统采用模块化设计，具备良好的可扩展性和可维护性；

⑥ 适用于个人博客搭建、小型内容发布平台或学习实验项目。

（3）项目于 2025 年 3 月启动，由软件大作战项目小组成员共同参与需求分析、系统设计与功能开发等全过程。目前系统仍处于开发阶段，正在逐步实现博客内容发布、用户注册登录、评论互动、管理员后台管理等核心功能。

（4）本系统采用前后端分离架构，前端使用 Vue 框架进行开发，后端采用 Flask 框架构建 RESTful API，并结合 MySQL 实现数据持久化存储。系统开

发过程大致分为以下几个阶段：

① 需求分析阶段：明确系统的目标用户与功能需求，制定整体开发计划；

② 设计阶段：完成系统架构、数据库模型、接口规范及前端界面原型的设计；

③ 开发阶段：各模块正在开发中，前端与后端协同进行接口联调；

④ 测试与部署阶段（计划中）：完成主要功能后将进行系统测试与优化，并部署到本地或云服务器运行。

（5）相关方

投资方/需方：软件大作战团队；

用户：普通用户、自媒体人；

开发方：软件大作战团队。

运行现场：当前本系统主要在开发者本地环境中运行，前端使用 Vue 运于浏览器环境中，后端基于 Flask 框架运行于本地 Python 环境，MySQL 数据库部署在本地数据库服务器。当前开发和测试均在 Windows 操作系统下进行。

（6）其他有关的文档

《可行性分析（研究）报告》（FAR）1.0

1.3 文档概述

（1）用途：评估项目的技术、经济、操作和法律可行性，为后续开发提供决策依据。

（2）内容：包括背景、技术选择、成本估算、风险分析等多方面分析，确定项目的可行性。

（3）保密性：本文档仅限项目团队及投资方内部使用，禁止外传。

1.4 基线

编写本系统设计说明书所依据的设计基线为：

《可行性分析（研究）报告》（FAR）1.0

2 引用文件

《MySQL 8.0 参考手册》

编号：DOC-1 版本：8.0 发行日期：2022-11-18

《中华人民共和国个人信息保护法》

编号：DOC-2 版本：1.0 发行日期：2021-11-01

《中华人民共和国网络安全法》

编号：DOC-3 版本：1.0 发行日期：2016-11-07

3 需求

以下是关于 CSCI 需求的详细描述，这些需求将作为 CSCI 验收的条件，确保每个 CSCI 能够满足分配给它的系统需求。每个需求都将被赋予一个项目唯一标识符，以便于测试和追踪。需求将以可定义客观测试的方式进行陈述，确保其明确性和可验证性。如果某些需求的合格性方法和对系统需求的可追踪性在其他章节中未提供，将在本章中进行注解。需求的描述将遵循既定规则，既包含构成 CSCI 验收条件的特性，也包括需方愿意推迟到设计阶段由开发方说明的特性。如果某条中没有需求，将如实陈述。若某个需求在多条中出现，将只陈述一次，并在其他条中直接引用。

3.1 所需的状态和方式

若 CSCI 需在不同状态及方式下运行，且各状态、方式有独特需求，则需明确标识与定义每种状态和方式。例如，状态可包括空闲、准备就绪、活动、事后分析、培训、降级、紧急情况和后备等；方式则可依具体场景而定。状态与方式的区分并无固定标准，可根据实际情况灵活选择描述方式，如仅用状态描述 CSCI，或仅用方式，亦或结合方式中的状态、状态中的方式等其他有效方式。若 CSCI 无需多个状态和方式，则无需人为区分，应如实说明；若需多个状态或方式，则应将本规格说明中的每个需求或需求组与这些状态和方式相关联。关联可

通过本条或引用附录中的表格等方法表示，也可在需求出现处加以注解。

3.1.1 CSCI 交付前的测试与准备

在将系统交付给最终用户之前，必须完成一系列严格的测试流程，确保系统满足所有既定的功能和性能标准。这些测试包括单元测试、集成测试以及系统级测试，旨在全面验证系统的各项功能和性能指标。此外，系统应具备高可靠性和可用性，能够在用户操作过程中保持稳定运行，即使遇到异常情况也能迅速恢复。系统的设计和开发必须严格遵循既定的软件架构和规范，确保其能够无缝集成到现有的系统环境中。同时，系统应配备详尽的用户手册和技术文档，帮助用户快速掌握系统的使用和维护方法。此外，系统还应具备良好的可扩展性和升级性，以便在未来能够灵活地进行功能扩展和性能提升。

3.1.2 CSCI 正常运行时的要求

当系统处于正常运行状态时，必须展现出高性能和高可靠性，能够迅速响应用户的请求，并确保数据的完整性和一致性。系统的设计应充分考虑用户的实际需求和系统架构的要求，确保其能够与其他系统组件实现无缝协作。系统的用户界面应简洁直观，符合用户的使用习惯，提供良好的交互体验。此外，系统还应具备稳定的运行环境，以及良好的可维护性和可扩展性，以便能够及时响应用户的需求变化和系统的升级需求。

3.1.3 CSCI 异常与故障处理

在面对异常情况和系统故障时，系统必须具备有效的错误处理机制，能够及时检测并捕获错误，并向用户提供清晰的提示信息和相应的解决方法。系统应配备完善的日志记录功能，详细记录运行过程中的所有日志信息，包括正常运行日志和异常日志，以便于后续的故障排查和系统优化。同时，系统必须具备快速恢复能力，能够在发生故障时迅速恢复运行，并确保数据的完整性和安全性。此外，系统还应具备完善的安全保护措施，以保障系统的安全性，确保符合相关的法规

和标准要求。

3.1.4 用户交互与操作体验

在用户与系统交互的过程中，系统必须能够稳定运行，并对用户的操作做出及时响应。系统的用户界面应简洁明了，易于操作，尽量降低用户的学习成本和使用难度。系统应能够准确处理用户的输入信息，并根据用户的需求提供相应的反馈和结果。同时，系统还必须确保用户操作的安全性和可靠性，防止因用户误操作而导致系统崩溃或数据丢失。

3.1.5 多用户并发访问管理

当系统面临多用户并发访问时，必须能够有效地处理各种请求，确保系统的整体稳定性和性能。系统应具备合理的并发访问控制机制，确保不同用户之间的数据安全性和隔离性，防止数据泄露和交叉干扰。系统还应能够根据用户的访问需求，灵活地进行资源分配和调度，避免资源争用和性能瓶颈。此外，系统还应配备有效的负载均衡机制，合理分配各个用户请求的处理负荷，确保系统在高并发情况下的整体性能和稳定性。

3.2 需求概述

3.2.1 目标

a.开发意图、应用目标及作用范围

(1) 现有产品存在的问题

用户互动体验差:当前许多博客平台的用户界面设计较为陈旧，操作流程复杂。用户在发表文章时，需要经过繁琐的步骤，如多次点击、填写大量表单等，这使得用户在发布内容时感到不便，降低了用户的创作积极性。而且评论功能不够友好，用户在评论他人文章时，可能需要等待较长时间才能看到自己的评论是

否成功提交，或者在回复评论时操作不便捷，影响了用户之间的互动交流。

管理功能不完善:对于博客网站管理员来说，现有的管理工具功能有限。在用户管理方面，管理员无法快速准确地查询到用户的相关信息，如用户发表文章的数量、评论记录等。在文章管理上，管理员难以对大量文章进行高效分类和筛选，对于一些违规或低质量的文章，删除和修改操作不够便捷，导致网站内容质量参差不齐，影响网站的整体形象。

数据安全与隐私问题:部分博客网站对用户数据的保护措施不足。用户注册时提供的个人信息，如邮箱、手机号码等，存在泄露风险。而且在文章存储和传输过程中，没有采用有效的加密措施，可能导致用户创作的内容被非法篡改或窃取，损害用户的权益。

(2) 建议产品所要解决的问题

提升用户体验:提供简洁、直观的用户界面，优化用户发表文章和评论的操作流程。例如，采用所见即所得的编辑器，让用户在撰写文章时能够方便地添加图片、格式化文本等。同时，改进评论系统，实现即时评论显示和便捷的回复功能，增强用户之间的互动性。

增强管理功能:为管理员提供强大的后台管理工具，实现用户信息的快速查询、文章的高效分类和筛选。管理员可以方便地对违规文章进行删除和修改，并且能够对用户行为进行有效监控，维护网站的良好秩序。

保障数据安全:加强用户数据的保护措施，采用加密技术对用户个人信息和文章内容进行存储和传输加密。确保用户数据的隐私性和完整性，防止数据泄露和非法篡改，让用户能够放心地使用博客平台。

(3) 应用目标与作用范围

应用目标:本博客网站旨在为用户提供一个便捷、安全、互动性强的创作和交流平台。对于用户来说，能够轻松地发表自己的见解、分享生活点滴和专业知识等。同时，通过优化管理功能，吸引更多的高质量创作者和活跃用户，提升网站的知名度和影响力。并且，通过保障数据安全，赢得用户的信任，使用户愿意长期使用该平台。

作用范围:该博客网站适用于各种类型的用户，包括个人创作者、专业博主、

行业专家等。它可以涵盖多个领域的内容，如文学创作、科技分享、生活感悟、旅游攻略等。并且，网站的后台管理功能可以满足不同规模的网站运营需求，无论是小型的个人博客网站还是中大型的综合博客平台都可以使用。

b.本系统的主要功能、处理流程、数据流程及简要说明

(1) 主要功能概述

用户注册及登录功能:用户可以通过填写用户名、邮箱、密码等信息进行注册。系统会对注册信息进行验证，如检查邮箱格式是否正确、用户名是否唯一等。登录时，用户输入用户名和密码，系统进行身份验证，验证成功后，用户进入个人博客页面。同时，系统提供忘记密码功能，用户可以通过绑定的邮箱或手机号码找回密码。

管理员后台管理功能:管理员可以对用户信息进行修改、查询和删除操作。例如，修改用户的权限级别、查询用户发表文章的数量和评论记录、删除违规用户账号等。管理员还可以对文章进行分类管理，添加、修改和删除文章分类。并且，管理员能够查看网站的整体运行数据，如用户访问量、文章发表数量等统计信息。

用户文章管理功能:用户可以发表文章，包括输入文章标题、内容，上传图片、视频等多媒体资源。用户还可以修改自己发表的文章，如修改文章标题、内容、分类等。并且，用户能够删除自己的文章。在浏览博客时，用户可以对个人及他人博客的文章进行评论，评论内容可以包含文字、表情等。

(2) 处理流程

用户注册及登录流程:用户打开网站的注册页面，填写注册信息（用户名、邮箱、密码等）。点击注册按钮后，系统对注册信息进行验证。如果验证通过，系统将用户信息存储到数据库，并发送一封验证邮件到用户注册的邮箱。用户点击邮件中的验证链接完成邮箱验证，之后就可以登录。登录时，用户输入用户名和密码，系统验证用户名和密码是否正确。如果正确，用户进入个人博客页面；如果错误，提示用户用户名或密码错误。

管理员后台管理流程:管理员登录后台管理系统。在用户管理界面，管理员可以通过搜索框输入用户名、邮箱等条件查询用户信息。点击用户信息旁边的修改按钮，进入用户信息修改页面，管理员可以修改用户权限等信息。点击删除按钮，可以删除违规用户账号。在文章分类管理界面，管理员可以添加新的分类，输入分类名称等信息后点击保存按钮添加分类；也可以修改分类名称等信息；还可以删除不再需要的分类。在数据分析界面，管理员可以查看网站的运行数据，如通过图表展示用户访问量随时间的变化趋势等。

用户文章管理流程:用户进入个人博客页面，点击发表文章按钮。在文章编辑页面，用户输入文章标题、内容，上传图片、视频等资源。点击发表按钮，系统对文章内容进行审核（如检查是否包含违规内容等）。审核通过后，文章发布成功。用户在文章列表中找到自己发表的文章，点击编辑按钮，可以修改文章的标题、内容等信息。点击删除按钮，可以删除自己的文章。在浏览博客文章时，用户在文章下方的评论框输入评论内容，点击发表评论按钮，评论显示在文章评论区。

(3) 数据流程

用户注册及登录数据流程:用户在注册页面输入注册信息后，这些信息通过网络传输到服务器。服务器端的程序对注册信息进行验证，验证通过后，将用户信息（用户名、加密后的密码、邮箱等）存储到用户数据库表中。同时，服务器生成一封验证邮件，通过邮件服务器发送到用户注册的邮箱。用户点击邮箱中的验证链接后，验证链接中的信息（如用户标识、验证令牌等）通过网络传输到服务器，服务器验证令牌的有效性，验证通过后，将用户的邮箱验证状态更新为已验证。用户登录时，输入的用户名和密码通过网络传输到服务器，服务器从数据库中查询对应的用户信息，对密码进行解密验证。如果验证通过，服务器生成一个会话标识（session ID）发送给用户浏览器，用户浏览器存储该会话标识，用户就可以进入个人博客页面。

管理员后台管理数据流程:管理员在后台管理界面进行用户管理操作时，输入的查询条件（用户名、邮箱等）通过网络传输到服务器。服务器从用户数据库表中查询符合条件的用户信息，将查询结果（用户列表，包括用户名、邮箱、权

限等信息）通过网络传输回管理员界面显示。管理员修改用户信息时，修改后的信息通过网络传输到服务器，服务器更新用户数据库表中对应的用户记录。管理员删除用户账号时，服务器从用户数据库表中删除对应的用户记录。在文章分类管理中，管理员添加分类时，输入的分类信息（分类名称等）通过网络传输到服务器，服务器将分类信息存储到文章分类数据库表中。修改分类信息时，更新数据库表中对应的分类记录；删除分类时，从数据库表中删除对应的分类记录。管理员查看网站运行数据时，服务器从数据库中提取相关的数据（如用户访问记录、文章发表记录等），进行统计分析后，将统计结果（如图表数据）通过网络传输到管理员界面显示。

用户文章管理数据流程:用户发表文章时，输入的文章信息（标题、内容、图片、视频等）通过网络传输到服务器。服务器对文章内容进行审核，审核通过后，将文章信息存储到文章数据库表中。用户修改文章时，修改后的文章信息通过网络传输到服务器，服务器更新文章数据库表中对应的记录。用户删除文章时，服务器从文章数据库表中删除对应的记录。用户评论文章时，评论内容通过网络传输到服务器，服务器将评论信息（评论内容、评论时间、评论者信息等）存储到评论数据库表中。

c.表示外部接口和数据流的系统高层次图

外部接口

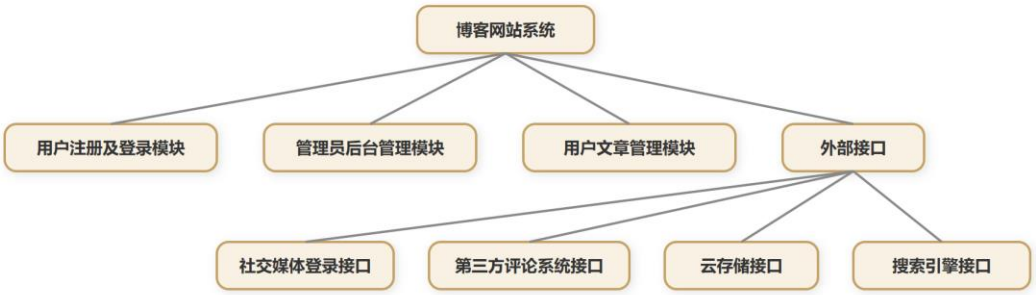
社交媒体登录接口（如 OAuth/OpenID Connect 服务）：允许用户使用社交媒体账号（如微信、微博等）快速登录博客网站，方便用户注册和登录，提高用户体验。

第三方评论系统接口（如 Disqus）：集成第三方评论系统，提供更丰富的评论功能，如评论嵌套、评论点赞等，增强用户之间的互动交流。

云存储接口（如阿里云 OSS）：用于存储用户上传的图片、视频等多媒体资源，确保资源的稳定存储和快速访问，减轻服务器存储压力。

搜索引擎接口（如百度搜索引擎）：将博客网站的文章提交到搜索引擎，提高网站的曝光率和搜索引擎优化（SEO）效果，吸引更多用户访问。

博客网站系统结构图



3.2.2 运行环境

(1) 硬件环境

操作系统：要求设备搭载 Windows 7 或以上版本、macOS Mojave 或以上版本，以确保兼容性。

处理器：至少要求双核处理器，以保证系统运行的流畅性。

内存：系统要求至少具备 4GB RAM，以确保应用程序的正常运行。

存储空间：至少需要 8GB 的空闲存储空间，用于安装和存储系统数据。

(2) 支持环境

云存储服务：开发者可能会使用云存储服务如 AWS S3 或阿里云 OSS，以存储用户上传的图片、视频等多媒体资源。

数据库：系统会使用关系型数据库如 MySQL 或 PostgreSQL，用于存储用户信息、文章内容、评论记录等数据。

加密库：系统可能会使用加密库来加密用户敏感信息，如密码、邮箱等，确保数据的安全性。

第三方 API：系统集成第三方 API，如社交媒体登录接口（如 OAuth/OpenID Connect 服务）、云存储接口、搜索引擎接口等，用于实现便捷的用户登录、资源存储和内容分发等功能。

日志记录工具：系统可能会使用日志记录工具来记录用户操作和系统运行状

态，以便进行故障排查和系统优化。

安全性验证服务：系统可能会使用身份验证服务如 LDAP/AD 服务，用于用户登录认证和权限管理。

3.2.3 用户的特点

(1) 博客网站管理员

网站内容管理：拥有对网站内容的全面管理权限，包括审核、编辑、删除用户发布的文章和评论，确保网站内容的合规性、质量和一致性。能够对文章进行分类管理，添加、修改和删除文章分类，以便用户更方便地浏览和查找感兴趣的内容。

用户账号管理：负责创建、编辑和删除用户账号，设置用户权限级别，如普通用户、VIP 用户、博主等，根据用户的不同角色分配相应的功能权限，保障网站的用户管理体系有序运行。

系统配置与维护：对博客网站的各项功能进行配置和优化，如设置网站的主题风格、布局、功能模块等，以提升用户体验和网站的吸引力。同时，需要定期进行系统维护，包括更新软件版本、修复漏洞、优化性能等，确保网站的稳定运行。

数据备份与安全：负责网站数据的备份和恢复工作，定期将用户文章、评论、用户信息等重要数据进行备份，防止数据丢失。并且，要关注网站的安全状况，采取措施防范黑客攻击、数据泄露等安全威胁，保障用户数据的安全和隐私。

(2) 博客网站普通用户

文章浏览与评论：可以浏览网站上的各类文章，根据自己的兴趣和需求选择阅读不同主题、不同作者的文章。在阅读文章后，能够发表自己的看法和意见，对文章进行评论，与其他用户进行互动交流，分享观点和感受。

个人博客管理：拥有自己的个人博客空间，可以发表文章，分享自己的生活点滴、专业知识、兴趣爱好等内容。能够对已发表的文章进行编辑和删除操作，修改文章的标题、内容、分类等信息，管理自己的博客文章。同时，可以查看自己发表文章的浏览量、评论数等统计信息，了解文章的受欢迎程度。

社交互动：可以通过关注其他用户、与其他用户互相关注等方式，建立自己的社交圈子。能够查看关注用户的最新文章和动态，与其他用户进行点赞、评论、私信等互动，增进彼此之间的交流和联系。

(3) 博客网站博主（高级用户）

优质内容创作：作为网站上的活跃创作者，博主通常具有较高的创作能力和专业素养，能够定期发布高质量、有深度、有吸引力的文章，吸引大量用户关注和阅读。其文章内容可能涵盖特定领域或多个领域的专业知识、见解、经验分享等，对其他用户具有一定的指导和启发作用。

粉丝互动与管理：拥有一定数量的粉丝群体，需要与粉丝进行积极的互动，回复粉丝的评论和私信，解答粉丝的问题，增强与粉丝之间的粘性和信任。同时，博主可以通过发布专属内容、举办线上活动等方式，提升粉丝的活跃度和忠诚度，维护良好的粉丝关系。

品牌建设与推广：博主往往会注重个人品牌的建设和推广，通过优化个人博客页面的设计、提升文章质量、参与行业交流等方式，树立自己在特定领域的专业形象和影响力。并且，博主可能会利用社交媒体等渠道，将自己的博客内容进行推广，吸引更多潜在读者，扩大自己在博客领域的知名度和影响力。

(4) 博客网站 IT 管理员

网站安全防护：负责制定和实施网站的安全策略，包括用户身份验证机制的设置和管理，如密码加密、登录限制等，防止非法用户入侵。对网站进行访问控制管理，根据用户角色和权限设置，限制用户对不同功能模块和数据的访问权限，确保网站数据的安全性和保密性。同时，要关注网络安全动态，及时发现和修复网站的安全漏洞，防范黑客攻击、恶意软件入侵等安全威胁，保障网站的安全稳定运行。

系统运维与优化：负责博客网站的日常运维工作，包括服务器的安装、配置和维护，确保服务器的正常运行和性能稳定。对网站进行性能监控，分析网站的访问流量、响应时间等指标，及时发现和解决性能瓶颈问题，优化网站的性能，提升用户体验。并且，要定期对网站进行升级，包括软件版本更新、功能模块优

化等，以适应不断变化的用户需求和互联网技术发展。

技术支持与故障排除：为网站用户提供技术支持，解答用户在使用过程中遇到的问题，如账号登录问题、文章发表问题等。当网站出现故障时，能够迅速定位问题并进行有效的故障排除，及时恢复网站的正常运行，减少故障对用户的影响。

3.2.4 关键点

(1) 关键功能

用户注册与登录：提供用户注册、登录功能，支持邮箱、手机号码、社交媒体账号等多种登录方式，方便用户快速注册和登录。

文章发布与管理：用户可以发布、编辑、删除文章，支持图文混排、视频嵌入等多种内容形式，同时提供文章分类、标签等功能，方便用户管理和读者查找。

评论互动：用户可以在文章下发表评论，支持回复、点赞、举报等功能，促进用户之间的互动交流。

个人博客管理：用户可以管理自己的个人博客页面，包括设置头像、个人简介、自定义页面布局等，展示个性化的博客风格。

文章推荐与搜索：根据用户的浏览历史、兴趣爱好等，为用户推荐相关的文章，同时提供强大的搜索功能，方便用户快速找到感兴趣的内容。

社交功能：用户可以关注其他用户，查看关注用户的最新文章和动态，与其他用户进行点赞、评论、私信等互动，建立社交圈子。

后台管理：管理员可以对用户账号、文章、评论等进行管理，包括审核、编辑、删除等操作，同时提供数据统计和分析功能，帮助管理员了解网站的运行情况。

(2) 关键算法

内容推荐算法：根据用户的浏览历史、兴趣爱好、行为数据等，通过机器学习和数据分析技术，为用户推荐个性化的文章内容，提高用户发现优质内容的效率。

评论审核算法：采用自然语言处理技术，对用户发表的评论进行自动审核，

识别和过滤违规、垃圾评论，维护良好的社区环境。

文章排序算法：根据文章的发布时间、浏览量、点赞数、评论数等多维度数据，对文章进行智能排序，确保优质内容能够得到更多的曝光和关注。

用户画像算法：通过收集和分析用户的注册信息、浏览行为、发表内容等数据，构建用户画像，为用户提供更精准的内容推荐和个性化服务。

(3) 关键技术

数据库技术：选择适合的数据库系统，如 MySQL、MongoDB 等，用于存储用户信息、文章内容、评论数据等，确保数据的高效存储和快速查询。

前端技术：采用现代化的前端框架和技术，如 React、Vue.js 等，实现用户友好的界面和流畅的交互体验，提升用户满意度。

后端技术：使用稳定可靠的后端框架和技术，如 Node.js、Django 等，实现系统的业务逻辑和数据处理功能，确保系统的稳定运行。

安全技术：采用 SSL 加密协议、数据加密存储、用户身份验证、访问控制等安全技术，保障用户数据的安全和隐私，防止数据泄露和非法访问。

云计算技术：利用云存储服务（如 AWS S3、阿里云 OSS）存储用户上传的多媒体资源，提供弹性可扩展的存储空间，同时利用云服务器（如 AWS EC2、阿里云 ECS）部署网站应用，确保网站的高可用性和性能稳定性。

搜索引擎技术：集成搜索引擎（如 Elasticsearch）实现高效的全文搜索功能，帮助用户快速找到所需的文章和内容。

(4) 用户特点

博客网站管理员：负责网站的整体管理和运营，需要系统提供强大的后台管理功能，如用户账号管理、内容审核、数据统计分析等，以便高效地管理网站内容和用户，确保网站的正常运行和健康发展。

博主（内容创作者）：专注于创作和发布优质内容，需要系统提供便捷的文章编辑和发布工具，支持多种内容形式，同时希望获得更多的曝光和关注，因此需要系统具备良好的内容推荐和推广机制。

普通用户（读者）：主要通过浏览和阅读文章获取信息和娱乐，需要系统界

面简洁、操作便捷，能够快速找到感兴趣的内容。同时，普通用户也希望能够方便地与其他用户互动交流，发表自己的看法和意见。

IT 技术人员：负责网站的技术开发和维护，需要系统具备良好的技术架构和可扩展性，便于进行功能开发、性能优化和故障排查等工作。

3.2.5 约束条件

(1) 经费限制

预算规划：开发团队需在有限经费内完成开发与维护，需制定详细开发计划，合理分配经费至各阶段与模块，如前端设计、后端开发、测试、服务器租赁等，控制成本，确保高效完成任务。

质量保障：经费有限，但不能忽视软件质量与用户满意度。需在有限资源下，优化开发流程，提高开发效率，确保软件功能完善、性能稳定、界面友好，满足用户基本需求。

(2) 开发期限

进度安排：为满足市场需求，团队需按约定时间完成开发并上线。应根据开发计划合理安排工作进度，分解任务，明确各阶段时间节点，定期检查进度，及时调整计划，确保按时交付。

功能与性能平衡：开发期限紧张，需合理安排功能开发，优先实现核心功能，如用户注册登录、文章发布与浏览等，确保软件基本可用。同时，保证软件性能，避免因赶进度而忽视性能优化，影响用户体验。

(3) 技术限制

现有技术应用：团队需使用现有成熟技术和工具开发，如常见的编程语言、数据库、前端框架等，遵循技术标准和规范，确保开发和维护质量稳定。同时，关注技术更新，适时引入新技术提升软件竞争力。

技术熟练度：团队成员需熟练掌握所用技术和工具，熟悉开发流程，避免因技术生疏导致开发效率低下和质量问题。必要时，进行技术培训和学习，提高团队整体技术水平。

(4) 法律限制

法律法规遵守：开发团队需严格遵守相关法律法规，如个人信息保护法、著作权法、网络安全法等。在用户注册、文章发布、评论互动等环节，确保用户信息收集、存储和使用合法合规，尊重用户隐私和知识产权，避免法律风险。

内容管理：博客网站内容丰富多样，需建立严格内容审核机制，防止违法、违规、侵权等不良信息传播，维护网络环境健康有序。

(5) 用户需求限制

需求调研与分析：开发前需深入调研用户需求，通过问卷调查、用户访谈等方式了解用户对博客功能、界面、交互等方面期望，从用户角度出发设计软件，满足用户基本需求和个性化需求。

用户体验优化：开发过程中，持续关注用户体验，根据用户反馈及时调整和优化功能，提高软件易用性和满意度，如优化文章发布流程、提升页面加载速度、增强评论互动功能等。

(6) 平台限制

多平台适配：博客网站需支持多种设备和平台，如电脑端、手机端、平板端等，开发团队需了解各平台特点和技术要求，采用响应式设计或开发不同平台专属版本，确保软件在不同设备上显示正常、操作流畅、功能完整。

兼容性测试：在开发过程中，需进行充分兼容性测试，确保软件在不同操作系统、浏览器、屏幕分辨率等环境下运行稳定，及时发现和修复兼容性问题，提高用户使用体验。

(7) 数据安全限制

数据加密与保护：用户在博客网站上传的文章、图片、评论等数据需确保安全，采用加密技术对数据存储和传输进行加密，防止数据泄露和被恶意篡改。同时，对敏感信息如用户密码、联系方式等采取严格访问控制策略，限制数据访问权限。

安全防护措施：加强网站安全防护，采用防火墙、入侵检测系统等技术手段，

防止黑客攻击和恶意软件入侵，保障网站和用户数据安全。

(8) 可用性限制

界面设计与交互优化：注重软件界面友好性和易用性，遵循用户界面设计规范和标准，采用简洁明了的布局、直观的操作流程、美观的图形化设计，提供清晰的导航和帮助信息，降低用户学习成本，提高用户满意度。

多设备适配与测试：考虑用户使用不同设备和平台的差异，进行适配和测试，确保软件在各种设备上操作一致、功能正常，为用户提供稳定、流畅的使用体验。

(9) 敏捷开发

迭代开发：采用敏捷开发方法，将开发过程划分为多个短迭代周期，每个迭代集中开发重要功能，快速响应用户需求变化，及时调整开发方向，提高开发效率和软件质量。

团队协作与沟通：敏捷开发强调团队协作和沟通，开发团队需与产品经理、设计师、测试人员等密切合作，定期召开会议，及时交流开发进展、问题和解决方案，确保团队成员目标一致，协同推进项目。

(10) 开源软件利用

开源资源利用：在开发过程中，可合理利用开源软件和开源库，如开源的前端框架、后端框架、数据库管理系统等，节省开发时间和成本，快速搭建软件基础架构。

开源社区参与：鼓励开发团队参与开源社区，积极贡献代码和经验，借助社区力量解决开发中遇到的问题，同时提升软件稳定性和完善度，促进软件持续发展。

(11) 互联网审查

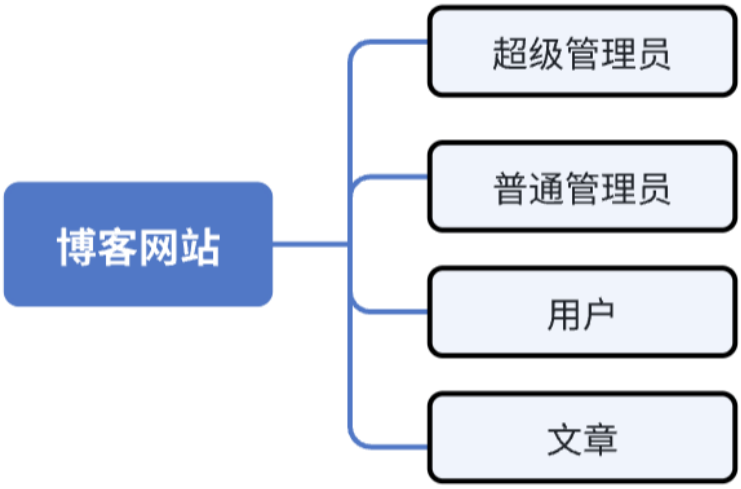
内容合规性：了解当地互联网审查要求，确保博客网站内容符合相关法律和条例，不涉及敏感、违法、违规等不良信息。建立内容审核机制，对用户发布的内容进行严格审核，及时处理违规内容，避免引发法律纠纷和审查风险。

文化适应性：不同地区文化背景差异，开发团队需考虑文化因素，使网站内容和功能符合当地文化习俗和价值观，避免因文化冲突导致用户反感或审查问题。

3.3 需求规格

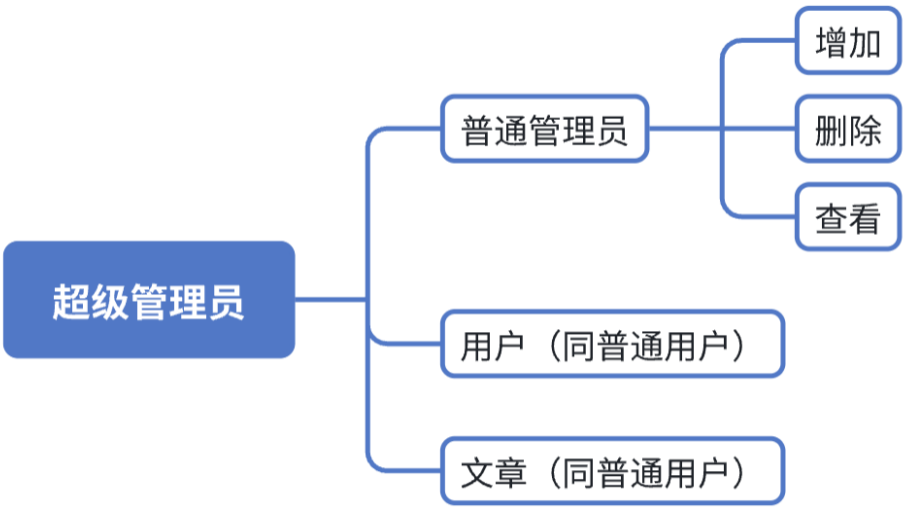
3.3.1 软件系统总体功能/对象结构

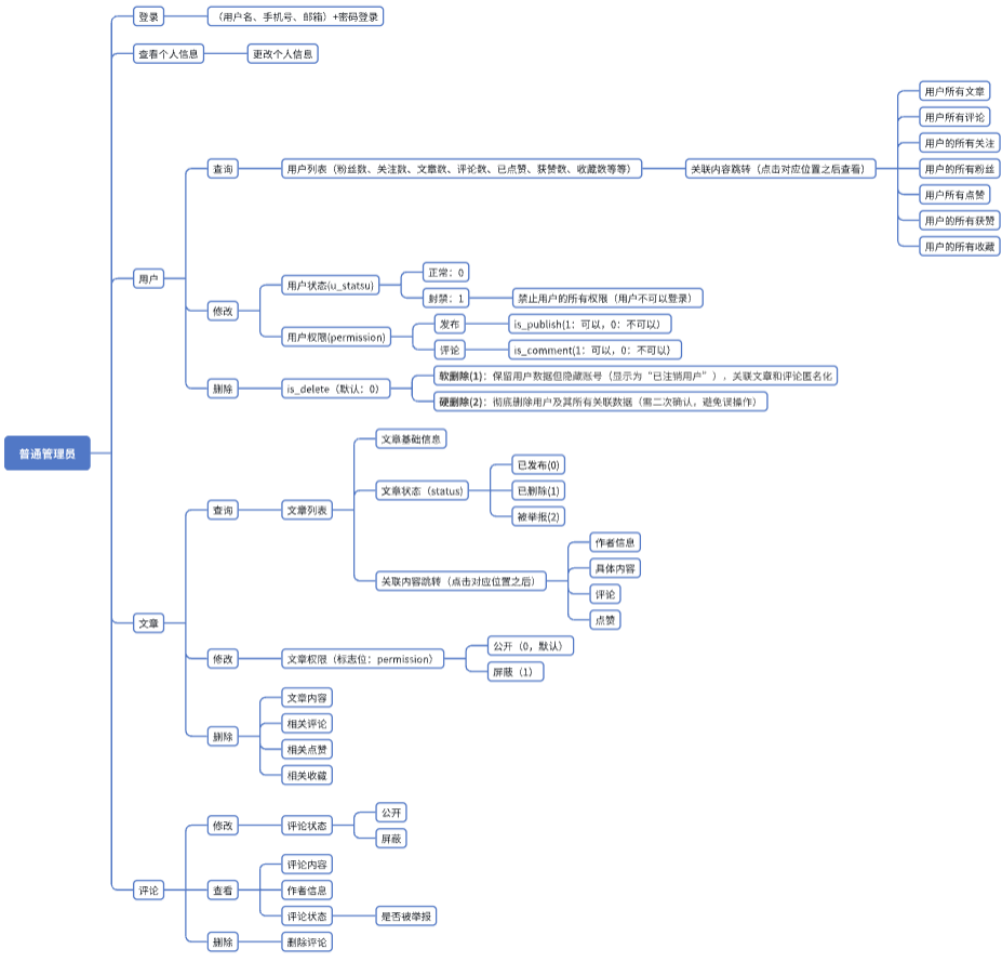
对软件系统总体功能/对象结构进行描述，包括结构图、流程图或对象图。

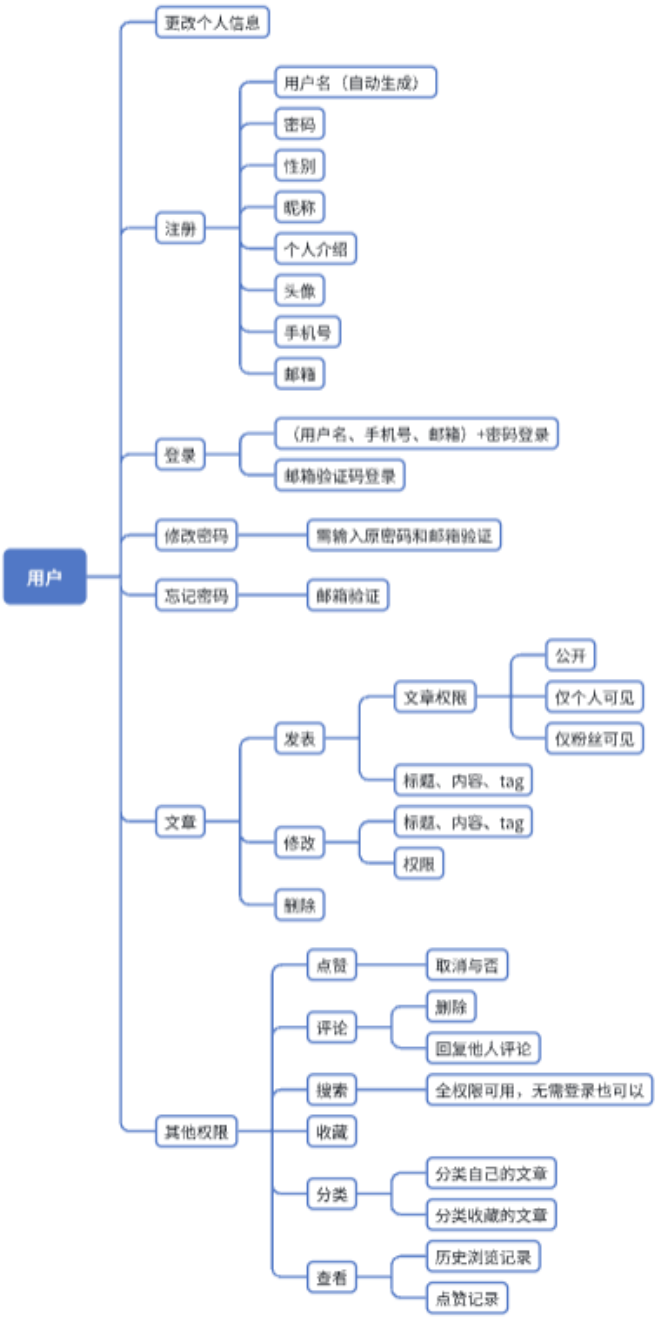


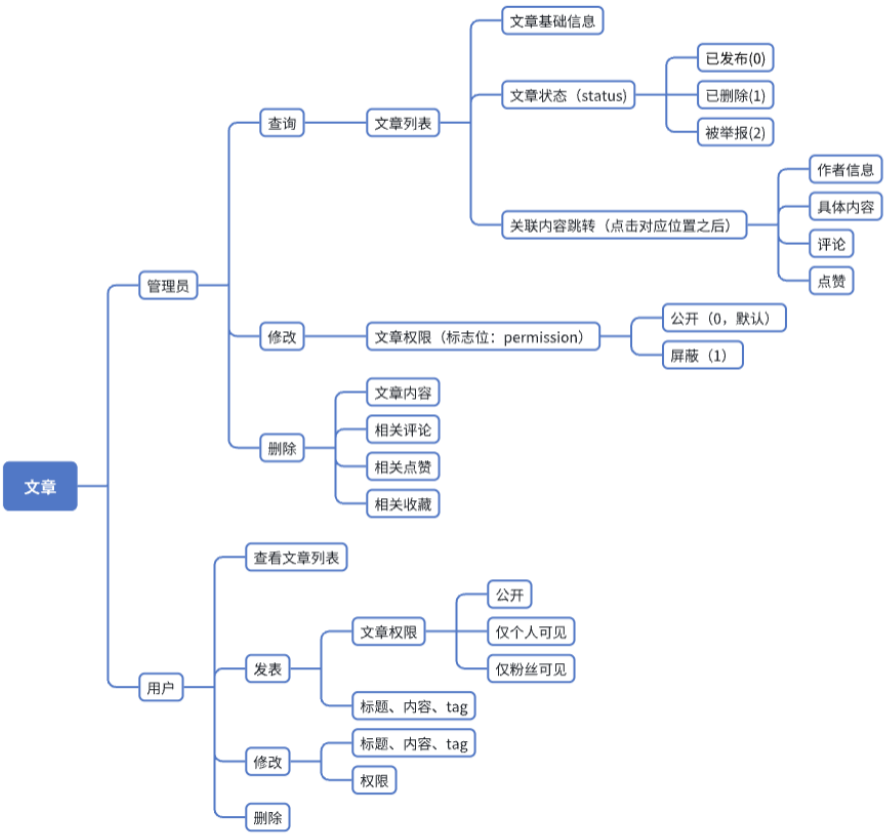
3.3.2 软件子系统功能/对象结构

对每个主要子系统中的基本功能模块/对象进行描述，包括结构图、流程图或对象图。


超级管理员
普通管理员
增加
删除
查看
用户（同普通用户）
文章（同普通用户）


普通管理员
登录
（用户名、手机号、邮箱）+密码登录
查看个人信息
更改个人信息
用户
查询
用户列表（粉丝数、关注数、文章数、评论数、已点赞、获赞数、收藏数等等）
关联内容跳转（点击对应位置之后查看）
用户所有文章
用户所有评论
用户的所有关注
用户的所有粉丝
用户所有点赞
用户的所有获赞
用户的所有收藏
修改
用户状态(u_statsu)
正常：0
封禁：1
禁止用户的所有权限（用户不可以登录）
用户权限(permission)
发布
is_publish(1：可以，0：不可以)
评论
is_comment(1：可以，0：不可以)
删除
is_delete（默认：0）
软删除(1)：保留用户数据但隐藏账号（显示为“已注销用户”），关联文章和评论匿名化
硬删除(2)：彻底删除用户及其所有关联数据（需二次确认，避免误操作）
文章
查询
文章列表
文章基础信息
文章状态（status）
已发布(0)
已删除(1)
被举报(2)
关联内容跳转（点击对应位置之后）
作者信息
具体内容
评论
点赞
修改
文章权限（标志位：permission）
公开（0，默认）
屏蔽（1）
删除
文章内容
相关评论
相关点赞
相关收藏
评论
修改
评论状态
公开
屏蔽
查看
评论内容
作者信息
评论状态
是否被举报
删除
删除评论


用户
更改个人信息
注册
用户名（自动生成）
密码
性别
昵称
个人介绍
头像
手机号
邮箱
登录
（用户名、手机号、邮箱）+密码登录
邮箱验证码登录
修改密码
需输入原密码和邮箱验证
忘记密码
邮箱验证
文章
发表
文章权限
公开
仅个人可见
仅粉丝可见
标题、内容、tag
修改
标题、内容、tag
权限
删除
其他权限
点赞
取消与否
评论
删除
回复他人评论
搜索
全权限可用，无需登录也可以
收藏
分类
分类自己的文章
分类收藏的文章
查看
历史浏览记录
点赞记录



3.3.3 描述约定

在本博客网站需求规格说明中，我们遵循以下描述约定：

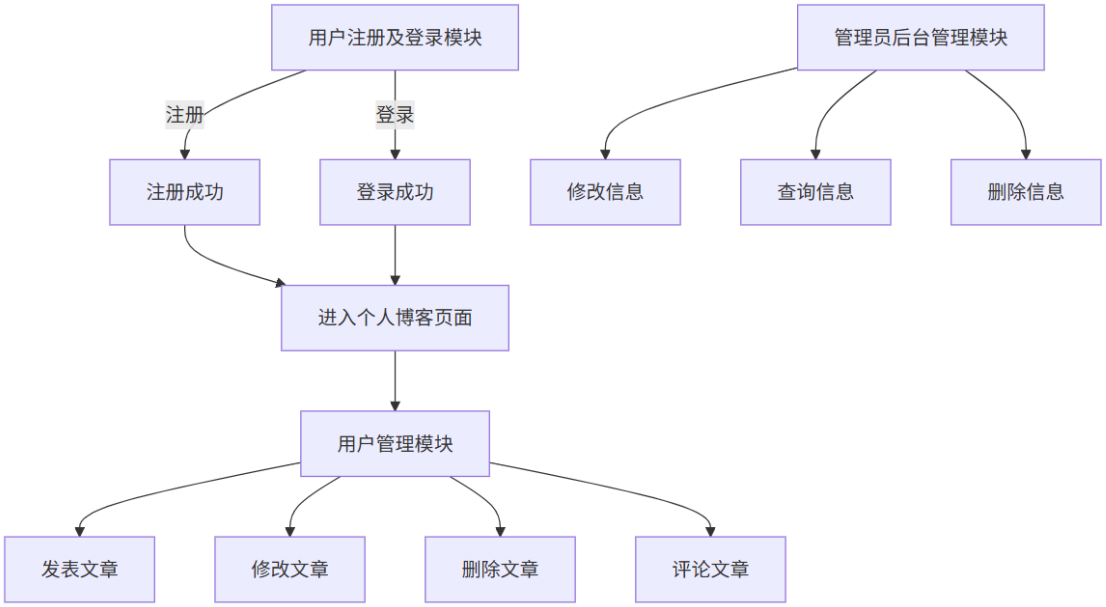
(1) 度量单位：所有的度量单位均为国际标准单位（SI 单位），除非特别指明。

(2) 缩写词和首字母缩略词：所有的缩写词和首字母缩略词在第一次出现时均需解释其含义。

(3) 图形符号和缩略语：本文档中使用的所有图形符号和缩略语的含义将在每个图形下方进行说明。

(4) 术语：本文档中使用的所有术语遵循业界通用定义，并在术语表中进行解释。

信息格式转换模块



3.4CSCI 能力需求

对于博客网站项目，我们可以将其划分为以下几个主要的 CSCI（Computer Software Component）能力需求：

(1) 前端应用程序

(2) 后端应用程序

(3) 数据库管理系统

(4) 用户认证系统

(5) 内容管理系统

对于每个 CSCI，我们可以定义相应的能力需求，以确保其实现符合预期。以下是该项目的 CSCI 能力需求：

(1) 前端应用程序

a. 说明

前端应用程序负责提供用户界面，包括文章浏览、发表、评论等功能。它需要支持响应式设计，适配不同的设备和屏幕尺寸。

b. 输入

•用户操作：如点击、输入文本、选择选项等。

•用户设备信息：屏幕尺寸、分辨率等。

c. 处理

•输入数据的有效性检查：如文章内容不能为空、评论长度限制等。

•操作的顺序：如先登录后发表文章。

•异常情况的响应：如网络中断、服务器错误等。

d. 输出

•显示内容：文章列表、文章详情、评论列表等。

•用户反馈：操作成功或失败的提示信息。

(2) 后端应用程序

a. 说明

后端应用程序负责处理前端发送的请求，包括用户认证、文章管理、评论管理等。

b. 输入

•前端发送的请求：如登录请求、发表文章请求等。

•用户数据：用户名、密码、文章内容等。

c. 处理

•输入数据的有效性检查：如用户名格式、密码强度等。

•数据处理：如验证用户身份、存储文章、处理评论等。

•异常情况的响应：如数据库错误、权限不足等。

d. 输出

•响应数据：如登录成功、文章发表成功等。

•错误信息：如用户名不存在、密码错误等。

(3) 数据库管理系统

a. 说明

数据库管理系统负责存储和管理用户信息、文章内容、评论等数据。

b. 输入

•后端应用程序发送的数据：如用户注册信息、文章数据等。

c. 处理

•数据存储：如创建用户记录、存储文章等。

•数据检索：如查询用户信息、获取文章列表等。

•数据一致性：确保数据的准确性和完整性。

d. 输出

•查询结果：如用户信息、文章内容等。

•错误信息：如查询失败、数据不存在等。

(4) 用户认证系统

a. 说明

用户认证系统负责验证用户身份，确保只有合法用户可以访问系统。

b. 输入

•用户提交的认证信息：如用户名、密码等。

c. 处理

•验证用户身份：如检查用户名和密码是否匹配。

•异常情况的响应：如密码错误、账户被锁定等。

d. 输出

•认证结果：如登录成功、登录失败等。

•错误信息：如用户名不存在、密码错误等。

(5) 内容管理系统

a. 说明

内容管理系统负责管理文章内容，包括文章的创建、编辑、删除等。

b. 输入

•用户提交的文章数据：如标题、内容、分类等。

c. 处理

•文章存储：如创建新文章、更新文章内容等。

•文章检索：如根据分类查询文章、获取文章详情等。

•异常情况的响应：如文章不存在、权限不足等。

d. 输出

•文章列表：如根据查询条件返回的文章列表。

•文章详情：如文章的标题、内容、作者等。

•错误信息：如文章不存在、操作失败等。

这些 CSCI 能力需求确保了博客网站项目的功能完整性、性能稳定性和用户

体验的优越性。

3.5CSCI 外部接口需求

本节详细描述博客网站项目的外部接口需求，包括用户接口、硬件接口、软件接口和通信接口。

1. 用户接口

(1) 用户登录接口:

•需求：提供用户登录账号渠道，为保证软件的高可用度，应至少支持以下登录形式：

•社交媒体账号登录（如微信、微博、GitHub 等）

•邮箱登录

•功能：用户可以通过以上任一方式登录博客网站，系统应验证用户身份并提供相应的访问权限。

(2) 用户注册接口:

•需求：提供用户注册渠道。用户需通过手机号码或邮箱进行注册。

•功能：用户可以通过填写注册信息（如用户名、密码、邮箱或手机号码等）进行注册，系统应验证信息的有效性并创建用户账号。

(3) 文章浏览接口:

•需求：普通用户可以浏览网站上的文章，包括标题、作者、发布时间、内容摘要等。

•功能：用户可以通过分类、标签、搜索等方式查找和阅读文章，系统应提供文章的详细信息和相关操作（如评论、点赞等）。

(4) 文章查询接口:

•需求：用户可以查询文章的详细信息，包括文章内容、评论、点赞数等。

•功能：用户可以通过文章链接或搜索结果查看文章的详细信息，系统应提供文章的完整内容和相关互动功能。

2. 硬件接口

(1) 移动设备接口:

•需求：博客网站应支持在各种移动设备（如智能手机、平板电脑）上访问和操作。

•功能：网站应采用响应式设计，确保在不同设备和屏幕尺寸上都能正常显示和操作。

(2) 服务器接口:

•需求：博客网站需要部署在服务器上，支持高并发访问和数据存储。

•功能：服务器应提供稳定的运行环境，支持网站的部署、运行和维护，确保数据的安全和可用性。

3. 软件接口

(1) 第三方登录接口:

•需求：支持通过第三方平台（如社交媒体）进行用户登录。

•功能：系统应集成第三方登录 SDK，提供用户授权和登录功能，确保用户身份验证的安全性和便捷性。

(2) 云存储接口:

•需求：支持用户上传的多媒体资源（如图片、视频）存储在云存储服务中。

•功能：系统应集成云存储服务 API，提供资源的上传、下载和管理功能，确保资源的稳定存储和快速访问。

(3) 搜索引擎接口:

•需求：支持将博客网站的文章提交到搜索引擎，提高网站的曝光率和 SEO 效果。

•功能：系统应集成搜索引擎 API，提供文章的提交和索引功能，确保文章能够被搜索引擎收录和检索。

4. 通信接口

(1) API 接口:

•需求：博客网站应提供 RESTful API 接口，供前端应用程序调用，实现数据的交互和功能的操作。

•功能：API 接口应支持常见的 HTTP 方法（如 GET、POST、PUT、DELETE 等），

提供数据的查询、创建、更新和删除功能，确保数据的一致性和安全性。

(2) WebSocket 接口:

•需求：支持实时通信功能，如实时通知、在线聊天等。

•功能：系统应集成 WebSocket 协议，提供实时数据传输和通信功能，确保用户能够及时获取信息和进行互动。

通过以上外部接口，博客网站项目能够确保与外部系统和设备的兼容性、互操作性和扩展性，为用户提供稳定、安全、便捷的服务。

3.5.1 接口标识和接口图

1. 接口标识

每个接口标识应包括项目唯一标识符，并应用名称、序号、版本和引用文件指明接口的实体（系统、配置项、用户等）。

(1) 用户登录接口

•用户需至少提供邮箱号、密码。

•接口示例：@user_bp.route('/user/login', methods=['POST'])

(2) 用户注册接口

•用户至少要提供验证码、手机号码、密码和昵称。

•接口示例：@user_bp.route('/user/register', methods=['POST'])

(3) 文章管理接口

•用户可以管理网站上自己的文章，包括标题、内容摘要等。

•接口示例：@artical_bp.route('/article/update/<int:article_id>', methods=['PUT'])

(4) 文章发布接口

•用户可以发布文章的详细信息。

•接口示例：@artical_bp.route('/article/create', methods=['POST'])

(5) 评论发布接口

•用户可以对文章进行评论，也可以对评论进行评论。

•接口示例：@comment_bp.route('/comment/<int:article_id>/create', methods=['POST'])

(6) 管理员管理用户接口：

•管理员可以对用户进行管理。

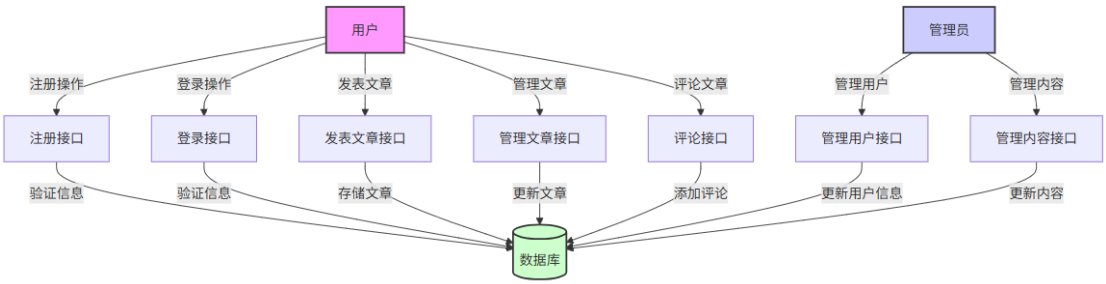
•接口示例：@manager_bp.route('/manager/user_list', methods=['GET'])

(7) 管理员管理内容接口：

•管理员可以对用户状态进行管理。

•接口示例：@manager_bp.route('/manager/update_user_status', methods=['POST'])

2. 接口图



3.6CSCI 内部接口需求

功能模块	函数名称	URL 路径	接口用途
用户管理	register_user	/user/register	添加新用户到系统
	login_user	/user/login	从系统中登录指定用户
	send_verification_code	/user/sendEmailCode	发送验证码接口
	verify_code	/user/verifyEmailCode	验证邮箱验证码
	update_profile	/user/updateProfile	更新用户信息
	logout_user	/user/logout	用户退出登录
管理员管理	manager_login	/manager/login	管理员登录
	add_manager	/manager/add_mng	增加管理员
	delete_manager	/manager/delete_mng	删掉管理员
	list_managers	/manager/manager_list	列出所有管理员
	get_manager_profile	/manager/profile	管理员查看个人信息接口
	update_manager_profile	/manager/update_profile	管理员修改个人信息接口
	list_users	/manager/user_list	管理员查看用户列表接口
	update_user_status	/manager/update_user_status	管理员修改用户状态

			接口
	update_user_permission	/manager/update_user_permission	管理员修改用户权限接口
文章管理	get_all_articles	/manager/article_list	获取所有文章列表（管理员专用）
	get_article_detail	/manager/article_detail/<int:article_id>	获取特定文章详情（管理员专用）
	change_article_status_by_admin	/article/manager/update/<int:article_id>/status	修改文章状态（管理员专用）
	change_article_permission_by_admin	/article/manager/update/<int:article_id>/permission	修改文章权限（管理员专用）
	delete_article_physically	/article/manager/de1/<int:article_id>	物理删除文章（管理员专用）
	soft_delete_article	/article/manager/de2/<int:article_id>	软删除文章（管理员专用）
	create_article	/article/create	创建文章（用户）
	get_articles	/article/list	获取特定用户文章列表（管理员和用户均可）
	update_article_by_user	/article/update/<int:article_id>	更新文章（用户）
	delete_article_by_user	/article/delete/<int:article_id>	删除文章（用户）
	like_article	/articles/<int:article_id>/like	点赞文章（用户）
	unlike_article	/articles/<int:article_id>/unlike	取消点赞（用户）
管理评论	create_comment	/comment/<int:article_id>/create	用户创建评论，已经实现父评论功能
	update_comment	/comment/update/<int:comment_id>	用户更新自己的评论
	delete_comment	/comment/delete/<int:comment_id>	用户删除自己的评论
	report_comment	/comment/report/<int:comment_id>	用户举报评论
	get_user_comments	/comment/listall	用户获取自己发布的所有评论
	get_user_reported_comments	/comment/reported_comments	用户获取自己所有被举报的评论
管理收藏	favorite_article	/article/favorite/<int:article_id>	收藏
	unfavorite_article	/article/unfavorite/<int:article_id>	取消收藏
	get_user_favorites	/user/favorites/<int:user_id>	获取某个用户的所有收藏
	get_article_favorites_count	/article/favorites/count/<int:article_id>	获取某个文章的收藏数
管理点赞	like_article	/alike/like/<int:article_id>	点赞

	unlike_article	/alike/unlike/<int:article_id>	取消点赞
	get_like_count	/alike/count/<int:article_id>	获取文章的点赞数
	get_likers	/alike/list/<int:article_id>	获取文章的点赞用户
	get_user_likes	/alike/user/records	获取用户的所有点赞记录
管理评论点赞	like_comment	/comment/like/<int:comment_id>	用户评论点赞
	unlike_comment	/comment/unlike/<int:comment_id>	用户取消点赞
	get_comment_likes_count	/comment/likes/count/<int:comment_id>	用户获得某个评论的所有点赞数
	get_comment_likes_users	/comment/likes/users/<int:comment_id>	用户获得某个评论所有点赞的用户

3.7CSCI 内部数据需求

3.7.1 静态数据

表 3 静态数据表

名称	定义	类型
用户信息表	包括用户的基本信息、登录方式等	数据表
文章信息表	包括文章的标题、内容、作者、分类等	数据表
评论信息表	包括评论的内容、作者、关联文章等	数据表
管理员信息表	包括管理员的基本信息、权限等	数据表
文章点赞表	包括文章点赞的用户名称信息等	数据表
评论点赞表	包括评论点赞的用户名称信息等	数据表

3.7.2 动态数据

表 4 动态数据表

名称	定义	类型
文章浏览记录	记录用户浏览文章的行为	记录
用户活动记录	记录用户的登录、发表文章、评论等行为	记录
系统日志	记录系统的操作日志，包括错误日志等	日志文件

3.8 适应性需求

3.8.1 安装数据要求

(1) 数据库

•类型：MySQL 数据库

•用途：用于存储系统所需的各种静态和动态数据，如用户信息、文章内容、评论等。

(2) 编程环境

•语言：Python 3.9 版本

•用途：用于运行 flask 框架，开发和部署博客网站。

(3) Web 框架

•框架:vue3 框架

•用途：提供了强大的 Web 开发功能，用于快速开发和部署博客网站。

3.8.2 运行参数要求

(1) 数据库连接参数

要求：MySQL 数据库的连接地址、用户名、密码等，确保应用程序能够正确连接和操作数据库。

(2) Web 服务器配置

端口：flask 框架的运行端口（默认为 8000）。

域名：网站的域名配置，用于访问博客网站。

静态文件路径：静态文件（如 CSS、JavaScript、图片等）的存储路径。

(3) 系统参数配置

日志级别：系统日志记录的详细程度，如 DEBUG、INFO、WARNING、ERROR 等。

调试模式：是否开启调试模式，影响错误报告和性能。

国际化语言设置:支持的语言和默认语言设置,以适应不同地区用户的需求。

(4) 硬件要求

服务器配置：CPU、内存、硬盘空间等硬件配置需要满足系统运行的基本要

求，以保证系统的流畅运行和数据处理能力。

(5) 软件兼容性

操作系统兼容性：系统需要与服务器的操作系统兼容，如 Linux、Windows Server 等。

集成需求：可能需要与其他软件或服务（如社交媒体登录、云存储服务等）进行集成，因此需要考虑软件之间的兼容性。

(6) 网络环境

稳定性：一个稳定且安全的网络环境是必要的，以确保数据可以实时同步和备份。

安全性：保障数据传输的安全性，如使用 SSL/TLS 加密通信。

(7) 数据安全

加密：系统应具备数据加密功能，以防止数据被非法访问。

备份和恢复：定期备份数据，并能够在数据丢失或损坏时恢复。

(8) 用户界面

友好性：系统的操作界面应友好易用，以便用户能够快速上手，提高使用体验。

博客网站的运行参数要求涉及多个方面，企业或开发者在选择时应根据自身的具体需求和预算进行综合考虑。随着技术的发展，一体化的解决方案越来越受到青睐，因为它们能够提供更加全面和智能化的服务。在考虑博客网站系统时，也应关注系统的数字化和智能化程度，以及是否能够支持组织的长期发展和变革需求。

3.9 保密性需求

为防止系统因操作失误或恶意行为对数据完整性、系统稳定性及用户隐私造成危害，博客系统（CSCI）必须满足以下保密性与安全性相关要求。

3.9.1 误操作防护机制

- 系统必须在执行高风险操作（如删除用户、修改权限、删除文章、封禁账户

等）前进行显式确认，防止误触；

- 每类高风险操作需具备防重复提交机制，例如：操作后按钮冷却时间、防抖处理；
- 若非当前操作人拥有执行权限，系统必须拒绝该请求并返回错误提示，防止无效动作发生。

3.9.2 身份验证机制

- 所有进入系统的用户必须通过有效的身份验证流程，包括密码校验与（可选）二次验证码校验；
- 系统应阻止非法请求（如未登录用户访问后台、权限不足用户修改他人数据等）；
- 登录失败超过设定次数，应启用验证码验证，防止暴力破解攻击。

3.9.3 权限控制机制

- 系统需在用户请求操作前进行权限验证，确认其是否具有访问或操作资源的权利；
- 对“仅自己可见”“仅粉丝可见”等权限状态的文章，系统必须进行访问校验，禁止未授权用户查看；
- 普通管理员不得修改超级管理员信息，权限应层级隔离。

3.9.4 操作审计机制

- 系统需对所有敏感操作（如删除文章、修改权限、登录失败等）进行日志记录，便于问题追踪；
- 所有管理员操作日志应存档保留不少于 90 天，以便安全审计。

3.9.5 数据保护机制

- 所有用户密码在数据库中必须加密存储；
- 敏感数据（如邮箱、手机号）在页面展示时应进行脱敏处理；

•系统数据传输过程需启用 HTTPS，防止中间人攻击。

3.10 保密性和私密性需求

为保障用户隐私数据安全、确保系统运行的保密性与私密性，博客网站系统（CSCI）应满足如下保密性与私密性相关的需求：

3.10.1 保密性/私密性运行环境要求

•博客系统应部署于具备访问控制权限的服务器环境中；

•部署环境应支持 HTTPS 协议，以加密传输所有客户端与服务器之间的敏感数据；

•后台管理系统仅对授权用户（如管理员）开放，并通过身份认证及权限控制保护访问。

3.10.2 保密性和私密性的类型与程度

•保密性要求：防止用户敏感信息（如邮箱、手机号、IP 地址、私密文章等）在未授权的情境下泄露；

•私密性要求：允许用户自定义文章的可见范围（如仅自己可见、仅粉丝可见），并保证该设定严格生效；

•系统应限制数据库直接访问，仅通过受控后端接口进行数据操作。

3.10.3 面临的保密性/私密性风险

•未授权用户访问敏感文章或评论；

•数据库信息泄露（如通过 SQL 注入、越权操作）；

•网络通信被中间人窃听；

•管理员误操作导致用户隐私数据泄露或误封账号；

•用户密码泄露导致账户被冒用。

3.10.4 降低风险所需的安全措施

•所有敏感数据应通过加密（如密码哈希加密）存储；

•前后端通信采用 TLS/SSL 保障加密通道；

•每个用户操作需经过权限验证，避免水平或垂直越权；

•后台敏感操作应有双重确认机制（如删除、封禁等）；

•对异常行为（如频繁登录失败）应有日志记录与警告机制；

•用户登录、注册、敏感信息修改等接口应具备验证码防护和限流机制。

3.10.5 保密性/私密性相关政策

•系统应遵守《中华人民共和国网络安全法》《数据安全法》《个人信息保护法》等法规；

•用户信息（如头像、昵称、手机号、邮箱）仅在用户授权或用于功能实现的前提下展示；

•管理员不得私自浏览、导出或滥用用户数据，后台操作行为需记录并审计。

3.10.6 审计机制

•系统应记录所有访问敏感资源的行为日志（如访问私密文章、修改权限）；

•管理员操作日志应详细记录操作时间、操作者、目标对象、操作类型；

•日志数据应保存至少 90 天并定期备份，以供安全审计和追责。

3.10.7 确证/认可准则

•系统上线前应通过完整的安全性测试，包括但不限于权限测试、越权测试、SQL 注入测试等；

•系统部署后，至少每季度进行一次安全审查；

•若涉及第三方数据处理，应确保其遵循同等的保密性/私密性标准。

3.11 CSCI 环境需求

类别	环境	计算机硬件需求	操作系统
服务器	交互服务器	CPU：双核及以上； 内存：8GB 以上； 存储空间：8GB 以上	Windows7 及以上、Linux
客户端	Web 客户端	CPU：双核及以上 内存：1GB 及以上； 存储空间：8GB 及以上	Andriod、Linux、Windows7 及以上

3.12 计算机资源需求

本条应分以下各条进行描述。

3.12.1 计算机硬件需求

类别	配置要求	说明
处理器（CPU）	双核及以上	满足高性能需求
内存	4GB 及以上	保证多用户访问下系统稳定运行
存储设备	8GB 及以上	支持数据持久化、日志、缓存、用户上传资源
辅助存储	SSD 固态硬盘（可选）	提高数据库和系统的访问速度
输入/输出设备	显示器、键盘、鼠标、移动终端适配自身交互方式	适应不同平台管理与使用
通信/网络设备	支持 Wi-Fi	保证数据通信安全与高可用性

3.12.2 计算机硬件资源利用需求

类别	最大许可使用能力	推荐使用上限	说明

处理器（CPU）	双核至八核（多核心并行任务支持）	不超过 85%	高并发访问、峰值使用，推荐预留缓冲防止阻塞
内存	4GB～32GB	不超过 80%	避免内存溢出，保障缓存、数据库及中间处理稳定性
存储设备	8GB～2TB	不超过 90%	高容量文件上传、日志积压等情况须预留空间
辅助存储	SSD 固态硬盘（可选）	不超过 85%	数据库频繁访问时负载较高，推荐使用 RAID 或分区机制以分担压力
输入/输出设备	常规输入设备+移动端适配	不超过 70%（交互频率）	保障多用户同时交互流畅，尤其在高峰期需避免延迟
通信/网络设备	千兆以太网 / Wi-Fi	带宽使用不超过 75%	上传下载、前后端通信及数据库同步，保持带宽余量以防抖动或中断

测量资源利用的条件：

(1) CPU 利用率通过 Linux top/Windows 任务管理器观察。

(2) 内存占用参考 free/Task Manager 内存指标。

(3) 磁盘占用定期通过 df 或系统盘监控工具检测。

(4) 网络带宽利用率通过交换机监控端口或 iftop/nload 工具测量。

(5) 以上数据基于每日平均访问并发 200 次、操作任务每小时 500 次的标准负

载条件测试得出。

3.12.3 计算机软件需求

本系统的开发与运行依赖多种计算机软件资源，涵盖操作系统、数据库管理系统、前后端开发环境、通信/网络中间件、测试软件及实用工具。下表列出了各项软件的名称、版本、来源及用途说明，确保系统在指定环境下稳定运行并便于维护。

软件类别	软件名称	版本/要求	文档引用	用途说明
操作系统	Microsoft Windows 10 / 11	64 位专业版或企业版	微软官网	当前开发与测试环境，兼容主流软件及工具
数据库管理系统	MySQL	8.0 或以上	MySQL 官方文档	用于用户数据、日志数据存储
通信/网络中间件	无（Flask 自带服务）	——	——	前后端通过 Flask 提供的 HTTP 接口通信
	Apache HTTP Server/Nginx	最新稳定版	Apache 官网及 Nginx 官网	用于处理前后端通信、负载均衡
实用软件	Git	2.30 或以上	Git 官网	源代码版本管理
	Visual Studio Code	1.60 或以上	VS Code 官网	用于前端 Vue 项目的开发与调试
	Python	3.8 或以上	Python 官网	脚本开发、测试工具、服务编排
	PyCharm	2021 或以上	PyCharm 官网	编写、调试和管

				理基于 Flask 的后端服务代码
测试软件	Apipost	最新稳定版	Apipost 官网	用于后端 API 接口测试与调试
生产用软件	Docker	20.10 或以上	Docker 官网	可选用于后期部署集成，便于环境统一和快速交付

3.12.4 计算机通信需求

本系统基于 B/S 架构，前端页面通过 HTTP/HTTPS 协议与后端服务进行通信，后端与本地 MySQL 数据库之间通过标准数据库连接协议进行数据交互。系统通信需求主要体现在前端访问后端接口、后端与数据库之间的数据传输、以及本地网络通信的稳定性保障。其详细通信要求如下表所示：

通信要素	通信需求描述
连接地理位置	当前系统组件部署在本地局域网环境中。后期数据库计划部署至云服务器（如阿里云、腾讯云等），前端用户可从任意地点通过浏览器远程访问系统，后端通过公网连接云数据库。
网络配置与拓扑结构	系统采用典型的“浏览器—服务器—数据库”拓扑结构：前端浏览器 → 后端 Flask 接口 → 云数据库。当前阶段数据库本地部署，后期调整为云端部署后，需配置数据库公网地址及访问权限。
传输技术	使用基于 TCP 的 HTTP 协议进行通信，接口遵循 RESTful API 标准，数据采用 JSON 格式传输。数据库连接将使用 MySQL 的标准协议（TCP 3306 端口），远程连接时采用加密传输（SSL/TLS）。
数据传输速率	局域网内通信速率不少于 100 Mbps，公网通信速率

	依赖云服务器与客户端带宽，建议带宽不低于 10 Mbps，以保障访问体验。
通信网关	当前开发环境下未配置网关。正式部署阶段，可配置 Nginx 作为前端反向代理，实现 HTTPS 通信、安全认证、负载均衡等功能；云数据库服务需开启远程访问白名单控制及身份认证。
系统使用时间要求	系统需支持 24 小时持续运行能力。云部署后，需保障数据库服务同样具备高可用性。
传输数据类型和容量	JSON 格式数据、图片文件，单次传输数据量通常在 5MB 以内。上传图片等大文件时，系统支持文件大小限制、压缩与分块机制，防止通信瓶颈
响应时间限制	100ms
数据的峰值	每小 时 1000 次数 据传 输的 峰值
通信诊断功能	开发阶段通过 Apipost 工具对接口进行测试；后端 Flask 提供日志记录机制；部署后可借助 Nginx、云监控平台（如阿里云 CloudMonitor）进行网络状态监控和故障排查。

3.13 软件质量因素

本系统为典型的基于 B/S 架构的博客网站，旨在提供良好的用户体验、高效的交互逻辑、可扩展的功能模块以及可靠的后台支持。为保证系统软件质量，在系统开发与运行过程中，对以下几个软件质量因素提出如下需求：

（1）功能性

系统应完整实现博客网站的核心功能，包括但不限于：用户注册与登录、文章发布与管理、评论交互、权限控制、内容展示、搜索功能等。所有功能需在符合业务逻辑的前提下稳定运行，确保数据传输正确、前后端联动准确。

（2）可靠性

系统在大多数情况下（正常、异常输入、恶意请求）应返回一致且准确的处理结果，后端需具备基础容错能力（如数据库连接失败的重试机制、异常日志记录等）。系统每日连续运行可靠性应达到 99.9%。

（3）可维护性

代码结构应清晰，前后端模块需分离，具有良好的注释和文档。应支持功能模块化与热更新，便于后期修改、调试和优化。项目中使用的框架和库版本应规范记录，确保长期维护可控。

（4）可用性

系统应支持 7×24 小时持续运行，用户界面响应时间在普通网络条件下不超过 3 秒。前端页面应自适应主流浏览器（Chrome、Edge、Firefox）和不同分辨率，后端 API 响应速度控制在 1 秒内。

（5）灵活性

应支持快速增加新功能模块，例如新增标签管理、用户通知、文章分类等，具备较高的扩展性和业务适应性。系统配置参数（如数据库地址、端口、秘钥等）应集中管理、便于修改。

（6）可移植性

系统应支持从 Windows 环境迁移到 Linux 环境，并兼容多种主流浏览器。数据库后期应支持迁移至远程云数据库（如阿里云、腾讯云），前后端可通过配置切换实现部署环境迁移。

（7）可重用性

系统中常用组件（如用户权限校验模块、通用接口封装、数据库操作工具等）应具备可复用性，可在其他 Web 应用中复用或作为独立中间件抽离使用。

（8）可测试性

系统应提供完整的接口文档（如基于 Apipost 管理），支持自动化接口测试和单元测试。各模块功能应具备独立可测试性，后端代码应具备 80% 以上的测试覆盖率。

（9）易用性

前端应提供简洁直观的用户界面，支持中文提示、按钮引导和异常反馈机制，普通用户应在首次使用系统时即可完成核心操作。后台管理应具备图形化界面辅助使用。

（10）安全性

系统应具备基本的安全防护机制，包括：

- 登录接口采用 Token 验证或 Session 控制防止未授权访问；
- 所有输入均进行服务器端校验，防止 SQL 注入、XSS 攻击；
- 重要接口应使用 HTTPS 加密通信；
- 密码信息在数据库中必须加盐加密存储；
- 后台管理界面应限制仅管理员角色访问；
- 提供基础操作日志记录与异常行为监控能力；用户登录状态应定期过期刷新，防止盗用。

3.14 设计和实现的约束

本系统为典型的基于 B/S 架构的博客管理系统，系统前端采用 Vue 框架运行于浏览器环境中，后端基于 Flask 框架运行于本地 Python 环境，数据库使用 MySQL 并部署在本地数据库服务器上，未来计划迁移至远程云数据库（如阿里云）。系统的设计和实现需遵循如下约束条件：

a. 特殊体系结构与资源使用约束

- **体系结构：**系统基于 B/S（Browser/Server）架构设计，前后端分离，采用 RESTful API 进行通信。

•**数据库配置项：**MySQL 作为系统唯一数据库，要求支持 UTF-8 编码，具备事务机制和远程访问支持能力。未来支持迁移至云数据库平台（如阿里云 RDS）。

•**组件复用：**系统应尽可能复用标准组件和框架模块，如：

•Vue 的组件化机制（如 ElementPlus UI 组件库）

•Flask 的蓝图机制和中间件支持

•**需方资源使用：**开发阶段使用需方提供的本地测试环境（Windows 操作系统），包括 VSCode（前端开发）、PyCharm（后端开发）、Apipost（接口测试工具）等。

b. 设计与实现标准

•**编程语言要求：**

•前端使用 JavaScript（ES6+）配合 Vue 框架；

•后端使用 Python（版本 ≥3.8）配合 Flask 框架；

•数据库脚本使用标准 SQL 语言。

•**通信格式标准：**前后端采用 JSON 格式 进行数据传输；

•**数据命名标准：**数据库字段和接口字段命名采用统一小写加下划线风格（snake_case），避免大小写混用；

•**接口规范：**后端接口符合 RESTful 设计风格，具有统一的请求响应结构和状态码规范；

•**前端样式标准：**统一使用 ElementPlus 样式规范，布局响应式兼容多设备。

c. 可扩展性与灵活性

为适应系统后期的功能扩展、部署调整和数据增长，系统设计需具备如下灵活性与可扩展性要求。

•**功能扩展：**

•系统模块应模块化设计，支持热插拔式扩展（如新增评论审核、标签管理等功能）；

•接口设计预留扩展字段，避免频繁修改结构。

•部署灵活性：

•后端可部署于本地服务器或云服务器；

•数据库支持远程连接配置，支持快速迁移至云平台；

•支持容器化部署（Docker）以实现跨平台运行。

•并发与性能：

•系统应支持用户并发量动态扩容能力，如未来部署 Nginx + Gunicorn/Wsgi 等服务器；

•数据库索引和分页机制应具备处理大数据量能力。

•技术更新适应性：

•框架版本更新应可控，代码尽量与上游兼容；

•前端构建工具（如 Vite/Webpack）应支持快速重构。

•安全性扩展：

•安全机制模块化设计，后期可集成验证码、双因素认证、日志审计等扩展能力。

3.15 数据

3.15.1 输入数据

系统输入数据主要来源于用户操作及管理端输入，具体如下表。

输入来源	数据内容描述	格式类型
用户注册	用户名、密码、邮箱等	JSON + 图片文件
用户登录	用户登录	JSON
文章发布	文章标题、正文内容、标签、分类、封面图	JSON + 图片文件（Base64 或表单上传）
评论提交	评论内容、所属文章、用户信息等	JSON
点赞提交	所属文章、用户信息等	JSON
收藏提交	所属文章、用户信息等	JSON

用户信息更新	昵称、头像、个人简介等	JSON + 图片文件
搜索/筛选	关键词、筛选条件（标签、时间、用户等）	JSON 查询参数
管理员注册	名字、密码、头像等	JSON + 图片文件
管理员登录	名字、密码	JSON
管理员操作	超级管理员对普通管理员的增/删/改操作、用户管理、文章审核、评论管理等操作指令	JSON + 图片文件/JSON

3.15.2 输出数据

系统向用户或管理端返回的数据包括：

输出对象	数据内容描述	格式类型
页面展示	文章内容、评论信息、用户资料、热门标签等	JSON
登录结果	验证状态、登录成功与否、会话 Token（如 JWT）	JSON
操作反馈	各类操作结果反馈（如发布成功、删除成功等）	JSON
管理端界面	用户列表、文章审核列表、评论列表等管理数据展示	JSON
搜索结果	满足条件的文章列表及相关内容	JSON

3.15.3 数据处理能力（吞吐与性能）

类别	处理能力要求
并发支持	支持不低于 100 并发用户同时访问与操作
数据响应时间	页面加载响应 ≤ 2 秒，接口响应时间 ≤ 1 秒
接口调用频率	支持每秒 ≥ 300 次查询类 API 请求
高峰处理	在高峰期（如热点文章访问）支持峰值 5000 次访问/分钟

3.15.4 数据存储与安全管理

(1) 数据库类型：MySQL，支持事务、安全访问控制、索引优化。

(2) 数据一致性：所有关键数据操作需具备事务一致性保障（ACID 特性）。

(3) 数据备份与恢复：提供定时自动备份策略，支持按天/按周备份及一键恢复机制。

(4) 远程访问支持：支持将数据库部署至云端（如阿里云），通过安全通道进行远程数据访问。

(5) 敏感信息加密：如用户密码需进行加盐哈希处理存储，接口数据传输使用 HTTPS 保障传输安全。

(6) 日志数据记录：系统操作日志与访问日志持久化存储，便于审计与回溯。

3.16 操作

本节描述系统在常规操作、特殊操作、初始化操作和恢复操作等方面的需求。考虑到本系统为一个基于 B/S 架构的博客管理系统，具备用户注册、登录、文章发布、评论互动、权限管理、数据审核、状态切换、用户行为记录等多模块复杂交互逻辑，特提出如下操作要求。

3.16.1 常规操作要求

系统需满足以下常规操作场景的高可用性与稳定性：

1. **用户操作：**

•注册并填写个人信息（用户名、密码、昵称、性别、手机、邮箱、头像等）；

•登录支持多种方式（用户名/手机号/邮箱 + 密码，或邮箱验证码）；

•修改密码（输入原密码或者邮箱验证）、忘记密码找回（邮箱验证）、退出登录；

•查看和修改个人信息；

•用户无需登录即可浏览、搜索文章或用户；

•撰写、修改、删除文章，支持分类、设置权限（公开、私有、粉丝可见）；

•点赞、评论、收藏、查看记录等互动行为；

•查看与管理个人发布内容及评论、点赞、收藏历史。

2. **普通管理员操作：**

•登录后台管理页面，查看和修改个人信息；

•审核用户权限与状态（禁言、恢复发布权等）；

•超级管理员可增删查管理员；

•管理用户文章状态（发布、屏蔽、删除）；

•评论状态管理（隐藏、恢复、删除）；

•查看和修改个人信息查询统计文章/评论点赞收藏数；

•审核被举报内容，决定处理方式；

•支持文章权限切换（公开、私有）。

3. **系统后台操作：**

•日志记录：记录所有操作行为日志（如登录、修改数据、异常尝试）；

•数据统计：用户活跃度、文章数量、点赞量等统计数据可导出；

•自动缓存与分页机制以保证前端高效响应。

3.16.2 特殊操作要求

系统应具备以下特殊操作支持，以应对突发情况与系统维护：

(1) 用户举报功能：支持用户举报他人评论，系统需生成对应举报记录，并推送给管理员审核；

(2) 系统审计：所有后台管理操作应记录日志，并允许追踪；

(3) 异常登录检测与限制：支持基于 IP/设备指纹的安全策略，异常登录行为触发验证码验证或二次验证；

(4) 操作限频机制：对评论、点赞、登录尝试等操作设置频率限制，防止恶意刷操作；

(5) 内容屏蔽与恢复：管理员可临时隐藏文章或评论，待复审后再恢复显示；

(6) 账号封禁与回退机制：禁用用户后其数据仍可留存，恢复账号后数据可继续展示；

(7) 权限即时生效：用户权限更改（如禁言、禁止发布）需立即对系统行为生效。

3.16.3 初始化操作要求

系统在首次部署或新节点部署时应具备如下初始化能力：

(1) 初始化数据库结构与默认数据（如初始超级管理员账号、默认分类标签）；

(2) 自动检测当前环境是否已初始化，避免重复配置；

(3) 提供初始化脚本和界面（如安装向导），指导部署步骤；

(4) 创建系统日志目录、配置缓存路径等基础资源；

(5) 可配置管理员邮箱，用于接收系统重要通知（如用户举报、宕机预警等）。

3.16.4 恢复操作要求

为保障系统稳定性和可用性，应支持以下恢复机制：

1. **数据备份与恢复：**
   - 系统支持周期性自动备份用户、文章、评论等关键数据；
   - 支持云数据库部署，保障远程访问与云端数据安全；
   - 一旦发生异常或数据丢失，可通过备份数据一键恢复；
2. **删除机制：**
   - 管理员可查看并恢复已删除的数据（如文章、评论等）；
3. **系统容错处理：**
   - 针对崩溃、异常终止等情况自动记录崩溃日志；
   - 用户操作中断后自动保存草稿内容，下一次登录可继续编辑。
4. **账号恢复机制：**
   - 被封禁或误禁的账号可通过管理员操作恢复；
   - 邮箱验证支持重新绑定和验证以找回账号权限。

3.17 故障处理

本博客系统在运行过程中，必须具备一定的容错能力和故障响应能力，确保在出现软硬件故障时系统能够及时发现问题、准确提示错误，并采取有效补救措施以保障用户体验和数据安全。

3.17.1 定义故障类型和级别

本系统可能出现的故障类型和级别如下：

- ✧ 硬件故障：严重故障
- ✧ 软件故障：重要故障
- ✧ 网络故障：一般故障

3.17.2 软件系统类故障分析

本系统可能出现的软件类故障主要包括以下几种类型：

(1) 后端服务异常：如程序崩溃、空指针引用、逻辑错误等导致服务不可用；

(2) 数据库访问异常：如连接失败、数据查询错误、主键冲突、事务失败等；

(3) 接口调用失败：前端与后端、后端与第三方接口之间调用异常；

(4) 配置错误：配置文件缺失、参数错误等导致服务启动失败或运行异常；

(5) 前端运行错误：JavaScript 报错、资源加载失败、UI 卡死等；

(6) 权限控制异常：如访问未授权页面、权限未校验、角色配置错误等。

3.17.3 错误信息说明

系统在出现错误时，应向用户或管理员显示明确的错误提示信息，错误信息应包括但不限于以下内容。

(1) 用户可见错误信息（前端）：

“页面加载异常，请刷新重试。”

“无权限访问此页面，请登录或联系管理员。”

“服务器发生内部错误，请稍后再试。”

“数据库连接失败，请联系管理员。”

(2) 管理员可见日志信息（后台）：

NullPointerException at BlogController.java:56

Database connection timeout - jdbc:mysql://localhost:3306/blog

API Error 502: Failed to fetch user info from AuthService

Unauthorized access attempt from IP 192.168.1.5

3.17.4 设计故障处理策略

针对上述错误类型，系统可采取如下故障处理措施：

(1) 对用户可见错误提供友好提示，并避免系统崩溃；

目标：提升用户体验，防止因错误导致整个应用崩溃。

实现方式：

- ✧ 使用统一的错误页面（如 HTTP 404、500）或组件，提示用户“系统繁忙，请稍后重试”等友好语句。
- ✧ 前端使用 try-catch 包裹重要逻辑，配合全局错误捕捉（如 Vue/React 的 error boundary）。
- ✧ 后端使用全局异常处理机制(如 Flask 的 errorhandler、Spring 的 @ControllerAdvice）拦截错误，返回结构化的 JSON 响应，避免将堆栈信息暴露给用户。
- ✧ 保持主线程/主流程稳定，局部错误不影响全局运行。

(2) 对后端服务错误自动记录日志，并可自动重启服务（使用守护进程如 Supervisor、systemd）；

目标：增强后端服务的自愈能力，缩短故障恢复时间。

实现方式：

- ✧ 使用守护进程管理服务进程，如 Supervisor、systemd、PM2：
- ✧ 在服务发生异常退出前通过日志模块记录错误(如 Log4j、Winston、Python logging）。
- ✧ 日志中记录时间、服务名、错误堆栈、请求参数、调用链路等。
- ✧ 可扩展为与监控系统（如 Prometheus + Alertmanager）联动，主动触发告警或自动恢复。

(3) 出现数据库错误时，可进行重试机制或切换至备用数据库；

目标：避免数据库临时故障导致请求失败，确保高可用。

实现方式：

- ✧ 使用数据库连接池（如 SQLAlchemy）提供连接可用性检查。
- ✧ 对于临时连接失败或超时类错误，设置有限次自动重试机制，如重试 3 次，每次间隔 200ms~1s。

- ✧ 支持主从切换或多活架构，配置备用数据库连接信息，当主库连接失败时自动切换。对写操作需考虑数据一致性策略，如通过中间队列（Kafka）缓冲写入，待主库恢复后重放。

(4) 网络调用失败时，可延迟重试并记录错误以便后续修复；

目标：降低对外部服务临时故障的敏感性，提升系统稳定性。

实现方式：

- ✧ 实现带退避策略的重试机制（如指数退避 exponential backoff + jitter），避免请求风暴。
- ✧ 对多次失败的调用记录入日志或存入队列，供后续人工或自动补偿处理。
- ✧ 使用熔断器（如 Netflix Hystrix）在调用方自动熔断连续失败服务，避免持续调用无响应接口。
- ✧ 可设置超时时间，防止长时间阻塞主业务流程。

(5) 配置错误时，禁止启动服务并给出明确报错信息供管理员修改；

目标：确保服务在配置异常时不以“错误状态”运行，避免引发更大问题。

实现方式：

- ✧ 在服务启动阶段进行配置校验，若配置缺失或非法，则抛出异常终止启动。
- ✧ 提供清晰明了的启动日志或控制台输出，标明配置错误项、期望格式、建议修改方式。
- ✧ 可配合配置中心（如 Apollo）实现实时配置校验与推送机制。

(6) 在前端出现资源加载或脚本错误时，可自动刷新页面或回退至默认状态；

目标：避免因前端脚本崩溃导致页面不可用。

实现方式：

- ✧ 使用前端全局错误监听（如 window.onerror）监控脚本错误。
- ✧ 发生关键资源加载失败（如 JS、CSS）时，可设置超时重载页面或加载默认资源。

- ✧ 引入前端监控 SDK（如 Sentry、Fundebug）上报错误堆栈，分析错误来源和影响范围。
- ✧ 对用户操作频繁页面可设置本地缓存和状态回滚机制，避免错误后丢失数据。

(7) 所有异常均记录入日志系统，并发送警报通知管理员进行处理。

目标：建立完备的可观测性体系，实现快速定位和响应问题。

实现方式：

- ✧ 日志系统使用结构化日志，分级管理（如 info/warn/error），便于检索与分析。
- ✧ 配合日志聚合工具（如 ELK、EFK、Loki）进行集中管理。
- ✧ 异常日志可接入监控告警系统，配置通知渠道。
- ✧ 可配置错误处理工单系统，便于问题分派与跟踪处理进度。

3.18 算法说明

3.18.1 文章推荐算法

本系统采用基于内容的推荐算法，通过分析用户历史阅读行为与文章内容之间的相似性，为用户推荐与其兴趣相关的文章。

算法的主要原理为：采用 TF-IDF 将文章内容进行向量化，再将用户兴趣建模为向量，通过余弦相似度计算用户与文章之间的相似性，并按照相似度排序推荐文章。

用途：根据用户浏览记录和文章标签，为用户推荐相关文章。

输入：用户历史浏览记录、文章标签向量。

输出：推荐文章列表（按相关度排序）

详细步骤如下：

1. 构建文章内容向量（TF-IFD）

设文章集合为 D={d1,d2,...,dn}，每篇文章 di 用词向量表示：

$$\vec{d_i} = (w_{i1}, w_{i2}, ..., w_{im})$$

其中 w_{ij} 表示词 tj 在文章 di 中的 TF-IDF 权重：

$$w_{ij} = tf_{ij} \cdot \log\left(\frac{N}{df_j + 1}\right)$$

其中 tfij：词 tj 在文章 di 中出现的次数

dfj： 词 tj 在多少篇文章中出现

N： 总文章数

2. 构建用户兴趣向量

用户兴趣向量 u⃗为用户已读文章的平均内容向量：

$$\vec{u} = \frac{1}{|R_u|} \sum_{d_i \in R_u} \vec{d_i}$$

其中 Ru：用户 u 浏览过的文章集合

3. 余弦相似度计算：

对候选文章 dj，计算其与用户兴趣向量之间的余弦相似度：

$$\text{Similarity}(u, d_j) = \frac{\vec{u} \cdot \vec{d_j}}{\|\vec{u}\| \cdot \|\vec{d_j}\|}$$

点积计算两个向量的相似度，范围为 [0,1]

4. 排序与推荐

对所有未读文章 dj，计算相似度后按得分排序，返回 Top-k 推荐列表：

$$\text{Recommend}(u) = \text{Top-}k\left(\text{Similarity}(u, d_j)\right)$$

3.18.2 文章热度算法

本系统采用基于热度的推荐算法，根据文章的访问量、点赞量、评论量等指标，计算文章的综合热度得分，并根据该热度进行排序推荐。

为了更真实地反映内容近期的热度情况，本算法引入时间衰减因子，使得旧文章热度逐渐下降、近期热门内容获得优先展示。

算法的主要原理为：将文章的各类交互数据（如浏览、点赞、评论）进行加权整合，形成热度得分，并结合文章发布时间进行时间衰减处理。最终按照热度分数高低排序，向用户推荐当前流行的内容。

用途：在用户未登录、无历史行为、或系统冷启动阶段，根据整体热度向所有用户推荐当前最受欢迎的文章。

输入：文章的浏览量、点赞数、评论数、发布时间、当前时间。

输出：推荐文章列表（按热度得分排序）。

详细步骤如下：

1. 基础热度得分计算

每篇文章的初始热度由其访问量（Views）、点赞量（Likes）、评论数（Comments）组成：

$$\text{RawScore}(a) = \alpha \cdot V_a + \beta \cdot L_a + \gamma \cdot C_a$$

其中 a：文章编号

Va：文章访问次数

La：点赞数

Ca：评论数

α，β，γ：各项指标的权重，满足：

$$\alpha + \beta + \gamma = 1$$

2. 时间衰减模型的建立

为使热度随时间自然衰减，引入指数型时间衰减因子：

$$\text{Decay}(a) = e^{-\lambda \cdot (T - t_a)}$$

3. 最终热度得分

综合初始热度和时间衰减，得到文章的最终热度得分：

$$\text{HotScore}(a) = \text{RawScore}(a) \cdot \text{Decay}(a)$$

4. 排序与推荐输出

系统根据每篇文章的热度分数排序后，向用户展示 Top-k 热门文章：

$$\text{Recommend}(u) = \text{Top-}k\,(\text{HotScore}(a))$$

3.18.3 文章搜索算法

本系统采用关键词匹配搜索算法，通过构建倒排索引结构，实现对文章内容的高效搜索与精准匹配，提升用户在博客系统中的检索体验。

算法的主要原理为基于关键词反向索引技术，通过预处理将每篇文章的关键词与其所在文章建立映射关系，实现对关键词的快速查找。用户输入搜索词时，

系统根据倒排索引快速定位包含该关键词的文章列表，并按相关性进行排序返回。

输入：用户输入的查询关键词 Query

输出：文章匹配列表（按匹配度排序）

详细步骤如下：

1. 倒排索引构建

   对文章内容进行分词、去停用词等预处理，构建关键词到文章的倒排映射表。

$$\text{InvertedIndex}[term] = \{doc\_id_1, doc\_id_2, ..., doc\_id_n\}$$

2. 查询解析与匹配

   对用户输入的 Query 进行分词与清洗，在倒排索引中查找所有包含关键词的文档集合，计算每篇文章与查询词的匹配度（匹配词数量 / 查询词数量）。

3. 匹配度排序

   根据文章与查询的匹配度计算结果，按照相关性排序：

$$\text{Score}(doc) = \sum_{t \in query} \text{tf-idf}(t, doc)$$

   也可结合词频（TF）和逆文档频率（IDF）进行加权排序。

4. 搜索过程

   当用户输入查询词时，系统通过倒排索引定位包含相关关键词的文章，计算每篇文章的匹配度，对结果进行排序并返回前 N 条最相关内容。

3.18.4 信息加密算法

本系统采用基于 Werkzeug 的哈希加密算法的加密算法，用于保护用户账户密码等敏感信息的安全。通过不可逆的哈希加密技术，即使数据泄露，也能有效防止明文密码暴露，保障系统整体的安全性。

算法的主要原理为使用 PBKDF2 哈希算法（基于 SHA256）对密码进行不可逆加密。每次加密都会引入随机盐值（Salt），确保即使两个用户使用相同密码，其加密结果也完全不同，从而防止彩虹表攻击。

输入：用户原始密码（明文）

输出：加密后的密码（密文）

详细步骤如下：

1. 生成加密密码，使用 generate_password_hash 将用户原始密码加密：

```
from werkzeug.security import generate_password_hash

hashed_password = generate_password_hash(raw_password)
```

2. 密码校验，在用户登录时使用 check_password_hash 进行密码验证：

```
from werkzeug.security import check_password_hash

is_valid = check_password_hash(hashed_password, input_password)
```

3.19 有关人员需求

本系统在开发、部署和运行过程中，涉及若干与 CSCI 使用和支持相关的人员需求，具体说明如下：

1. 人员数量与角色职责

人员角色	所需人数	技能要求	责任期	培训需求
系统用户	不定	具备基本的网页操作能力，能够阅读与发布文章、评论	系统全生命周期	系统全生命周期
系统管理员	2 人	熟悉后台管理、系统配置和日志审查	维护期全程	初始培训与操作文档
技术	2 人	掌握故障排查、数据库维	运维期	提供系统部

支持人员		护、服务部署		署说明和维护手册

2. 培训与帮助功能需求

(1) 系统应提供可视化用户帮助系统（如“使用说明”、“功能提示”）与内提示文字。

(2) 管理端应配套操作视频或文档手册，便于非技术人员掌握使用流程。

(3) 支持用户首次使用时的引导模式（如“欢迎引导页”）。

3. 人因工程与人为错误防控需求

系统设计考虑用户操作的易用性与错误预防，具体要求如下：

(1) 颜色提示：错误信息以红色提示框显示，持续时间至少 5 秒，同时支持手动关闭;

(2) 表单校验：对用户输入如标题、内容、标签等进行实时验证，避免留空或格式错误;

(3) 物理位置优化：系统界面按钮如“发布”“删除”等关键操作放置在分离位置，避免误触;

(4) 确认机制：对不可逆操作（如删除文章）提供二次确认。

3.极端条件下的人为错误应对

(1) 对于系统长时间无响应、网络中断等极端环境，需提供错误页面与自动恢复机制;

(2) 用户频繁刷新、误操作等行为应由系统限流并提示，如“操作过于频繁，请稍后重试”。

3.20 有关培训需求

本系统为确保用户和管理人员能够正确使用系统功能，设定如下培训要求：

1. 面向用户的培训

本系统面向普通用户（包括博客作者和浏览者）提供友好的操作引导与帮助功能，帮助其快速了解系统操作流程，具体措施包括：

(1) 用户首次登录系统时，将自动弹出交互式新手引导，包括账户管理、文章浏览、收藏、评论等基本操作说明。

(2) 设置“常见问题”页面，整理用户易混淆或常见的使用问题，并提供详细解答。

(3) 对于用户输入错误、无效操作等行为，系统通过清晰的提示语、颜色标识、图标符号等方式给出引导与建议。

2. 面向管理员的培训

本系统为后台管理员提供系统维护与管理方面的培训支持，帮助其掌握网站日常管理与监控操作，包括：

(1) 编写《博客后台管理使用手册》提供详细操作文档，涵盖用户账户管理、评论审核、文章管理、标签配置、系统设置等功能模块。

(2) 说明各类管理员权限划分与操作边界，引导正确使用高权限操作，避免误删等严重后果。指导管理员定期备份数据库、日志文件，并说明系统异常后快速恢复数据的流程。

3. 面向技术员的培训

为保障系统的稳定运行与后期维护，系统面向技术支持与运维人员提供部署、更新与排障方面的专业培训，内容包括：

(1) 编写《系统部署与维护手册》，包含系统所需运行环境配置、依赖包安装、数据库初始化、配置文件说明、接口配置等详细步骤。

(2) 提供标准部署脚本与一键初始化配置，减少部署误差，提高上线效率。

(3) 提供日志分析工具及使用说明，辅助技术人员进行异常排查与性能分析。

3.21 有关后勤需求

为确保本博客系统的正常运行与可持续维护，系统在后勤支持方面提出以下需求：

1. 系统维护与软件支持

系统需配备定期的维护机制，包括服务器运行状态监控、数据库备份、日志清理及安全漏洞修补。应由专人负责系统维护，确保软件持续更新、Bug 快速修复，并提供必要的远程技术支持服务。

2. 部署与运输方式

本系统采用 Web 架构部署，无需实体运输。系统可通过远程服务器上传部署，支持使用自动化脚本（如 Docker、CI/CD 工具）完成快速上线与更新。

3. 供应系统需求

系统运行需依赖稳定的服务器资源、数据库服务（如 MySQL 或 PostgreSQL）、对象存储服务（如阿里云 OSS、MinIO）、以及日志与监控系统（如 ELK、Prometheus+Grafana）。需确保上述软硬件环境持续可用。

4. 对现有设施的影响

系统运行所需服务器可使用现有云主机或本地机房资源部署，不对原有办公设施产生实质性影响。如采用云端部署，仅需保证运维人员具备远程访问权限。

5. 对现有设备的影响

用户侧访问系统仅需具备现代浏览器的终端设备（如 PC、手机或平板），对现有终端设备无特殊硬件要求。管理员使用后台管理系统时，建议使用带有外接键盘与大屏显示器的终端设备以提升操作效率。

3.22 其他需求

3.22.1 安全性需求

说明系统在数据保护、用户隐私、防止非法访问等方面的安全措施，如：

(1) 用户密码加密存储（如采用 bcrypt）

(2) 防止 SQL 注入、XSS、CSRF 等常见攻击

(3) 管理员权限校验机制

(4) 数据访问日志记录与追踪

3.22.2 性能需求

描述系统在响应时间、并发支持、数据处理速度等方面的性能目标，如：

(1) 页面响应时间应小于 2 秒

(2) 同时支持至少 1000 个并发用户在线

(3) 每天支持处理约 2000 篇文章及相关评论

3.22.3 可用性与可靠性需求

强调系统的稳定性和可用率，例如：

(1) 系统年平均可用率不低于 99%

(2) 提供自动故障恢复机制（如服务重启）

(3) 每天自动备份数据，并保留至少 7 天

3.23 包装需求

本系统在交付时应具备良好的可部署性与完整的发布包。软件发布应包含完整的安装程序、使用说明文档、部署配置文件及必要的依赖库，确保用户能够在目标环境中顺利完成安装与配置。系统应支持以压缩包（如 .zip 或 .tar.gz）形式发布，内部应包括：

(1) 系统安装文件及执行入口

(2) 数据库初始化脚本

(3) 配置说明文档

(4) 用户使用手册与管理员操作指南

(5) 升级日志及版本说明文档（如 CHANGELOG.md

(6) 所需依赖环境说明（如 Python 版本、第三方库依赖）

系统应提供图形化或命令行的安装向导，帮助用户完成部署。对于部署到服务器环境的场景，应支持容器化（如 Docker）或一键部署脚本，以便快速上线系统。

3.24 需求的优先次序和关键程度

为确保系统开发顺利进行，并在资源有限的情况下保障关键功能优先实现，本系统将各项需求按其对系统运行的影响程度划分为三个优先级等级：高（关

键）、中（重要）、低（可选），具体如下：

1. 高优先级

这些需求直接关系到系统基本功能与安全性，是系统上线前必须完成的部分。包括：

(1) 用户注册与登录（含身份验证）

(2) 文章发布、浏览与评论功能

(3) 后台管理（用户管理、内容审核、日志管理）

(4) 权限控制与访问管理

(5) 数据存储与备份机制

(6) 用户密码加密、接口权限校验等安全措施

2. 中优先级

此类需求在提升系统智能化、增强用户体验方面具有重要作用，建议在基础功能完成后优先实现。包括：

(1) 基于内容的文章推荐算法

(2) 文章热度排序与相关推荐机制

(3) 站内全文搜索功能

(4) 评论点赞与排序

(5) 管理员操作日志统计与分析

(6) 低优先级（可选增强需求）

这些属于辅助性功能，对系统基本运行无直接影响，根据开发时间与用户反馈可灵活实现。包括：

(1) 用户个人主页装修功能

(2) 夜间模式、界面主题切换

(3) 动态背景与动画效果

4 合格性规定

需求	测试方法
添加新用户到系统	演示+测试+审查
从系统中登录指定用户	演示+测试+审查
发送验证码接口	演示+测试+审查

验证邮箱验证码	演示+测试+审查
更新用户信息	演示+测试+审查
用户退出登录	演示+测试+审查
管理员登录	演示+测试+审查
增加管理员	演示+测试+审查
删掉管理员	演示+测试+审查
列出所有管理员	演示+测试+审查
管理员查看个人信息接口	演示+测试+分析+审查
管理员修改个人信息接口	演示+测试+分析+审查
管理员查看用户列表接口	演示+测试+分析+审查
管理员修改用户状态接口	演示+测试+分析+审查
管理员修改用户权限接口	演示+测试+分析+审查
获取所有文章列表（管理员专用）	演示+测试+审查
获取特定文章详情（管理员专用）	演示+测试+审查
修改文章状态（管理员专用）	演示+测试+审查
修改文章权限（管理员专用）	演示+测试+审查
物理删除文章（管理员专用）	演示+测试+审查
软删除文章（管理员专用）	演示+测试+审查
创建文章（用户）	演示+测试+审查
获取特定用户文章列表（管理员和用户均可）	演示+测试+审查
更新文章（用户）	演示+测试+审查
删除文章（用户）	演示+测试+审查
点赞文章（用户）	演示+测试+审查
取消点赞（用户）	演示+测试+审查
用户创建评论，已经实现父评论功能	演示+测试+审查
用户更新自己的评论	演示+测试+审查
用户删除自己的评论	演示+测试+审查
用户举报评论	演示+测试+审查
用户获取自己发布的所有评论	演示+测试+审查
用户获取自己所有被举报的评论	演示+测试+审查
收藏	演示+测试+审查
取消收藏	演示+测试+审查
获取某个用户的所有收藏	演示+测试+审查
获取某个文章的收藏数	演示+测试+审查
点赞	演示+测试+审查
取消点赞	演示+测试+审查
获取文章的点赞数	演示+测试+审查
获取文章的点赞用户	演示+测试+审查
获取用户的所有点赞记录	演示+测试+审查
用户评论点赞	演示+测试+审查
用户取消点赞	演示+测试+审查
用户获得某个评论的所有点赞数	演示+测试+审查

用户获得某个评论所有点赞的用户	演示+测试+审查

5 需求可追踪性

（1）a 部分

CSCI 模块	主要能力需求	所追踪的系统需求 / 设计目标
前端应用程序	响应式界面、用户交互（浏览、评论、点赞等）	系统功能需求：支持多平台访问，提供良好用户体验
后端应用程序	用户认证、评论管理、数据处理等	系统功能需求：处理用户请求、保障数据正确性和业务逻辑完整性
数据库管理系统	数据存储、查询、更新、一致性保障	系统功能需求：支持海量数据的持久化存储和快速访问
用户认证系统	支持多种登录注册方式、账号验证	系统安全性需求：身份验证、访问控制
内容管理系统	管理文章、评论、点赞收藏等	系统功能需求：内容创建与互动管理

（2） b 部分

系统需求编号	系统需求内容	所涉及的 CSCI 及其需求编号（简写）
SYS-01	系统应支持多种登录方式	用户认证系统（3.4.4）
SYS-02	支持用户注册、登录、资料管理	前端应用程序（3.4.1）、后端应用程序（3.4.2）、用户认证系统
SYS-03	用户可浏览、发表、编辑、删除文章	内容管理系统（3.4.5）、前端、后端
SYS-04	支持用户评论、点赞、收藏功能	内容管理系统、前端、后端
SYS-05	数据必须安全、可追溯、可恢复	数据库管理系统（3.4.3）

SYS-06	所有模块需通过接口协调通信	所有 CSCI（3.6 内部接口）
SYS-07	支持移动设备与浏览器访问	前端应用程序、硬件接口（3.5）
SYS-08	系统应可扩展并支持高并发	后端应用程序、数据库、服务器接口（3.5.2）

6 尚未解决的问题

如需要，可说明软件需求中的尚未解决的遗留问题。

（1）关于用户上传的图片、视频等资源，是否需要统一进行压缩、格式转换、内容审查等处理，目前尚未制定明确策略，可能影响前端加载速度和后端存储压力。

（2）用户发布的评论或文章是否需要自动或人工审核，以及举报后是否由系统自动屏蔽或人工介入，目前尚未形成完整机制，需进一步探讨合规性与平台治理策略。

（3）当前系统采用数据库模糊查询实现基础搜索，但是否引入专业搜索引擎（如 Elasticsearch）以提升搜索性能与精度，尚未作出决定。

（4）对用户上传资源的存储是否采用第三方云服务（如阿里云、七牛云），以及其计费模式、存储策略、安全机制等细节，尚在评估之中。

（5）系统尚未制定自动备份与数据恢复方案，存在在系统异常或数据丢失情况下无法快速恢复的潜在风险。

（6）是否在后续版本中加入如用户私信、标签管理、文章草稿箱等扩展功能，目前仍处于规划阶段，需结合开发周期和实际需求决定。

7 注解

（1）背景信息

本系统为一个面向公众的博客网站，旨在提供一个用户可以注册登录、发布文章、评论、点赞、收藏的互动平台。系统采用前后端分离的架构设计，前端使用 Vue 框架实现响应式页面交互，后端采用 Python 的 Flask 框架处理业务逻辑，数

据库使用 MySQL 存储各类数据。

（2）术语和定义

术语	定义
CSCI	计算机软件配置项，表示系统中的独立软件单元
API	应用程序编程接口，用于前后端或系统之间的通信
RESTful	一种基于 HTTP 协议的 Web 服务设计架构风格
JSON	一种轻量级数据交换格式
ORM	对象关系映射，后端用于操作数据库的技术
Token	用户登录后的身份验证令牌，用于保持会话状态

（3）缩略语表

缩略语	全称	含义
API	Application Programming Interface	应用程序接口
CSCI	Computer Software Configuration Item	软件配置项
DB	Database	数据库
JSON	JavaScript Object Notation	一种数据交换格式
ORM	Object-Relational Mapping	对象关系映射
REST	Representational State Transfer	一种 Web 服务架构风格
SDK	Software Development Kit	软件开发工具包
UI	User Interface	用户界面
UX	User Experience	用户体验

附录

附录可用来提供那些为便于文档维护而单独出版的信息(例如图表、分类数据)。为便于处理，附录可单独装订成册。附录应按字母顺序(A，B 等)编排。