1、需求分析

1.1 系统概述

该系统旨在为同一所学校的同学之间提供一个在线交流和信息共享的平台。在信息化水平持续提升的今天，打破信息差，加速信息的获取与分享至关重要。区别于市面上的社交平台，比如微信、微博、知乎等，该系统更加区域化和社区化，可以让用户更畅所欲言，拥有更舒适的交流体验。系统将支持多种交流形式，包括但不限于发帖、评论、点赞等，同时提供信息发布和资源共享的功能。系统将采用现代化的 Web 技术构建，以适应不断变化的教育需求和用户习惯。

1.2 用户角色

1.2.1 学生

- 注册并登录系统
- 参与论坛讨论和分享信息
- 发布和回复帖子
- 管理个人资料

1.2.2 管理员

- 具有最高权限，进行用户账户和权限管理
- 监管论坛内容，维护论坛健康
- 对系统设置和维护

1.3 功能需求

1.3.1 用户注册、登录和退出

- 用户注册：用户可以通过提供用户名、密码、电子邮件地址进行注册。
- 用户登录：已注册用户可以通过用户名和密码登录。
- 用户退出：用户可以选择退出账号，以保证信息安全。

1.3.2 编辑、发布、删除帖子

- 编辑、发布帖子：用户可以在特定模块内发布新帖子。帖子可以包含文字、图片和链接等内容。
- 浏览帖子：用户可以浏览所有公开的帖子，找到感兴趣的标题，并可以点击查看详细的内容，包括文字、图片等。
- 删除帖子：用户可以管理自己的内容，查看自己的点赞记录、删除自己发布过的帖子。

1.3.3 评论、点赞功能

- 回复帖子：用户可以在感兴趣的帖子评论区回复，且支持多级嵌套回复。
- 点赞帖子：用户可以对自己认可的帖子进行点赞表达喜欢和认可。
- 取消点赞帖子：用户可以取消之前的点赞。

1.3.4 帖子分类

- 系统支持将帖子按照不同的版块进行分类管理，便于用户查找和浏览。并提供热度较高的帖子榜单，帮助用户参与当前的热门话题。

1.3.5 用户个人资料管理

- 用户可以查看和编辑自己的个人资料，包括头像、用户名和联系邮箱等。

1.3.6 权限管理

- 权限设置：系统应提供细粒度的权限管理，确保不同用户角色具有适当的权限。
- 用户管理：管理员可以查看用户列表以及用户注册时填入的信息。
- 内容管理：管理员可以删除任意帖子，维护论坛秩序，防止违规信息传播。

1.4 非功能需求

1.4.1 系统性能要求

1. 快速响应：

系统应保证用户操作的及时响应，提升用户体验。要求在高并发情况下，用户请求的平均响应时间不超过2秒，页面加载时间不超过3秒。

2. 并发用户：

系统应能够支持大量用户同时访问，保证在高峰期也能稳定运行。要求支持至少1000名并发用户访问，确保系统的可用性和响应速度。

3. 可扩展性：

系统架构应具备良好的可扩展性，支持后续功能扩展和用户增长。要求系统设计应采用模块化、微服务架构，支持横向和纵向扩展。

1.4.2 系统安全性要求

1. 数据加密：

系统应确保用户数据的安全存储，防止用户数据的泄露和篡改。传输中的数据应使用 HTTPS 协议加密。

2. 用户隐私保护：

系统应采取措施保护用户的个人隐私，防止未经授权的访问。系统只允许注册有账户的用户进入和访问帖子。

3. 访问控制：

系统应具备完善的权限管理机制，确保不同角色的用户具有适当的权限。

4. 安全审计：

系统应具备安全审计功能，记录关键操作日志（如用户登录、数据修改、权限变更等），以便于追踪和监控异常行为。

1.4.3 可用性

1. 用户界面友好

系统应提供直观、简洁、美观的用户界面，提升用户体验。界面设计应符合用户使用习惯，采用响应式设计，支持多种设备（如PC、手机、平板）。

2. 易于导航

系统应具备良好的导航设计，帮助用户快速找到所需内容和功能，确保用户能够轻松浏览和操作。

3. 自适应设计

界面应自适应不同分辨率和设备类型，兼容桌面和移动设备，提供良好的用户体验。

1.4.4 可维护性和可扩展性

1. 代码质量 系统代码应具备良好的可读性、可维护性，便于后续开发和维护；应遵循统一的编码规范，具备良好的注释和文档；应采用模块化设计，便于代码复用和扩展。
2. 测试 系统应进行单元测试、集成测试、系统测试和用户验收测试，确保各模块和整体系统的功能符合需求。
3. 可扩展性 开发出来的程序在应对管理所需时，需要能根据对应的现实情况进行程序升级与更新。不论是功能完善还是数据库升级都能在原来的基础上对原有程序进行迭代升级。

2、总体设计

2.1 系统架构

该校园论坛系统采用典型的客户端-服务器架构，包括前端 Web 应用、后端服务器和数据库。前端提供用户界面，后端处理业务逻辑和数据管理，数据库存储用户信息、帖子内容等。

2.2 技术栈

1. 前端：html + java + CSS

- 使用 CSS3 实现高级视觉效果，提高样式的可维护性。

2. 后端：Spring Boot

3. 数据库：MySQL

- 使用关系型数据库 MySQL 作为主数据库，存储用户数据、帖子数据等结构化信息，提供 ACID 事务支持。
- 数据库设计：采用规范化设计，合理设置数据表和关系，确保数据的一致性和完整性。
- 索引优化：为常用的查询字段建立索引，提高查询效率。

4. 辅助工具：使用 Maven 作为项目构建和依赖管理工具。

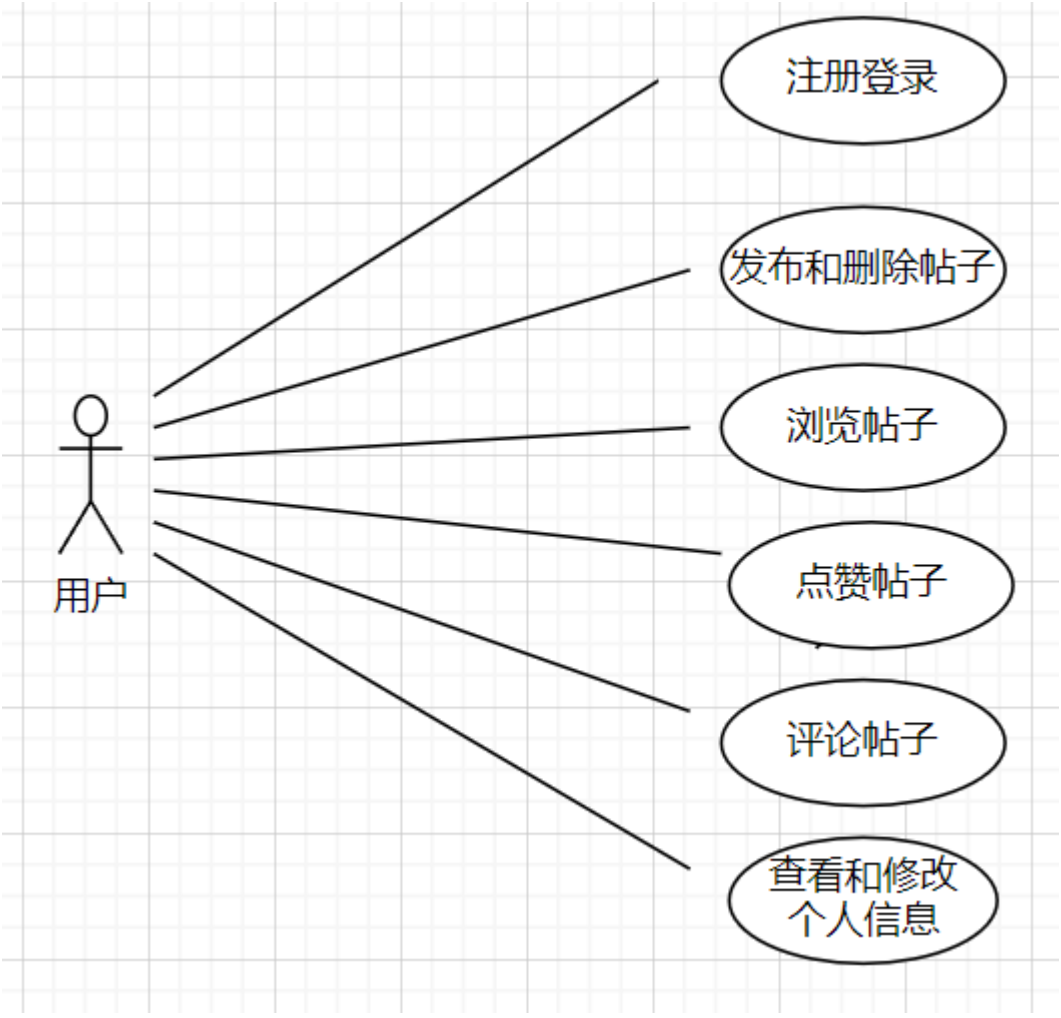
3、详细设计

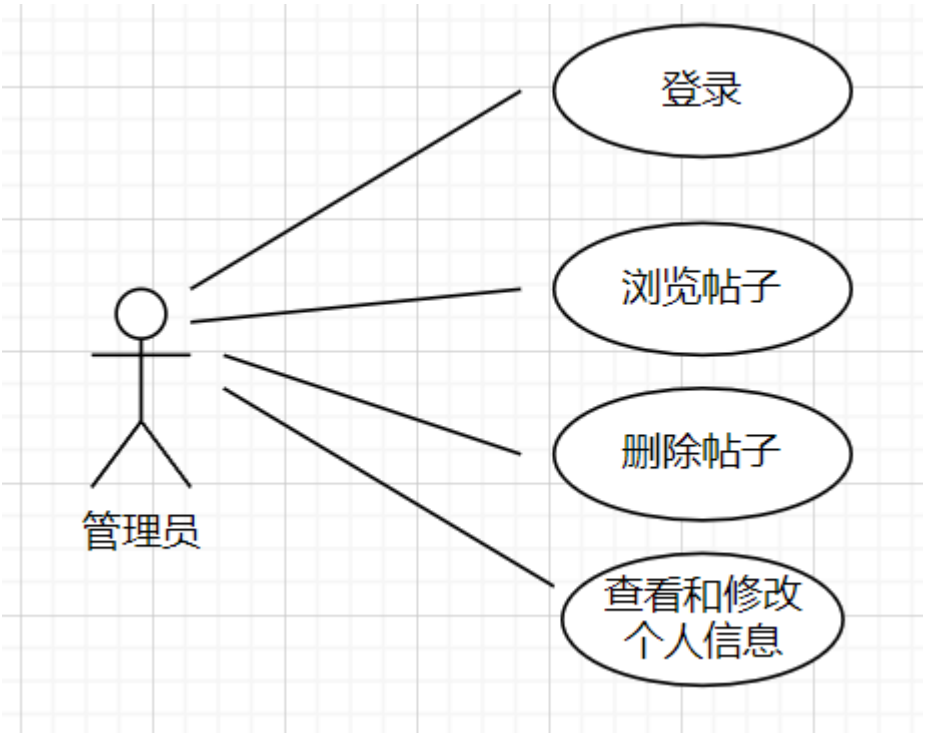
3.1 模块设计

3.1.1 用户模块

- 注册与登录：提供用户注册、登录功能。
- 个人资料：允许用户设置个人信息、头像等。
- 权限管理：区分普通用户和管理员，管理用户权限和角色

用例图：





3.1.2 帖子管理模块

- 发布、浏览帖子：用户可以发布新帖子、浏览、评论和点赞他人的帖子。
- 查看热门帖子：用户可以在热榜栏查看受他人喜爱、点赞数较多的帖子。
- 取消点赞和删除帖子：用户可以管理自己点赞或发布过的帖子，取消点赞或者删除。

3.1.3 社交互动模块

- 点赞与评论：用户可以对帖子点赞和评论。
- 回复：支持用户对帖子的评论进行回复

3.1.4 管理与运维模块

- 数据统计与分析：管理员可以查看帖子热度（点赞数）进行数据统计。
- 内容审核与管理：对用户发布的帖子和评论进行审核和管理，可以删除违规的帖子。

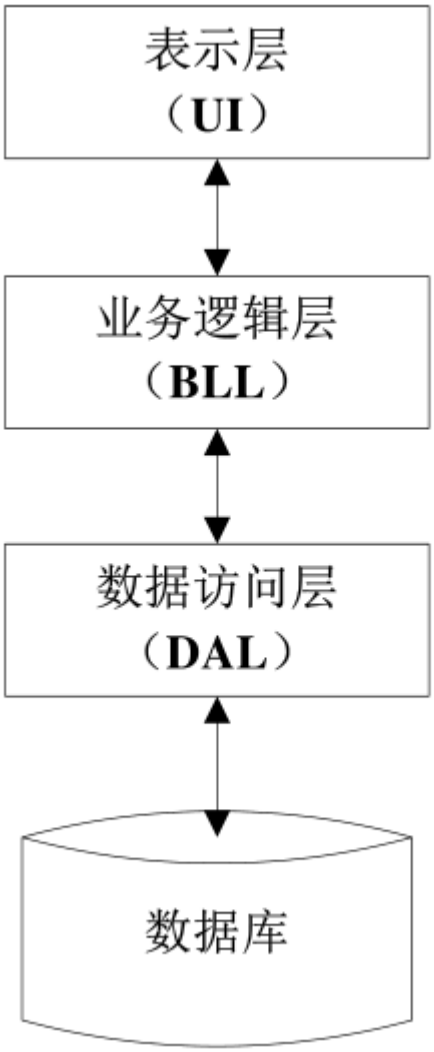
3.2 架构设计

3.2.1 软件架构

该校园论坛系统采用分层架构设计，主要包括表示层、业务逻辑层和数据访问层。这种设计模式有助于系统的模块化和可维护性。

- 表示层：负责与用户交互，显示信息并收集用户输入。主要包括网页前端和移动端应用。
- 业务逻辑层：处理所有的业务逻辑和规则，包括用户认证、权限管理、帖子处理等。

- 数据访问层：负责与数据库交互，执行数据的存储、检索和更新操作。



3.3 子系统及其接口设计

3.3.1 子系统划分

- 用户管理子系统：处理用户注册、登录、退出、密码管理和用户信息更新。
- 帖子管理系统：处理帖子编辑、发布、删除、浏览和统计点赞数等功能。
- 回复和互动系统：处理帖子回复、点赞、评论等互动操作。
- 权限管理系统：管理不同用户角色的权限，确保系统的安全性和数据隐私。

3.3.2 接口设计

用户服务接口（UserService）

- 用户服务接口负责定义用户管理相关的操作，包括获取用户列表、保存用户、根据用户名和密码查找用户等功能。
- 接口方法说明：
  - getAllUsers()：获取所有用户列表。
  - saveUser(User user)：保存用户到数据库。
  - findUserByNameAndPassword(String name, String password)：根据用户名和密码查找用户。
  - getLastRegisteredUser()：获取最近注册的用户。
  - isUsernameUnique(String username)：检查用户名是否唯一。

    - findUserByName(String name)：根据用户名查找用户。
    - updateUser(User user)：更新用户信息。
    - findUserByUid(Long uid)：根据用户 ID 查找用户。

帖子服务接口（PostService）

- 定义了对帖子的操作，包括保存帖子、根据作者 ID 获取帖子、删除帖子、根据帖子 ID 获取帖子等方法。同时使用了 PostRepository 和 LikeRepository 实现对帖子和点赞数据的持久化操作，包括保存、查找和统计等功能。
- 接口方法说明：
    - savePost()：用于保存帖子到数据库中。
    - getPostsByAuthorId()：根据作者 ID 获取该作者的所有帖子列表。
    - deletePost()：根据帖子 ID 删除帖子。
    - getPostByPostId()：根据帖子 ID 获取单个帖子的详细信息。
    - getAllPosts()：获取所有帖子的列表。
    - getLikeCount()：获取指定帖子的点赞数。
    - findHotPosts()：查找热门帖子，通过调用 LikeDAO 获取热门帖子列表，并将结果转换为 Post 对象返回。

点赞服务接口（LikeService）

- 定义了对点赞操作的方法，包括点赞、取消点赞、查找点赞记录、统计点赞数等。
- 接口方法说明：
    - likePost()：用于给帖子点赞，创建并保存点赞记录到数据库。
    - unlikePost()：用于取消对帖子的点赞，根据用户 ID 和帖子 ID 删除点赞记录。
    - findLikesByPostId()：根据帖子 ID 查找该帖子的所有点赞记录。
    - isPostLikedByUser()：判断指定用户是否已对某帖子点赞。
    - countLikes()：统计指定帖子的点赞数。
    - toggleLike()：实现点赞状态的切换，如果用户未点赞则点赞，如果已点赞则取消点赞。

评论服务接口（CommentService）

- 定义了对评论操作的方法，包括获取指定帖子的所有评论、根据评论 ID 获取评论、保存评论、回复评论和删除评论等功能。
- 接口方法说明：
    - getAllCommentsByPostId()：根据帖子 ID 获取所有相关评论。
    - getCommentById()：根据评论 ID 获取单个评论。
    - saveComment()：保存评论到数据库。
    - replyToComment()：用于回复评论，通过指定的帖子 ID、父评论 ID 和内容创建新的评论，并将其保存到数据库中。同时，还会更新父评论的子评论列表。
    - deleteComment()：根据评论 ID 删除评论。

3.4 类的设计

系统中的类设计包括用户类、帖子类、回复类、点赞类等，每个类包含相关的属性和方法。

3.4.1 用户类（users）

- 属性：

    - uid (Long): 用户 id，使用自动生成策略生成。
    - name (String): 用户名，不能为空。
    - password (String): 用户密码，不能为空。
    - email (String): 用户电子邮件，不能为空。
    - is_admin (Integer): 是否为管理员，默认为0（不是管理员）。
    - avatar (String): 用户头像。

- 方法（getter 和 setter）：

    - getUid(): 获取用户 id。
    - setUid(Long uid): 设置用户 id。
    - getName(): 获取用户名。
    - setName(String name): 设置用户名。
    - getPassword(): 获取用户密码。
    - setPassword(String password): 设置用户密码。
    - getEmail(): 获取用户电子邮件。
    - setEmail(String email): 设置用户电子邮件。
    - getIsAdmin(): 获取是否为管理员，如果 is_admin 为空，则返回0。
    - setIsAdmin(Integer is_admin): 设置是否为管理员。
    - getAvatar(): 获取用户头像。
    - setAvatar(String avatar): 设置用户头像。

3.4.2 帖子类（posts）

- 属性：

    - postId (int): 帖子 id，使用自动生成策略生成。
    - title (String): 帖子标题，不能为空。
    - content (String): 帖子内容，使用 TEXT 数据类型，不能为空。
    - authorId (Long): 帖子的作者id，不能为空。
    - createdAt (Date): 帖子的创建时间，使用时间戳记录，不能为空，并且在创建后不可更新。
    - updatedAt (Date): 帖子的更新时间，使用时间戳记录，不能为空。

- getter 和 setter 方法：

    - getPostId(): 获取帖子 id。
    - setPostId(int postId): 设置帖子 id。
    - getTitle(): 获取帖子的标题。
    - setTitle(String title): 设置帖子的标题。
    - getContent(): 获取帖子的内容。
    - setContent(String content): 设置帖子的内容。
    - getAuthorId(): 获取帖子作者的 id。
    - setAuthorId(Long authorId): 设置帖子作者的 id。
    - getCreatedAt(): 获取帖子的创建时间。
    - setCreatedAt(Date createdAt): 设置帖子的创建时间。
    - getUpdatedAt(): 获取帖子的更新时间。
    - setUpdatedAt(Date updatedAt): 设置帖子的更新时间。

- 生命周期回调方法：

- onCreate(): 在持久化之前调用，设置创建时间和更新时间为当前时间。
- onUpdate(): 在更新之前调用，设置更新时间为当前时间。

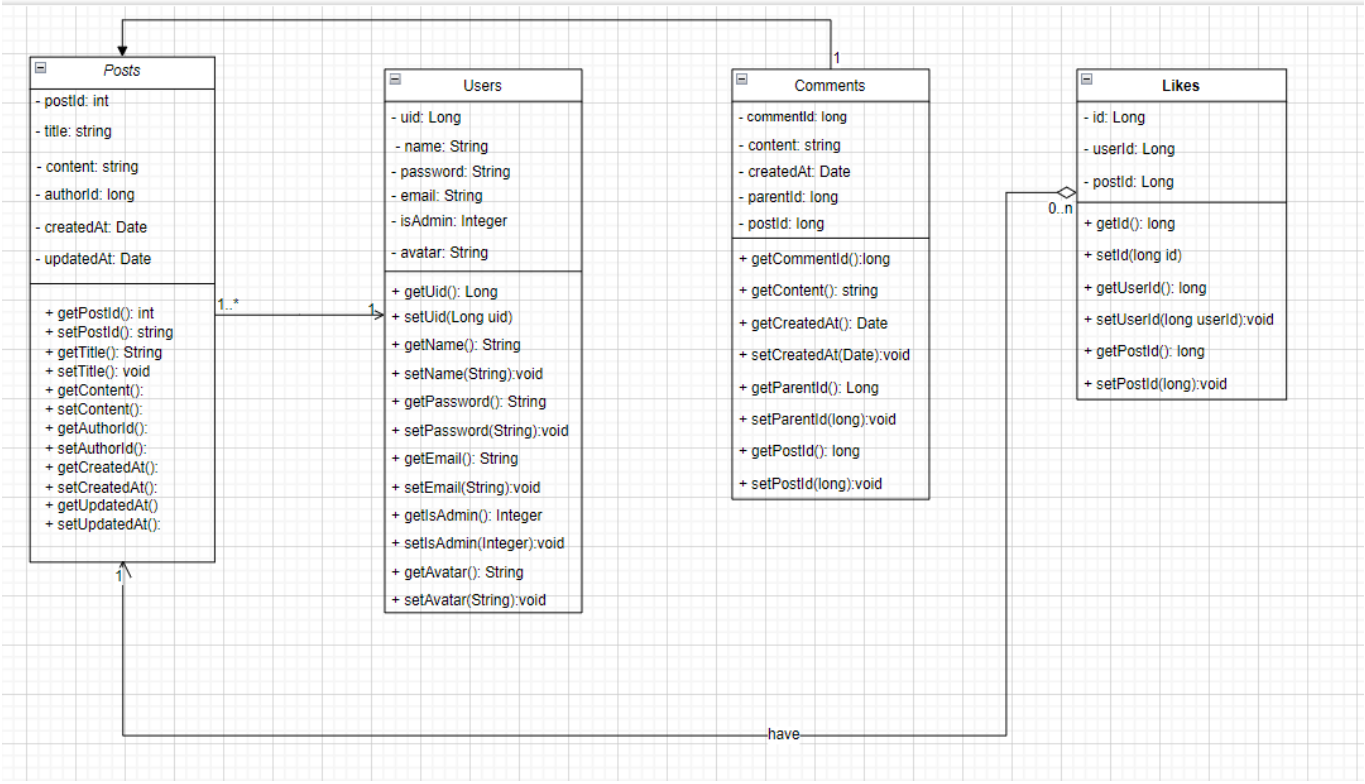
3.4.3 评论类（comments）

- 属性：
  - commentId: long 类型，评论的唯一标识符。
  - content: string 类型，评论的内容。
  - createdAt: DateTime，评论的创建时间。
  - parentId: long 类型，用于表示这是一个回复评论
  - postId: long 类型，评论所属帖子的标识符。
  - userId: long 类型，评论者的用户标识符。
- 方法：
  - getCommentId(): 获取评论 id。
  - setCommentId(Long commentId): 设置评论 id。
  - getContent(): 获取评论的内容。
  - setContent(String content): 设置评论的内容。
  - getCreatedAt(): 获取评论的创建时间。
  - setCreatedAt(DateTime createdAt): 设置评论的创建时间。
  - getParentId(): 获取上一级评论 id。
  - setParentId(Long parentId): 设置上一级评论的 id。
  - getPostId(): 获取评论所属帖子的 id。
  - setPostId(Long postId): 设置评论所属帖子的 id。
  - getUserId(): 获取评论者的 id。
  - setUserId(Long userId): 设置评论者的 id。

3.4.4 点赞类（likes）

- 属性：
  - id: 点赞记录的 id（类型 long）
  - postId: 被点赞帖子 id（类型 long）
  - userId: 点赞用户的 id（类型 long）
- 方法：
  - getId(): 获取点赞记录的 ID。
  - setId(Long id): 设置点赞记录的 ID。
  - getUserId(): 获取点赞用户的 ID。
  - setUserId(Long userId): 设置点赞用户的 ID。
  - getPostId(): 获取被点赞帖子的 ID。
  - setPostId(Long postId): 设置被点赞帖子的 ID。

3.4.5 类图

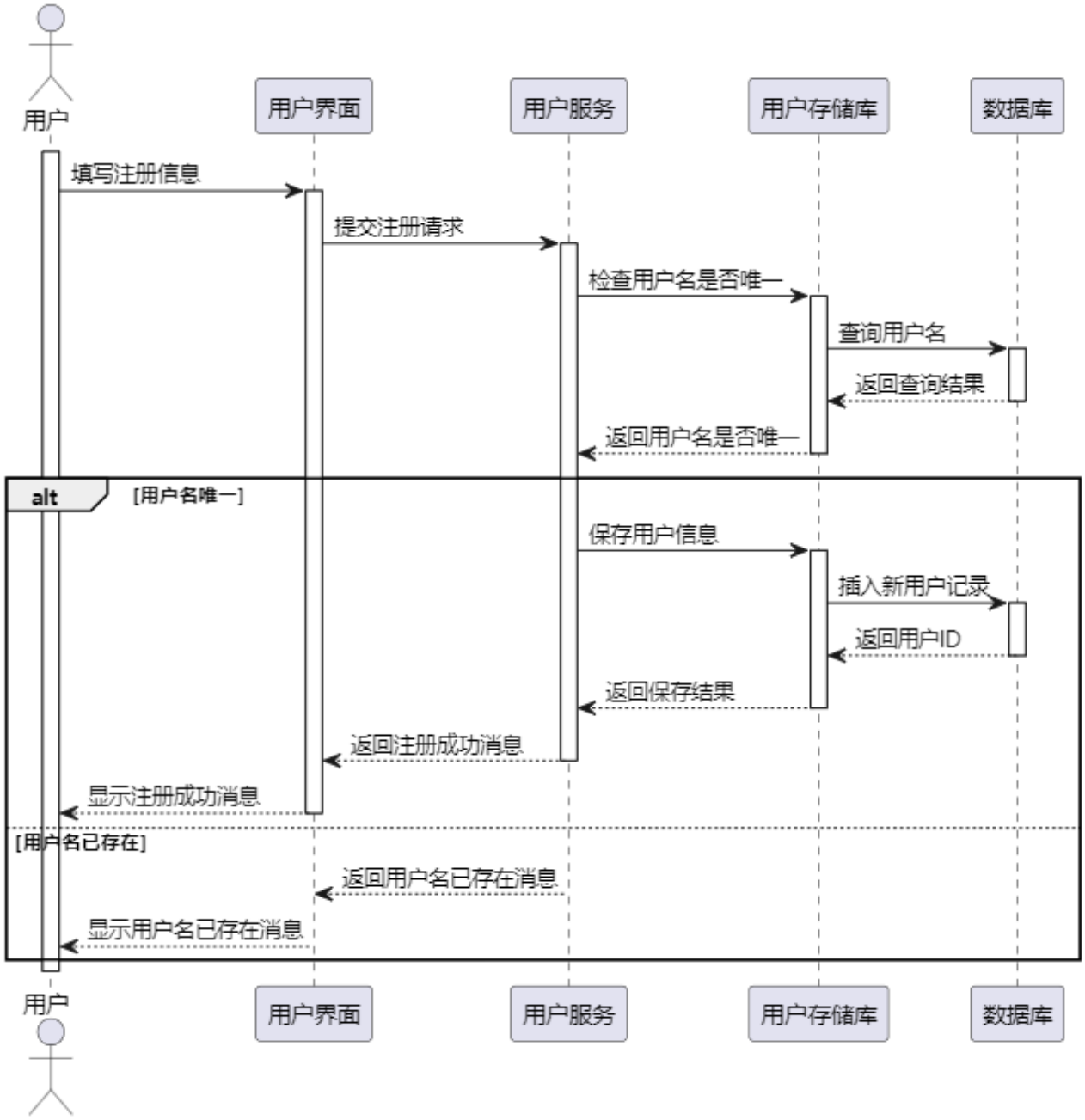


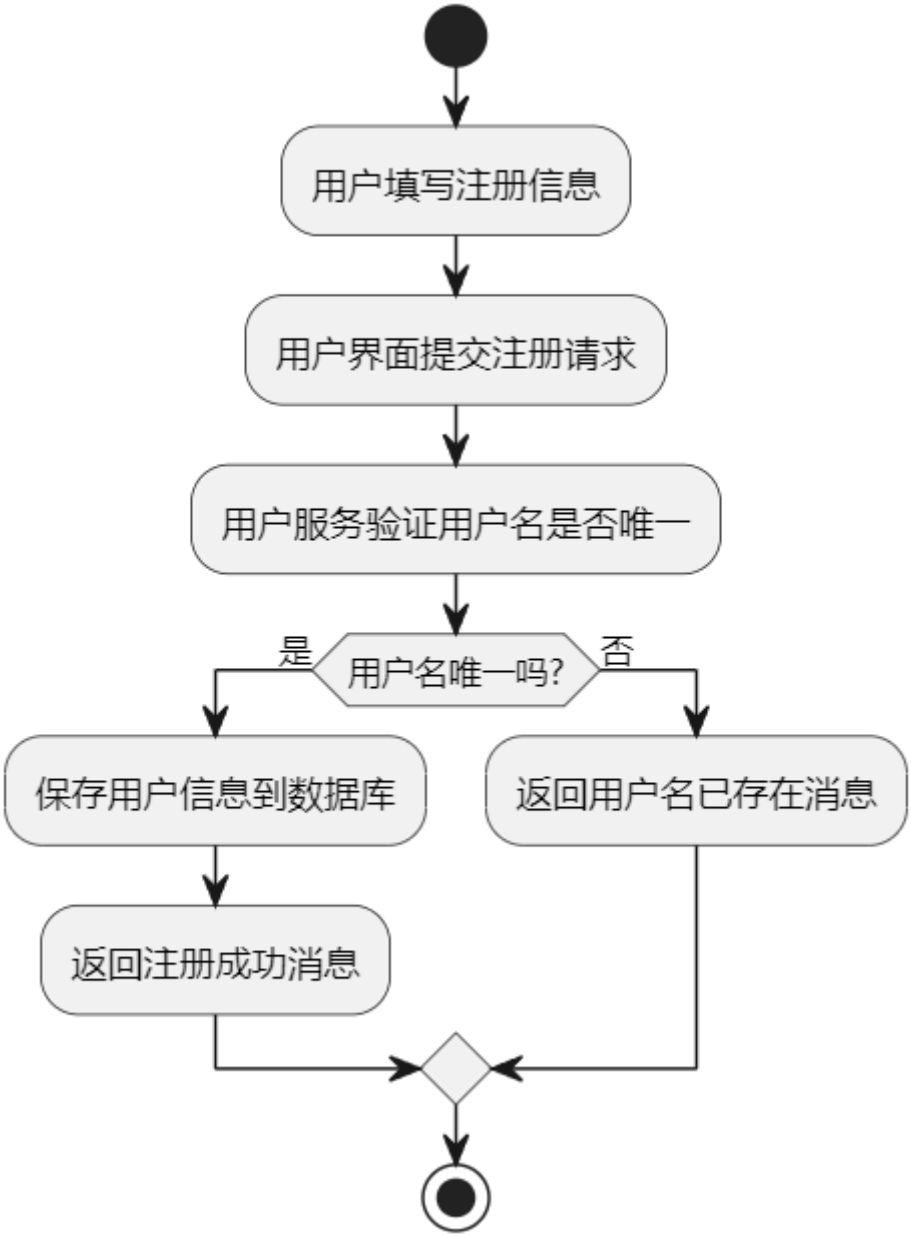
3.5 系统流程

3.5.1 注册流程

未有账号的用户可进行注册操作。

- 时序图：


用户
用户界面
用户服务
用户存储库
数据库
填写注册信息
提交注册请求
检查用户名是否唯一
查询用户名
返回查询结果
返回用户名是否唯一
alt
[用户名唯一]
保存用户信息
插入新用户记录
返回用户ID
返回保存结果
返回注册成功消息
显示注册成功消息
[用户名已存在]
返回用户名已存在消息
显示用户名已存在消息
用户
用户界面
用户服务
用户存储库
数据库

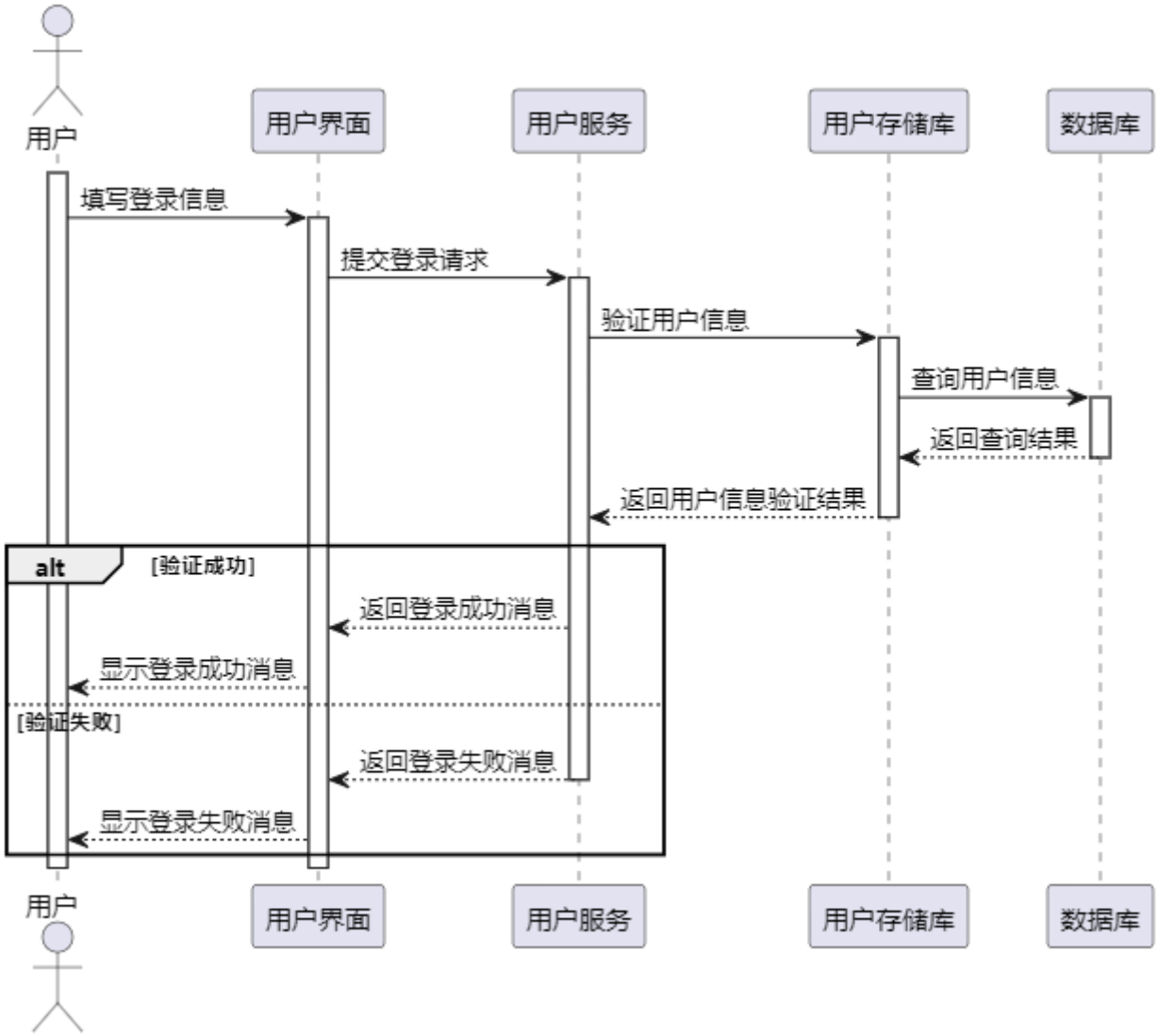


- 活动图：

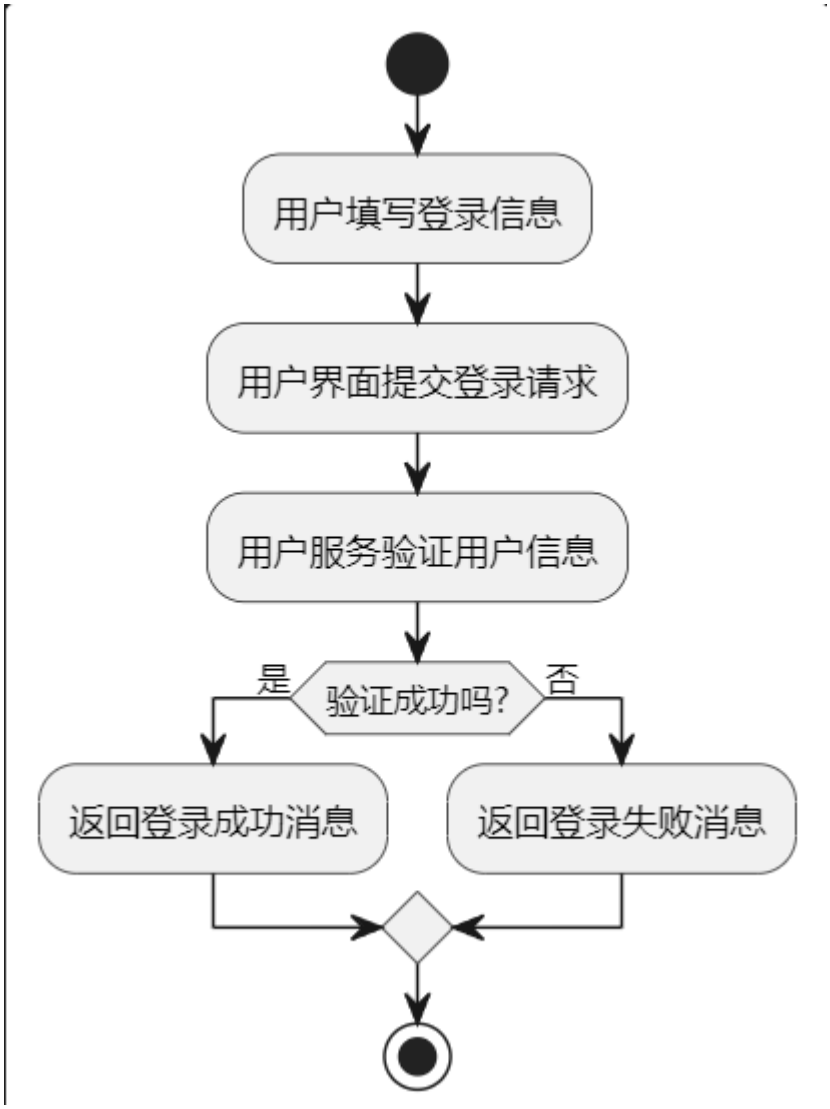
3.5.2 登录流程

登录模块主要满足了管理员和用户的权限登录。

- 时序图：


用户
用户界面
用户服务
用户存储库
数据库
填写登录信息
提交登录请求
验证用户信息
查询用户信息
返回查询结果
返回用户信息验证结果
alt
[验证成功]
返回登录成功消息
显示登录成功消息
[验证失败]
返回登录失败消息
显示登录失败消息
用户
用户界面
用户服务
用户存储库
数据库

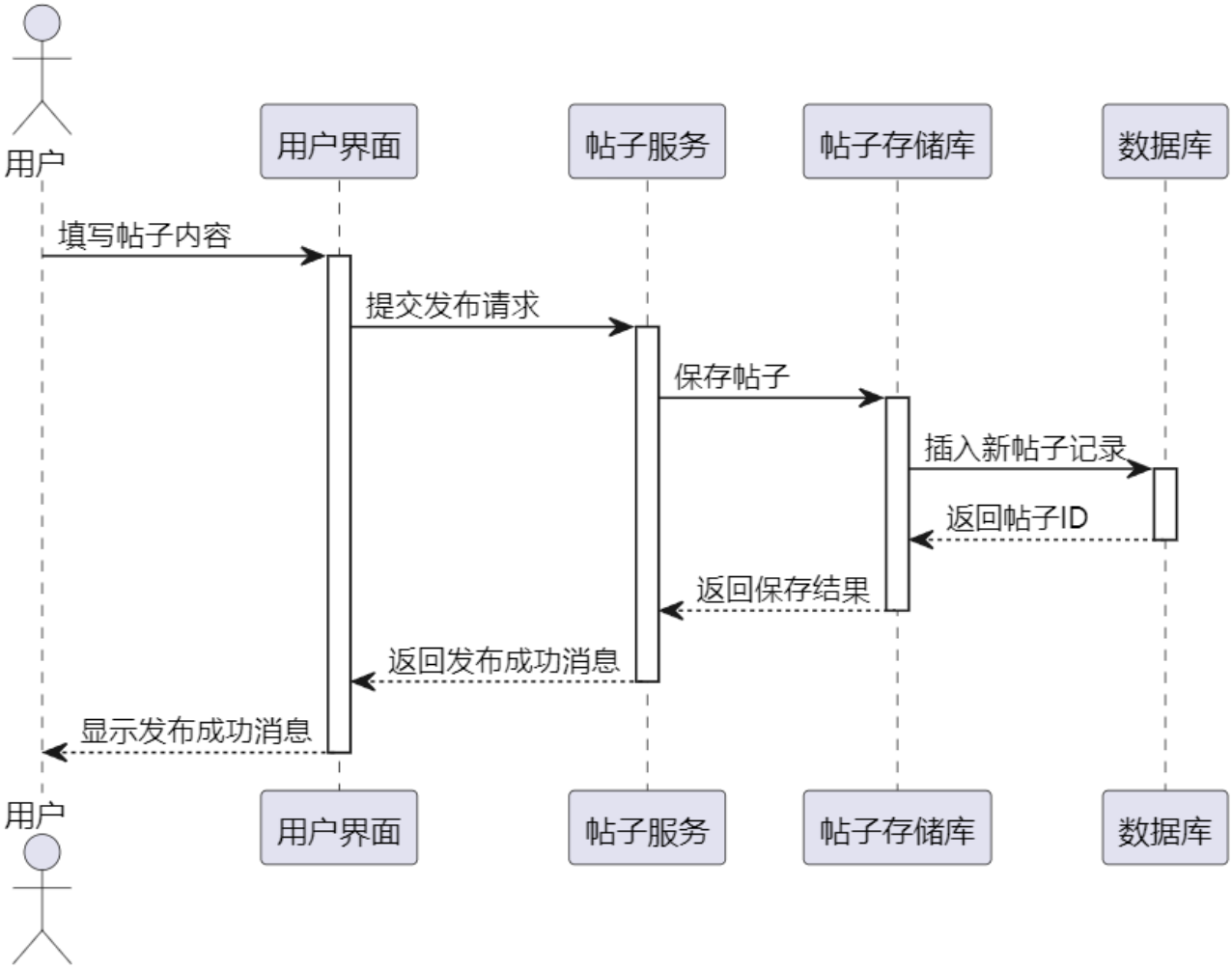
- 活动图：



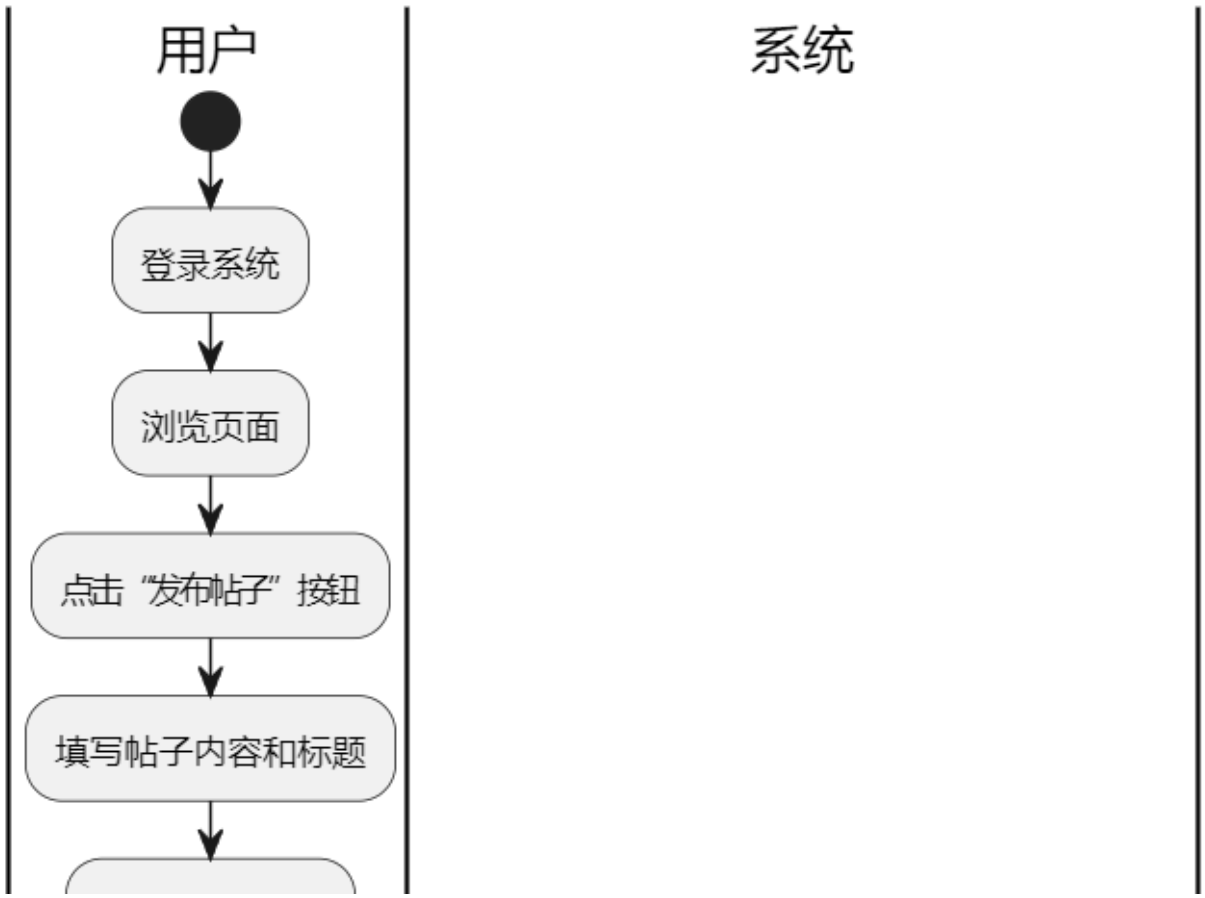
3.5.3 发帖流程

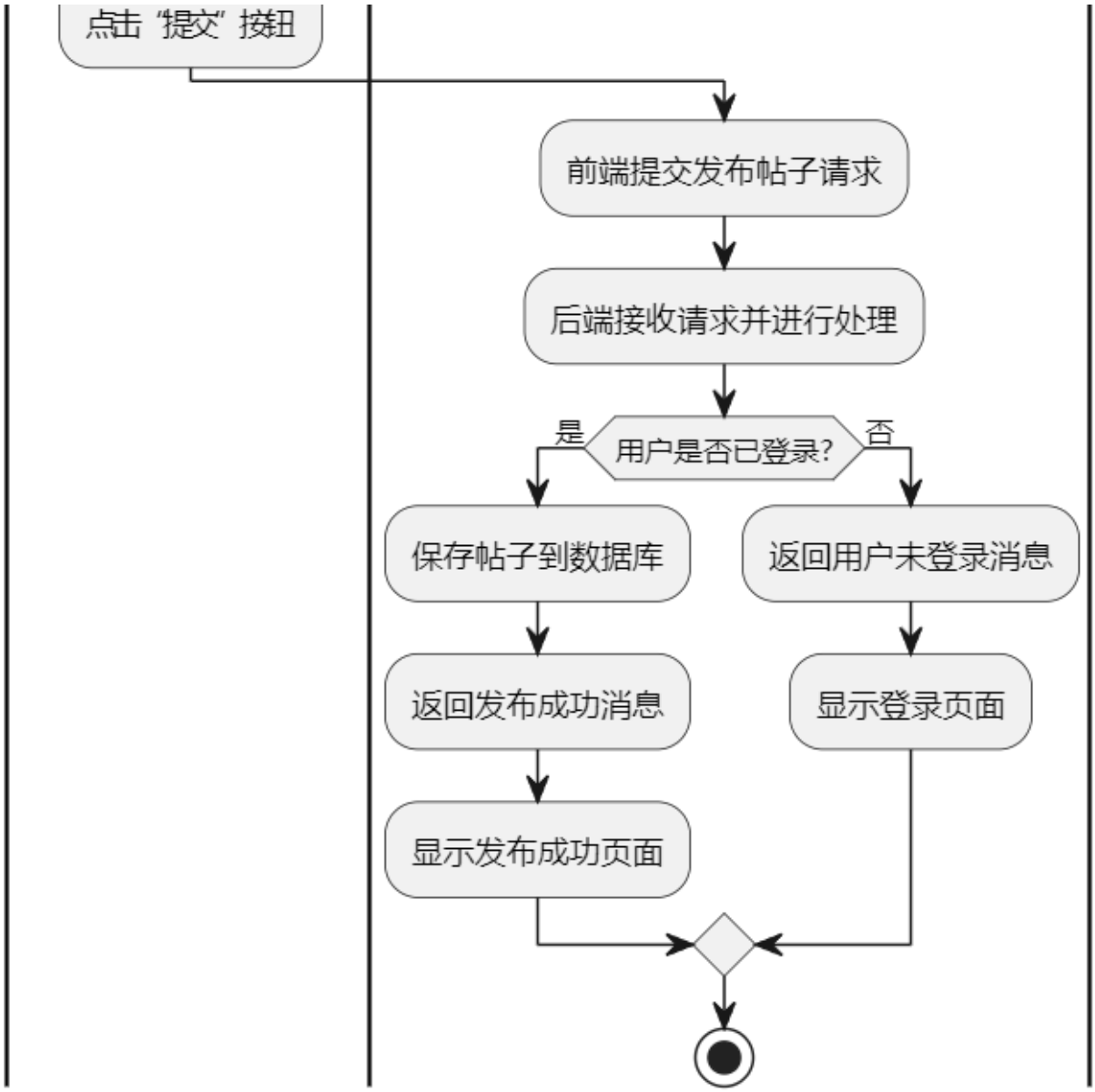
用户论坛发帖前提必须是登录后才能进行，发帖内容不能为空，且合法才能进行提交发表成功。

- 时序图


用户
用户界面
帖子服务
帖子存储库
数据库
填写帖子内容
提交发布请求
保存帖子
插入新帖子记录
返回帖子ID
返回保存结果
返回发布成功消息
显示发布成功消息
用户
用户界面
帖子服务
帖子存储库
数据库

- 活动图

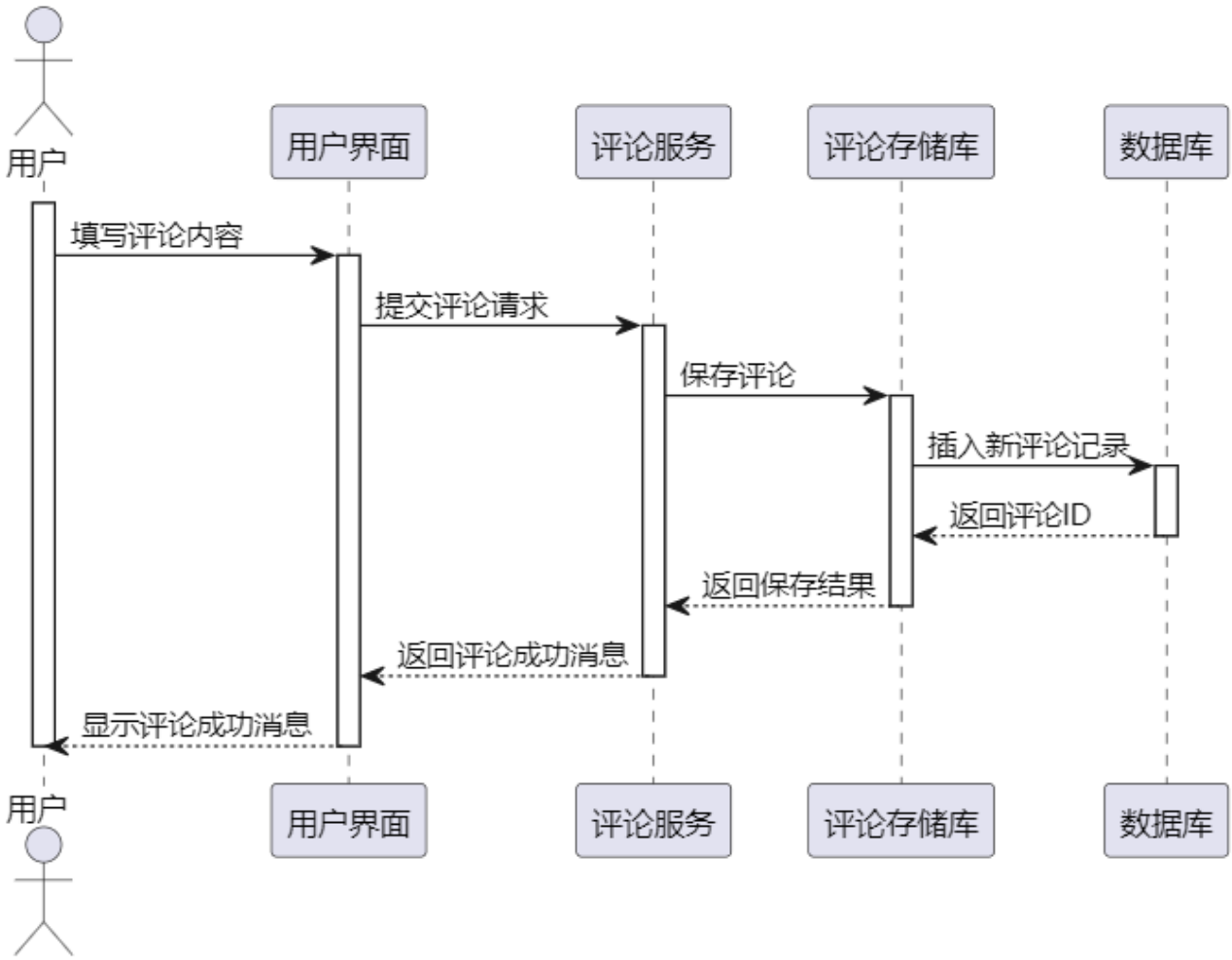

用户
系统
登录系统
浏览页面
点击“发布帖子”按钮
填写帖子内容和标题

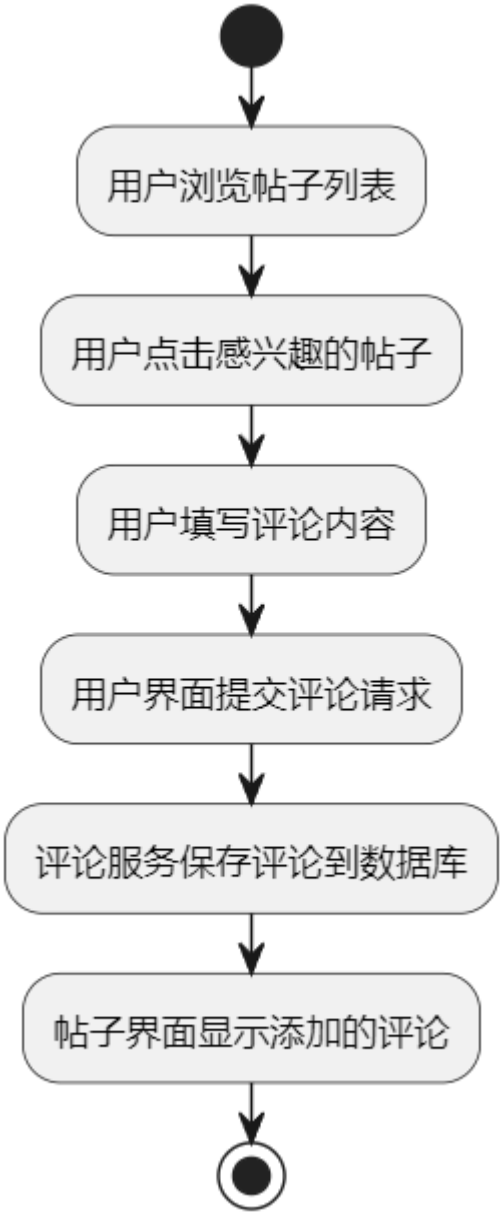


3.5.4 评论流程

用户可以对平台里的帖子进行评论。

- 时序图：

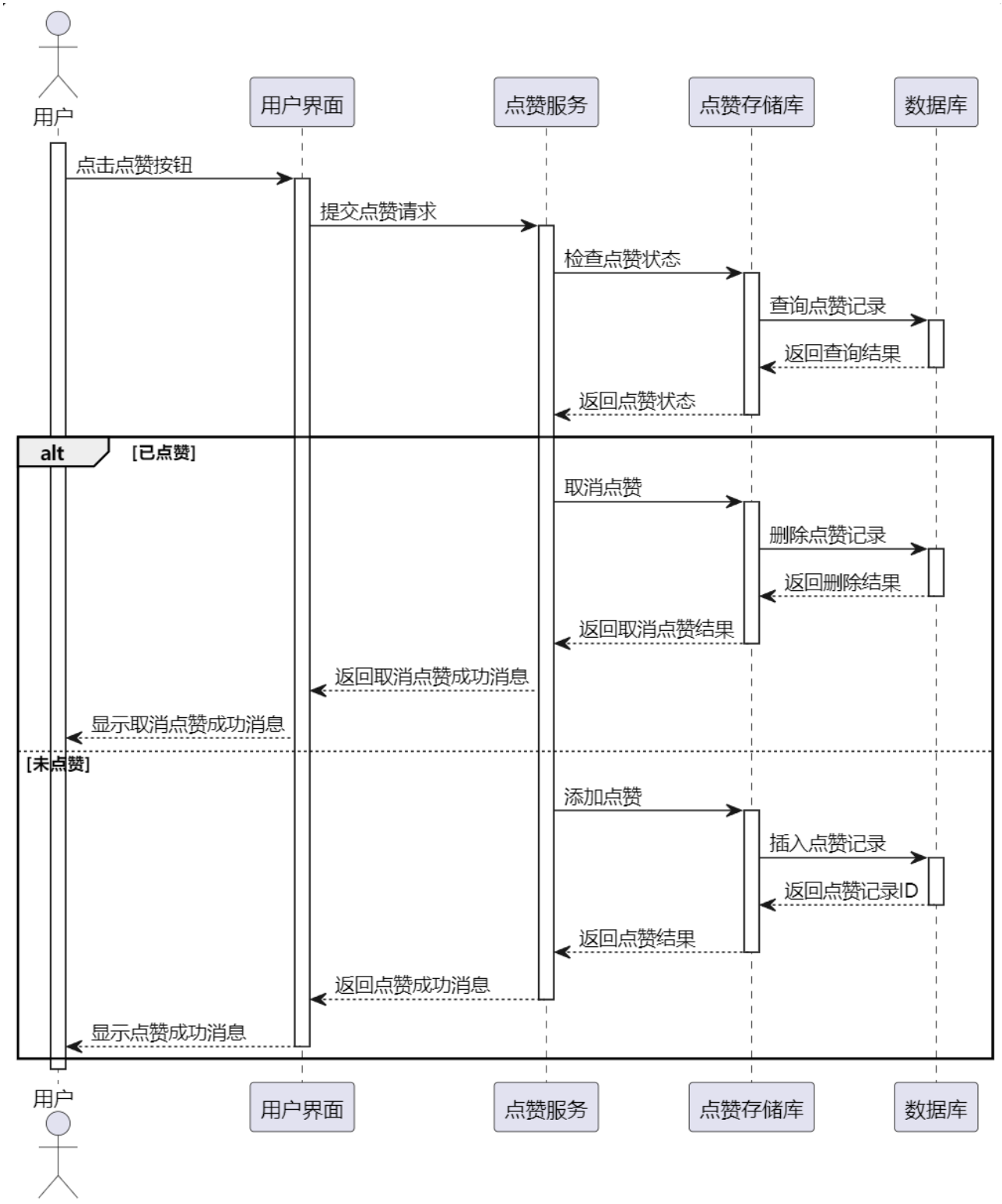

用户
用户界面
评论服务
评论存储库
数据库
填写评论内容
提交评论请求
保存评论
插入新评论记录
返回评论ID
返回保存结果
返回评论成功消息
显示评论成功消息
用户
用户界面
评论服务
评论存储库
数据库

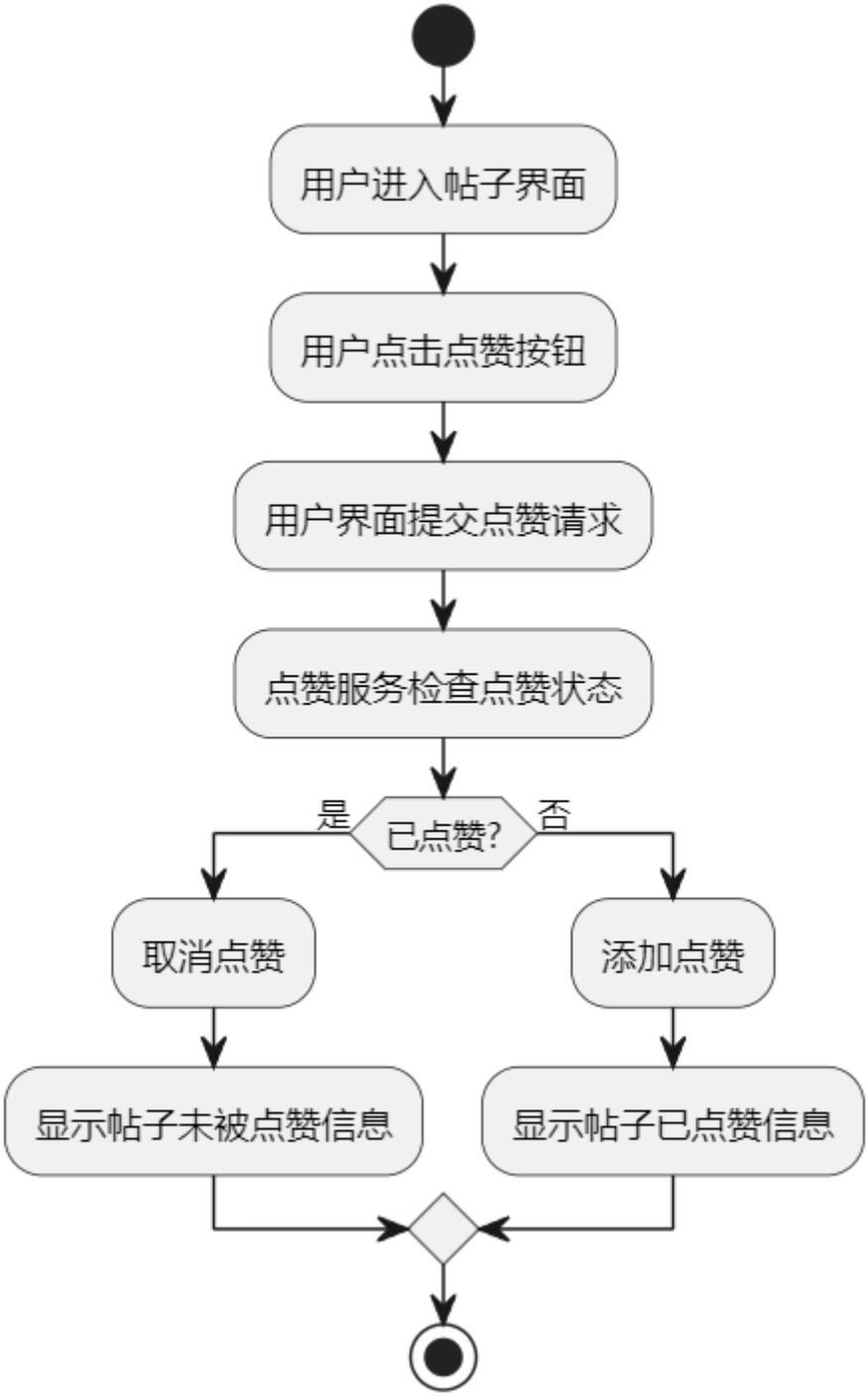

用户浏览帖子列表
用户点击感兴趣的帖子
用户填写评论内容
用户界面提交评论请求
评论服务保存评论到数据库
帖子界面显示添加的评论

- 活动图:

3.5.5 点赞流程

- 时序图：


用户
用户界面
点赞服务
点赞存储库
数据库
点击点赞按钮
提交点赞请求
检查点赞状态
查询点赞记录
返回查询结果
返回点赞状态
alt
[已点赞]
取消点赞
删除点赞记录
返回删除结果
返回取消点赞结果
返回取消点赞成功消息
显示取消点赞成功消息
[未点赞]
添加点赞
插入点赞记录
返回点赞记录ID
返回点赞结果
返回点赞成功消息
显示点赞成功消息
用户
用户界面
点赞服务
点赞存储库
数据库

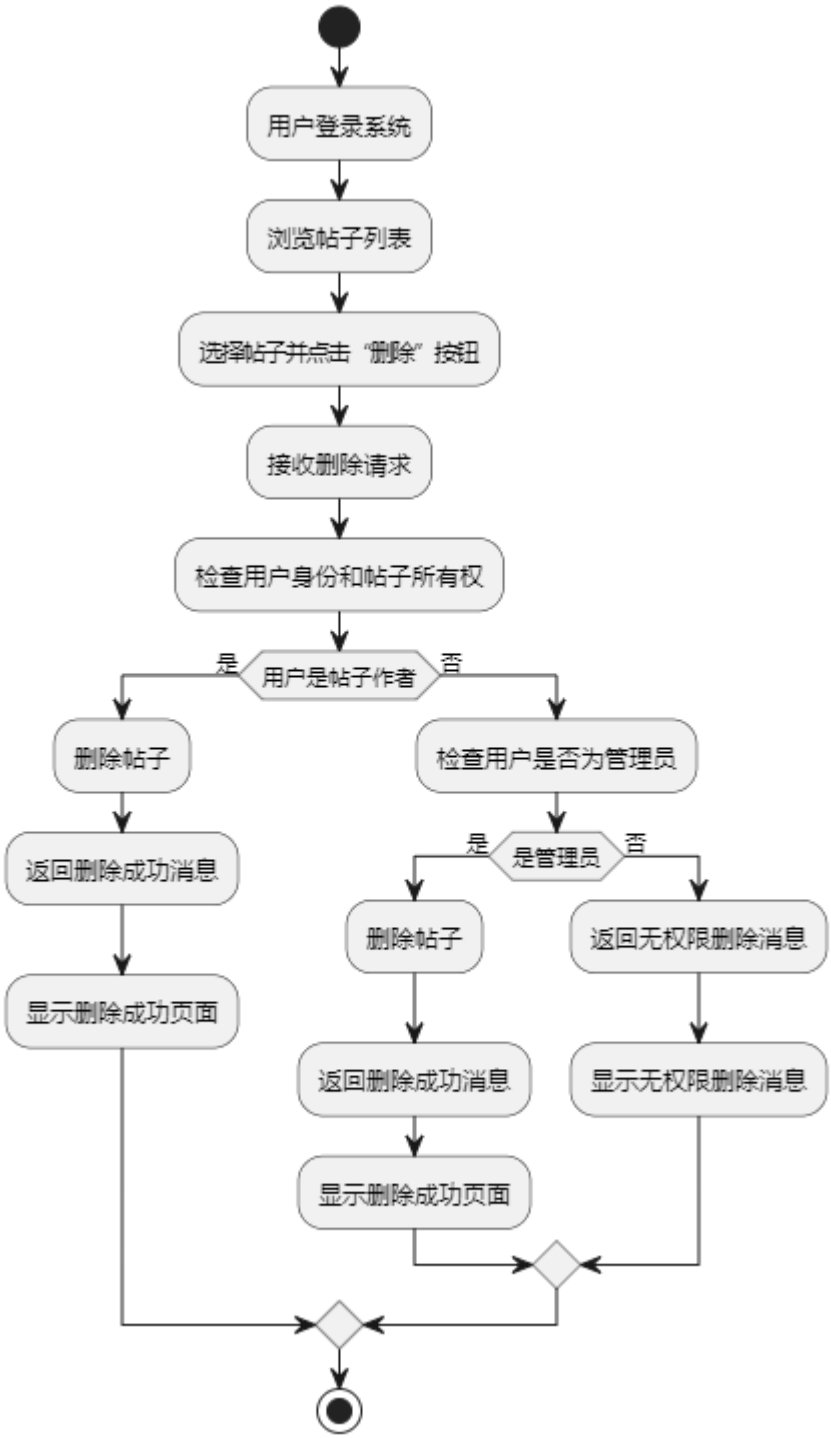

用户进入帖子界面
用户点击点赞按钮
用户界面提交点赞请求
点赞服务检查点赞状态
已点赞?
是
否
取消点赞
添加点赞
显示帖子未被点赞信息
显示帖子已点赞信息

- 活动图：

3.5.6 删帖流程

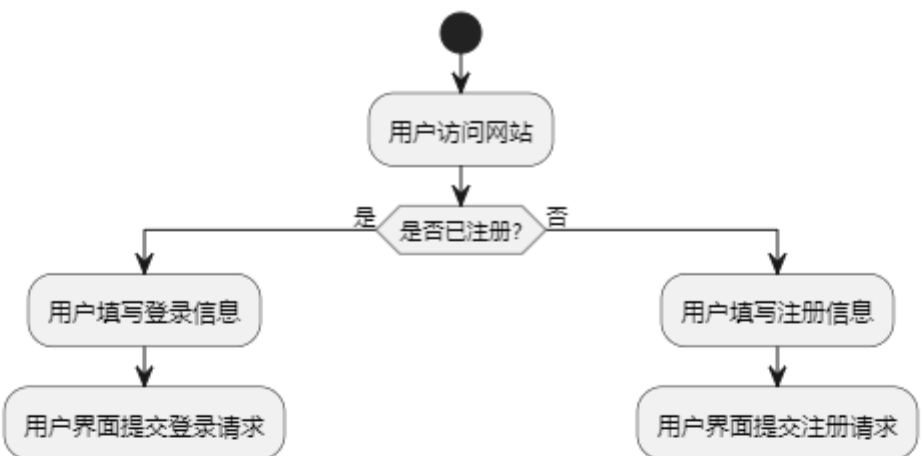
- 时序图：

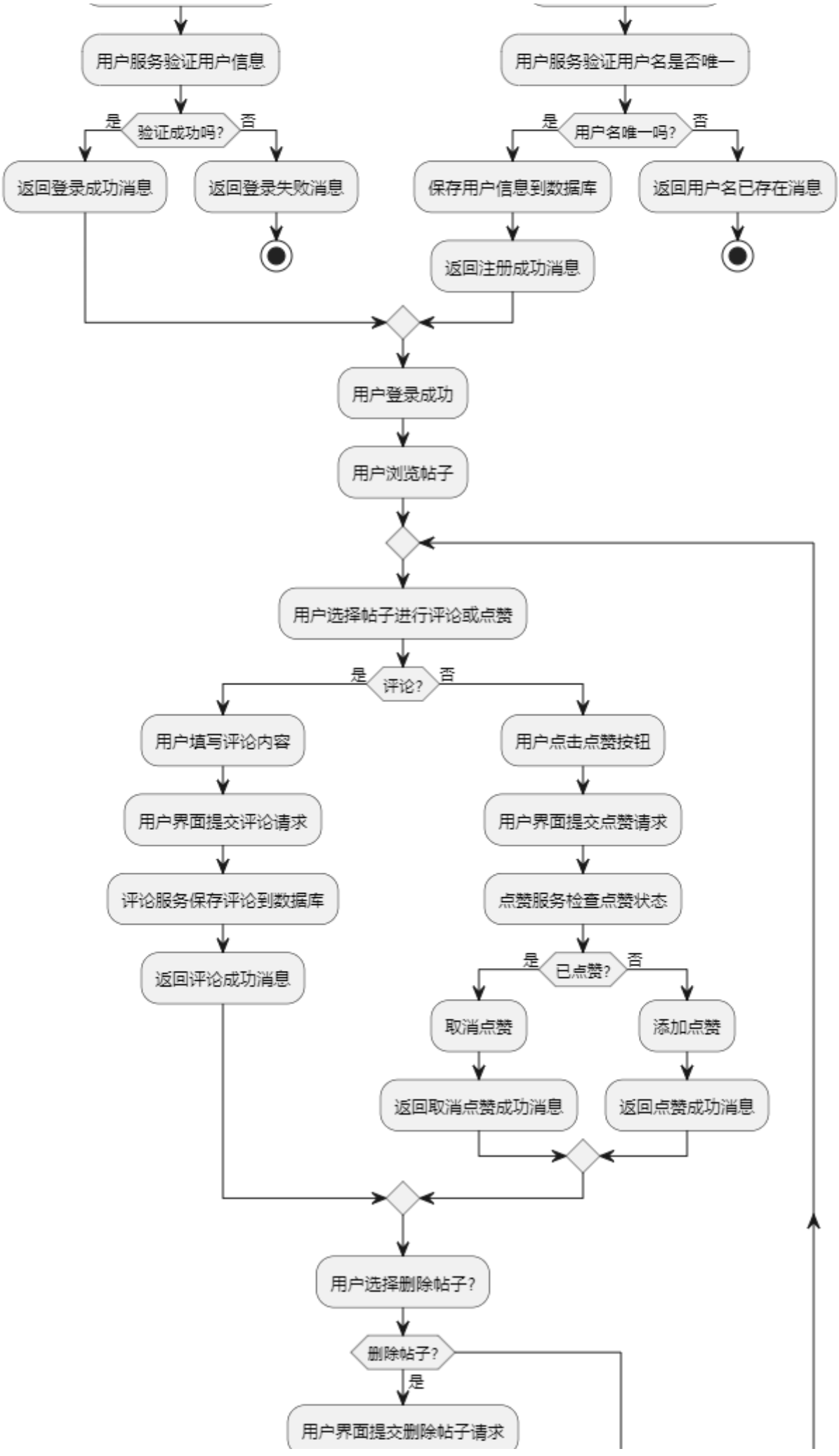

用户
管理员
前端
后端
数据库
点击“删除帖子”按钮
发送删除帖子请求
检查用户身份和帖子所有权
alt
[用户是帖子作者]
删除帖子
确认删除成功
返回删除成功消息
显示删除成功页面
[用户不是帖子作者]
返回无权限删除消息
显示无权限删除消息
点击“删除帖子”按钮
发送删除帖子请求
检查管理员身份
alt
[是管理员]
删除帖子
确认删除成功
返回删除成功消息
显示删除成功页面
[不是管理员]
返回无权限删除消息
显示无权限删除消息
用户
管理员
前端
后端
数据库

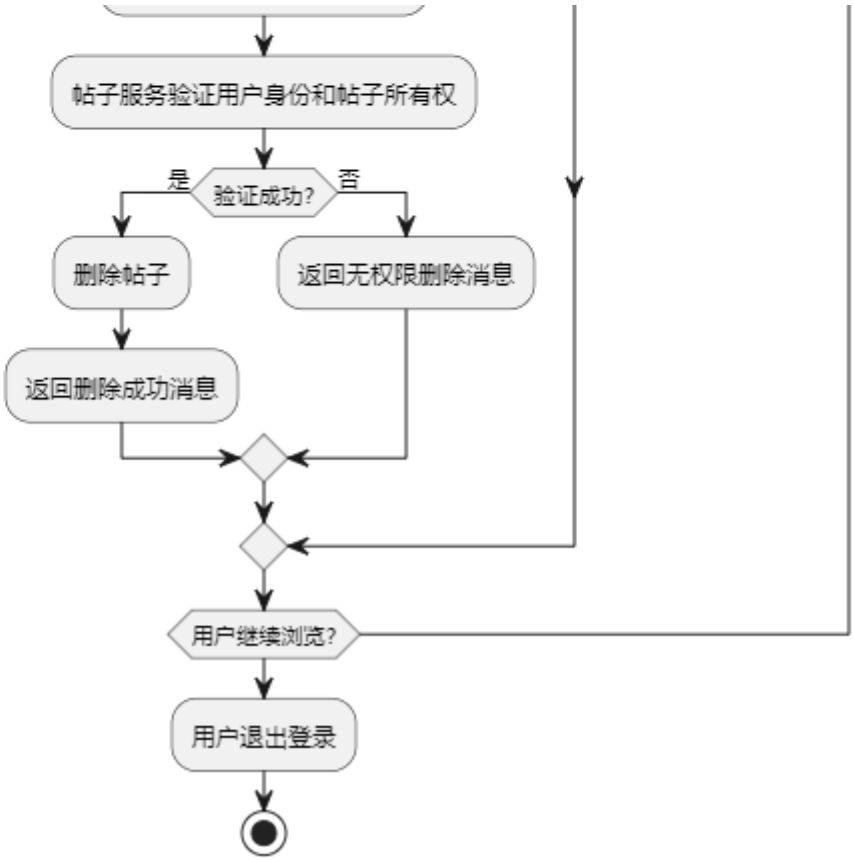

用户登录系统
浏览帖子列表
选择帖子并点击“删除”按钮
接收删除请求
检查用户身份和帖子所有权
是
用户是帖子作者
否
删除帖子
返回删除成功消息
显示删除成功页面
检查用户是否为管理员
是
是管理员
否
删除帖子
返回删除成功消息
显示删除成功页面
返回无权限删除消息
显示无权限删除消息

- 活动图：

3.5.7 完整流程：


用户访问网站
是
是否已注册?
否
用户填写登录信息
用户界面提交登录请求
用户填写注册信息
用户界面提交注册请求


用户服务验证用户信息
是
验证成功吗?
否
返回登录成功消息
返回登录失败消息
用户服务验证用户名是否唯一
是
用户名唯一吗?
否
保存用户信息到数据库
返回用户名已存在消息
返回注册成功消息
用户登录成功
用户浏览帖子
用户选择帖子进行评论或点赞
是
评论?
否
用户填写评论内容
用户界面提交评论请求
评论服务保存评论到数据库
返回评论成功消息
用户点击点赞按钮
用户界面提交点赞请求
点赞服务检查点赞状态
是
已点赞?
否
取消点赞
添加点赞
返回取消点赞成功消息
返回点赞成功消息
用户选择删除帖子?
删除帖子?
是
用户界面提交删除帖子请求



4、部件设计

4.1 前端组件设计

用户界面组件设计：

1. 导航栏组件

- 功能：显示网站导航链接，帮助用户快速导航到主要页面。
- 与其他组件关系：与帖子列表组件和用户信息组件有交互，例如显示当前用户信息（用户名和邮箱）等。

2. 帖子列表组件

- 功能：展示帖子的摘要信息，本系统采用标题来代替帖子信息作为索引。
- 与其他组件关系：点击帖子标题可以导航到帖子详情页；

3. 帖子详情组件

- 功能：展示完整的帖子信息，包括帖子内容、评论和点赞。
- 与其他组件关系：允许用户在此页面进行评论、点赞的操作。

4. 用户信息组件

- 功能：展示用户的个人信息，如用户名、头像、邮箱等。
- 与其他组件关系：支持用户修改个人信息或上传新头像；显示用户的帖子发布情况和点赞、评论情况。

帖子管理组件设计

1. 创建帖子组件

- 功能：提供界面让用户填写帖子内容并提交。
- 与其他组件关系：与帖子列表组件交互，创建成功后刷新帖子列表。

2. 删除帖子组件

- 功能：允许用户删除自己发布的帖子。
- 与其他组件关系：与帖子详情组件和帖子列表组件有关联，删除后更新界面和数据。

4.2 后端组件设计

用户服务组件设计

1. 用户注册服务

- 功能：处理用户提交的注册信息，验证用户名的唯一性，保存用户信息到数据库。
- 与其他组件关系：与用户登录服务和用户信息更新服务有交互，确保用户注册成功后能够登录和更新信息。

2. 用户登录服务

- 功能：验证用户提交的登录信息，如果验证成功则生成登录状态并返回给前端。
- 与其他组件关系：与用户信息服务和帖子管理服务有交互，通过登录状态控制用户权限和操作。

3. 用户信息更新服务

- 功能：允许用户修改个人信息或更新头像。
- 与其他组件关系：与用户登录服务和帖子管理服务交互，确保只有已登录用户才能更新个人信息。

帖子服务组件设计

1. 创建帖子服务

- 功能：接收用户提交的帖子内容，保存到数据库并返回创建成功的消息。
- 与其他组件关系：与帖子管理组件和用户服务组件有交互，确保只有登录用户才能创建帖子。

2. 删除帖子服务

- 功能：验证用户身份和帖子所有权，删除数据库中的帖子记录。
- 与其他组件关系：与帖子管理组件和用户服务组件有交互，确保只有帖子作者或管理员能删除帖子。

5、数据库设计

ER图

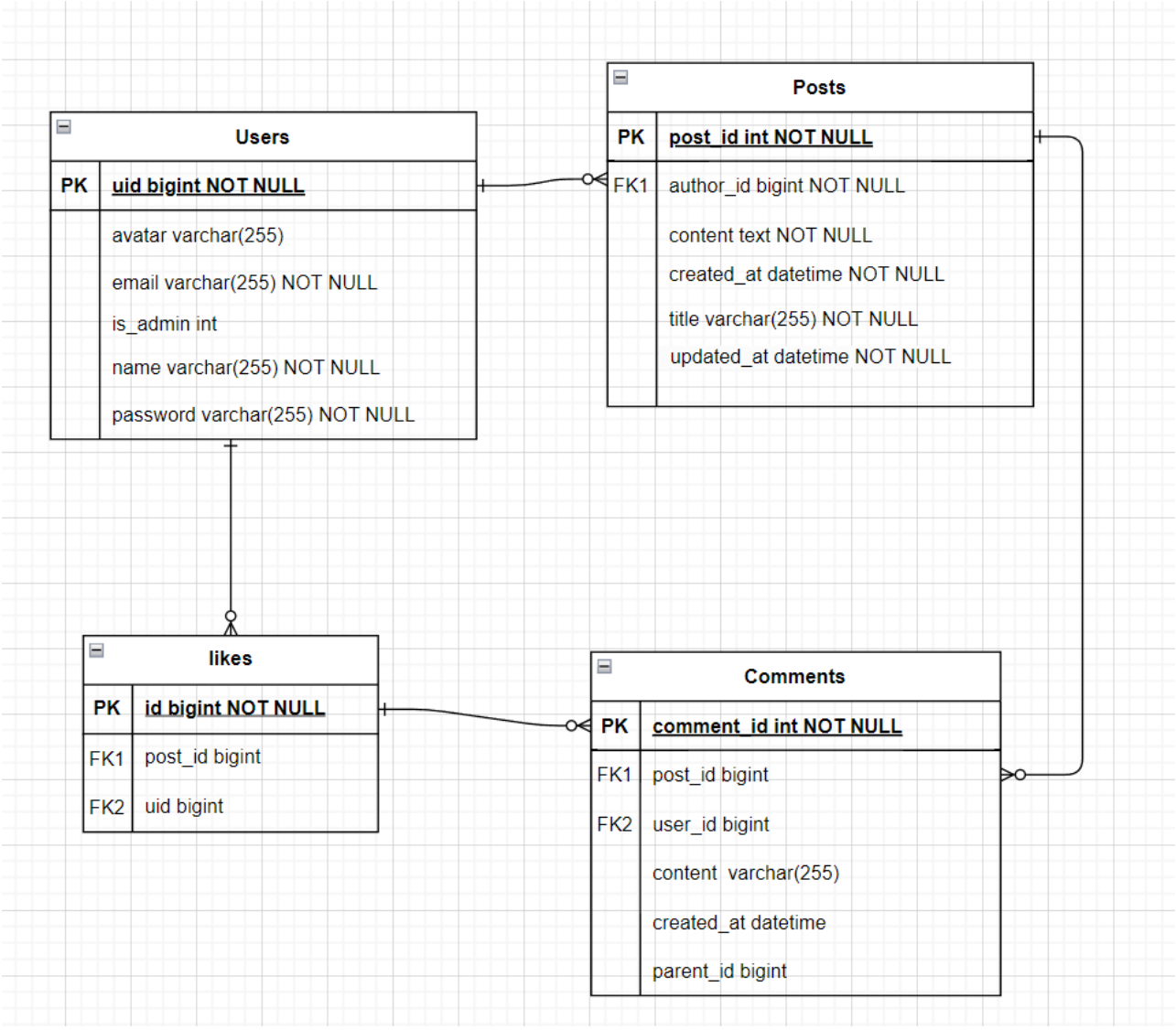
- 实体：
    - 用户（User）：
        - 属性：uid，avatar，email，is_admin，name，password
    - 帖子（Post）：
        - 属性：post_id，author_id，content，created_at，title，updated_at

    - 回复（Comment）：
        - 属性：comment_id，content，created_at，parent_id，post_id，user_id
    - 点赞（Like）：
        - 属性：id，post_id，uid

- 关系：

    - 用户和帖子之间是一对多关系，一个用户可以发布多个帖子。
    - 帖子和回复之间是一对多关系，一个帖子可以有多个回复。
    - 用户和回复之间是一对多关系，一个用户可以发表多个回复。
    - 用户和点赞之间是一对多关系，一个用户可以对多个帖子点赞。

- 图示：



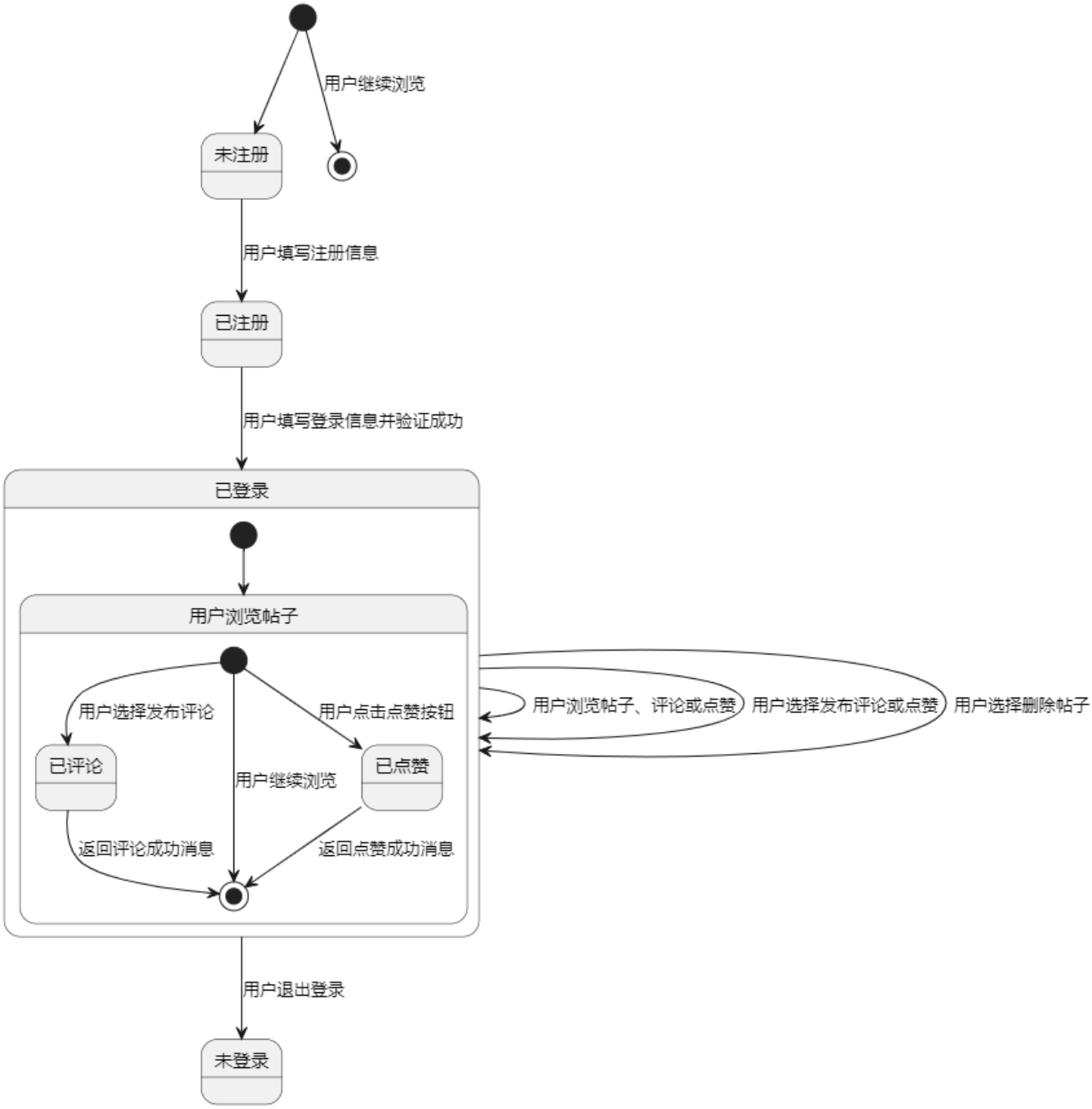
状态图

主要有以下状态和状态转换：

- 用户状态：

    - 未注册：用户尚未注册网站账号。
    - 已注册：用户已完成注册，可以选择登录或浏览帖子。
    - 已登录：用户已成功登录，可以进行评论、点赞和删除帖子等操作。

- 帖子状态：
    - 存在：帖子在系统中存在，用户可以浏览、评论和点赞。
    - 已删除：帖子被作者或管理员删除，不再显示在帖子列表中。
- 点赞状态：
    - 已点赞：用户已对帖子进行点赞。
    - 未点赞：用户未对帖子进行点赞。
- 评论状态：
    - 已评论：用户已发布评论。
    - 未评论：用户尚未发布评论。



6、数据表设计

- 用户表：
  - uid: 用户ID（主键，bigint）
  - avatar: 头像URL（varchar(255)）
  - email: 邮箱（varchar(255)）
  - isAdmin: 是否管理员（int）
  - name: 用户名（varchar(255)）
  - password: 密码（varchar(255)）

```
mysql> desc users;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| uid      | bigint       | NO   | PRI | NULL    | auto_increment |
| avatar   | varchar(255) | YES  |     | NULL    |                |
| email    | varchar(255) | NO   |     | NULL    |                |
| is_admin | int          | YES  |     | NULL    |                |
| name     | varchar(255) | NO   |     | NULL    |                |
| password | varchar(255) | NO   |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

- 帖子表：
  - post_id: 帖子ID（主键，int）
  - author_id: 作者ID（bigint）
  - content: 帖子内容（text）
  - created_at: 创建时间（datetime）
  - title: 帖子标题（varchar(255)）
  - updated_at: 更新时间（datetime）

```
mysql> desc posts;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| post_id    | int          | NO   | PRI | NULL    | auto_increment |
| author_id  | bigint       | NO   |     | NULL    |                |
| content    | text         | NO   |     | NULL    |                |
| created_at | datetime     | NO   |     | NULL    |                |
| title      | varchar(255) | NO   |     | NULL    |                |
| updated_at | datetime     | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

- 评论表：
  - comment_id: 回复ID（主键，bigint）
  - content: 回复内容（varchar(255)）
  - created_at: 创建时间（datetime）
  - parent_id: 父回复ID（bigint）
  - post_id: 所属帖子ID（bigint）
  - user_id: 用户ID（bigint）

```
mysql> desc comments;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| comment_id | bigint       | NO   | PRI | NULL    | auto_increment |
| content    | varchar(255) | YES  |     | NULL    |                |
| created_at | datetime     | YES  |     | NULL    |                |
| parent_id  | bigint       | YES  |     | NULL    |                |
| post_id    | bigint       | YES  |     | NULL    |                |
| user_id    | bigint       | YES  | MUL | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
```

- 点赞表：
  - id: 点赞ID（主键，bigint）
  - post_id: 帖子ID（bigint）
  - uid: 用户ID（bigint）

```
mysql> desc likes;
+---------+--------+------+-----+---------+----------------+
| Field   | Type   | Null | Key | Default | Extra          |
+---------+--------+------+-----+---------+----------------+
| id      | bigint | NO   | PRI | NULL    | auto_increment |
| post_id | bigint | YES  |     | NULL    |                |
| uid     | bigint | YES  |     | NULL    |                |
+---------+--------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

以上就是需求分析和设计规格文档的全部内容了。