**需求分析规约**

目录

[需求分析规约 4](#_Toc14389)

[修订历史记录 4](#_Toc32004)

[1. 引言 7](#_Toc14752)

[1.1 背景 7](#_Toc677)

[1.1.1 软件系统的名称 7](#_Toc2872)

[1.1.2 项目的任务提出者、开发者、用户及实现该软件的计算中心或计算机网络 7](#_Toc26125)

[1.1.3 软件系统与其他系统的相互关系 7](#_Toc22935)

[1.2 参考资料 8](#_Toc3530)

[1.3 假定和约束 8](#_Toc13293)

[1.3.1 假定 8](#_Toc27590)

[1.3.2 约束 8](#_Toc11810)

[1.4 用户的特点 9](#_Toc12100)

[1.4.1 操作人员 9](#_Toc366)

[1.4.2 维护人员 9](#_Toc16307)

[1.4.3 预期使用频度 9](#_Toc1606)

[2. 功能需求 9](#_Toc3924)

[2.1 系统范围 9](#_Toc31497)

[2.1.1 概要说明 9](#_Toc3004)

[2.1.2 作用范围 10](#_Toc25822)

[2.1.3 系统组成与接口关系 11](#_Toc2769)

[2.2 系统总体流程 12](#_Toc7426)

[2.3 总类图 13](#_Toc4353)

[2.4 需求分析建模 14](#_Toc6655)

[2.4.1 功能建模 14](#_Toc25090)

[2.4.2 数据建模 45](#_Toc3093)

[2.4.3 行为建模 80](#_Toc1907)

[3.1 性能要求 114](#_Toc2985)

[3.1.1 精度 114](#_Toc8064)

[3.1.2 时间特性要求 114](#_Toc9399)

[3.1.3 输入输出要求 115](#_Toc3728)

[3.2 数据管理能力要求 116](#_Toc29048)

[3.3 安全及保密性要求 117](#_Toc10996)

[3.4. 灵活性要求 118](#_Toc1105)

[3.5. 其他专门要求 119](#_Toc2144)

[4. 运行环境规定 119](#_Toc24235)

[4.1. 设备 119](#_Toc15252)

[4.2. 支持软件 120](#_Toc12273)

[4.3. 与其他系统的接口 120](#_Toc29249)

[4.4. 控制 124](#_Toc30290)

[5. 需求跟踪管理 124](#_Toc3792)

**修订历史记录**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **编写日期** | **SEPG** | **版本** | **说明** | **作者** | **评审时间** | 评审参与人员 | 评审后修改批准日期 | 确认签字人员 | | 2024/11/05 | Air Rowing | v0.1 | 需求调研，团队进行不限于问卷、访谈等方式的调研 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2024/11/15 | 章卓源 | 2024/11/15 | 章卓源 | | 2024/11/15 | Air Rowing | v0.2 | 前期需求调研结果讨论及书写 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2024/11/27 | 陈茁扬 | 2024/11/29 | 陈茁扬 | | 2024/11/27 | Air Rowing | v0.3 | 软件技术栈讨论，市场竞品分析 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2024/12/12 | 李沅衡 | 2024/11/15 | 李沅衡 | | 2024/12/12 | Air Rowing | v1.0 | 初步需求规约 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2024/12/20 | 杨宇琨 | 2024/12/22 | 杨宇琨 | | 2024/12/20 | Air Rowing | v1.1 | 初步需求分析规约  ，需求规约补充 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2024/12/28 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2024/12/28 | 章卓源，陈茁扬，李沅衡，杨宇琨 | | 2024/12/28 | Air Rowing | v1.2 | 初步概要设计规约 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2024/12/31 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2024/12/31 | 章卓源，陈茁扬，李沅衡，杨宇琨 | | 2024/12/31 | Air Rowing | v1.3 | 初步详细设计规约，概要设计规约完善 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2025/1/1 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2025/1/1 | 章卓源，陈茁扬，李沅衡，杨宇琨 | | 2025/1/1 | Air Rowing | v2.0 | 详细设计规约完善 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2025/1/1 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2025/1/2 | 章卓源，陈茁扬，李沅衡，杨宇琨 | | 2025/1/2 | Air Rowing | v3.0 | 和杜老师讨论过后，进行整体升级 | 章卓源，陈茁扬，李沅衡，杨宇琨 | 2025/1/2 | 章卓源，陈茁扬，李沅衡，杨宇琨，杜庆峰 | 2025/1/2 | 章卓源，陈茁扬，李沅衡，杨宇琨 |   同步自文档: <https://n133h6cwp1u.feishu.cn/docx/NvMWdvBx3ou1oLxdcCicIOEqnNV#CNOedQ9TasWH98b7U7Ecznilnmc> |

1. **引言**

**1.1 背景**

**1.1.1 软件系统的名称**

**AirRowing**项目旨在开发一款智能赛艇应用软件及数字化赛艇辅助训练平台，专为赛艇运动员、教练员及爱好者设计，旨在提升赛艇训练的科学性和效率，促进赛艇运动的普及与发展。

**1.1.2 项目的任务提出者、开发者、用户及实现该软件的计算中心或计算机网络**

* **任务提出者：** 本项目由同济大学赛艇队提出，旨在解决赛艇训练中存在的技术指导不足、数据管理不完善以及缺乏交流平台等问题，推动赛艇运动的数字化进步。
* **开发者：** 开发团队由**同济大学计算机科学与技术学院软件工程专业**的学生及赛艇队中具备开发能力的队员组成，结合项目组成员的专业知识和实际需求，共同开发和维护AirRowing平台。
* **用户：**
* **核心用户：** 高校赛艇队的队员和教练员，主要使用平台进行训练记录、数据分析和俱乐部管理。
* **主要用户：** 赛艇运动爱好者，通过平台学习划船姿态、记录日常训练并参与社区交流。
* **潜在用户：** 资深赛艇运动员，寻求更专业的训练辅助和数据分析支持。
* **计算中心与计算机网络：** 软件系统将在云服务器（如阿里云、AWS）上部署，确保高可用性和可扩展性。前端用户通过互联网访问平台，无需依赖特定的内部网络环境，支持跨地域的用户访问和数据同步。

**1.1.3 软件系统与其他系统的相互关系**

AirRowing平台计划与以下系统和机构进行基本的相互联系与集成：

* **大语言模型：**
* 利用大语言模型强大的自然语言处理与多模态分析能力，实现对赛艇训练数据的综合解读。
* 在用户对训练问题、训练记录等提出疑问或希望获得指导时，大语言模型可根据上下文理解并反馈专业建议。
* 针对用户上传的赛艇姿态图片或视频，LLM 与图像/视频处理算法配合，给出更精准的姿态纠正和优化思路。
* **社交媒体平台：** 预期支持将训练成果和社区互动内容分享到主流社交媒体（如微博、微信），扩大平台影响力和用户基础。
* **第三方设备接口：** 预留开放API，支持与可穿戴设备（如智能手环）、智能划桨等硬件设备的数据交互，丰富训练数据来源和分析维度。
* **数据分析与云服务：** 利用云计算和大数据分析服务，提升平台的数据处理能力和智能分析水平，确保系统的高效运行和数据安全。

**1.2 参考资料**

* Design Patterns
* 软件工程——实践者的研究方法（8th）

**1.3 假定和约束**

**1.3.1 假定**

* **用户参与度：** 假定目标用户（赛艇队员、教练员及爱好者）积极参与平台使用，定期记录训练数据并参与社区互动。
* **技术可行性：** 假定现有的姿态识别算法和AI技术能够满足平台对训练分析和指导的需求，且能够在预定的硬件环境下高效运行。
* **数据可用性：** 假定能够获取足够量且质量良好的训练数据，用于算法训练和系统测试，确保系统功能的有效性和准确性。

**1.3.2 约束**

* **经费限制：** 项目预算有限，需合理分配资源，优先保证核心功能的开发和系统的稳定运行。
* **开发期限：** 项目计划在12个月内完成初步开发和测试，确保在下一赛季开始前投入使用。
* **设备条件：** 开发和部署需要配备高性能服务器和必要的硬件设备，如GPU加速卡，以支持姿态识别算法的运行。
* **用户资料准备：** 用户需提供准确的训练数据和个人信息，确保系统数据的完整性和可靠性。
* **交流与协作：** 开发团队需与赛艇队员和教练员保持密切沟通，及时反馈和调整需求，避免信息不对称导致的开发偏差。

**1.4 用户的特点**

**1.4.1 操作人员**

* **教育水平：** 用户主要为高校学生和教练员，具备基本的计算机操作能力，能够熟练使用Web应用和移动设备。
* **技术专长：** 普通用户不要求具备专业的技术背景，平台设计需简洁易用，降低操作门槛；维护人员需具备一定的软件维护和数据管理能力，熟悉平台的基本操作和常见问题处理。

**1.4.2 维护人员**

* **教育水平：** 维护人员多为具备计算机相关专业知识的项目组成员或赛艇队中具备开发能力的队员，能够进行日常的系统维护和技术支持。
* **技术专长：** 需要熟悉Java编程、数据库管理（MySQL）、前后端框架（Spring Boot、React/Vue.js），具备基本的故障排除和系统优化能力。

**1.4.3 预期使用频度**

* **日常使用：** 训练分析模块和俱乐部管理模块在用户进行日常训练和活动安排时频繁使用，预计主要在白天高频使用。
* **社区互动：** 社区模块的使用频度较为均匀，用户在任何时间段均可能进行帖子发布、浏览和互动，整体使用具有一定的平稳性，无明显高峰。
* **维护与管理：** 维护人员定期进行系统检查、数据备份和功能更新，使用频率较低但需保持持续关注，确保系统的稳定性和安全性。

2. **功能需求**

**2.1 系统范围**

**2.1.1 概要说明**

**AirRowing**系统旨在开发一款智能赛艇应用软件及数字化辅助训练平台，专注于满足赛艇运动员、教练员及爱好者的多样化需求。系统的开发意图在于通过集成先进技术（如大语言模型和人工智能智能指导），提升赛艇训练的科学性和效率，推动赛艇运动的普及与发展。

系统的应用目标包括：

* **提升训练效率与专业性**：通过实时姿态纠正和详尽的数据分析，帮助用户快速掌握正确的划船技巧，减少受伤风险，优化训练计划。
* **推动俱乐部数字化管理**：提供精准的训练和比赛记录与分析工具，辅助俱乐部管理者制定科学的训练计划，监控运动员进步，简化日常管理流程。
* **构建活跃的赛艇社区**：为赛艇爱好者、运动员和教练员提供信息共享与交流的平台，促进知识传播和经验交流，增强社区凝聚力和影响力。
* **促进赛艇运动发展**：通过智能化和数字化手段，吸引更多人参与赛艇健身和赛艇运动，提升赛艇运动的社会知名度和参与度。

**2.1.2 作用范围**

**AirRowing**系统的作用范围涵盖以下几个核心功能模块：

1. **账号管理模块**：

* 支持用户通过用户名和邮箱进行注册和登录。
* 提供密码修改、个人信息查看和编辑功能。
* 确保用户数据的安全性和隐私保护。

1. **训练分析模块**：

* 用户可以添加和管理训练记录，支持多种训练数据的输入与存储。
* 通过大语言模型分析训练数据，提供训练效果评估，帮助用户了解自身训练进展与不足。
* 通过大语言模型分析用户上传的赛艇图片或视频，提供精准的姿态评估与纠正建议。
* 利用人工智能RAG技术，为用户提供赛艇训练的智能指导。
* 用户可将训练成果分享到社区，与其他用户交流经验。

1. **俱乐部模块**：

* 用户可以通过平台报名参与俱乐部组织的训练和比赛，方便查看报名情况。
* 提供俱乐部信息编辑功能，管理者可更新俱乐部动态、成员信息及训练数据统计。
* 系统自动统计俱乐部成员的训练数据与比赛成绩，辅助管理者制定科学的训练计划。
* 促进成员间的互动与竞争，增强团队凝聚力。

1. **赛艇社区模块**：

* 用户可以发布、管理和浏览帖子，分享训练心得、赛艇资讯等内容。
* 支持帖子互动（如评论、点赞）、查看社交主页及搜索相关内容。
* 支持线上线下赛艇活动的组织与管理，提升社区的活跃度与用户粘性。
* 通过社区平台传播赛艇文化，提升赛艇运动的社会知名度与影响力。

**2.1.3 系统组成与接口关系**

以下为系统组成的示意方框图及其说明：



**说明：**

* **用户端**：用户通过PC或移动设备访问**AirRowing**平台，进行账号管理、训练记录、俱乐部管理和社区互动等操作。
* **AirRowing**：作为核心Web应用平台，集成各功能模块，处理用户请求，进行数据分析与反馈。
* **后端服务与数据库**：负责数据的存储与管理，运行姿态识别算法和AI智能指导模块，确保数据的安全性和处理效率。
* **第三方系统与设备接口**：**预期**与大语言模型、社交媒体平台和可穿戴设备进行数据交换和功能集成，扩展系统的应用范围和功能。

**2.2 系统总体流程**

**业务流 & 总体流程（图+文本示例）：**

**用户注册/登录：**

* 用户通过用户名或邮箱注册账户，完成登录后进入系统主界面。

**账号管理：**

* 用户可查看和编辑个人信息，确保账户信息的准确性和安全性。

**俱乐部管理：**

* 用户可以创建和管理自己的俱乐部，包括俱乐部信息编辑、成员管理等。

**训练分析：**

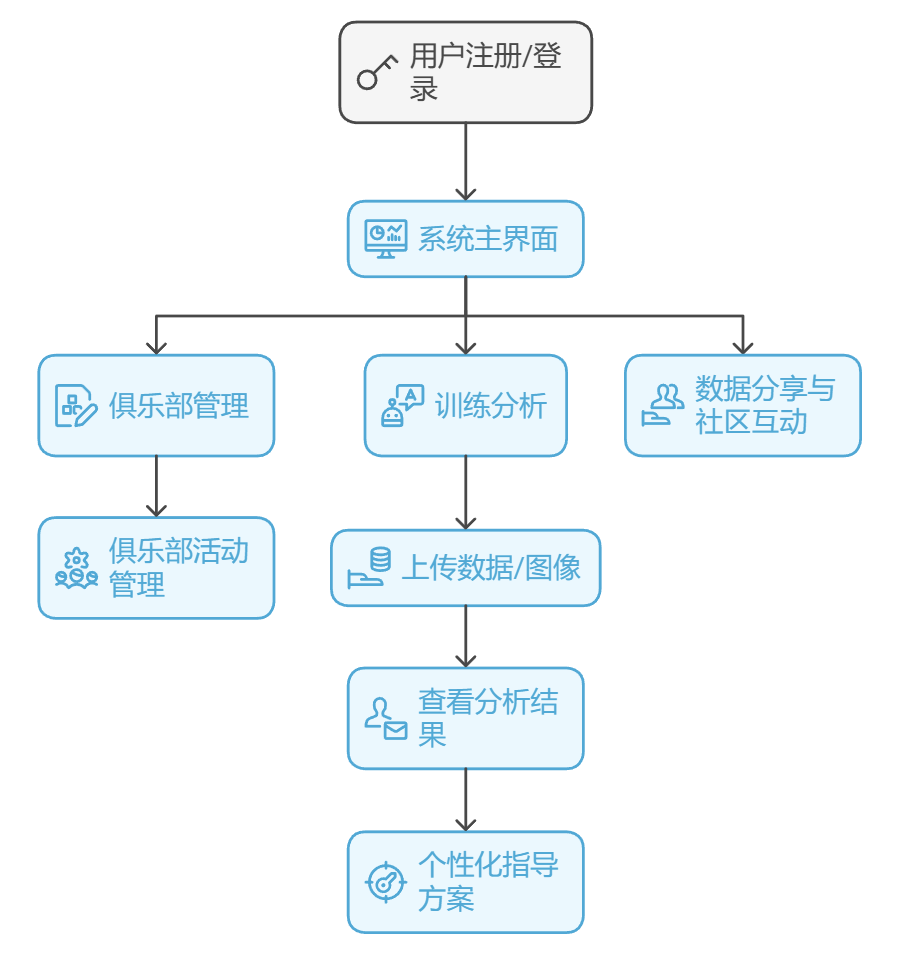
* 用户上传训练视频，系统调用大语言模型进行分析。
* 用户可查看训练数据并分析，获取个性化指导方案。

**数据分享与社区互动：**

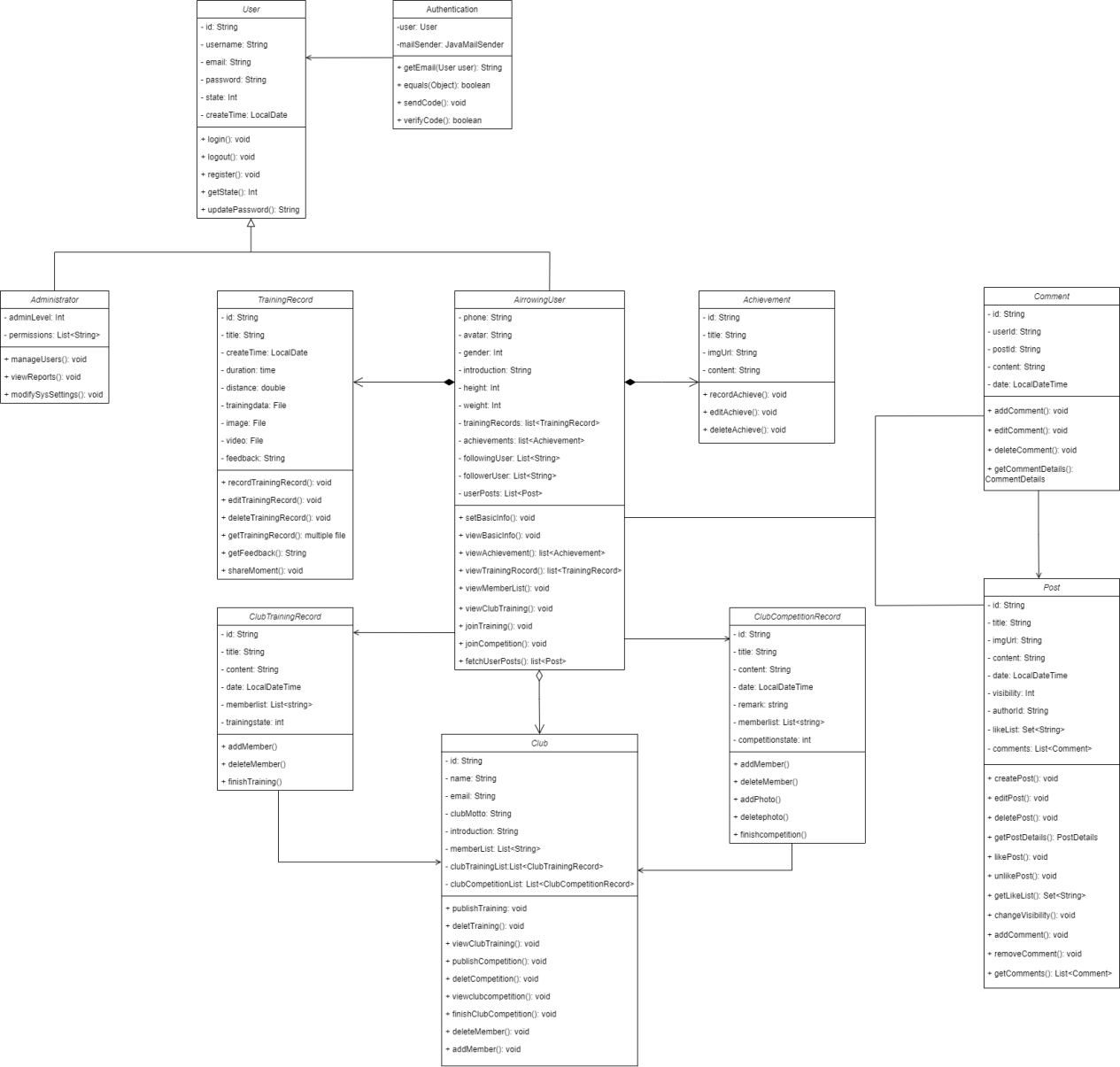
* 用户可将训练成果分享到赛艇社区，与其他用户互动交流。

**俱乐部活动管理：**

* 用户可在俱乐部模块中报名训练和比赛，查看报名情况，参与俱乐部活动。



**2.3 总类图**

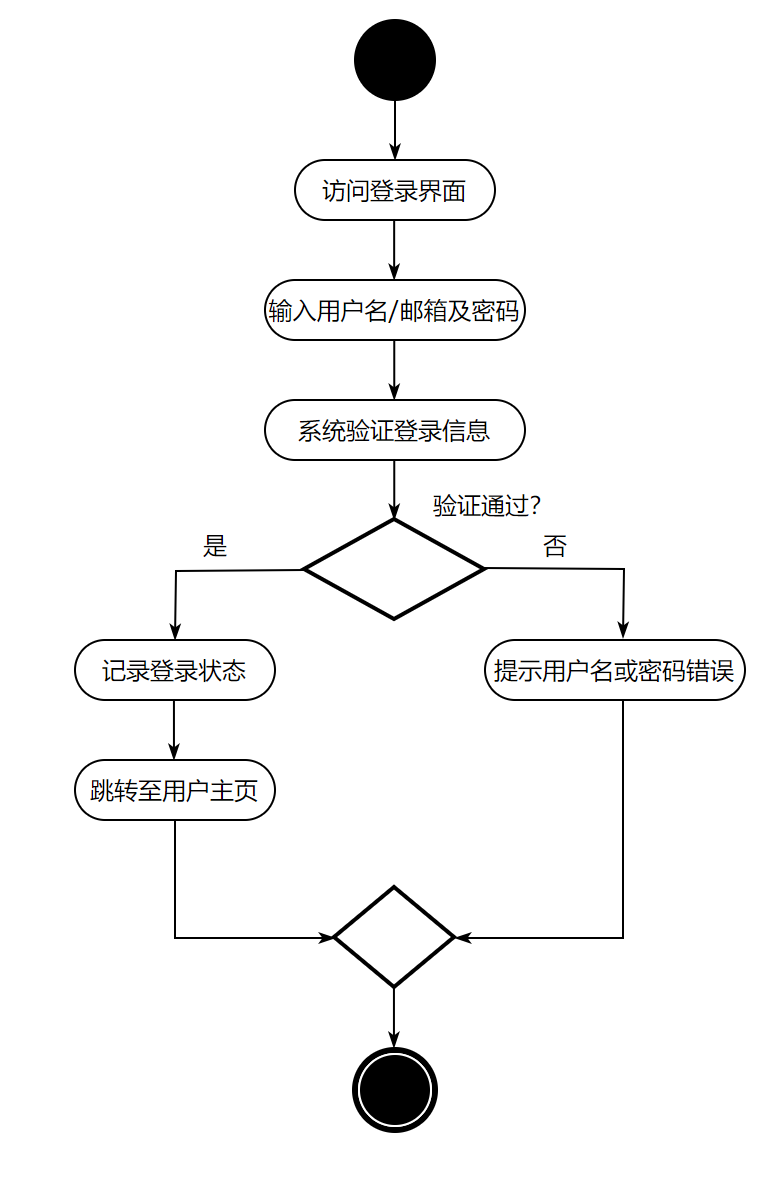


**2.4 需求分析建模**

**2.4.1 功能建模**

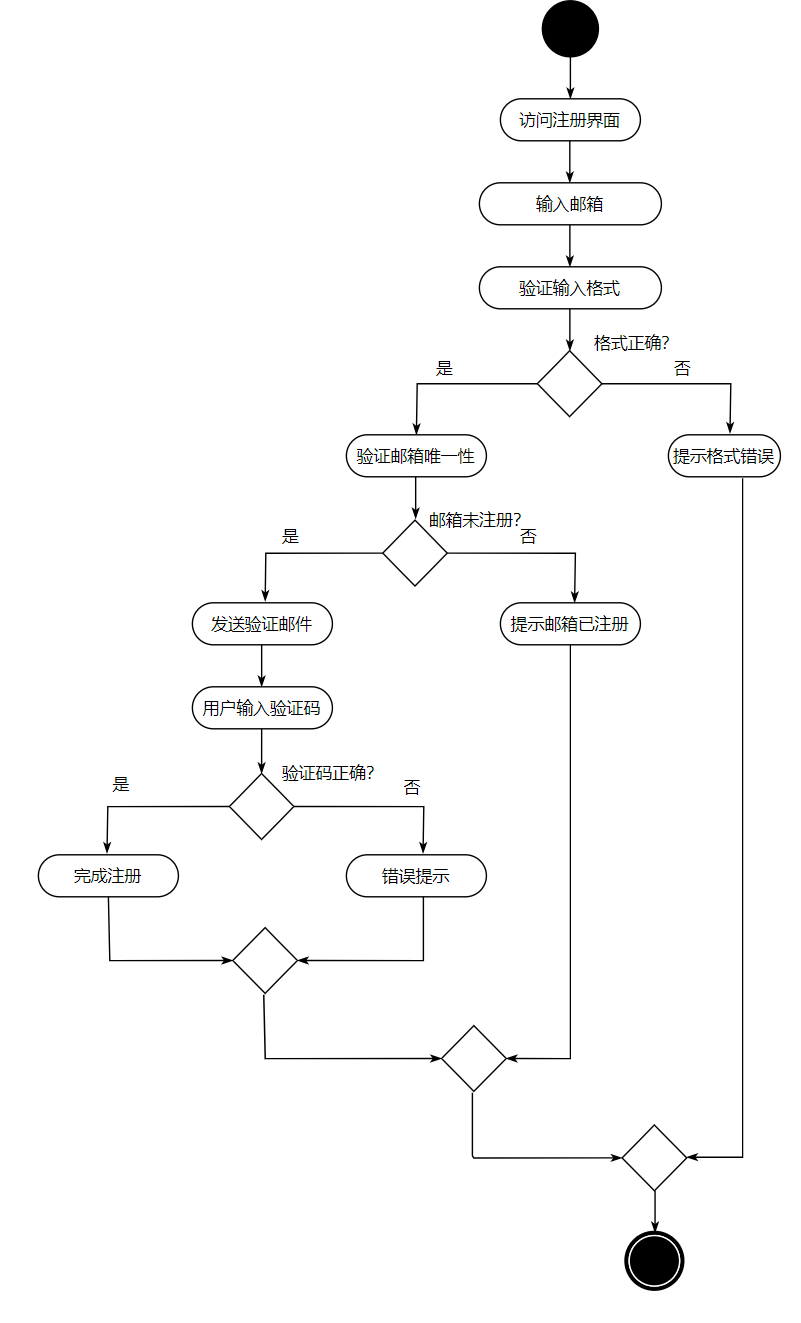
**2.4.1.1 账号管理模块**

**2.4.1.1.1 用户登录**



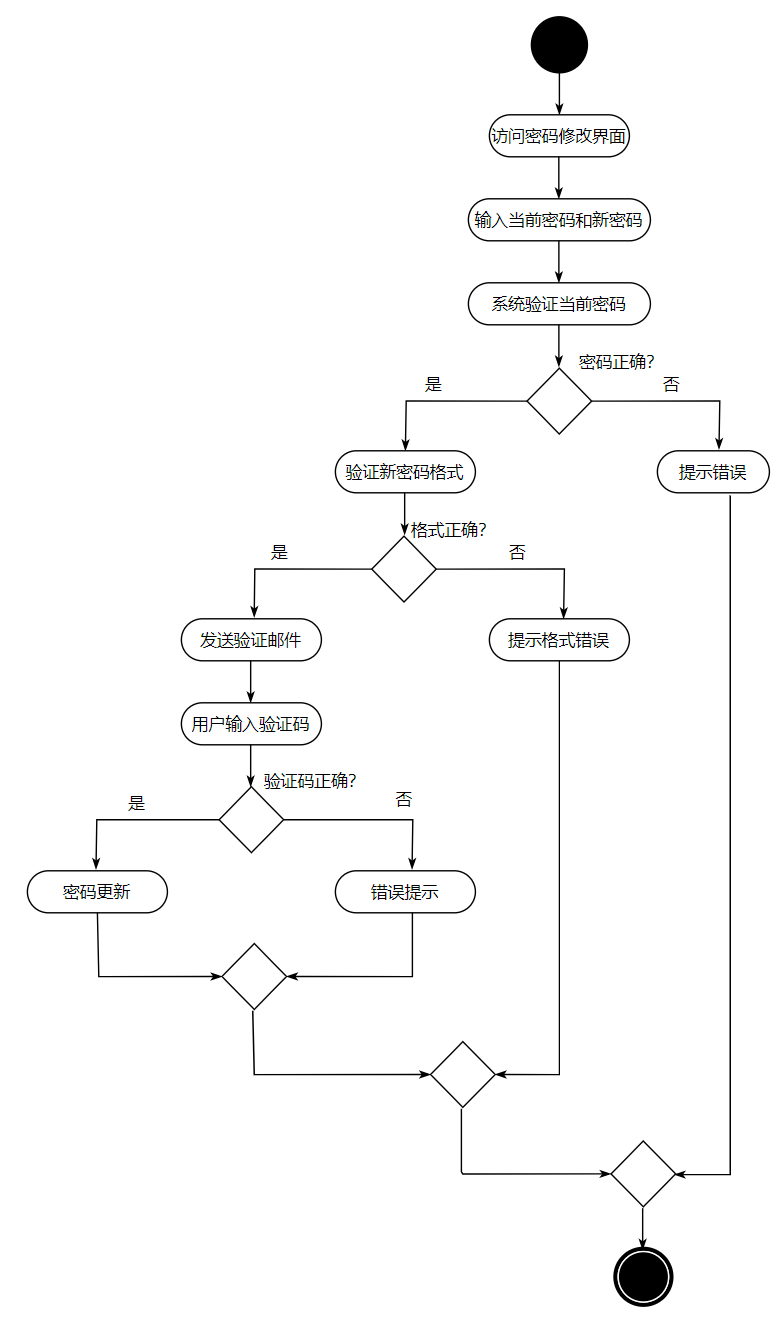
用户访问登录界面并输入用户名/邮箱和密码后，系统验证登录信息。如果验证通过，系统记录登录状态并跳转至用户主页；若验证失败，系统提示“用户名或密码错误”，用户可重新输入登录信息。

**2.4.1.1.2 账号注册**



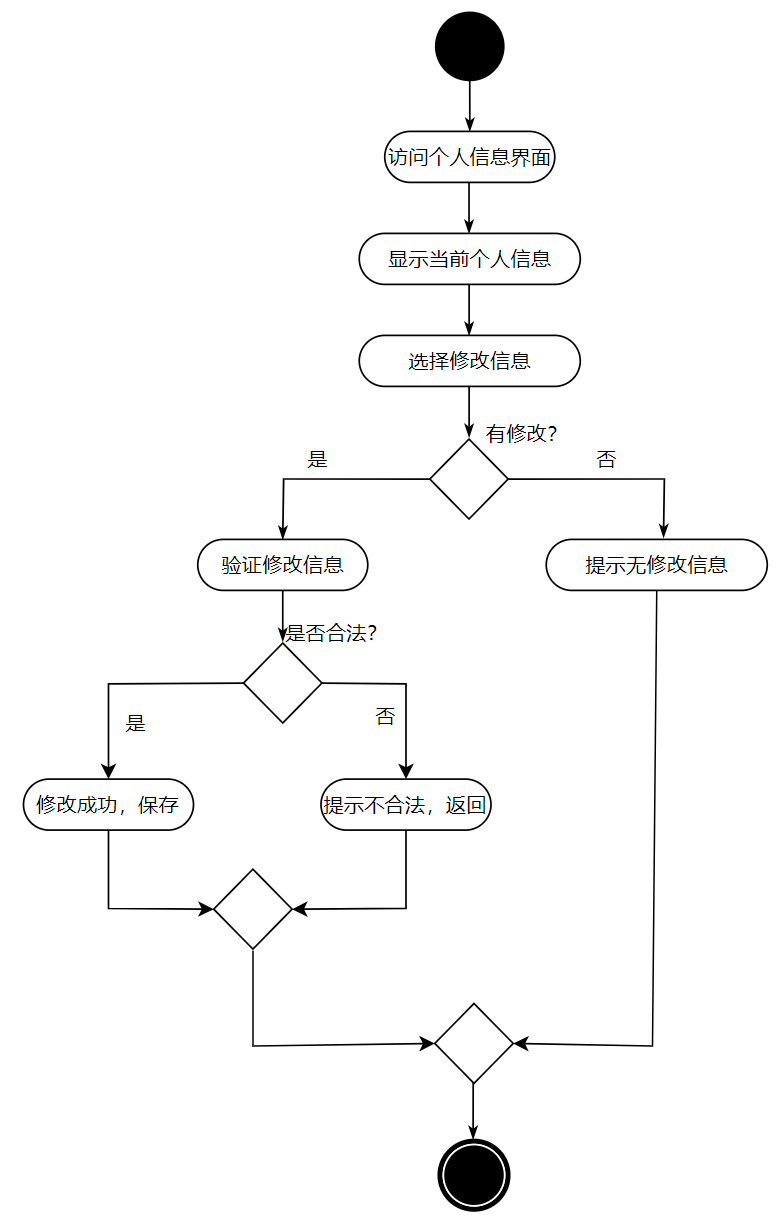
用户访问注册界面并输入邮箱后，系统验证输入格式。如果格式不正确，提示“格式错误”；若格式正确，系统检查邮箱是否未注册。如果邮箱已注册，提示“邮箱已注册”；若邮箱未注册，系统发送验证邮件。用户输入收到的验证码，系统验证其正确性。若验证码正确，完成注册；若验证码错误，系统提示错误。

**2.4.1.1.3 密码修改**



用户访问密码修改界面，输入当前密码和新密码后，系统首先验证当前密码的正确性。如果当前密码错误，系统提示错误并返回输入界面；如果密码正确，系统进一步验证新密码的格式是否符合要求。若格式正确，系统发送验证码到用户邮箱，用户需输入验证码并验证正确性。通过验证后，系统更新密码；若验证码验证失败，系统提示错误。

**2.4.1.1.4 个人信息查看和编辑**

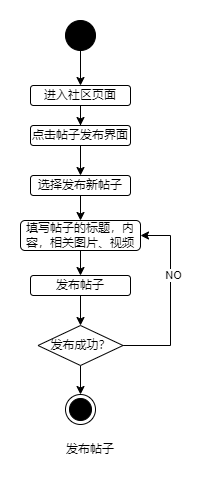


用户访问个人信息界面后，系统会展示当前的个人信息。当用户选择修改信息时，系统检查是否有修改内容。如果没有修改，系统提示“无修改信息”；如果有修改，系统验证输入的修改信息是否合法。如果信息合法，则保存修改并提示修改成功；若信息不合法，则提示“信息不合法”并返回修改界面。

**2.4.1.2 社区模块**

iteration1：(旧版）

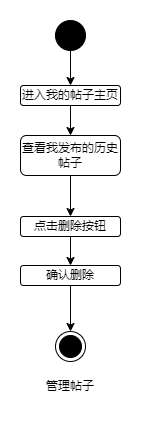
**2.4.1.2.1 发布帖子**



1. **进入社区页面**：用户进入社区主页面。
2. **点击帖子发布界面**：用户选择进入帖子发布界面。
3. **选择发布新帖子**：用户点击“发布新帖子”按钮，进入帖子编辑页面。
4. **填写帖子内容**：用户填写帖子标题、正文内容，并上传相关的图片或视频附件。
5. **发布帖子**：用户确认内容后点击发布按钮提交帖子。
6. **检查是否发布成功**：

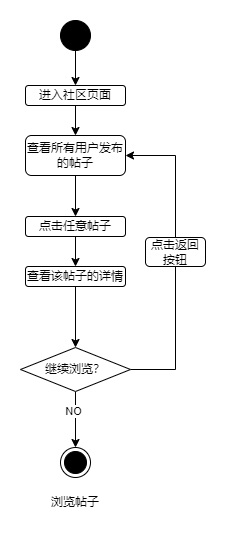
* 若发布成功，系统提示用户“发布成功”，帖子显示在社区中。
* 若发布失败，系统提示失败原因（如内容不合规），用户可重新编辑并提交。

**2.4.1.2.2 管理帖子**



1. **进入我的帖子主页**：用户进入“我的帖子”页面，查看个人发布的帖子列表。
2. **查看历史帖子**：用户浏览自己发布的历史帖子。
3. **点击删除按钮**：用户选择要删除的帖子，并点击对应的“删除”按钮。
4. **确认删除**：系统弹出确认框，用户确认删除操作。
5. **管理帖子**：系统删除指定帖子，并更新帖子列表显示。1

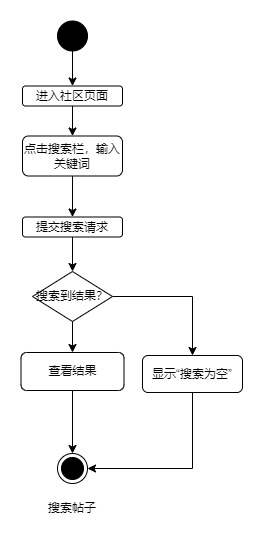
**2.4.1.2.3 浏览帖子**



1. **进入社区页面**：用户进入社区主页面，查看所有用户发布的帖子列表。
2. **查看所有帖子**：用户浏览社区中所有帖子，并选择感兴趣的帖子。
3. **点击任意帖子**：用户点击选中的帖子进入帖子详情页面。
4. **查看帖子详情**：系统显示帖子的详细内容，包括正文、图片、视频及评论等信息。
5. **判断是否继续浏览**：

* 若用户选择继续浏览，则返回帖子列表并重复查看其他帖子。
* 若用户停止浏览，点击返回按钮退出详情页面，回到社区主页面。

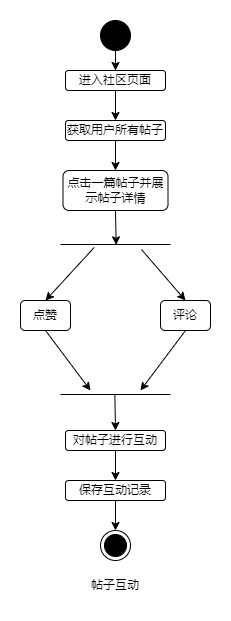
**2.4.1.2.4 搜索帖子**



1. **进入社区页面**：用户进入社区主页面。
2. **点击搜索栏**：用户选择搜索栏并输入关键词。
3. **提交搜索请求**：用户点击搜索按钮提交请求，系统根据关键词进行搜索。
4. **判断是否搜索到结果**：

* **有结果**：系统显示符合关键词的帖子列表，用户可浏览搜索结果。
* **无结果**：系统提示“搜索为空”，用户可修改关键词重新搜索。

**2.4.1.2.5 帖子互动**

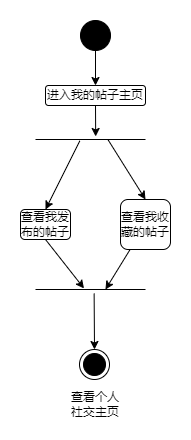


1. **进入社区页面**：用户进入社区主页面，获取所有帖子列表。
2. **获取用户所有帖子**：系统加载并展示用户发布的所有帖子。
3. **点击帖子并查看详情**：用户选择一篇帖子，进入帖子详情页面，查看帖子的完整内容。
4. **互动操作**：用户可以对帖子进行以下操作：

* **点赞**：点击点赞按钮，为帖子点赞。
* **收藏**：点击收藏按钮，将帖子保存至收藏列表。
* **评论**：输入评论内容并提交，与帖子进行互动。

1. **保存互动记录**：系统记录用户的所有互动行为（点赞、收藏、评论），并更新数据库。

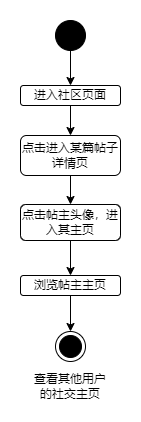
**2.4.1.2.6 查看个人社交主页**



1. **进入社区页面**：用户进入我的帖子页面，获取与我相关的所有帖子列表。
2. **获取我的所有帖子**：

* 查看我的我发的帖子
* 查看我收藏的帖子

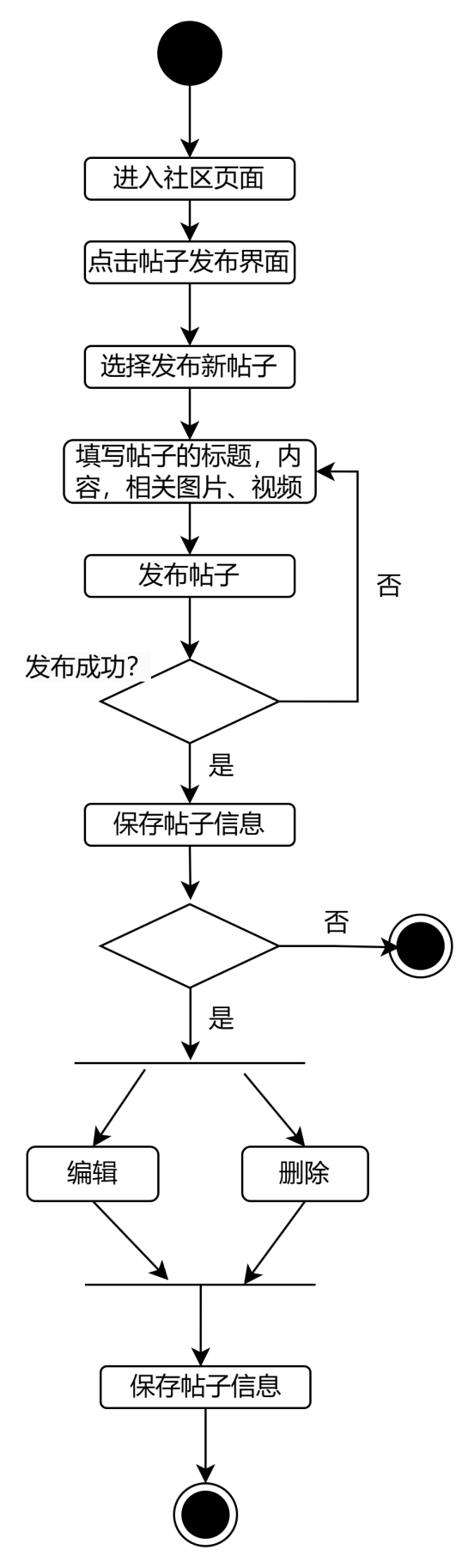
**2.4.1.2.7 查看其他用户的社交主页**



1. **进入社区页面**：用户进入社区页面，获取所有帖子列表。
2. **点击进入某篇帖子详情页**：进入帖子的详情页，有帖主的头像显示
3. **点击帖主头像：**进入其主页
4. **浏览帖主主页：**浏览帖主发过的帖子以及收藏的帖子

iteration2：

**2.4.1.2.1 发布帖子**



1. **进入社区页面**：用户首先进入社区页面，为后续操作做好准备。
2. **点击帖子发布界面**：用户点击进入帖子发布的相关界面。
3. **选择发布新帖子**：用户在帖子发布界面中选择创建一个新帖子。
4. **填写帖子信息**：用户填写帖子标题、内容，并可以添加相关的图片或视频。
5. **发布帖子**：用户完成填写后，点击“发布帖子”按钮提交。
6. **判断是否发布成功**：

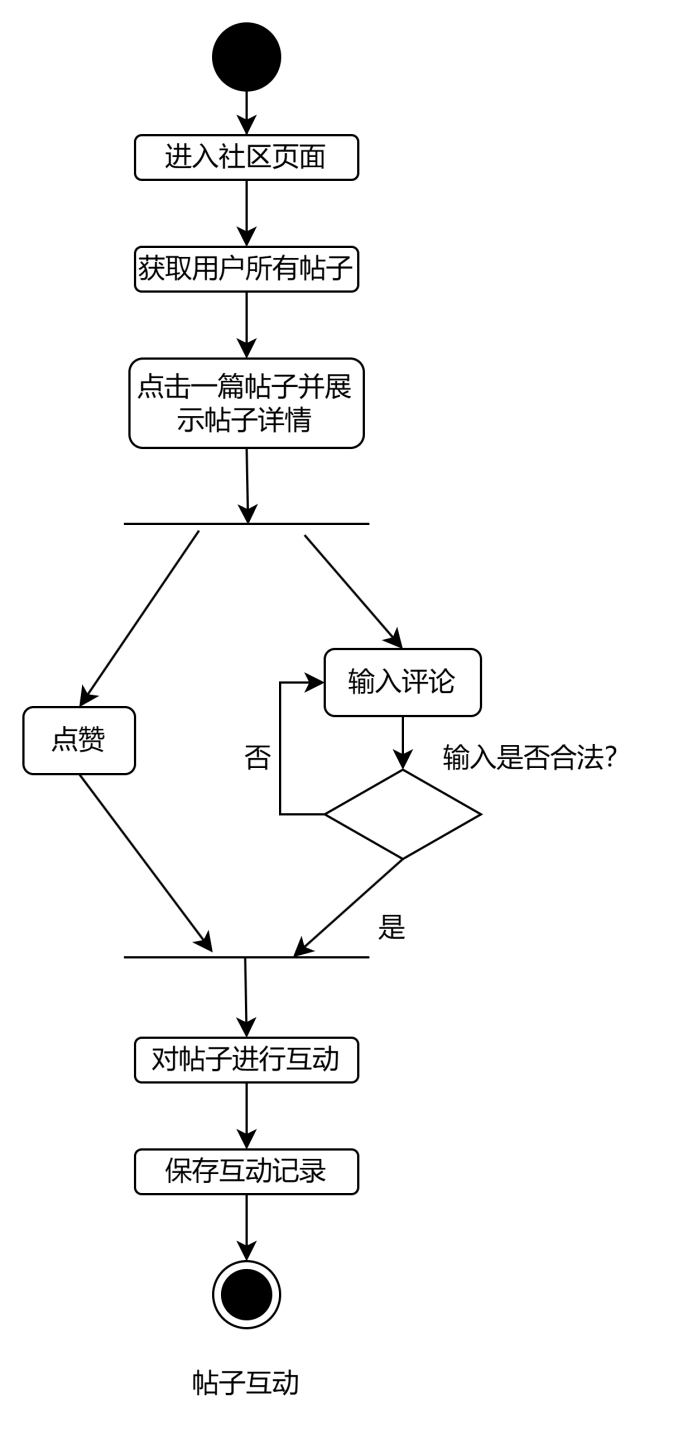
* 如果帖子发布成功，进入下一步。
* 如果发布失败，提示错误信息，重新填写帖子信息。

1. **保存帖子信息**：成功发布的帖子会保存其相关信息到系统中。
2. **编辑或删除**：用户可以对已发布的帖子进行后续的编辑或删除操作：

* **编辑**：用户修改帖子内容后，保存更新的帖子信息。
* **删除**：用户删除帖子后，相关信息从系统中移除。

1. **保存变更**：无论是编辑还是删除操作，都会触发保存变更的步骤，更新或移除帖子信息。

**2.4.1.2.2 浏览帖子**

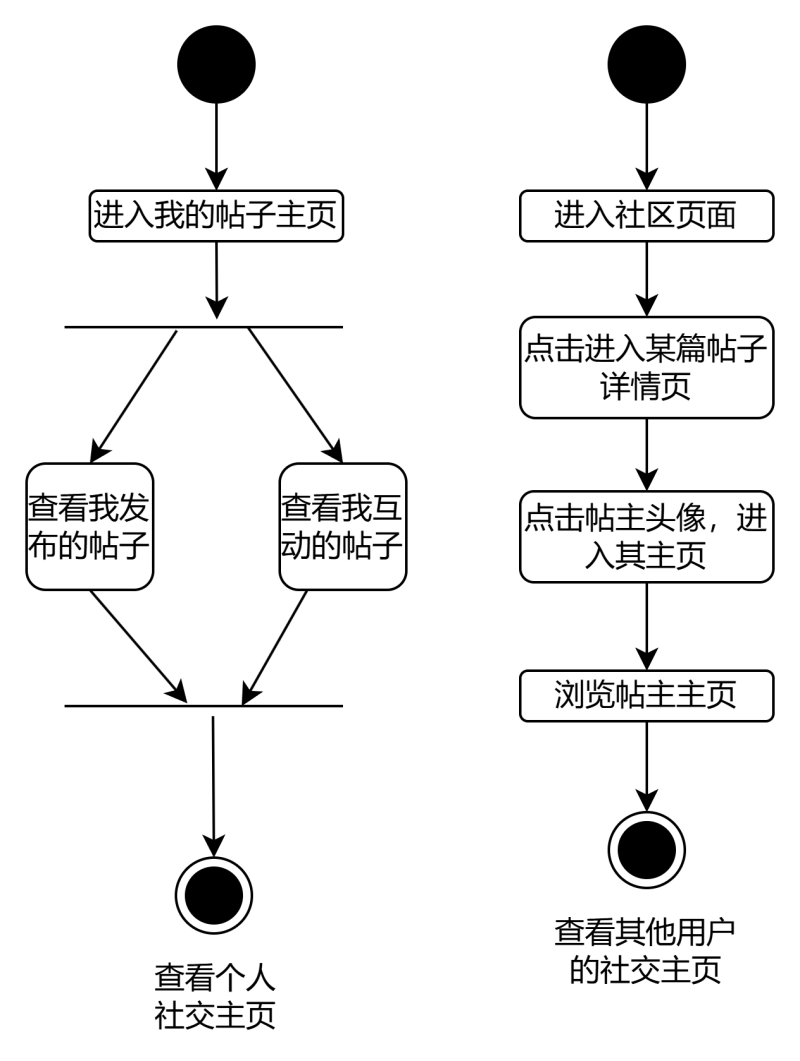


1. **进入社区页面**：用户进入社区主页面，获取所有帖子列表。
2. **获取用户所有帖子**：系统加载并展示用户发布的所有帖子。
3. **点击帖子并查看详情**：用户选择一篇帖子，进入帖子详情页面，查看帖子的完整内容。
4. **互动操作**：用户可以对帖子进行以下操作：

* **点赞**：点击点赞按钮，为帖子点赞。
* **评论**：输入评论内容并提交，需要判断输入的内容是否合法：
* 若内容为空则需要重新输入。
* 若内容合法则可以与帖子进行互动。

1. **保存互动记录**：系统记录用户的所有互动行为（点赞、评论），并更新数据库。

**2.4.1.2.3 浏览用户主页**



1. 查看个人主页
2. **进入我的帖子主页**：

* 用户进入自己的帖子主页，查看与自己相关的内容。

1. **查看我的帖子操作**：

* **查看我发布的帖子**：用户可以查看自己创建的所有帖子。
* **查看我互动的帖子**：用户可以查看自己参与互动（如评论、点赞等）的帖子。

1. 查看其他用户主页
2. **进入社区页面**：

* 用户进入社区页面，访问与其他用户相关的内容。

1. **浏览帖子详情**：

* 用户在社区页面中点击某篇帖子，进入该帖子的详情页面。

1. **进入帖子主页面**：

* 用户可以进一步点击帖主的头像，进入该帖主的个人主页。

1. **浏览帖主主页**：

* 用户在帖主主页上可以查看该帖主发布的其他帖子及相关信息。

**2.4.1.2.4 搜索帖子**



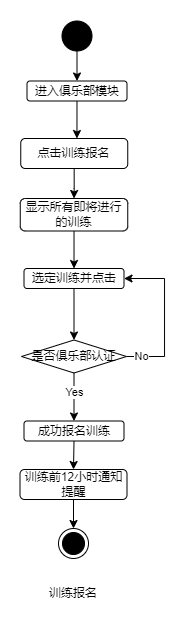
1. **进入社区页面**：用户进入社区主页面。
2. **点击搜索栏**：用户选择搜索栏并输入关键词。
3. **提交搜索请求**：用户点击搜索按钮提交请求，系统根据关键词进行搜索。
4. **判断是否搜索到结果**：

* **有结果**：系统显示符合关键词的帖子列表，用户可浏览搜索结果。
* **无结果**：系统提示“搜索为空”，用户可修改关键词重新搜索。

**2.4.1.3 俱乐部模块**

iteration1：（旧版）

**2.4.1.3.1 训练报名**

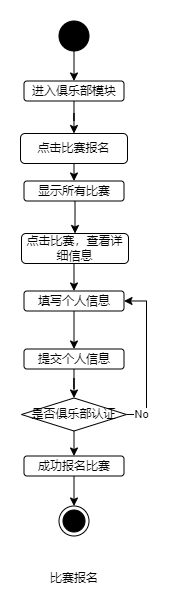


1. **进入俱乐部模块**：用户首先进入系统的俱乐部模块。
2. **点击比赛报名**：用户选择“训练报名”功能，触发报名流程。
3. **显示所有比赛**：系统向用户展示当前可报名的训练列表。
4. **点击比赛，查看详细信息**：用户选择感兴趣的训练，并点击查看具体的训练详细信息。
5. **选定训练并点击**：用户报名该训练。
6. **检查俱乐部加入状态**：

* 系统检查用户是否已加入训练所属的俱乐部。
* 如果用户尚未加入俱乐部，流程终止，提示用户加入俱乐部后再报名。

1. **成功报名比赛**：如果用户已经加入相关俱乐部，系统完成报名操作并提示成功。

**2.4.1.3.2 比赛报名**

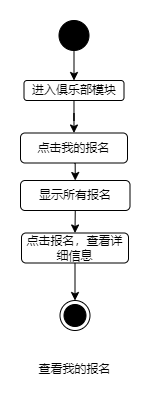


1. **进入俱乐部模块**：用户首先进入系统的俱乐部模块。
2. **点击比赛报名**：用户选择“比赛报名”功能，触发报名流程。
3. **显示所有比赛**：系统向用户展示当前可报名的比赛列表。
4. **点击比赛，查看详细信息**：用户选择感兴趣的比赛，并点击查看具体的比赛详细信息。
5. **填写个人信息**：用户在系统提供的表单中填写所需的个人信息（如姓名、联系方式等）。
6. **提交个人信息**：用户完成信息填写后，将信息提交给系统。
7. **检查俱乐部加入状态**：

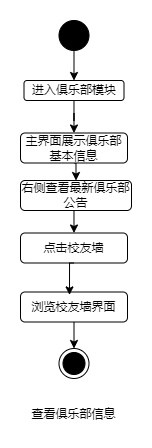
* 系统检查用户是否已加入比赛所属的俱乐部。
* 如果用户尚未加入俱乐部，流程终止，提示用户加入俱乐部后再报名。

1. **成功报名比赛**：如果用户已经加入相关俱乐部，系统完成报名操作并提示成功。

**2.4.1.3.3 查看我的报名**

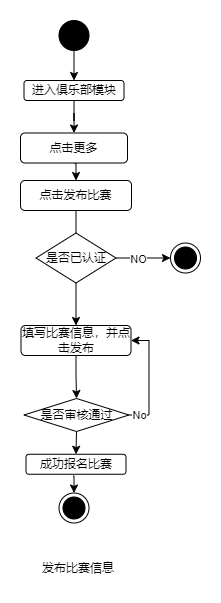


**2.4.1.3.4 查看俱乐部信息**



1. **进入俱乐部模块**：  
   用户从系统主界面选择并进入俱乐部模块，进入后将展示俱乐部的相关信息。
2. **主界面展示俱乐部基本信息**：  
   系统加载并展示俱乐部的基本信息，包括俱乐部名称、简介、成员人数等关键信息，以便用户快速了解俱乐部的核心内容。
3. **右侧查看最新俱乐部公告**：  
   系统在主界面右侧显示俱乐部的最新公告，用户可以浏览公告内容，了解俱乐部的最新动态或通知。
4. **点击校友墙**：  
   用户选择“校友墙”功能，并点击进入相关页面。
5. **浏览校友墙界面**：  
   系统加载校友墙页面内容，展示俱乐部校友的相关信息，例如校友列表、成就展示或互动留言等内容。用户可以浏览校友墙以了解更多校友动态。
6. **查看俱乐部信息**：  
   用户可以进一步查看更详细的俱乐部信息，如俱乐部的活动历史、发展规划等，或者返回主界面继续浏览其他信息。

**2.4.1.3.5 发布比赛信息**



1. **进入俱乐部模块**：用户进入俱乐部模块。
2. **点击更多**：用户选择“更多”功能。
3. **点击发布比赛**：用户点击“发布比赛”按钮进入比赛发布流程。
4. **检查是否已认证**：

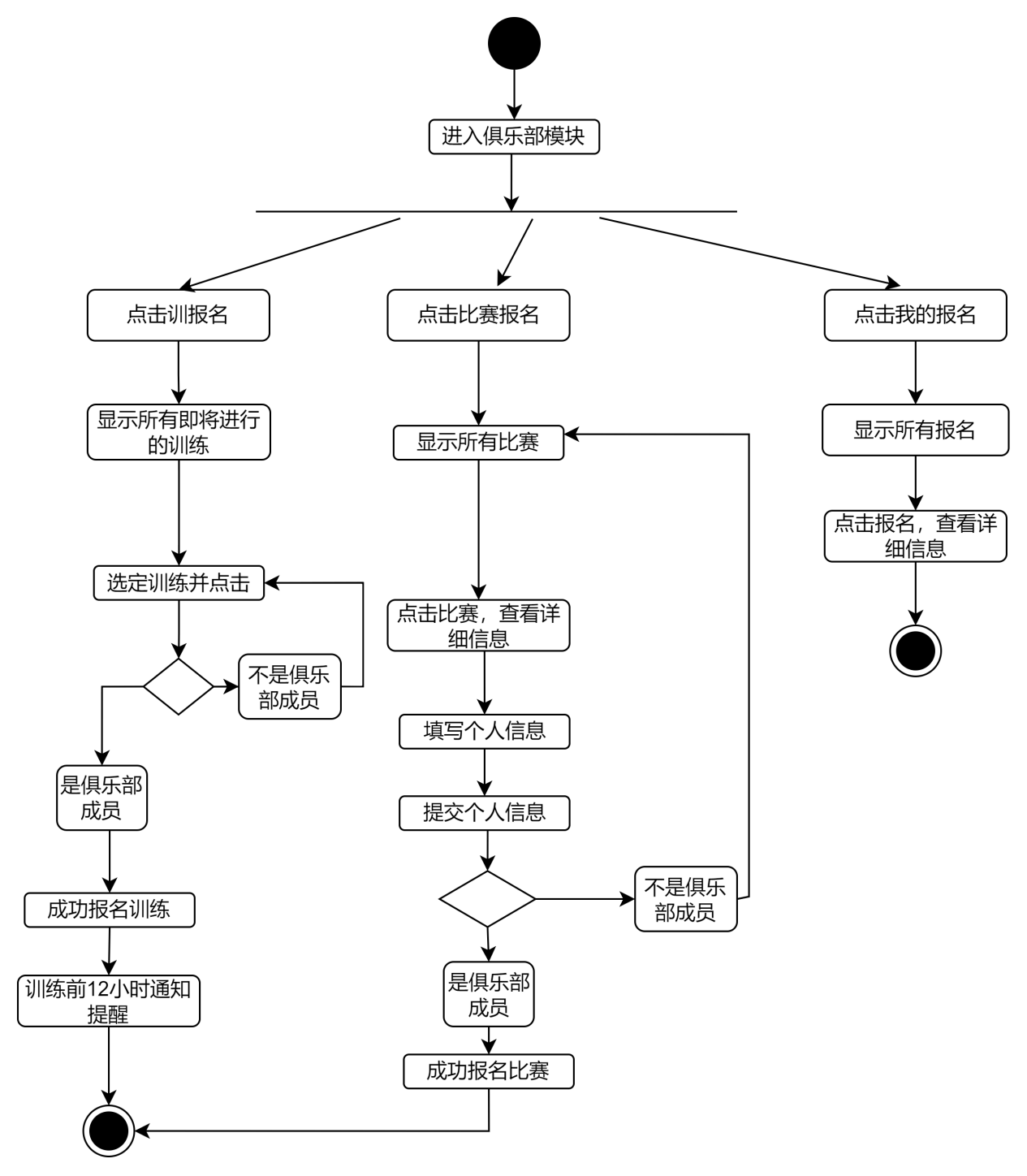
* 若用户未认证，终止流程，提示用户完成认证。
* 若已认证，继续下一步。

1. **填写比赛信息并点击发布**：用户填写比赛信息后提交发布申请。
2. **检查审核状态**：

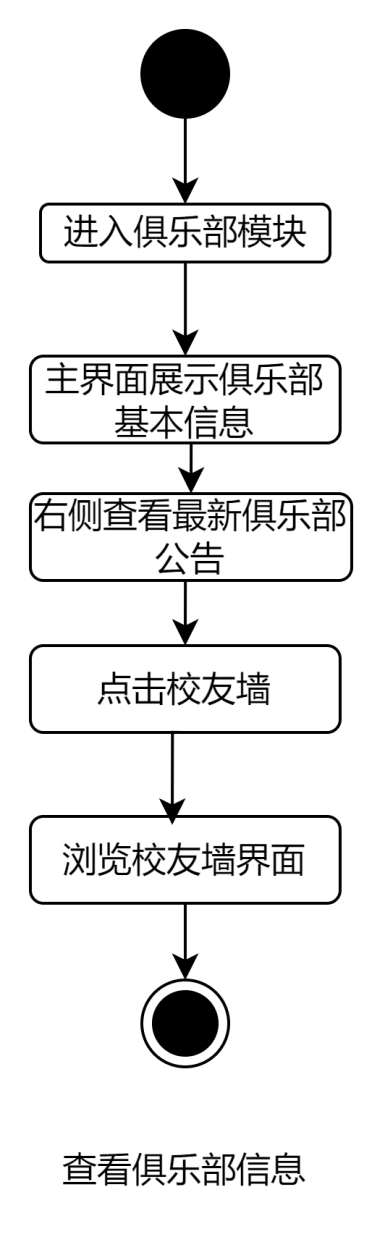
* 若审核未通过，提示用户修改后重新提交。
* 若审核通过，比赛发布成功。

iteration2：

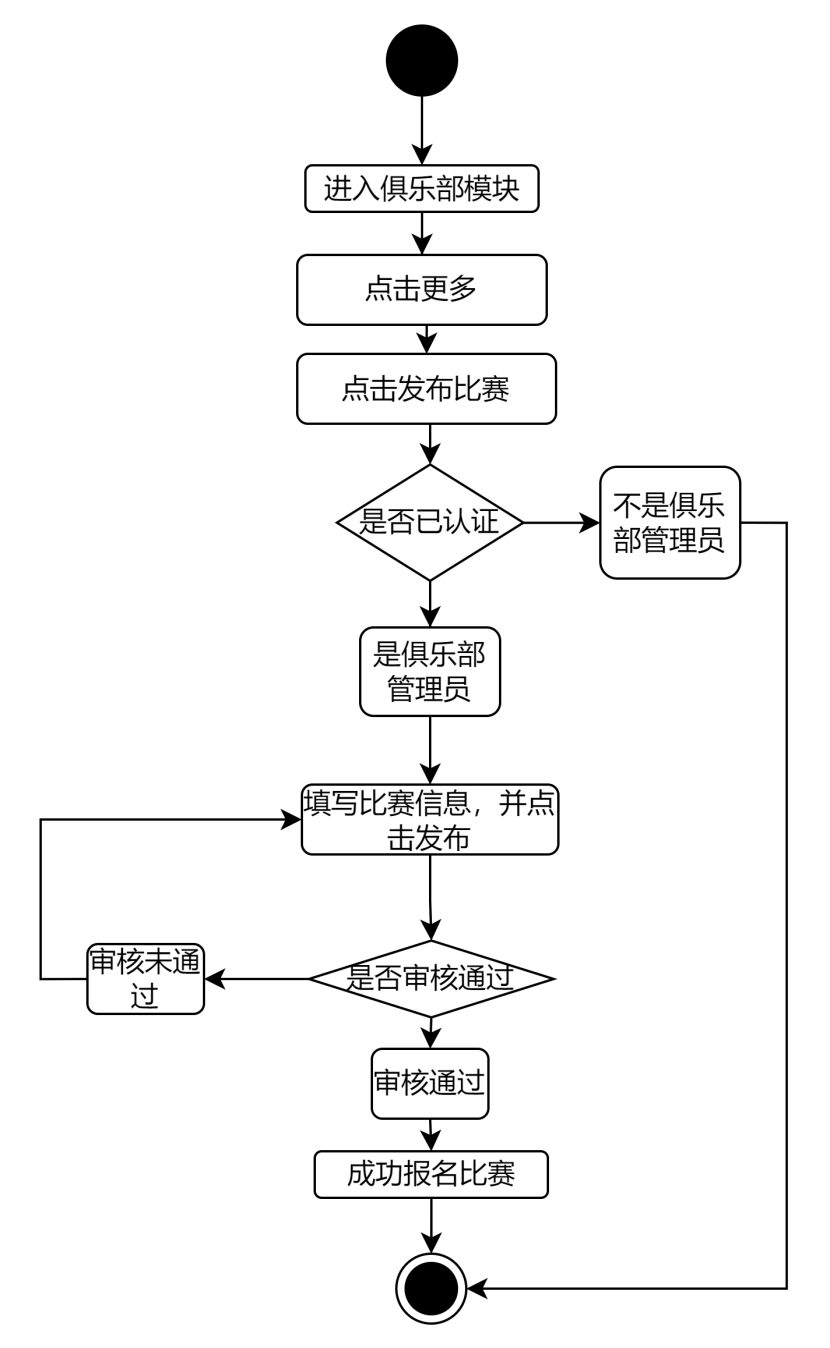
**2.4.1.3.1 训练、比赛报名**



**2.4.1.3.2 查看俱乐部信息**

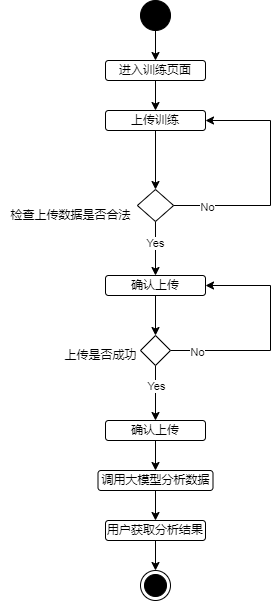


**2.4.1.3.3 发布比赛信息**



**2.4.1.4 训练分析模块**

**2.4.1.4.1 训练数据分析**



1. **登录系统并进入训练页面**  
    用户首先登录系统，然后进入训练页面。
2. **选择上传训练数据**  
    用户可以选择上传需要用于训练的数据。
3. **点击“上传训练”按钮**  
    用户点击“上传训练”按钮，开始上传操作。
4. **系统检查上传行为**  
    系统检查用户是否选择了需要上传的训练数据。

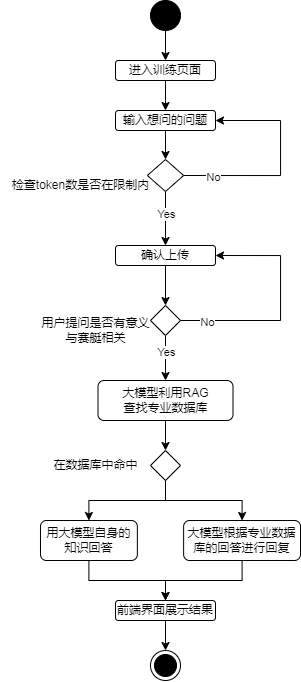
* **如果未选择数据**：系统提示用户选择数据并进行上传。
* **如果已选择数据**：系统继续进行下一步。

1. **用户确认上传操作**  
    在确认上传数据后，系统提示用户确认上传操作，用户点击确认按钮。
2. **验证上传数据的完整性和正确性**  
    系统检查上传数据是否符合要求，验证其完整性和正确性。

* **如果数据不符合要求**：系统拒绝上传操作，并提示用户检查数据格式，重新选择或填写数据。
* **如果数据有效**：系统继续进行下一步。

1. **正式接收数据并调用大模型进行分析**  
    系统接收有效的训练数据并调用大模型进行分析。
2. **生成分析结果与训练建议**  
    大模型对训练数据进行处理后，生成分析结果和个性化的训练建议。
3. **展示分析结果给用户**  
    系统将分析结果和训练建议展示给用户，供用户参考。

**2.4.1.4.2 智能指导**



1. **登录系统并进入训练页面**  
    用户首先登录系统并进入训练页面。
2. **进入智能指导区域**  
    用户在训练页面找到智能指导区域，准备与大模型互动。
3. **提交问题**  
    用户在输入框中输入与训练相关的问题并提交。
4. **系统检查问题内容**  
    系统自动检查用户输入的问题，确保其与训练相关。

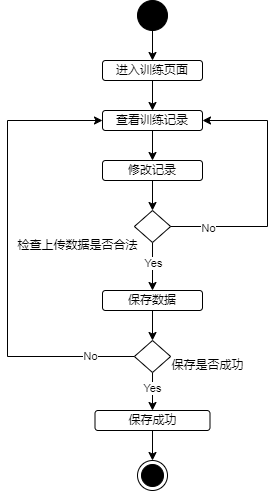
* **如果问题无关**：系统提示用户输入与训练相关的问题，并提供修改建议。
* **如果问题有效**：系统继续进行下一步。

1. **深度分析与解答**  
    系统利用 RAG（检索增强生成）技术，通过查询专业数据知识库对用户的问题进行深度分析和解答。
2. **选择最适合的响应方式**  
    系统根据问题内容，选择最适合的响应方式：

* **直接解答**：利用大模型自身的知识直接回答用户问题。
* **结合专业知识库**：结合检索到的专业知识库内容，提供更准确的建议和指导。

1. **展示解答结果**  
    系统在前端界面上展示解答结果，提供个性化的训练指导和优化建议。

**2.4.1.4.3 查看与编辑历史训练记录**



1. **进入训练记录管理页面**  
    用户登录系统后进入训练记录管理页面，系统展示用户以往上传的所有历史训练记录。
2. **浏览历史训练记录**  
    用户可以浏览所有历史记录，查看每条记录的训练时长、心率等关键信息。
3. **选择查看具体训练记录**  
    用户可以选择查看某条具体的训练记录，详细了解该记录的相关信息。
4. **点击“编辑”按钮进入编辑模式**  
    如果用户希望修改某条记录中的信息，可以点击该记录的“编辑”按钮，进入编辑模式。
5. **编辑训练记录**  
    在编辑模式下，用户可以修改训练记录的相关数据，如训练时长、心率等。
6. **检查修改数据的有效性**

* **如果数据有效且符合格式要求**：用户点击“保存”按钮更新记录。
* **如果数据无效**：系统会提示用户“请输入有效的心率数据”或“数据格式错误，无法保存”，要求重新输入有效数据。

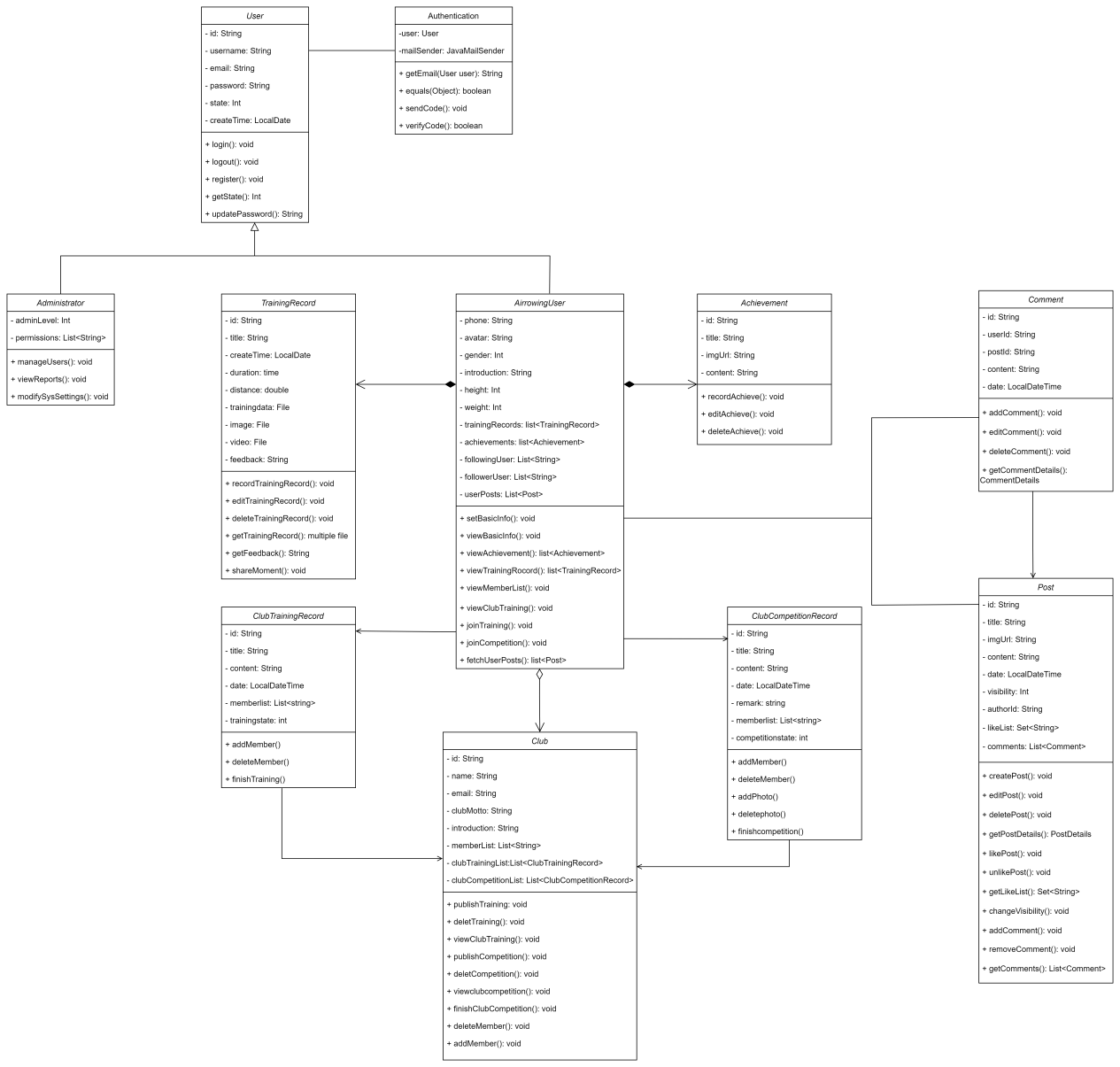
1. **保存修改后的记录**  
    系统检查修改后的数据，确保其完整性和准确性。

* **如果数据有效且已保存**：系统提示“更新成功”，并将用户重定向回训练记录列表页面。
* **如果数据格式无法识别或存在缺失**：系统提示“数据格式错误，无法保存”，要求用户重新提交数据。

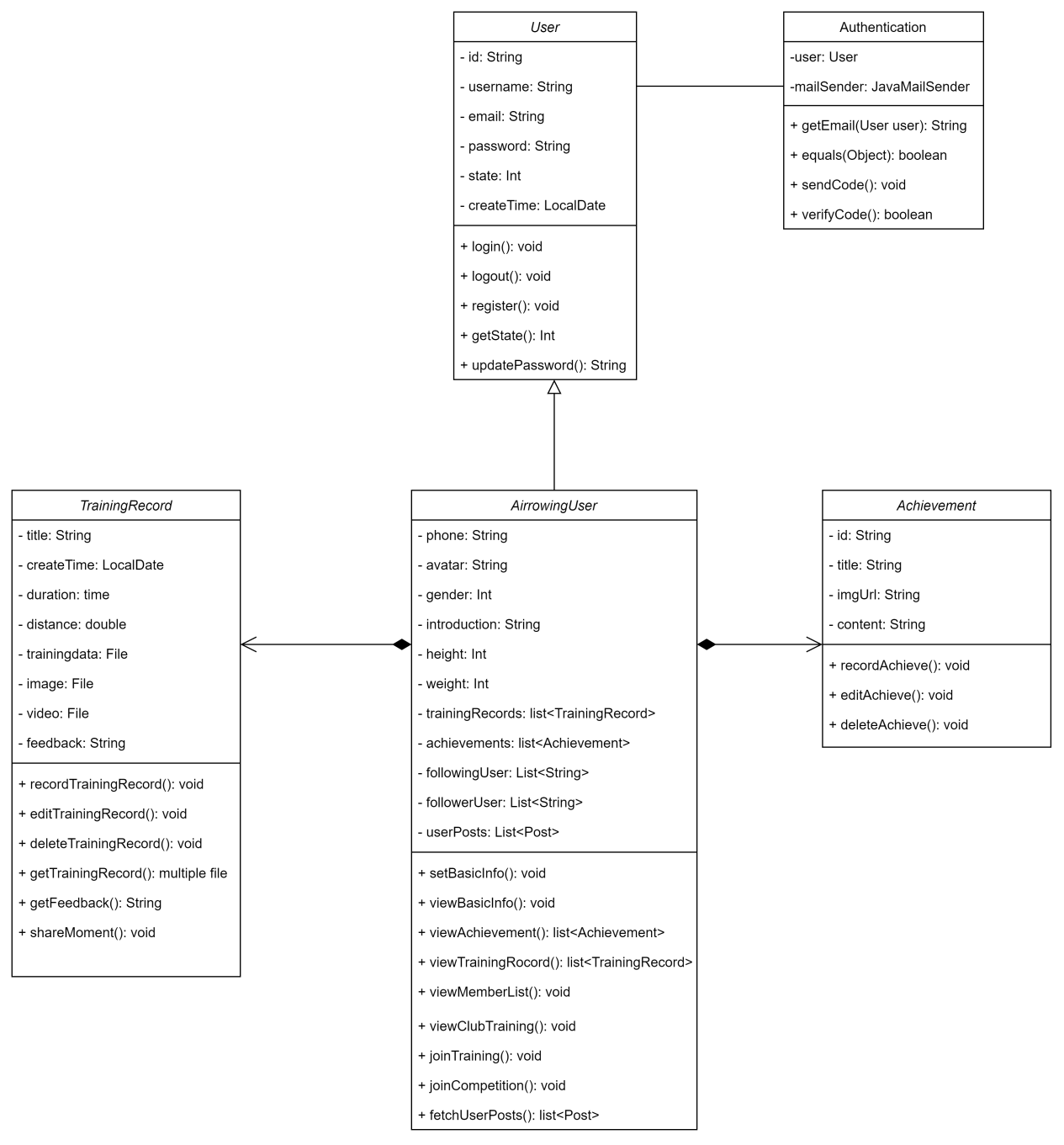
1. **未作修改时点击“保存”**  
    如果用户在编辑过程中未作任何修改，点击“保存”时，系统会提示“没有更改任何内容”，要求用户进行修改后再保存。
2. **返回历史记录列表页面**  
    用户可以继续编辑其他历史记录，或返回训练记录列表页面，查看所有记录。

**2.4.2 数据建模**

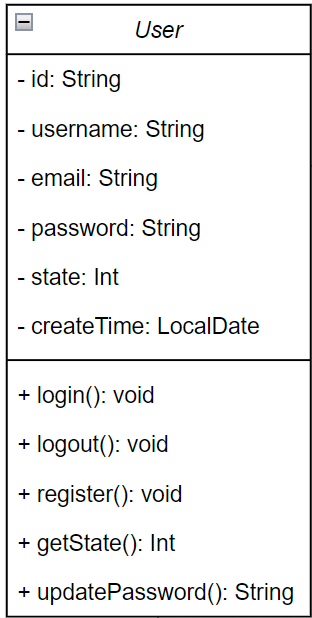
**总类图**



**2.4.2.1 账号管理模块**



**2.4.2.1.1 User 类**



**类名**：User

**简介**：

User 类表示系统中的基础用户，管理用户的基本信息和账户操作。用户可通过该类进行登录、注册、修改密码等操作。User 类是整个系统的基础类，提供了用户认证和管理的核心功能。

**属性**：

* **id** *(String)*：用户的唯一标识符。
* **username** *(String)*：用户的名称，最长50字符。
* **email** *(String)*：用户的电子邮件地址，用于登录和接收通知。
* **password** *(String)*：用户账户的密码。
* **state** *(int)*：用户账户状态（例如 0 表示未激活，1 表示已激活）。
* **createTime** *(LocalDate)*：账户的创建时间。

**方法**：

* **login()**：用户登录系统，验证邮箱和密码是否匹配。
* **logout()**：用户登出系统，清除登录状态。
* **register()**：注册新用户，保存用户基本信息。
* **getState()**：获取当前用户的账户状态。
* **updatePassword(newPassword)**：更新用户密码。

**关系**：

* 与 Authentication 类有关联关系，用于认证用户身份。
* User 是 AirrowingUser 的父类，提供基础功能供子类扩展。

**业务规则：**

1. 用户名必须唯一，不能重复。
2. 邮箱地址格式必须正确，否则会报错。
3. 密码必须包括字母和数字，最少 6 位长。
4. 用户必须先注册才能登录。
5. 只有激活状态的用户才能进行登录和修改操

**用例**：

1. **用户登录**：

* 用户输入邮箱和密码并提交登录请求。
* 系统调用 login() 方法验证用户输入的凭据。
* 验证成功后，系统记录用户的登录状态并跳转到主页面。

1. **用户注册**：

* 用户通过界面提交注册信息（用户名、邮箱、密码）。
* 系统调用 register() 方法保存用户数据。
* 系统返回注册成功的消息。

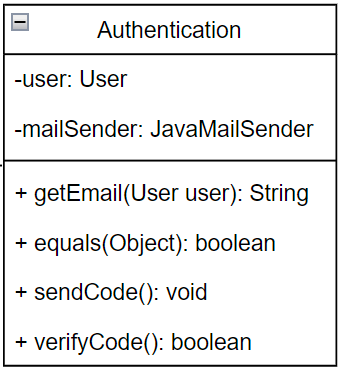
1. **修改密码**：

* 用户通过界面提交旧密码和新密码。
* 系统调用 updatePassword() 方法更新密码。

1. **用户登出**：

* 用户点击登出按钮。
* 系统调用 logout() 方法清除用户登录状态并跳转到登录页面。

**2.4.2.1.2 Authentication 类**



**类名**：Authentication

**简介**：

Authentication 类负责用户身份验证和账户安全，提供了验证码验证、密码匹配验证等功能，是用户登录和注册模块的核心部分。

**属性**：

* **user** *(User)*：与用户关联，用于验证身份。
* **mailSender** *(JavaMailSender)*：邮件发送组件，用于发送验证码。

**方法**：

* **getEmail(User user)**：获取用户的邮箱地址。
* **equals(Object obj)**：比较对象，判断身份是否一致。
* **sendCode()**：发送验证码到用户邮箱。
* **verifyCode(code)**：验证用户输入的验证码是否正确。

**关系**：

* 与 User 类有直接关联，依赖其提供的用户信息。
* 在注册和修改密码场景中充当身份验证模块。

**业务规则：**

1. 用户只能注册一次，已注册邮箱不能重复。
2. 验证码有效期为 3 分钟，过期验证码将不再有效。
3. 密码格式必须满足规则：含大写字母、小写字母和数字，最少 8 位长。
4. 验证成功后，才能进入下一步操作。

**用例**：

1. **用户登录**：

* 调用 equals() 方法验证用户凭据。
* 验证成功后，允许用户登录系统。

1. **注册账号**：

* 调用 sendCode() 方法向用户邮箱发送验证码。
* 用户输入验证码后，系统调用 verifyCode() 验证正确性。

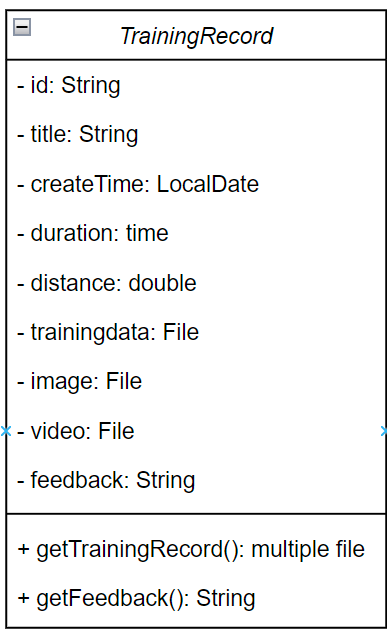
1. **修改密码**：

* 系统调用 verifyCode() 验证当前用户输入的旧密码。

1. **验证邮箱格式**：

* 系统调用 getEmail() 方法获取用户邮箱地址，并进行格式校验。

**2.4.2.1.3 TrainingRecord 类**



**类名**：TrainingRecord

**简介**：

TrainingRecord 类表示用户的训练记录，存储有关训练的详细数据，并支持数据分析、修改和分享。它是训练管理模块的核心类。

**属性**：

* **id** *(String)*：训练记录的唯一标识符。
* **title** *(String)*：训练记录的标题或描述。
* **createTime** *(LocalDate)*：训练记录的创建时间。
* **duration** *(time)*：训练的持续时间（分钟）。
* **distance** *(double)*：训练的总距离（公里）。
* **trainingData** *(File)*：包含详细训练数据的文件。
* **image** *(File)*：训练过程中拍摄的图片。
* **video** *(File)*：训练过程中录制的视频。
* **feedback** *(String)*：系统分析返回的训练反馈。

**方法**：

* **getTrainingRecord()**：获取训练记录详细信息。
* **getFeedback()**：调用分析模型获取训练反馈。
* **recordTrainingRecord(trainingData, image, video)**：记录新的训练数据。
* **editTrainingRecord(newTrainingData)**：修改训练记录。
* **deleteTrainingRecord()**：删除训练记录。

**关系**：

* 与 AirrowingUser 类有“从属”关系，一个用户可以拥有多个训练记录。

**业务规则：**

1. 每条训练记录必须包含至少一个有效的训练数据（如时长、距离或心率）。
2. 用户只能查看属于自己的训练记录。
3. 系统分析完成后才能生成反馈，并将反馈附加到训练记录中。
4. 用户可选择是否分享训练记录至社区，分享后记录可被其他用户查看和互动。

**用例**：

1. **上传训练记录**：

* 用户上传训练数据（如时长、图片、视频）。
* 系统保存数据并调用分析模型返回训练建议。

1. **编辑训练记录**：

* 用户修改已有记录，更新如时长或图像等信息。
* 系统重新分析并更新记录。

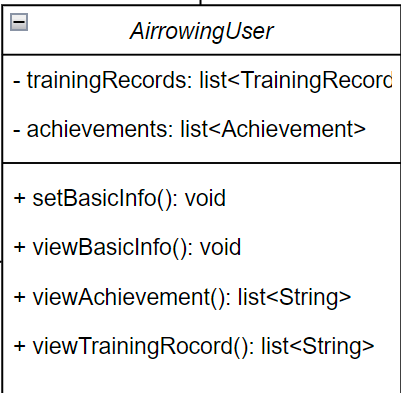
1. **删除训练记录**：

* 用户选择删除训练记录。
* 系统从数据库中移除记录。

1. **分享训练记录**：

* 用户选择分享训练记录。
* 系统将记录上传至社区，供其他用户查看。

**2.4.2.1.4 AirrowingUser 类**



**类名**：AirrowingUser

**简介**：

AirrowingUser 类继承自 User 类，表示系统中的扩展用户，包含与用户个性化相关的属性（如身高、体重等），并管理用户的训练记录和成就。

**属性**：

* **trainingRecords** *(list<TrainingRecord>)*：用户的训练记录列表。
* **achievements** *(list<Achievement>)*：用户的成就列表。

**方法**：

* **setBasicInfo()**：设置用户的基本信息（如身高、体重等）。
* **viewBasicInfo()**：查看用户的基本信息。
* **viewAchievement()**：获取用户的成就列表。
* **viewTrainingRecord()**：获取用户的训练记录。

**关系**：

* 继承自 User 类。
* 与 TrainingRecord 和 Achievement 类有直接关联。

**业务规则：**

1. 训练记录必须完整关联到用户，未关联记录无法保存。
2. 用户可以随时更新个人信息，但需验证其合法性（如身高、体重范围）。

**用例**：

1. **查看训练记录**：

* 用户调用 viewTrainingRecord() 方法查看所有训练记录。

1. **查看成就**：

* 用户调用 viewAchievement() 方法查看个人成就。

1. **更新个人信息**：

* 用户调用 setBasicInfo() 方法修改个人基本信息。

1. **查看基本信息**：

* 用户调用 viewBasicInfo() 方法获取当前个人信息。

**2.4.2.1.5 Administrator 类**

**类名**：Administrator

**简介**：

Administrator 类继承自 User 类，表示系统中俱乐部模块的管理员，主要负责俱乐部的管理工作。包括对俱乐部成员的管理、活动的创建与审批，以及俱乐部信息的维护。

**属性**：

* **adminLevel** *(Int)*：管理员的权限级别。
* **permissions** *(List<String>)*：管理员在俱乐部模块中拥有的具体权限。

**方法**：

* **manageUsers()**：管理俱乐部成员。
* **viewReports()**：查看俱乐部的信息报告。
* **modifySysSettings()**: 修改俱乐部的系统设置。

**关系**：

* Administrator 类继承自 User 类，复用了用户的基本信息和方法（如登录、登出等）。

**业务规则：**

1. 不同 adminLevel 和 permissions 决定不同管理员的操作权限。
2. 所有管理员的操作需记录日志，包括审批活动、修改俱乐部信息等，确保操作的透明性和可追溯性
3. 所有俱乐部活动必须经过管理员审批后才能发布。

**用例**：

1. **管理俱乐部成员**：

* 管理员调用 manageUsers() 方法，查看俱乐部成员列表。
* 管理员可以增加、移除成员，或修改成员的角色。

1. **审批俱乐部活动**：

* 管理员调用 viewReports() 方法，获取活动的统计数据或参与记录。
* 审批通过的活动会发布到俱乐部的活动页面。

1. **修改俱乐部信息**：

* 管理员调用 modifySysSettings() 方法，更新俱乐部的基本信息，如俱乐部名称或简介。

1. **生成俱乐部报告**：

* 管理员调用 viewReports() 方法，查看俱乐部成员活跃度和活动统计数据，支持俱乐部的日常运营分析。

**2.4.2.1.6 Achievement 类**

**类名**：Achievement

**简介**：

Achievement 类表示用户在系统中获得的成就，例如特定训练目标完成后的奖励。

**属性**：

* **id** *(String)*：成就的唯一标识符。
* **title** *(String)*：成就标题。
* **imgUrl** *(String)*：成就图片的 URL。
* **content** *(String)*：成就描述内容。

**方法**：

* **getAchieve()**：获取成就详情。

**关系**：

* 与 AirrowingUser 类有“从属”关系，一个用户可以拥有多个成就。

**业务规则：**

1. 成就标题必须简洁且具有描述性，最长不超过 50 字符。
2. 成就与用户关联后无法更改归属，但可被用户删除。
3. 成就可以根据社区活动规则自动生成。

**用例**：

1. **查看成就详情**：

* 用户调用 getAchieve() 方法查看指定成就的详细信息。

1. **新增成就**：

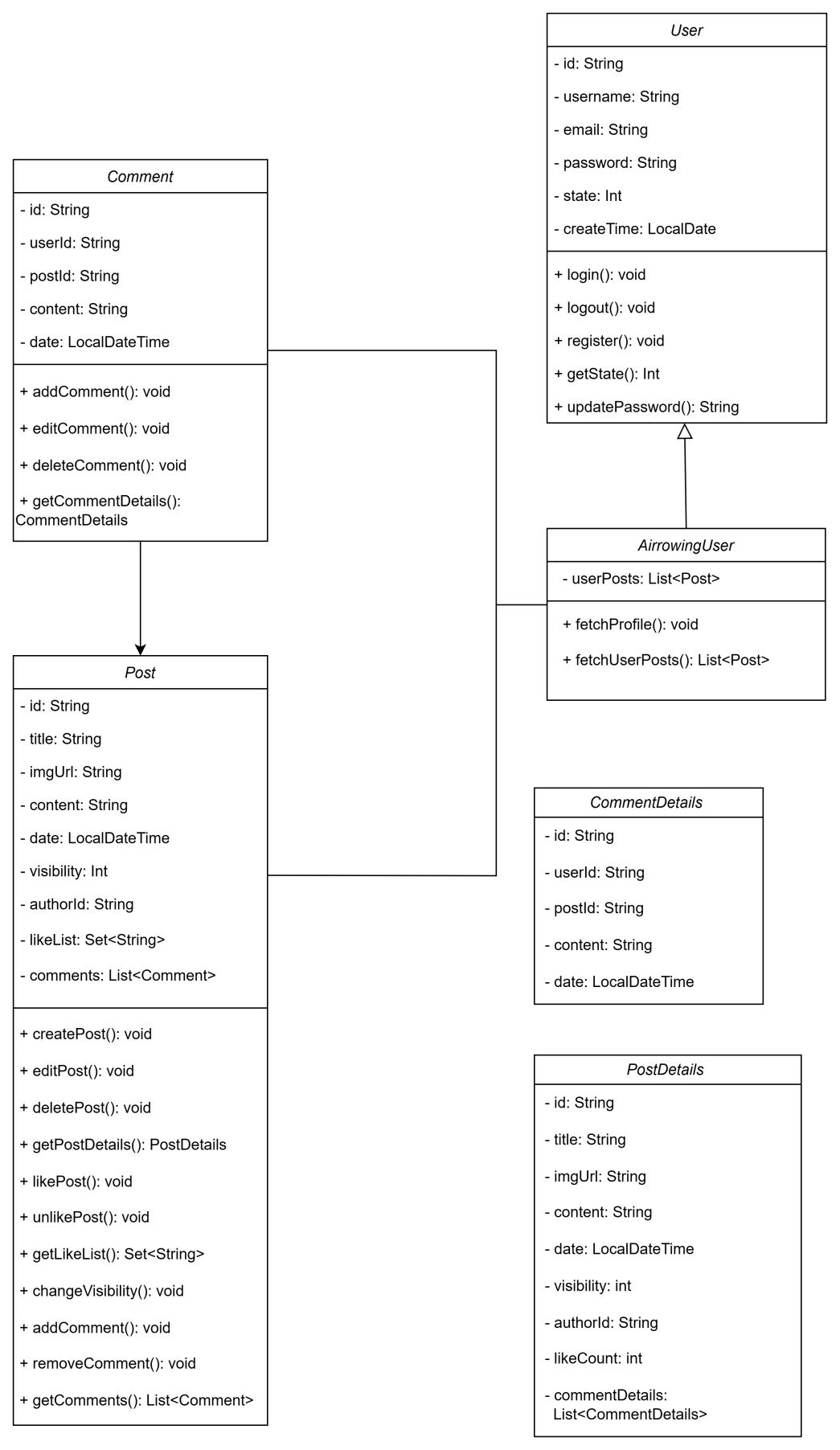
* 系统在用户完成目标时自动生成成就。

1. **删除成就**：

* 用户选择删除某个成就，系统从数据库中移除。

**2.4.2.2 社区模块**

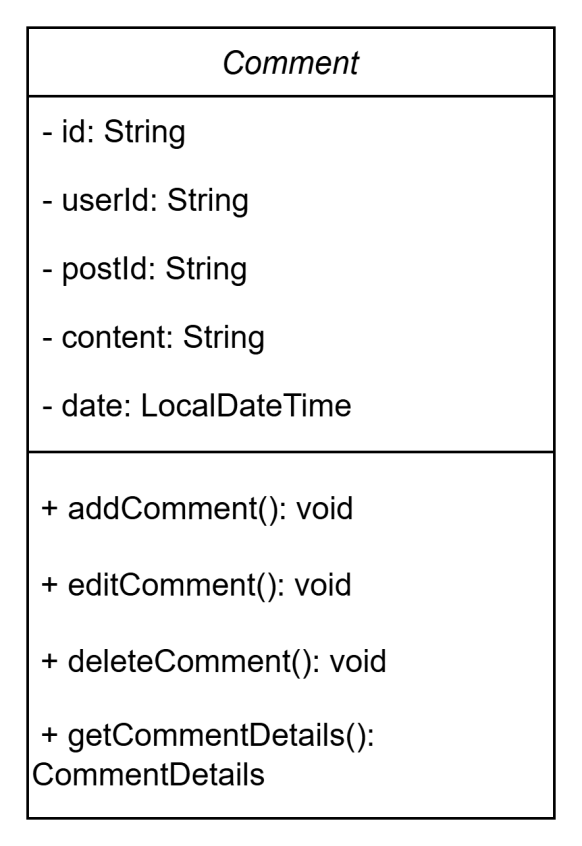
iteration1：（旧版）



iteration2:



**2.4.2.2.1 Comment类**



**类名：Comment**

**简介：**

Comment 类表示用户对帖子或其他内容的评论。每个评论包含评论的内容、发表时间、评论者等信息。该类用于社区互动模块，允许用户对帖子进行评论，并与其他用户进行互动。

**属性：**

* id (String): 评论的唯一标识符。
* userId (String): 发布该评论的用户的唯一标识符。
* postId (String): 评论所关联的帖子（或其他内容）的标识符。
* content (String): 评论的内容，文本类型，包含用户对帖子的意见或反馈。
* date (LocalDateTime): 评论发表的时间，使用 LocalDateTime 类型，精确到秒。

**方法：**

* addComment(String content, String postId, String userId): 添加新的评论，用户提供评论内容、帖子ID和用户ID。
* editComment(String newContent, String userId): 编辑已有的评论，用户可以修改评论内容，仅限于评论的发布者。
* deleteComment(String userId): 删除当前的评论，仅限于评论的发布者。
* getCommentDetails(): 获取当前评论的详细信息，包括评论内容、发表时间、评论者等。
* getPostId(): 获取评论关联的帖子ID，用于查找评论所在的帖子。

**关系：**

* **与 User 类的关联关系**：每个 Comment 都由一个用户发布，一个用户可以发布多个评论。即每个评论都与一个特定的用户（通过 userId）相关联。
* **与 Post 类的关联关系**：每个 Comment 都关联到一个具体的帖子或内容，一个帖子可以有多个评论。即每个评论都与一个特定的帖子（通过 postId）相关联。

**业务规则：**

* 每个评论必须包含有效的评论内容（不能为空）。
* 用户只能编辑或删除自己的评论，不能修改或删除其他用户的评论。
* 每个评论都会有一个唯一的 ID，用于标识和查询评论。
* 评论在创建时会自动记录当前的时间。
* 如果评论内容为空，系统会拒绝提交并提示用户填写评论。
* 用户可以删除评论，删除后该评论将不再可见。

**用例：**

1. **用户添加评论：**

* 用户选择一个帖子，并提供评论内容。
* 系统验证评论内容的有效性。
* 系统将评论内容保存到数据库，关联到该帖子的 postId 和用户的 userId。
* 系统记录评论的发表时间。
* 评论成功添加后，用户可以查看自己的评论。

1. **用户编辑评论：**

* 用户选择修改自己的评论。
* 用户编辑评论内容。
* 系统验证用户身份，确保用户是评论的发布者。
* 系统更新评论内容，并记录修改后的时间。

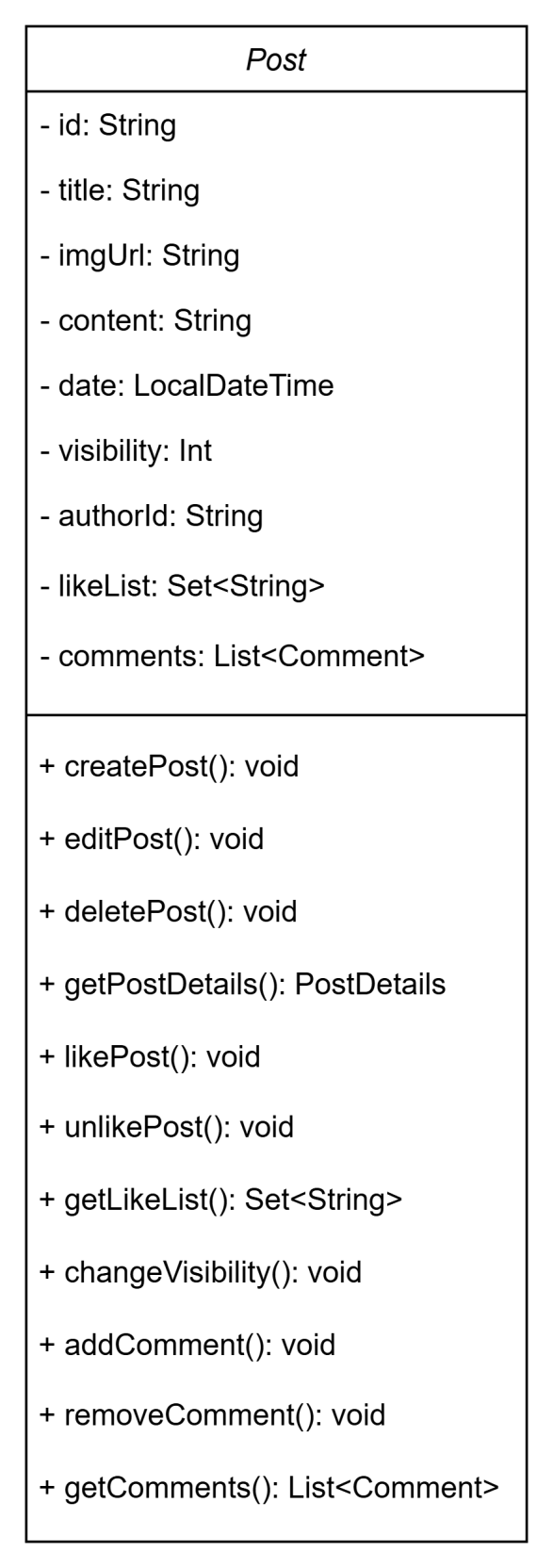
1. **用户删除评论：**

* 用户选择删除自己的评论。
* 系统验证用户身份，确保用户是评论的发布者。
* 系统删除评论，并确认删除操作。

1. **用户查看评论：**

* 用户浏览一个帖子时，系统显示该帖子的所有评论。
* 用户可以查看每条评论的内容、发表时间及评论者信息。

**2.4.2.2.2 Post类**



**类名：Post**

**简介：** Post 类表示一个用户发布的帖子，通常是用户在社区中分享的信息或内容。每个帖子包含标题、内容、发布的时间、可见性设置、点赞列表及评论列表等信息。该类用于社区互动模块，是用户与社区内容互动的核心部分。

**属性：**

* **id** (string): 帖子的唯一标识符，用于区分不同的帖子。
* **title** (string): 帖子的标题或描述，通常用于简要描述帖子内容。
* **imgUrl** (string): 帖子中包含的图片文件路径或URL，表示帖子相关的图像内容。
* **content** (string): 帖子的正文内容，包含帖子的详细描述或文本内容。
* **date** (LocalDateTime): 帖子创建的时间，使用 LocalDateTime 类型，精确到秒，记录发帖时间。
* **visibility** (int): 帖子的可见性设置，表示帖子是公开的还是私密的。通常使用数字表示不同的可见性级别（如 0 表示私密，1 表示公开等）。
* **authorId** (string): 发布该帖子的用户的唯一标识符。
* **likeList** (Set): 存储用户点赞的集合，包含所有点赞该帖子的用户的 userId。
* **comments** (List): 存储该帖子的所有评论。

**方法：**

* **createPost(title, content, imgUrl, visibility, authorId):** 创建一个新的帖子，用户提供帖子标题、内容、图片链接、可见性设置以及作者ID。
* **editPost(newTitle, newContent, newImgUrl):** 编辑已有的帖子，允许用户修改标题、内容或图片。
* **deletePost():** 删除当前的帖子。
* **getPostDetails():** 获取当前帖子的详细信息，包括标题、内容、图片、发布日期、作者信息等。
* **likePost(userId):** 用户点赞该帖子，将用户的 userId 添加到 likeList 中。
* **unlikePost(userId):** 用户取消点赞，将用户的 userId 从 likeList 中移除。
* **getLikeList():** 获取该帖子的所有点赞用户的 userId。
* **changeVisibility(newVisibility):** 修改帖子的可见性设置，控制帖子是公开还是私密。
* **addComment(Comment comment):** 添加一条新的评论到帖子中。
* **removeComment(commentId):** 从帖子中移除一条评论。
* **getComments():** 获取该帖子的所有评论列表。

**关系：**

* **与 User 类的关联关系：** 每个 Post 都是由一个特定的用户发布的，一个用户可以发布多个帖子。因此，每个帖子都与特定的用户关联（通过 authorId）。
* **与 Comment 类的关联关系：** 一个帖子可以包含多个评论。Comment 类和 Post 类之间存在“拥有”关系，意味着一个帖子可以拥有多个评论。

**业务规则：**

* 每个帖子必须有唯一的 ID。
* 帖子的内容、标题和可见性设置必须有效。帖子不能没有内容或标题。
* 用户可以编辑或删除自己的帖子，不能修改或删除其他用户的帖子。
* 用户可以对帖子进行点赞，点赞信息会记录在 likeList 中。
* 帖子的可见性可以设置为公开或私密，系统会根据可见性设置限制帖子的展示范围。
* 一个帖子可以拥有多个评论，评论必须关联到帖子和发布评论的用户。

**用例：**

1. **用户创建帖子：**

* 用户填写帖子标题、内容和图片，并设置帖子可见性。
* 系统创建并保存该帖子，将用户信息与帖子关联，并记录发布日期。
* 用户成功发布帖子后，其他用户可以查看帖子（如果该帖子的可见性是公开的）。

1. **用户编辑帖子：**

* 用户修改自己的帖子，如修改标题、内容或更新图片。
* 系统更新该帖子的内容，并重新记录更新时间。

1. **用户删除帖子：**

* 用户选择删除自己发布的帖子。
* 系统删除该帖子，并确认删除操作。

1. **用户点赞帖子：**

* 用户点击“点赞”按钮，系统将该用户的 userId 添加到 likeList 中。
* 帖子的点赞数将相应更新。

1. **用户取消点赞帖子：**

* 用户点击“取消点赞”按钮，系统将该用户的 userId 从 likeList 中移除。
* 帖子的点赞数将相应更新。

1. **获取帖子点赞列表：**

* 用户查看某帖子的点赞记录，系统显示所有点赞用户的 userId 及其点赞时间。

1. **添加评论到帖子：**

* 用户选择一个帖子，并提供评论内容。
* 系统验证评论内容的有效性。
* 系统将评论内容保存到数据库，关联到该帖子的 postId 和用户的 userId。
* 系统记录评论的发表时间。
* 评论成功添加后，用户可以查看自己的评论。

1. **删除评论：**

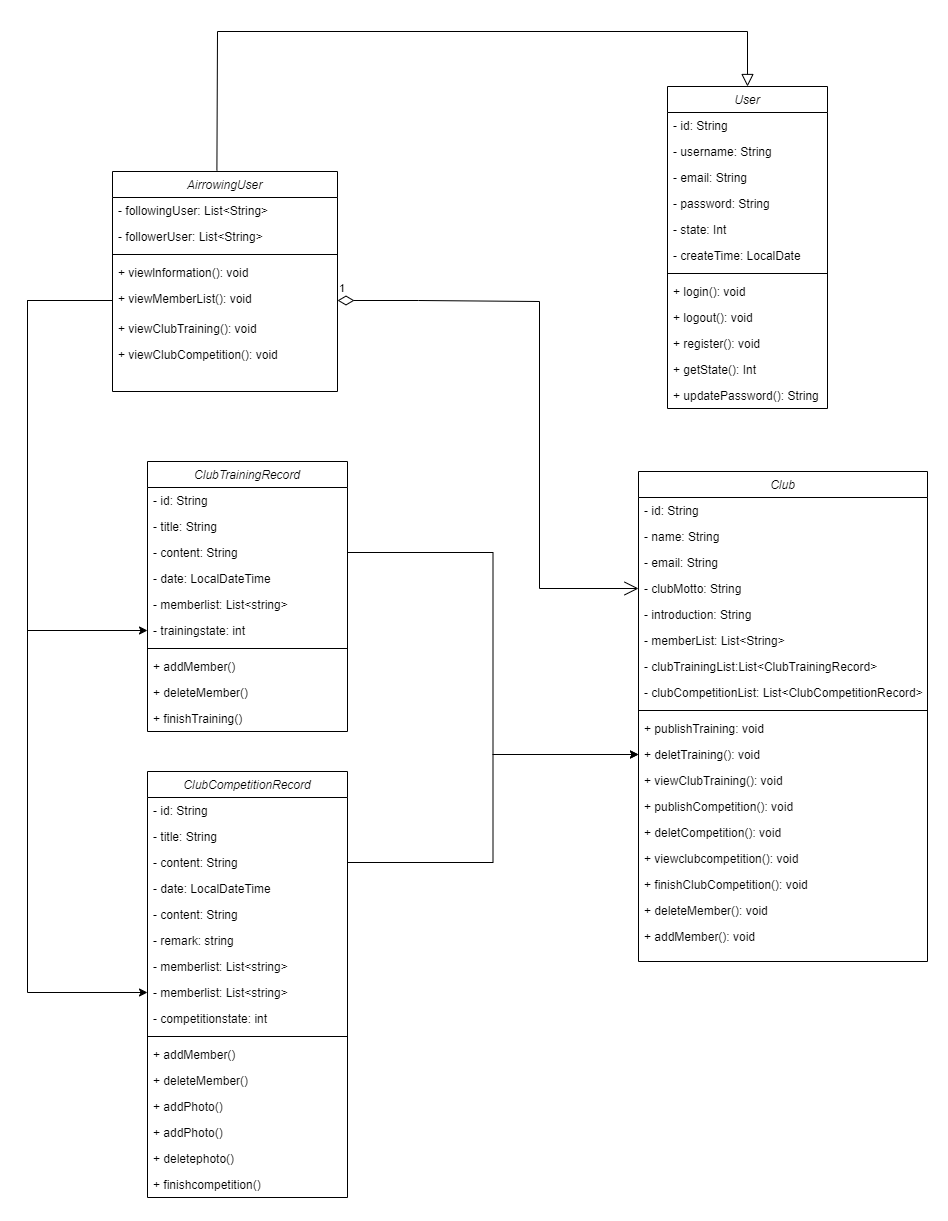
* 用户选择删除自己的评论。
* 系统删除评论，并确认删除操作。

1. **修改帖子可见性：**

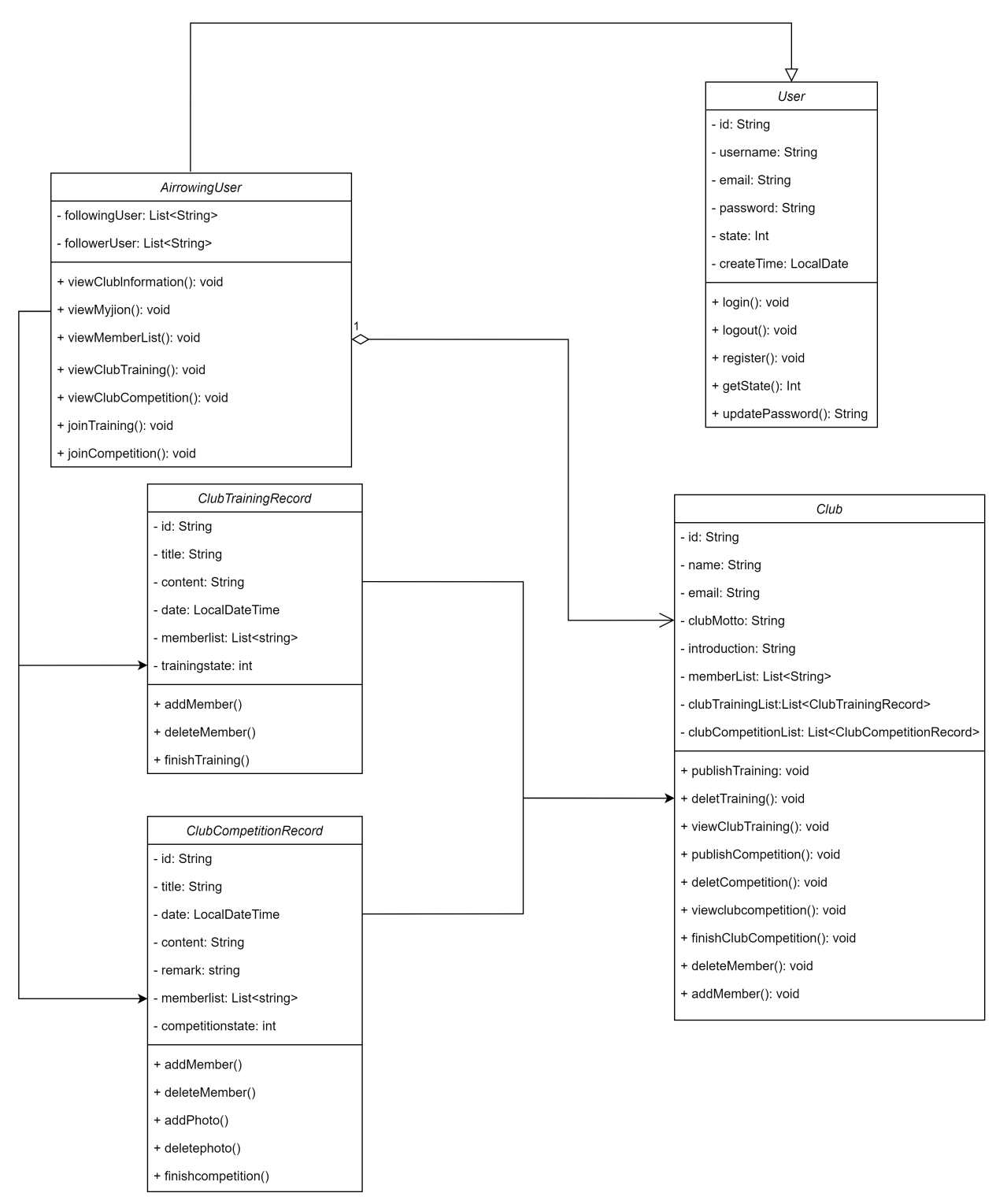
* 用户可以修改帖子可见性，将帖子设置为公开或私密。
* 系统更新帖子可见性，并按照新的可见性设置显示帖子。

**2.4.2.3 俱乐部模块**

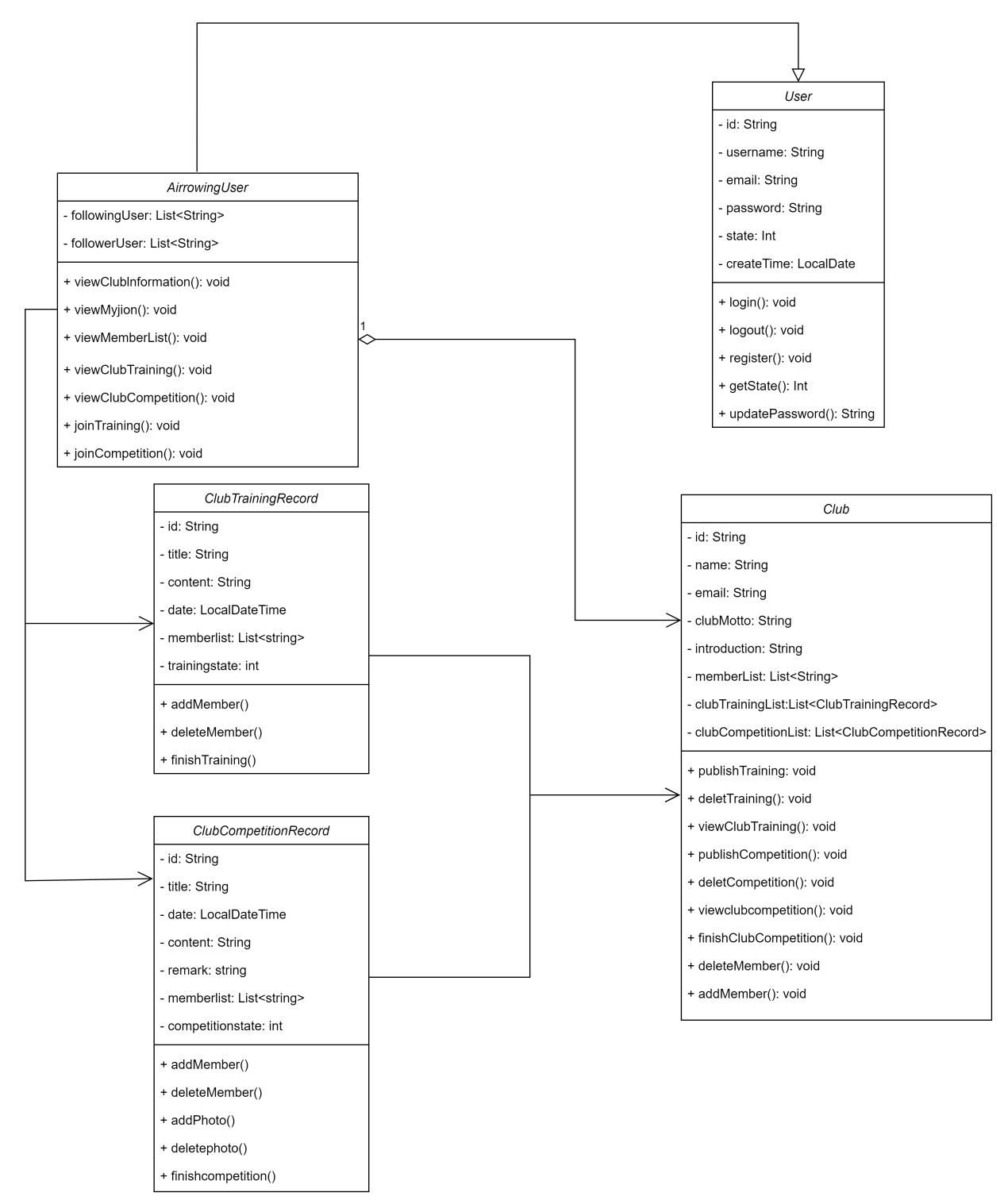
iteration1：（旧版）



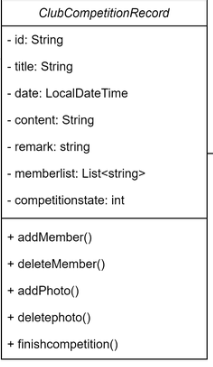
iteration2：（旧版）



iteration3：



**2.4.2.3.1 ClubCompetitionRecord类**



**类名：ClubCompetitionRecord**

**简介**：  
ClubCompetitionRecord 类表示俱乐部的比赛记录，包含比赛的基本信息、参赛成员、比赛状态、照片以及相关备注等内容。这个类用于记录俱乐部中的每个比赛的详细信息，并管理参赛成员和比赛过程中的照片。该类对于比赛的组织、成员管理和记录保存至关重要。

**属性**：

* **id (String)**: 比赛记录的唯一标识符，用于区分不同的比赛记录。
* **title (String)**: 比赛的标题或名称，简短描述比赛的主题或类型。
* **content (String)**: 比赛的详细内容或描述，通常包含比赛的目标、规则、目的等信息。
* **date (LocalDateTime)**: 比赛的开始时间或创建时间，使用 LocalDateTime 类型，记录比赛的日期和时间。
* **remark (String)**: 比赛的备注或附加说明，记录与比赛相关的其他信息或特别说明。
* **memberList (List<String>)**: 参赛成员的列表，存储参赛者的 ID 或名称，便于查看哪些成员参与了该比赛。
* **competitionState (int)**: 比赛状态，表示比赛的进展阶段。通常使用数字表示不同的状态（例如，0 表示未开始，1 表示进行中，2 表示已完成）。

**方法**：

* **addMember(memberId)**: 添加一个成员到比赛的成员列表中，memberId 是成员的唯一标识符。此方法可以让俱乐部组织者或管理员为比赛添加新的参赛成员。
* **deleteMember(memberId)**: 从比赛的成员列表中删除一个成员，memberId 是成员的唯一标识符。
* **addPhoto(photoUrl)**: 为比赛添加一张照片，photoUrl 是照片的路径或 URL。此方法用于将比赛的照片记录到系统中，便于赛后查看。
* **deletePhoto(photoUrl)**: 从比赛记录中删除一张照片，photoUrl 是照片的路径或 URL。此方法可以让管理员或组织者删除不必要的照片。
* **finishCompetition()**: 标记比赛为已完成，并更新比赛状态。此方法用于结束比赛并记录比赛的最终状态。

**关系**：

* 与 **AirrowingUser 类** 关联：memberList 存储的是成员信息，每个成员都是 AirrowingUser 类的一个实例。关联：比赛可以上传多张照片，每张照片对应一个 Photo 类的实例。ClubCompetitionRecord 类与 Photo 类之间是“拥有”关系。

**业务规则**：

* 比赛记录必须有唯一的 id。
* 比赛的 competitionState 必须有效（例如，0 表示未开始，1 表示进行中，2 表示已完成）。状态的变化必须符合比赛的实际进展。
* 参赛成员列表可以在比赛的整个过程中进行添加或删除，允许动态管理比赛成员。
* 比赛结束后，调用 finishCompetition() 方法更新比赛的状态为“已完成”。
* 每次比赛只能由相应的成员和照片进行关联。

**用例**：

1. **添加参赛成员**：

* 俱乐部管理员或组织者可以为比赛添加参赛成员，成员通过 addMember() 方法添加到 memberList。
* 系统成功更新参赛成员列表。

1. **删除参赛成员**：

* 俱乐部管理员或组织者可以从比赛中删除一个成员，使用 deleteMember() 方法。
* 系统更新成员列表，删除指定的成员。

1. **添加比赛照片**：

* 比赛过程中或比赛后，管理员可以添加照片，使用 addPhoto() 方法将比赛照片添加到比赛记录中。
* 系统保存照片并将其与比赛关联。

1. **删除比赛照片**：

* 比赛结束后，如果有不需要的照片，管理员可以使用 deletePhoto() 删除。
* 系统从比赛记录中删除指定的照片。

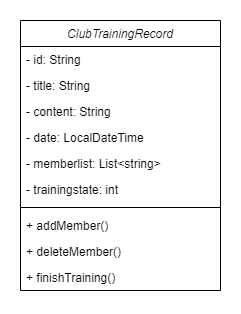
1. **结束比赛**：

* 当比赛结束时，俱乐部管理员调用 finishCompetition() 方法更新比赛的状态为“已完成”。
* 系统记录比赛结束时间，并更新比赛状态。

1. **查看比赛信息**：

* 用户（管理员、成员或其他参与者）可以查看比赛的详细信息，包括标题、内容、成员列表、照片和比赛状态。

**2.4.2.3.2 ClubTrainingRecord类**



**类名：ClubTrainingRecord**

**简介**：  
ClubTrainingRecord 类表示俱乐部的训练记录，记录训练的基本信息、参与的成员、训练状态以及相关的管理操作。这个类主要用于跟踪俱乐部中的各项训练活动，确保训练过程有序进行，并允许管理成员的加入与删除，同时能够标记训练的完成状态。

**属性**：

* **id (String)**: 训练记录的唯一标识符，用于区分不同的训练活动。
* **title (String)**: 训练的标题或名称，简短描述训练的内容或主题。
* **content (String)**: 训练的详细内容或描述，通常包括训练的目的、规则、计划等。
* **date (LocalDateTime)**: 训练的开始时间或创建时间，使用 LocalDateTime 类型来记录训练的日期和时间。
* **memberList (List<String>)**: 参与训练的成员列表，存储成员的 ID 或名称，便于查看哪些成员参与了该训练活动。
* **trainingState (int)**: 训练状态，表示训练的进展阶段。常用数字表示不同状态（例如，0 表示未开始，1 表示进行中，2 表示已完成）。

**方法**：

* **addMember(memberId)**: 将一个新的成员添加到训练活动的成员列表中，memberId 是成员的唯一标识符。这个方法允许管理员或组织者增加参与训练的成员。
* **deleteMember(memberId)**: 从训练记录的成员列表中删除一个成员，memberId 是成员的唯一标识符。这个方法可以让管理员或组织者移除某个成员。
* **finishTraining()**: 标记训练为已完成，更新训练的状态为“已完成”。该方法用于结束训练并记录训练的最终状态。

**关系**：

* 与 **User 类** 可能有关联：memberList 存储的是训练成员的信息，每个成员都是 User 类的实例。ClubTrainingRecord 类与 User 类之间有“成员”关系。
* 与 **Photo 类** 可能有关联：尽管目前没有明确提到，但如果训练过程中涉及照片或视频的上传，可能与 Photo 类存在关系。

**业务规则**：

* 每个训练记录必须有唯一的 id。
* 训练的 trainingState 必须有效（例如，0 表示未开始，1 表示进行中，2 表示已完成），且状态的变化应当符合训练的实际进展。
* 训练的成员列表可以随时进行更新，允许动态管理训练的参与人员。
* 训练完成后，调用 finishTraining() 方法更新训练状态为“已完成”。
* 每个训练记录只能由相应的成员参与，且训练过程需要遵循一定的顺序，不能跳过中间的步骤。

**用例**：

1. **添加训练成员**：

* 俱乐部管理员或组织者可以通过 addMember() 方法为训练活动添加成员，成员通过 memberId 被添加到 memberList。
* 系统成功更新成员列表。

1. **删除训练成员**：

* 俱乐部管理员或组织者可以通过 deleteMember() 方法将某个成员从训练活动中移除。
* 系统成功更新成员列表，删除指定成员。

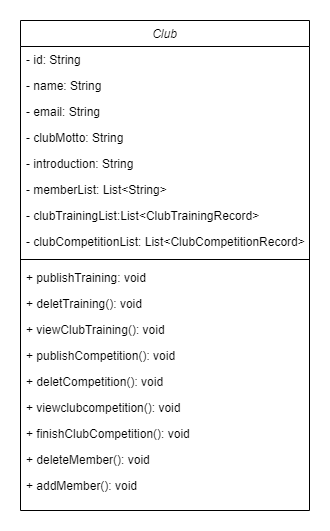
1. **结束训练活动**：

* 当训练活动完成时，俱乐部管理员调用 finishTraining() 方法更新训练状态为“已完成”。
* 系统记录训练结束时间，并更新训练状态。

1. **查看训练信息**：

* 用户（管理员、成员或其他参与者）可以查看训练的详细信息，包括标题、内容、参与成员列表和训练状态。

**2.4.2.3.3 Club类**



**类名：Club**

**简介**：  
Club 类表示一个俱乐部的基本信息和管理功能，包含了俱乐部的基础信息、成员、训练和比赛记录等。该类为俱乐部提供了管理训练和比赛活动、成员的加入与删除等功能。它是俱乐部运作的核心类，负责协调俱乐部内部活动、赛事以及成员管理。

**属性**：

* **id (String)**: 俱乐部的唯一标识符，通常由系统自动生成，用于区分不同的俱乐部。
* **name (String)**: 俱乐部的名称，表示俱乐部的标识或称号。
* **email (String)**: 俱乐部的联系方式（通常为邮箱），用于与外部或会员进行沟通。
* **clubMotto (String)**: 俱乐部的座右铭或口号，体现俱乐部的精神或宗旨。
* **introduction (String)**: 俱乐部的简要介绍，描述俱乐部的历史、宗旨和主要活动。
* **memberList (List<String>)**: 俱乐部的成员列表，存储每个成员的 ID 或名称，表示俱乐部的所有成员。
* **clubTrainingList (List<ClubTrainingRecord>)**: 俱乐部的训练记录列表，包含所有的训练活动记录，每个训练活动都是一个 ClubTrainingRecord 对象。
* **clubCompetitionList (List<ClubCompetitionRecord>)**: 俱乐部的比赛记录列表，包含所有的比赛活动记录，每个比赛活动都是一个 ClubCompetitionRecord 对象。

**方法**：

* **publishTraining()**: 俱乐部管理员可以发布新的训练活动，生成一个新的训练记录，并将其添加到 clubTrainingList 中。
* **deleteTraining()**: 俱乐部管理员可以删除已发布的训练活动，从 clubTrainingList 中移除指定的训练记录。
* **viewClubTraining()**: 允许用户查看俱乐部的所有训练活动，通常显示 clubTrainingList 中的所有训练记录。
* **publishCompetition()**: 俱乐部管理员可以发布新的比赛活动，生成一个新的比赛记录，并将其添加到 clubCompetitionList 中。
* **deleteCompetition()**: 俱乐部管理员可以删除已发布的比赛活动，从 clubCompetitionList 中移除指定的比赛记录。
* **viewClubCompetition()**: 允许用户查看俱乐部的所有比赛活动，通常显示 clubCompetitionList 中的所有比赛记录。
* **finishClubCompetition()**: 标记比赛活动为已完成，更新比赛的状态为“已完成”。
* **deleteMember()**: 俱乐部管理员可以删除某个成员，将其从 memberList 中移除。
* **addMember()**: 俱乐部管理员可以将新成员添加到 memberList 中，表示该成员加入了俱乐部。

**关系**：

* **与 ClubTrainingRecord 类有关系**：俱乐部的每个训练活动都通过 clubTrainingList 存储为 ClubTrainingRecord 类的实例，俱乐部可以管理和查看所有的训练记录。
* **与 ClubCompetitionRecord 类有关系**：俱乐部的每个比赛活动都通过 clubCompetitionList 存储为 ClubCompetitionRecord 类的实例，俱乐部可以管理和查看所有的比赛记录。
* **与 User 类有关系**：俱乐部通过 memberList 存储成员信息，每个成员都是 User 类的实例，表示俱乐部的成员。

**业务规则**：

* 每个俱乐部都有一个唯一的 id，用来区分不同的俱乐部。
* 俱乐部的成员列表、训练活动和比赛记录都可以进行动态管理。
* 训练活动和比赛活动的状态（如已发布、已完成等）需要根据进展更新。
* 训练和比赛记录只能由俱乐部管理员或特定权限的用户管理。
* 俱乐部的成员可以通过添加和删除操作动态变化，管理者可以控制谁加入或离开俱乐部。

**用例**：

1. **发布新的训练活动**：

* 俱乐部管理员通过 publishTraining() 方法发布新的训练活动。
* 训练活动的信息被添加到 clubTrainingList 中，并通知成员或参与者。

1. **删除训练活动**：

* 俱乐部管理员通过 deleteTraining() 方法删除某个已发布的训练活动。
* 该训练活动从 clubTrainingList 中移除，并不再可用。

1. **查看俱乐部的训练活动**：

* 用户或管理员可以通过 viewClubTraining() 方法查看当前俱乐部的所有训练记录，展示 clubTrainingList 中的内容。

1. **发布新的比赛活动**：

* 俱乐部管理员通过 publishCompetition() 方法发布新的比赛活动。
* 比赛活动的信息被添加到 clubCompetitionList 中，等待成员参与。

1. **删除比赛活动**：

* 俱乐部管理员通过 deleteCompetition() 方法删除某个已发布的比赛活动。
* 该比赛活动从 clubCompetitionList 中移除，系统不再显示或记录此比赛。

1. **查看俱乐部的比赛活动**：

* 用户或管理员可以通过 viewClubCompetition() 方法查看当前俱乐部的所有比赛记录，展示 clubCompetitionList 中的内容。

1. **结束比赛**：

* 俱乐部管理员通过 finishClubCompetition() 方法标记比赛活动为“已完成”。
* 比赛的状态被更新为“已完成”，表示比赛已结束。

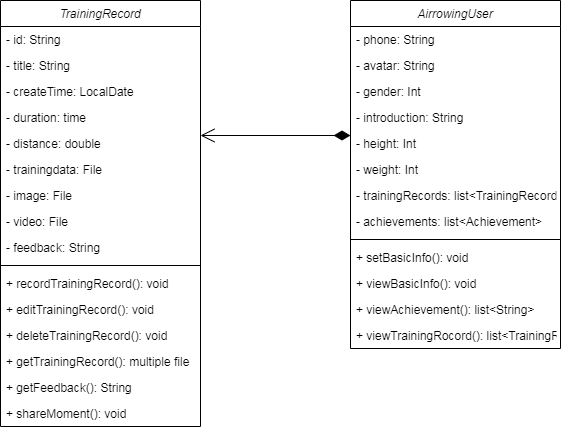
1. **添加成员**：

* 俱乐部管理员可以通过 addMember() 方法将一个新成员添加到俱乐部。
* 新成员的 id 被添加到 memberList 中，俱乐部成员数量增加。

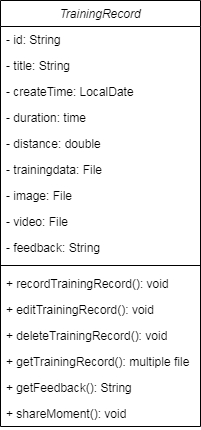
1. **删除成员**：

* 俱乐部管理员可以通过 deleteMember() 方法删除一个现有的成员。
* 被删除成员的 id 被从 memberList 中移除，俱乐部成员数量减少。

**2.4.2.4 训练分析模块**



**2.4.2.4.1 TrainingRecord 类**



**类名：**TrainingRecord

**简介：**TrainingRecord类表示用户的训练记录，包含有关每次训练的数据，如训练时长、距离、图片、视频、反馈等信息。该类是训练分析模块的核心，负责记录用户的训练数据并提供分析功能。用户可以上传训练数据、获取智能分析建议、并根据反馈调整训练方式，最终分享训练成果与社区进行互动。

**属性：**

* **id (string)**: 训练记录的唯一标识符。
* **title (string)**: 训练记录的标题或描述，最长100字符。
* **duration (int)**: 训练的持续时间，以分钟为单位。
* **distance (int)**: 训练的总距离，以公里为单位。
* **trainingData (File)**: 包含训练数据的字典，如心率、步频等。
* **image (string)**: 训练过程中拍摄的图像文件路径。
* **video (string)**: 训练过程中录制的视频文件路径。
* **feedback (string)**: 大模型分析给出的反馈意见和训练建议。

**方法：**

* **recordTrainingRecord(trainingData, image, video)**: 记录新的训练数据，包括训练时长、心率、图像、视频等，并将这些数据上传系统。
* **editTrainingRecord(newTrainingData)**: 修改已有的训练记录，更新训练数据或反馈信息。
* **deleteTrainingRecord()**: 删除当前的训练记录。
* **getTrainingRecord()**: 获取用户的训练记录信息，包括数据、图像、视频及反馈等。
* **getFeedback()**: 调用大模型分析上传的训练数据并返回智能分析和建议。
* **shareMoment()**: 将训练记录和分析结果分享至社区，供其他用户查看和交流。

**关系：**

* **与User类有“拥有”关系**：一个User可以拥有多个TrainingRecord，每个TrainingRecord都与特定的User关联。

**业务规则：**

* 每个训练记录必须包含至少一个有效的训练数据（如时长、距离等）。
* 如果用户上传的训练数据格式无效，系统会提示并要求重新上传。
* 用户可以编辑或删除任何自己的训练记录。
* 训练记录可以通过调用大模型分析后，获取反馈意见和个性化的训练建议。
* 训练记录可以选择是否分享至社区。分享后的记录可以被其他用户查看，并参与互动。

**用例：**

1. **用户上传训练数据并获取分析建议**：

* 用户上传训练数据（如时长、心率、图片或视频等）。
* 系统验证数据有效性，确保数据格式正确。
* 系统自动调用大模型进行分析，并返回个性化的训练建议。
* 用户查看分析结果，获得改进训练的建议。

1. **用户编辑训练记录**：

* 用户修改已有的训练记录，更新如训练时长或图像等信息。
* 系统更新训练记录，并重新调用大模型进行反馈分析。

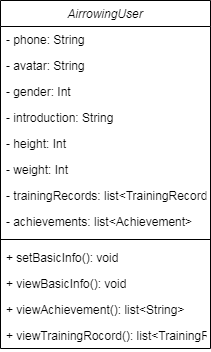
1. **用户删除训练记录**：

* 用户选择删除一个训练记录。
* 系统删除该记录，并确认删除操作。

1. **用户分享训练记录**：

* 用户上传训练数据并成功获得分析结果后，系统提示用户是否分享该记录至社区。
* 用户选择“分享”后，训练记录及分析结果被上传到社区，其他用户可以查看并互动。
* 用户选择“不分享”后，系统仅显示分析结果，不进行分享。

**2.4.2.4.2 AirrowingUser类**



**类名：**AirrowingUser

**简介：**  
AirrowingUser类表示参与空中划船训练的用户，包含与用户个人信息相关的属性（如性别、身高、体重）以及该用户的训练记录。该类是训练管理系统中的一个重要组成部分，负责存储用户的个人信息和训练历史。每个用户可以有多个训练记录，并通过viewTrainingRecord()方法查看自己的训练数据。

**属性：**

* **gender (string)**: 用户的性别。
* **height (float)**: 用户的身高，以厘米为单位。
* **weight (float)**: 用户的体重，以千克为单位。
* **trainingRecord (list)**: 一个包含多个训练记录的列表，记录了用户每次训练的数据。每个训练记录由TrainingRecord类实例表示。

**方法：**

* **viewTrainingRecord()**: 返回该用户的所有训练记录列表。用户可以查看每一条训练记录的详细信息，包括训练时长、距离、反馈等。
* **setBasicInfo()**: 用户修改自己的身高、体重或性别等信息。

**关系：**

* **与TrainingRecord类有“包含”关系**：一个AirrowingUser可以拥有多个TrainingRecord，表示该用户参与的不同训练记录。
* **与Feedback类有“间接关联”关系**：每个TrainingRecord都可能包含由大模型生成的反馈信息，帮助用户提升训练效果。

**业务规则：**

* 每个用户可以拥有零个或多个训练记录。
* 用户的身高、体重、性别等信息在创建用户时进行初始化，并可以进行修改。
* 用户可以随时查看自己所有的训练记录，以便跟踪训练进展和改进方向。

**用例：**

1. **用户查看训练记录：**

* 用户调用viewTrainingRecord()方法查看自己所有的训练记录。
* 系统返回一个包含多个TrainingRecord实例的列表，用户可以查看每个训练记录的详细数据，包括训练时长、反馈、图像和视频等。

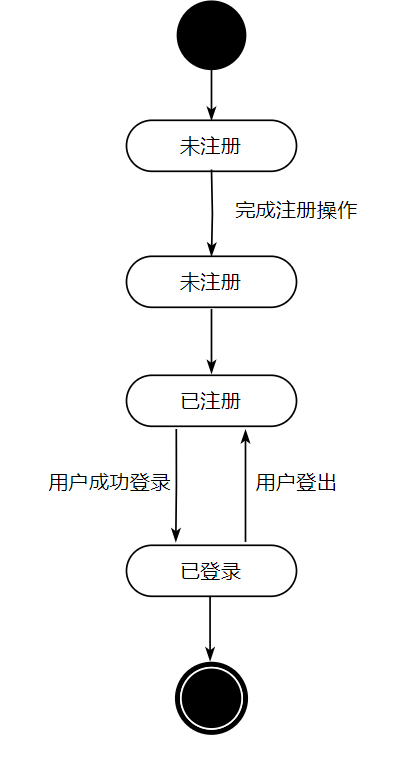
1. **用户更新个人信息：**

* 用户可以修改自己的身高、体重或性别等信息。
* 系统更新这些信息并保持训练记录不变。

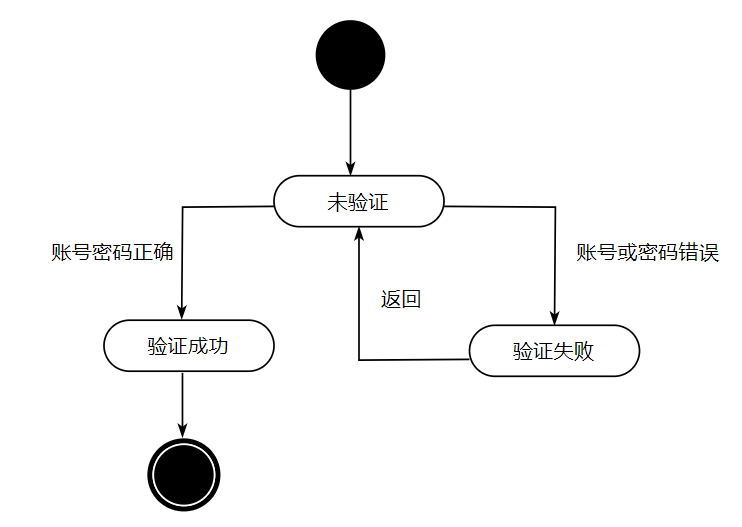
**2.4.3 行为建模**

2.1 **状态图**

**User类**



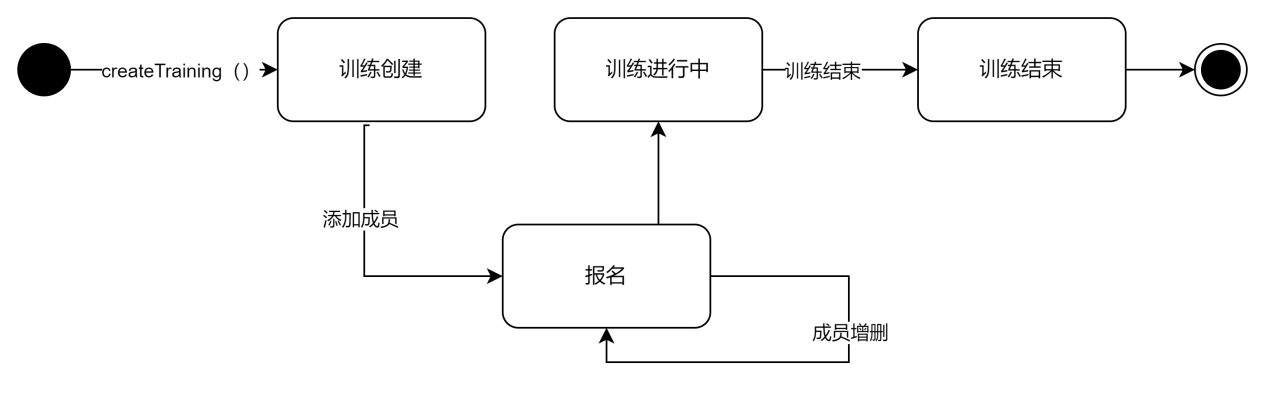
**Authentication类**



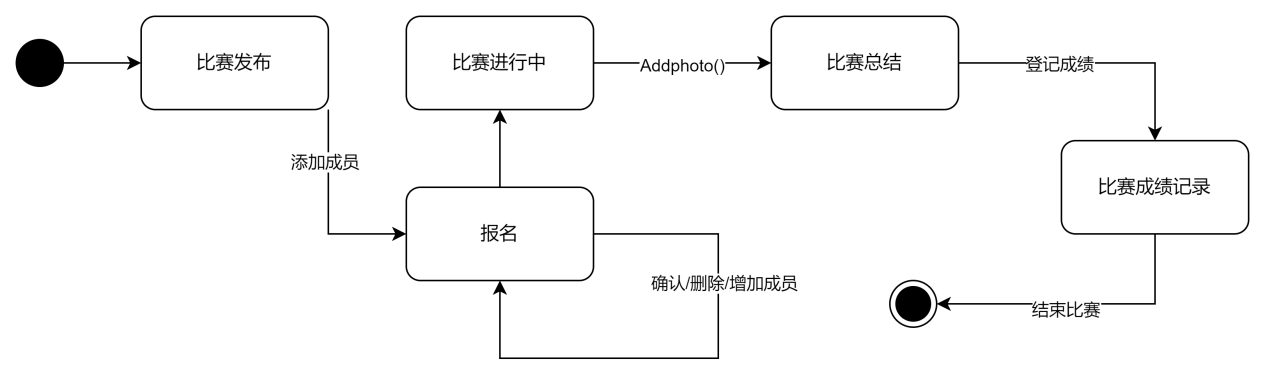
**Club类**



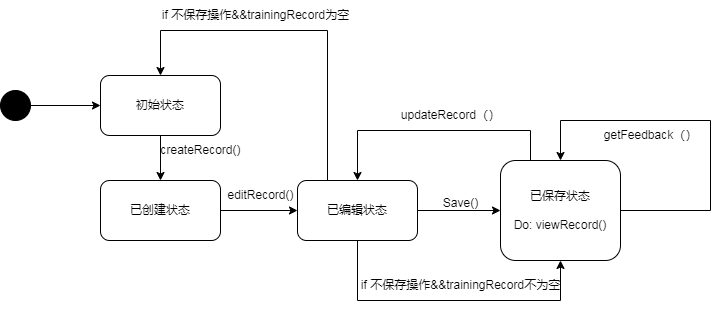
**ClubTrainingRecord类**



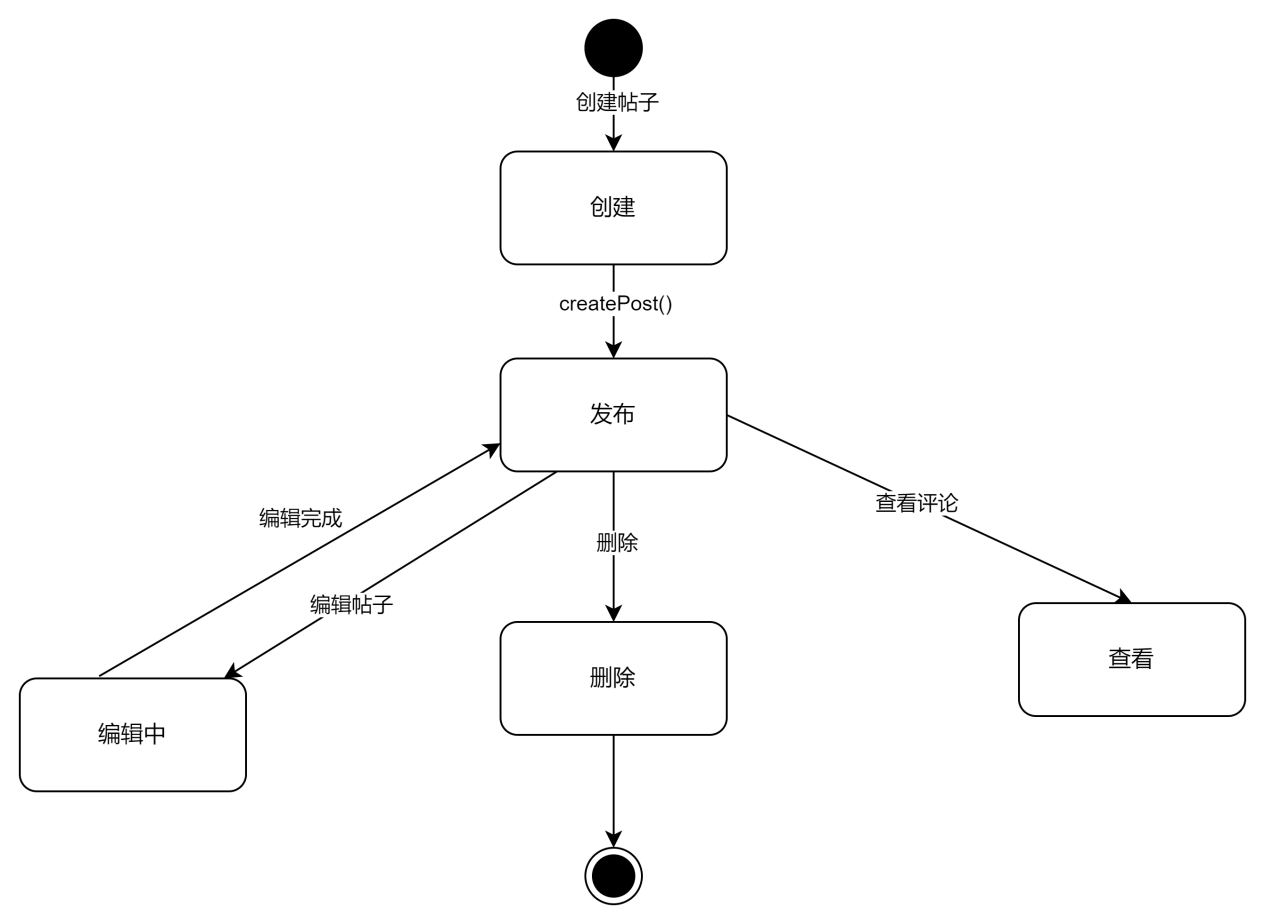
**ClubCompetitionRecord类**



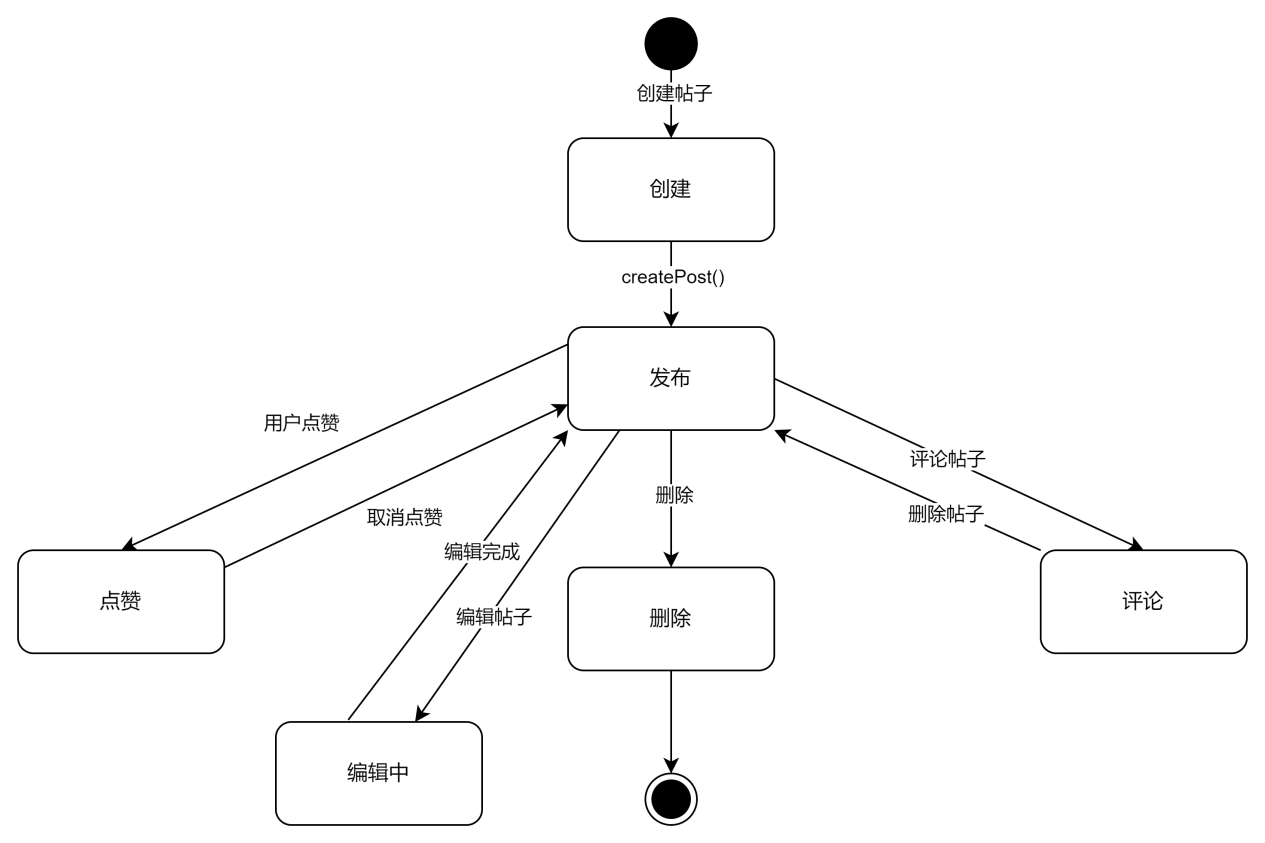
**trainingRecord类**



**Comment类**



**Post类**

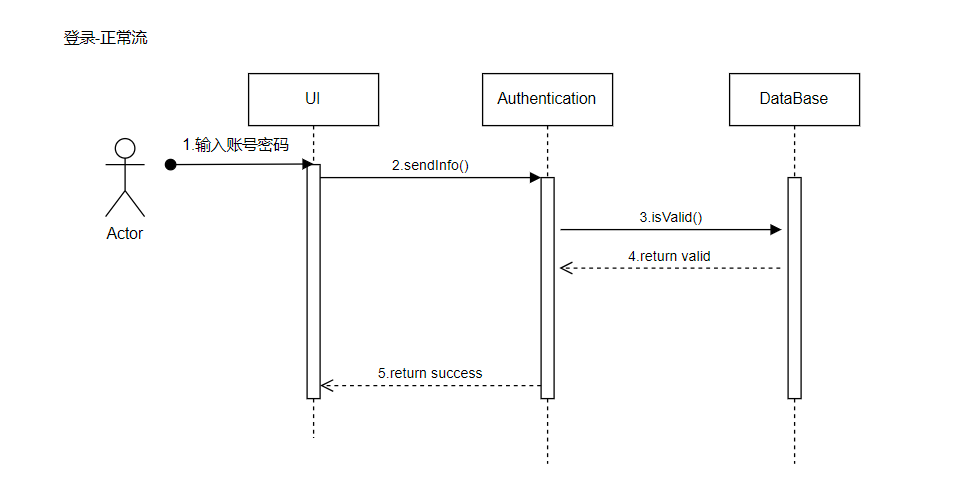


2.2 **时序图**

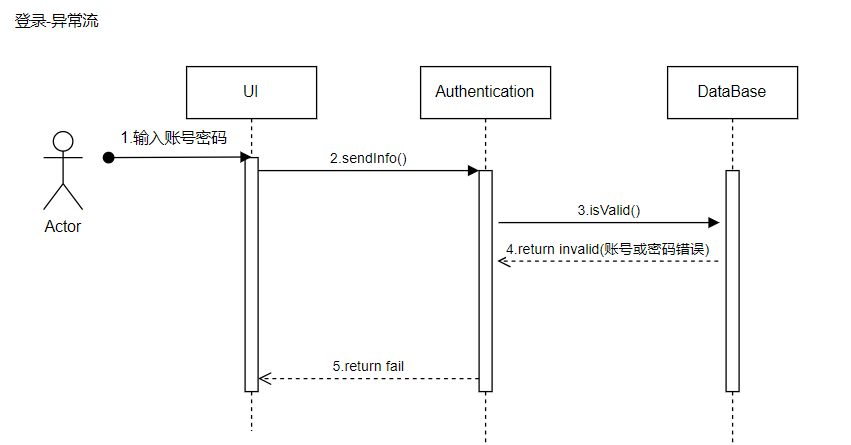
**2.4.3.1 账号管理模块**

**2.4.3.1.1 用户登录**

**正常流**



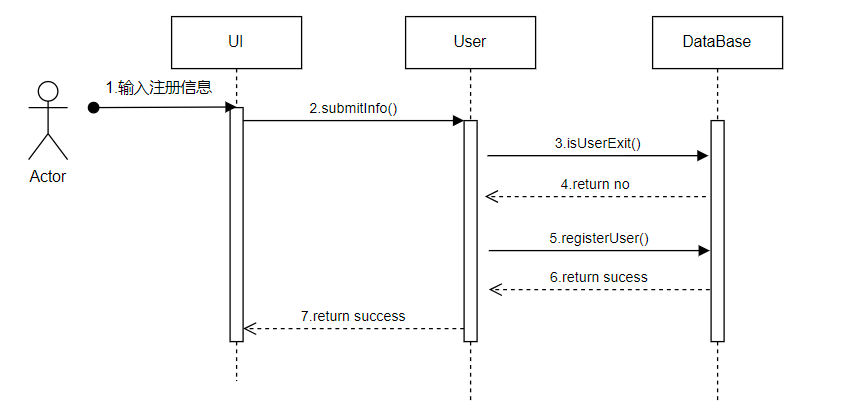
**异常流**



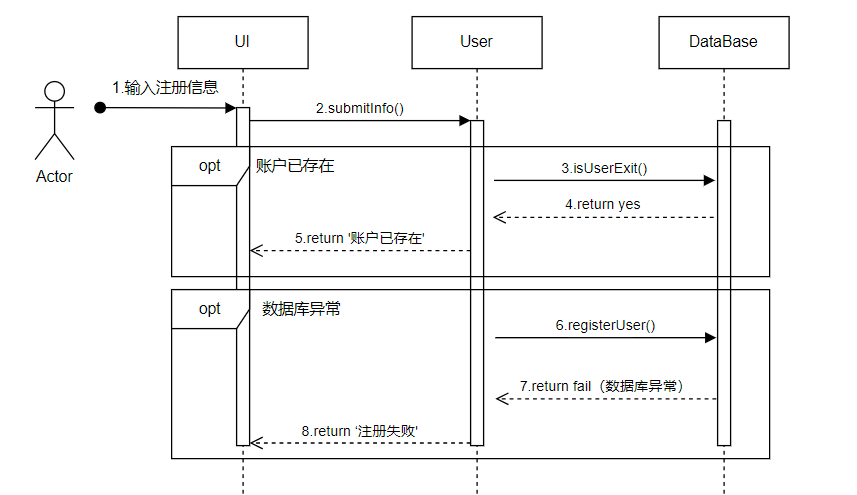
该序列图展示了用户登录的过程。用户在登录界面输入账号和密码后提交请求，系统接收到请求后调用验证模块检查用户的账号和密码是否匹配。若验证通过，系统记录用户的登录状态，并将用户引导到主页面。若验证失败，系统向用户返回错误提示信息，用户可以根据提示重新输入正确的登录信息完成登录。

**2.4.3.1.2 账号注册**

**正常流**

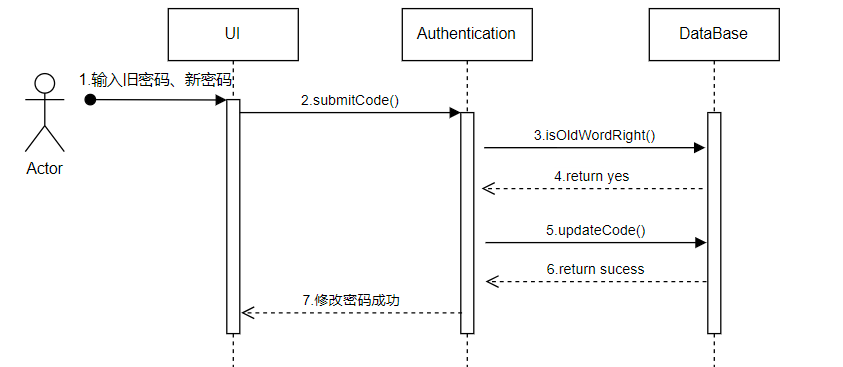


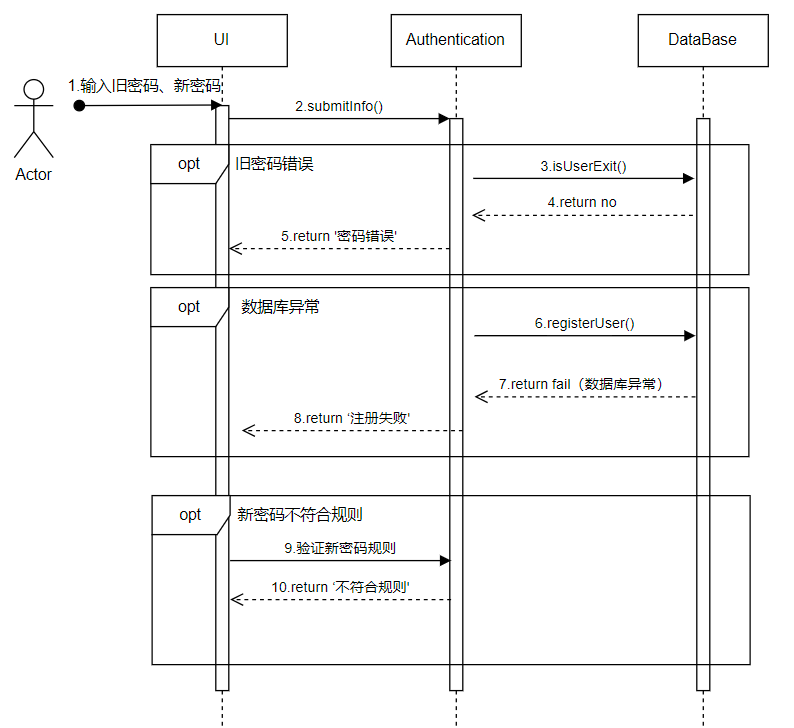
**异常流**



该序列图描述了用户注册账号的交互流程。用户在界面上输入注册信息并提交注册请求，系统首先验证用户输入的邮箱格式是否有效。若邮箱格式正确，系统生成验证码并通过邮件发送给用户。用户收到验证码后提交验证请求，系统对验证码进行校验，验证通过后将用户信息保存到数据库，并返回注册成功的消息给用户，完成注册流程。

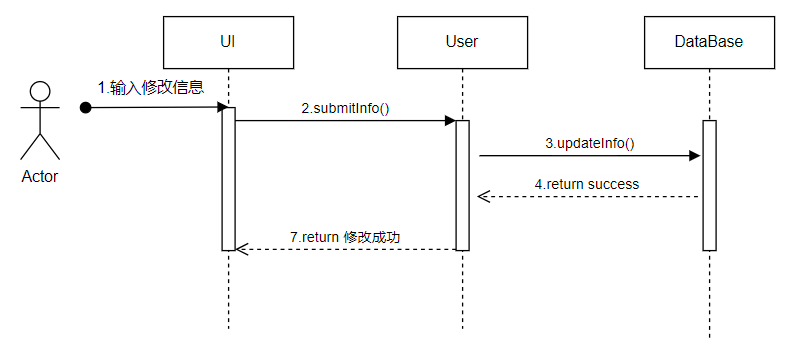
**2.4.3.1.3 密码修改**

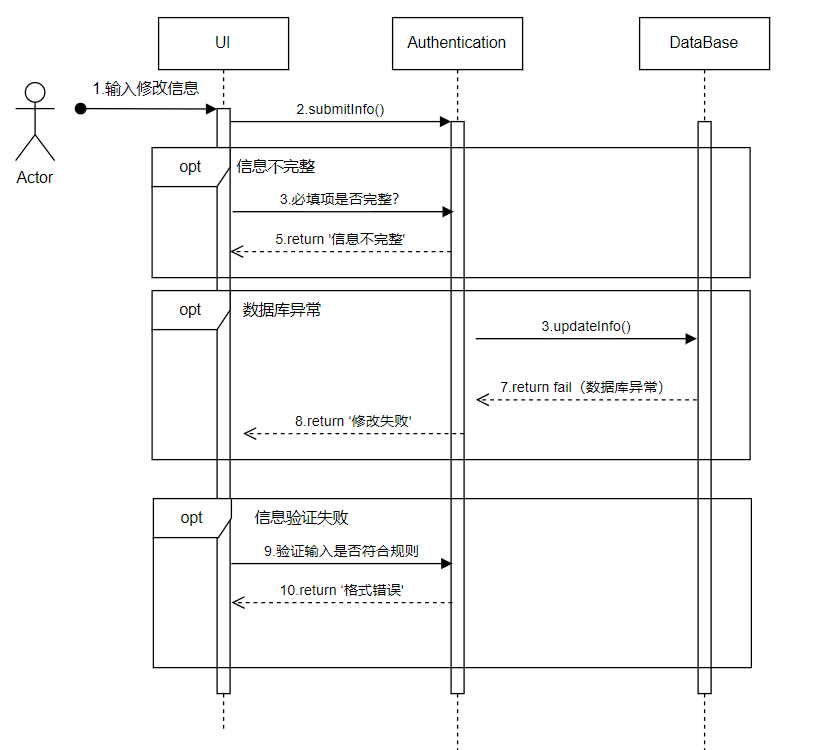




该序列图描述了用户在系统中修改密码的流程。用户首先访问密码修改界面，并提交旧密码和新密码。系统接收到请求后，调用验证模块检查旧密码是否正确，若验证通过，系统继续验证新密码的格式是否符合要求。当所有验证通过后，系统更新数据库中的用户密码信息，并向用户返回密码修改成功的消息，完成整个流程

**2.4.3.1.4 个人信息查看与修改**



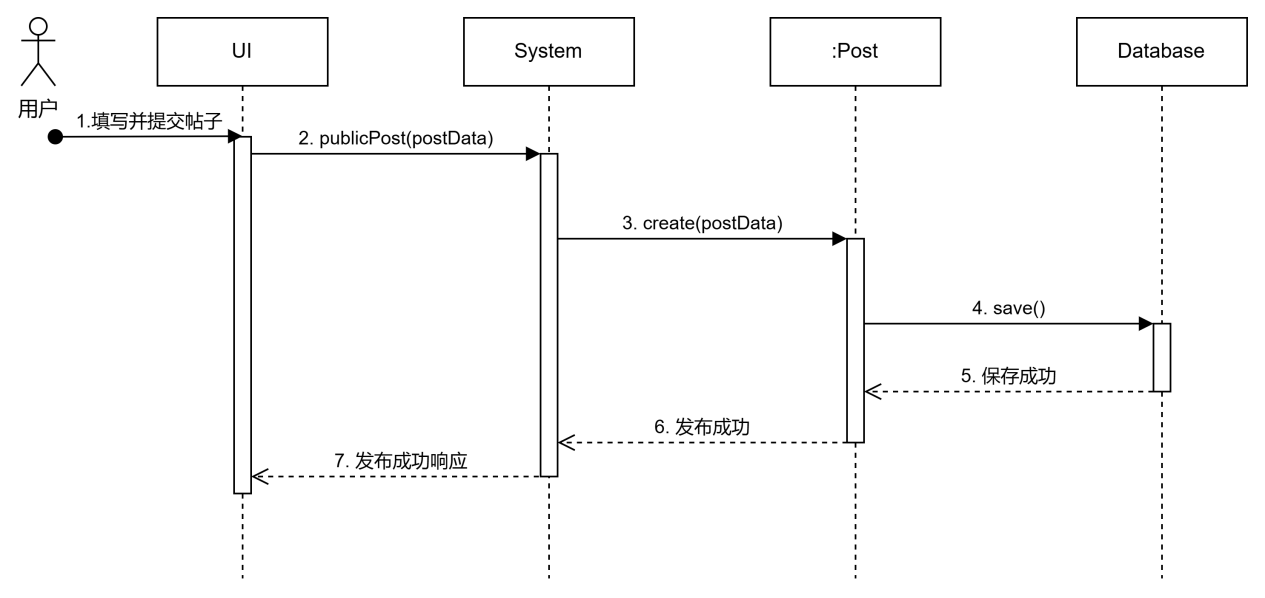


该序列图展示了用户修改个人信息的流程。用户通过界面发起请求查看当前的个人信息，系统从数据库中获取信息并展示给用户。用户修改个人信息后提交更改请求，系统验证提交的信息是否合法，验证通过后将修改后的信息更新到数据库。最后，系统向用户反馈修改成功的消息，从而完成个人信息的修改操作。

**2.4.3.2 社区模块**

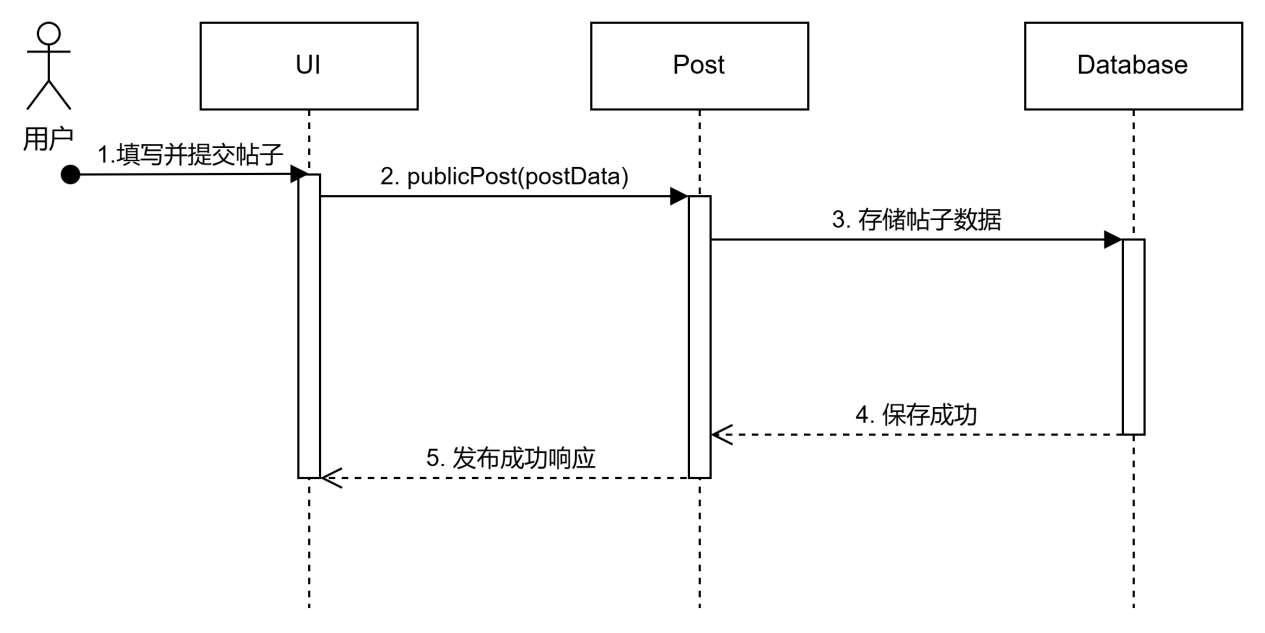
**2.4.3.2.1 发布帖子**

iteration1：（旧版）

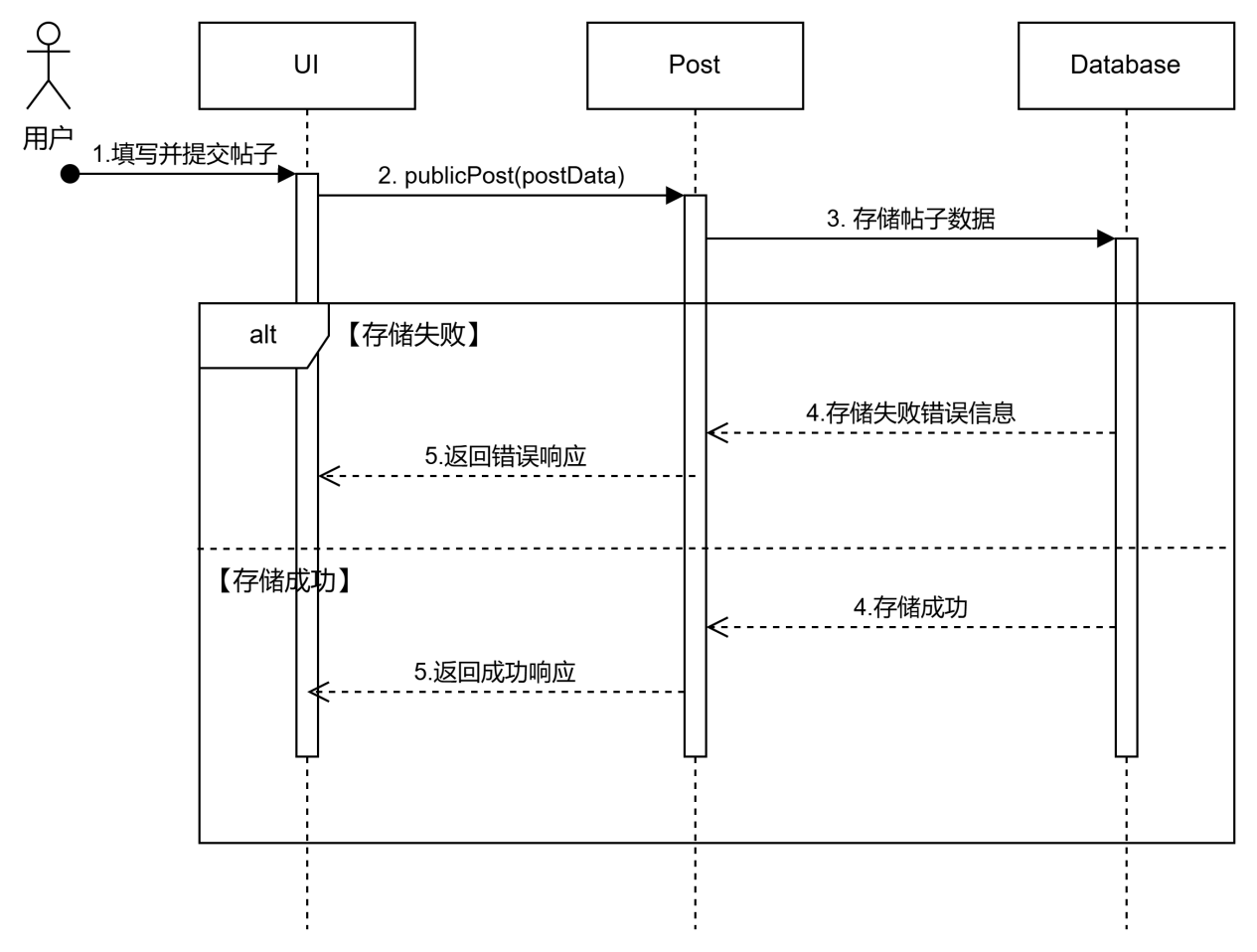


iteration2：

**正常流：**



**异常流：**



1. **参与者（Actors and Participants）**

* **用户（User）**：系统外部的终端用户，发起发布帖子请求。
* **UI**：用户界面，作为用户与系统交互的中介。
* **Post**：负责处理发布帖子的业务逻辑。
* **Database**：负责存储帖子数据的后端数据库系统。

1. **正常流程（Success Flow）**

* **步骤 1**：用户通过 **UI** 发起发布帖子请求。
* **步骤 2**：**UI** 将该请求转发给 **Post**。
* **步骤 3**：**Post** 处理该请求，并将帖子数据存储到 **Database**。
* **步骤 4**：**Database** 返回存储成功的确认。
* **步骤 5**：**Post** 将成功响应返回给 **UI**。

1. **异常流程（Exception Flow）**

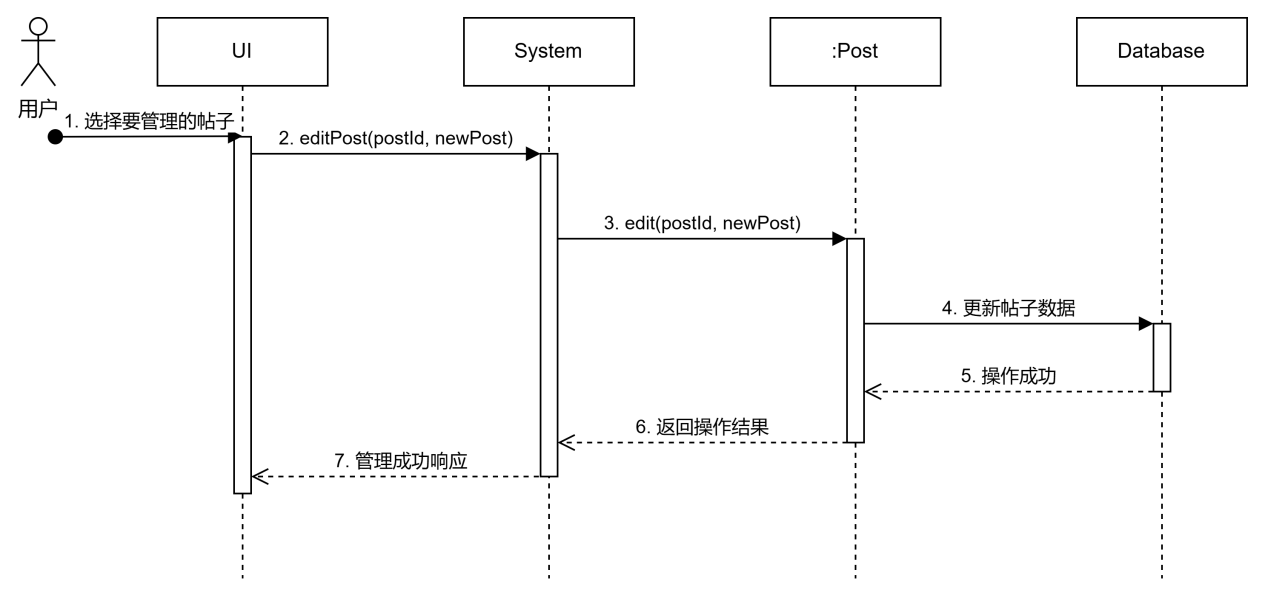
* **步骤 1-3**：与正常流程相同，用户通过 **UI** 发起请求，**UI** 转发给 **Post**，**Post** 尝试将数据存储到 **Database**。
* **步骤 4**：**Database** 返回存储失败的错误信息（如连接失败、数据验证错误等）。
* **步骤 5**：**Post** 捕获该错误，并生成错误响应。

1. **条件分支（alt-else）**

* 使用 alt 存储失败 分支来处理 **Database** 存储失败的情况。
* 使用 else 存储成功 分支处理存储成功的情况，以确保时序图涵盖所有可能的结果。

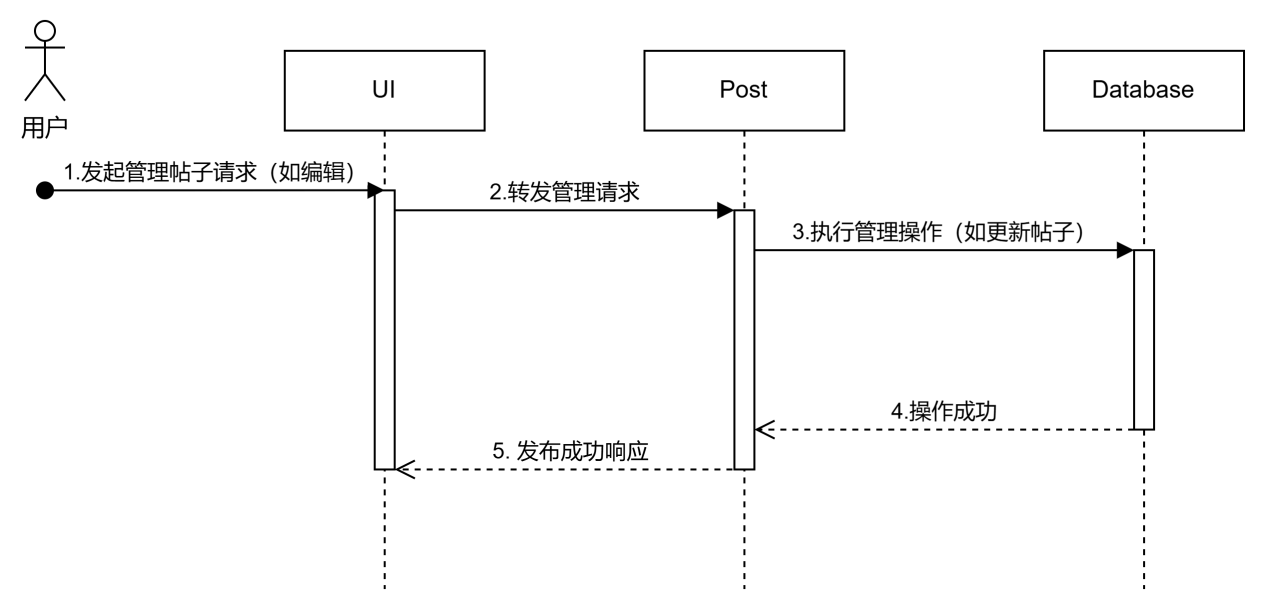
**2.4.3.2.2 管理帖子**

iteration1：（旧版）

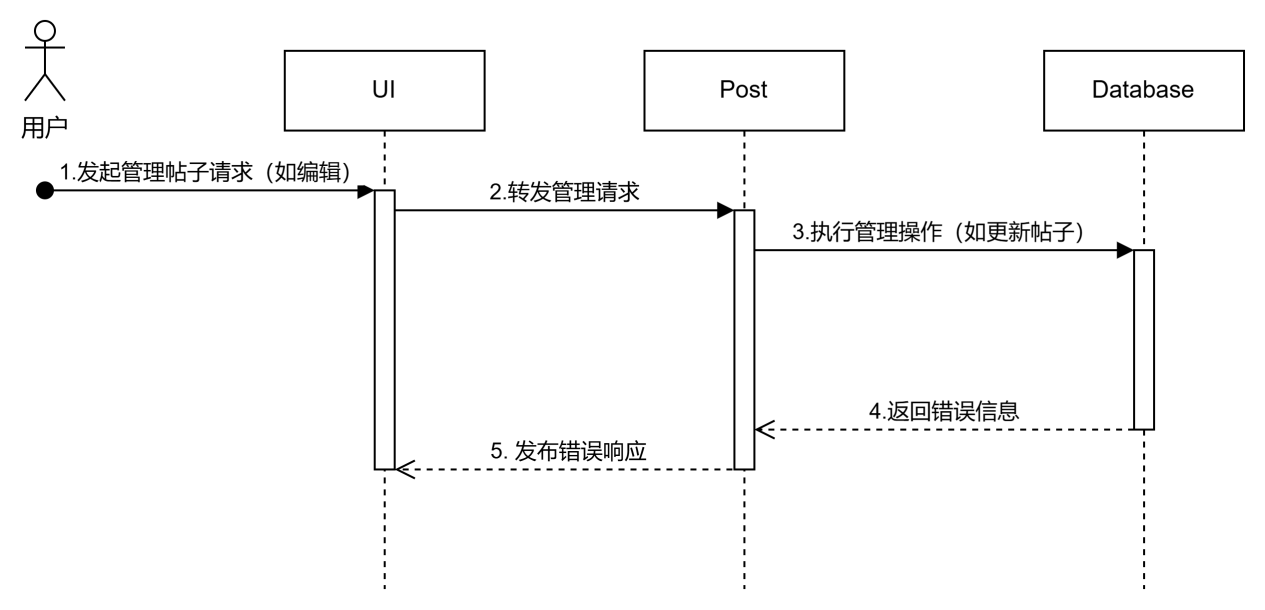


iteration2：

**正常流：**



**异常流：**



**参与者：**

* **用户（User）**：发起管理帖子的请求（如编辑、删除、查看）。
* **UI**：用户界面，负责接收用户输入并显示响应。
* **Post**：处理管理请求的实体，负责业务逻辑处理。
* **Database**：存储帖子数据的数据库系统。

**流程描述：**

1. **正常流程（Normal Flow）**：

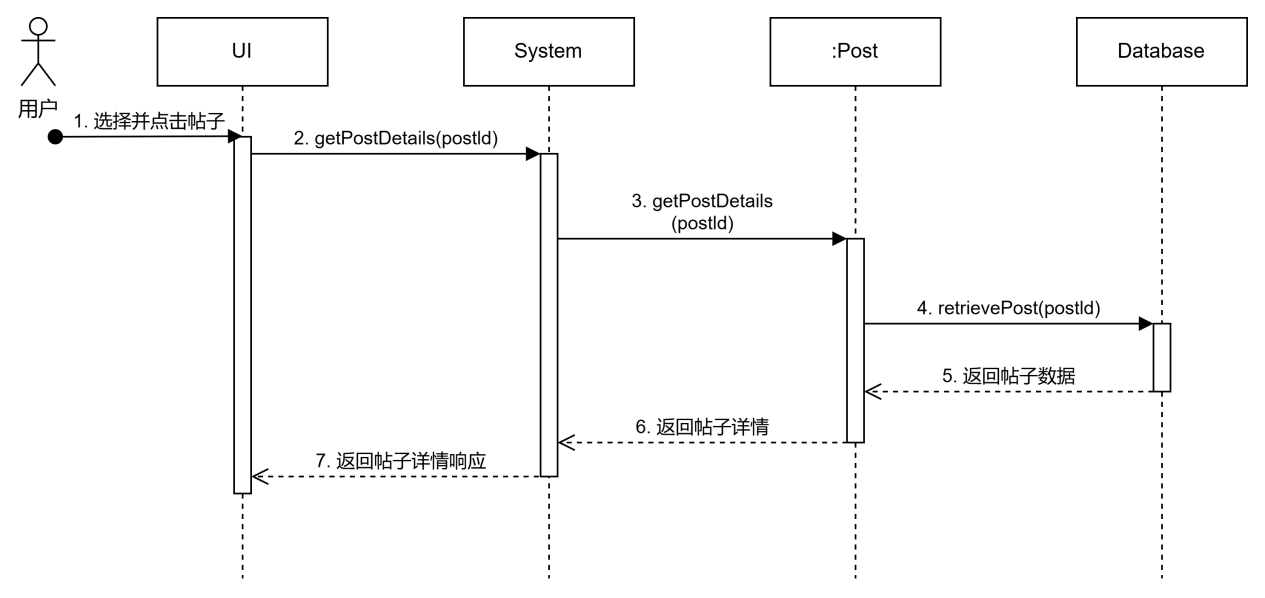
* 用户通过 **UI** 发起管理帖子请求（例如编辑帖子）。
* **UI** 将请求转发给 **Post**。
* **Post** 处理请求并与 **Database** 进行交互（如更新帖子数据）。
* **Database** 确认操作成功。
* **Post** 返回成功响应给 **UI**。
* **UI** 将成功信息展示给用户。

1. **异常流程（Exception Flow）**：

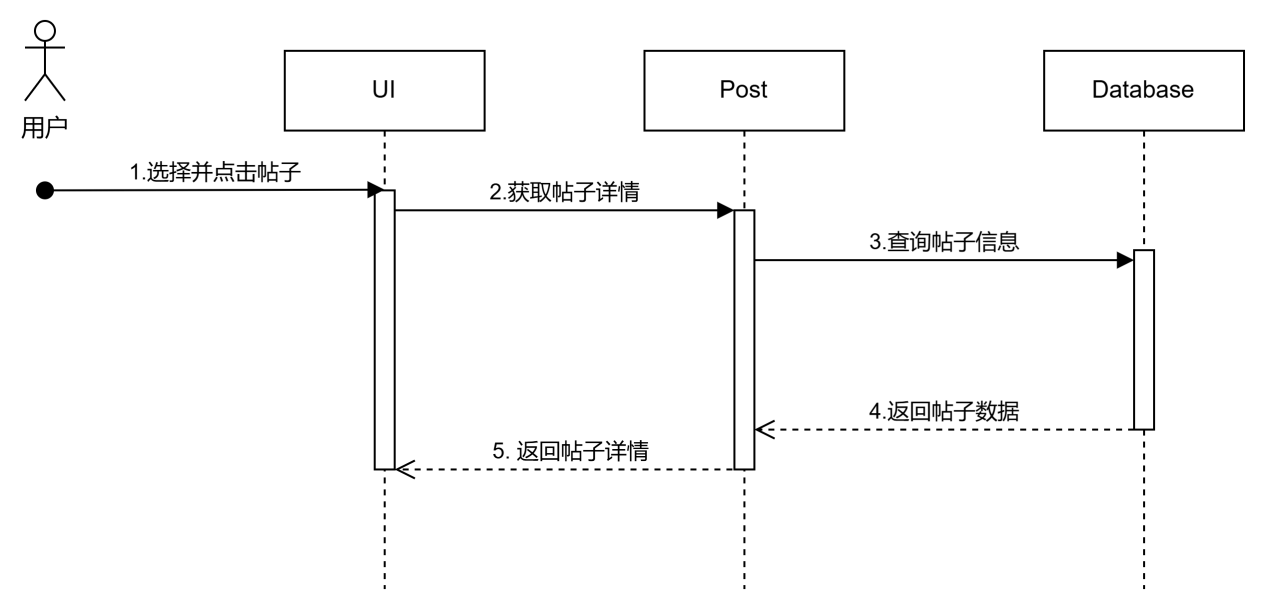
* 在正常流程的某个步骤中，如果发生异常（如数据库操作失败），则通过 opt 块表示异常处理流程。

**2.4.3.2.3 浏览帖子**

iteration1：（旧版）



iteration2：



**参与者：**

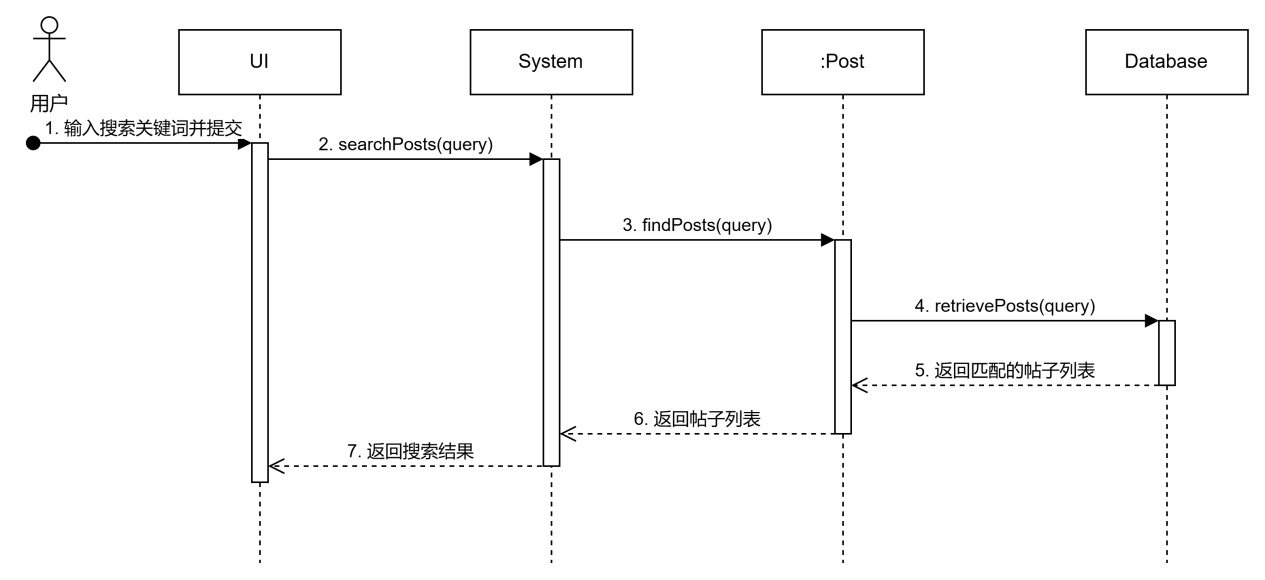
* **用户（User）**：发起浏览帖子请求的终端用户。
* **UI**：用户界面，负责接收用户输入并显示响应。
* **Post**：处理浏览请求的实体，负责业务逻辑处理。
* **Database**：存储帖子数据的数据库系统。

**正常流程（Normal Flow）**：

* 用户通过 **UI** 发起浏览帖子请求。
* **UI** 将请求转发给 **Post**。
* **Post** 处理请求并从 **Database** 获取帖子数据。
* **Database** 返回请求的数据。
* **Post** 返回帖子数据给 **UI**。
* **UI** 将帖子数据显示给用户。

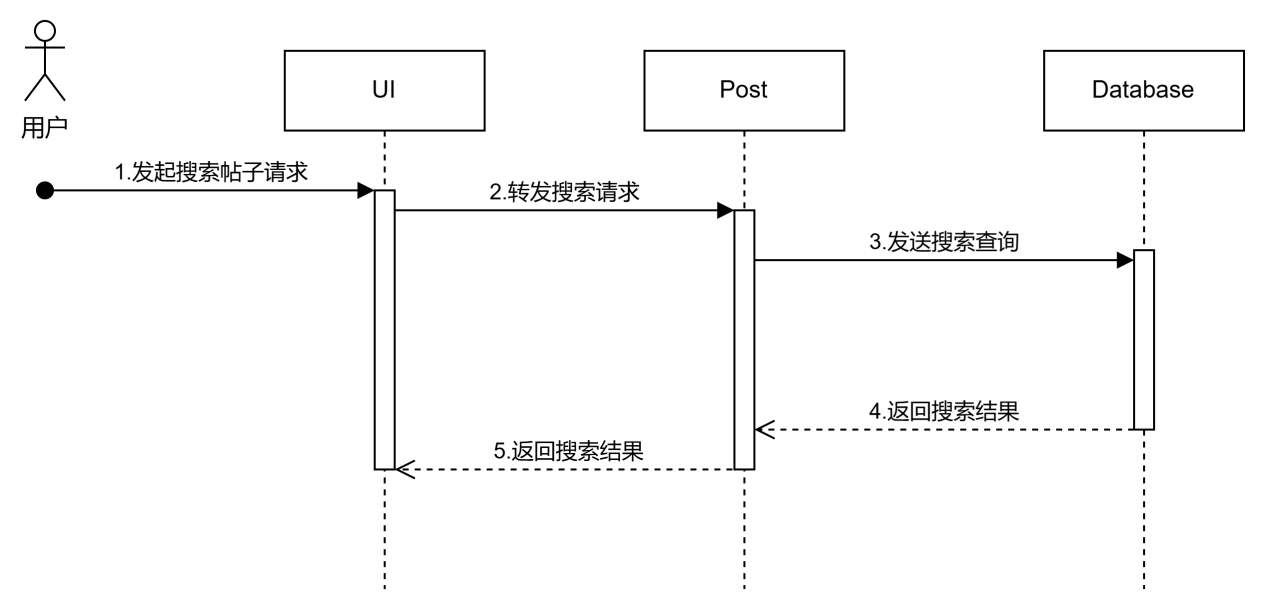
**2.4.3.2.4 搜索帖子**

iteration1：（旧版）



iteration2：

**正常流：**



**异常流：**



**参与者：**

* **用户（User）**：发起搜索帖子请求的终端用户。
* **UI**：用户界面，负责接收用户输入并显示响应。
* **Post**：处理搜索请求的实体，负责业务逻辑处理。
* **Database**：存储帖子数据的数据库系统。

**流程描述：**

1. **正常流程（Normal Flow）**：

* 用户通过 **UI** 发起搜索帖子请求。
* **UI** 将请求转发给 **Post**。
* **Post** 处理请求并向 **Database** 发送搜索查询。
* **Database** 返回匹配的帖子数据。
* **Post** 返回搜索结果给 **UI**。
* **UI** 将搜索结果展示给用户。

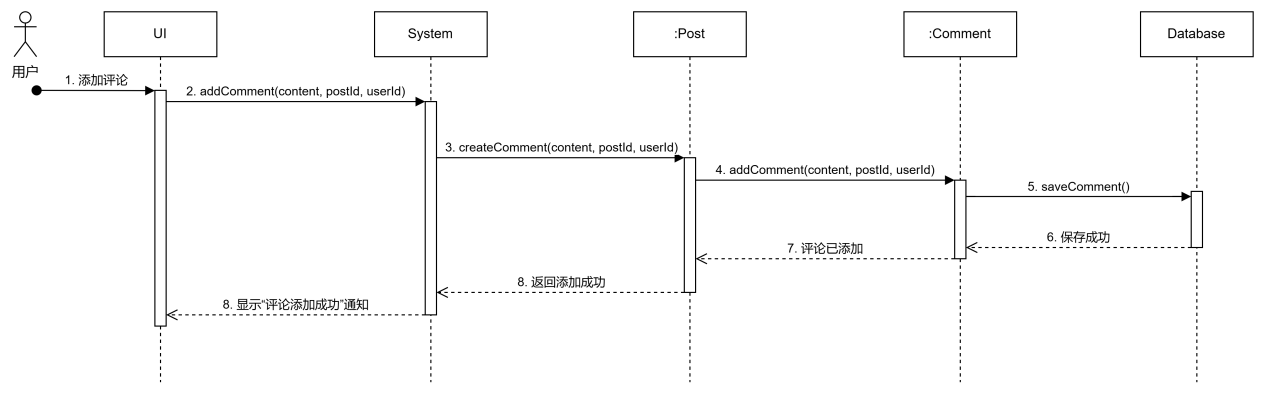
1. **异常流程（Exception Flows）**：

* **异常情况 1**：数据库连接失败。
* **异常情况 2**：查询结果为空（无匹配帖子）。
* **异常情况 3**：数据格式错误导致查询失败。

**2.4.3.2.5 帖子互动**

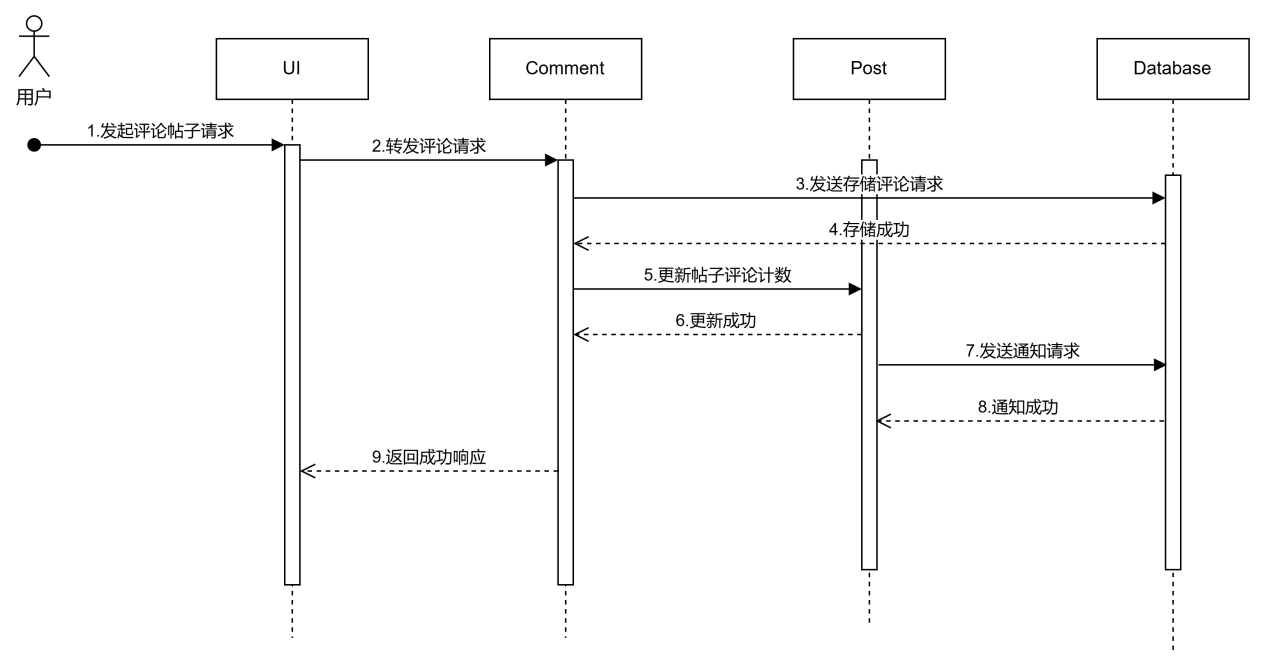
**a. 评论**

iteration1：（旧版）

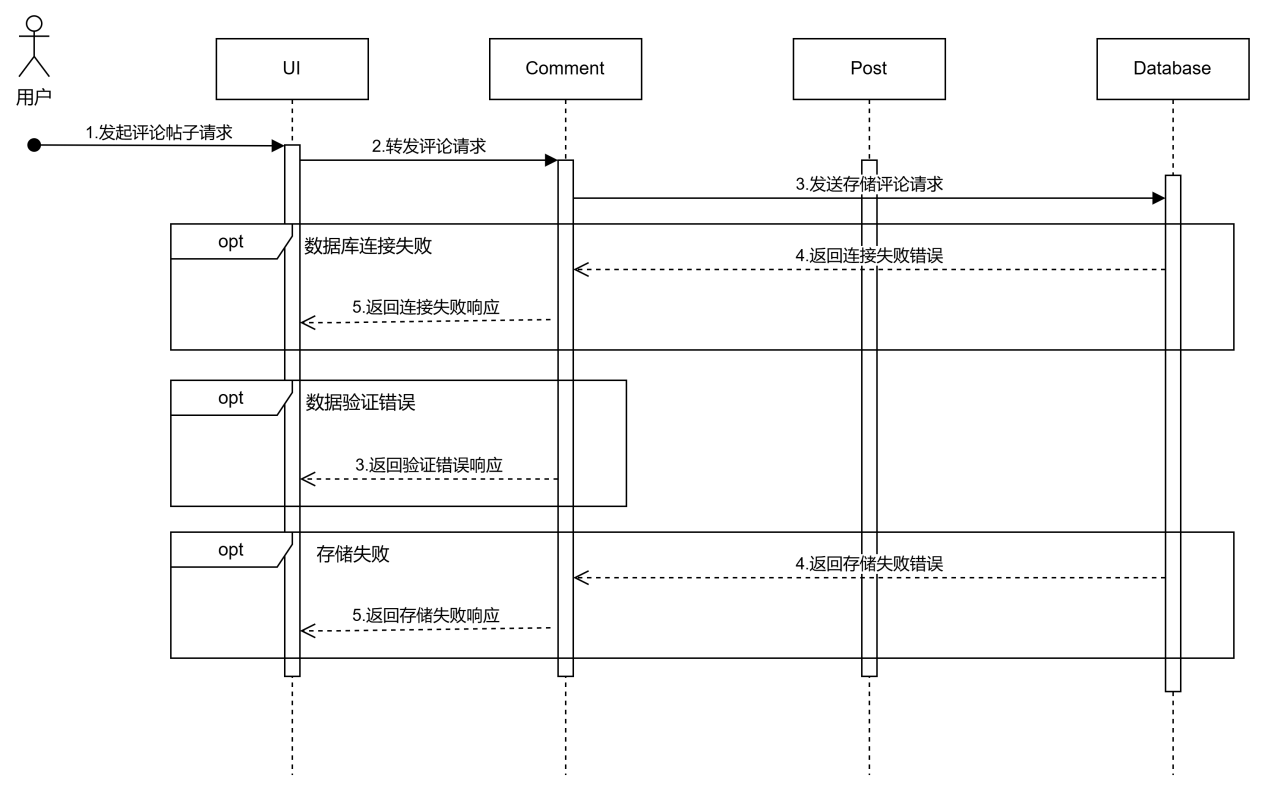


iteration2：

**正常流：**



**异常流：**



**参与者：**

* **用户（User）**：发起评论帖子请求的终端用户。
* **UI**：用户界面，负责接收用户输入并显示响应。
* **Comment**：处理评论请求的实体，负责业务逻辑处理。
* **Post**：处理与帖子相关的业务逻辑，如更新评论计数和通知作者。
* **Database**：存储评论数据的数据库系统。

**流程描述：**

1. **正常流程（Normal Flow）**：

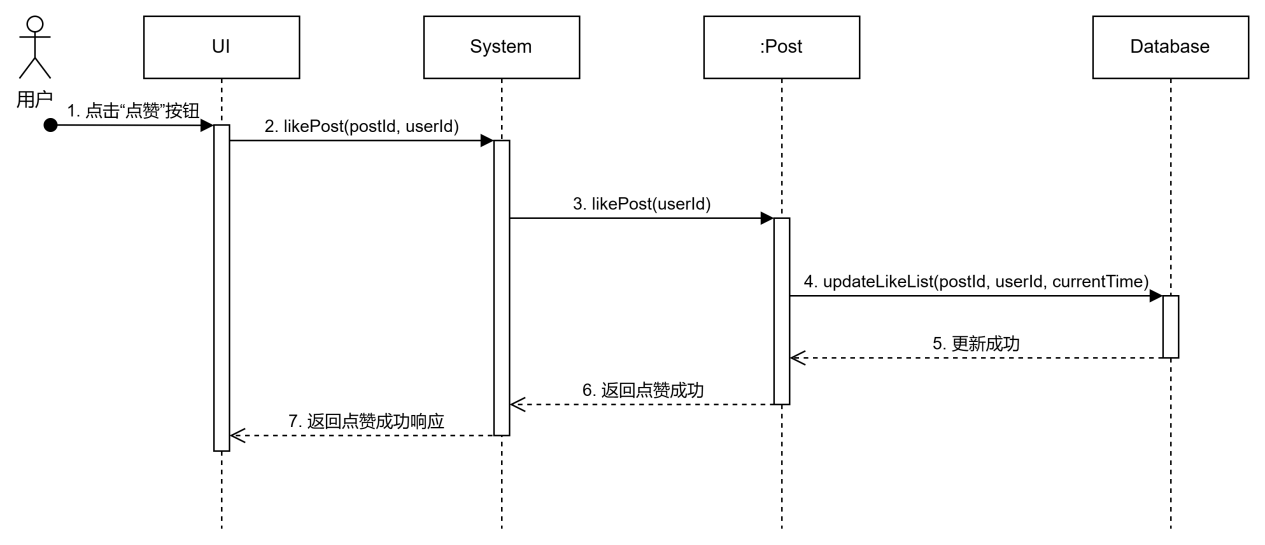
* 用户通过 **UI** 发起评论帖子请求。
* **UI** 将请求转发给 **Comment**。
* **Comment** 处理请求并向 **Database** 发送存储请求。
* **Database** 返回存储成功的确认。
* **Comment** 通知 **Post** 更新评论计数。
* **Post** 更新评论计数并向 **Database** 发送通知请求。
* **Database** 返回通知成功的确认。
* **Comment** 返回成功响应给 **UI**。
* **UI** 将评论成功的信息展示给用户。

1. **异常流程（Exception Flows）**：

* **异常情况 1**：数据库连接失败。
* **异常情况 2**：数据验证错误（如评论内容为空或超长）。
* **异常情况 3**：存储失败（如数据库操作错误）。

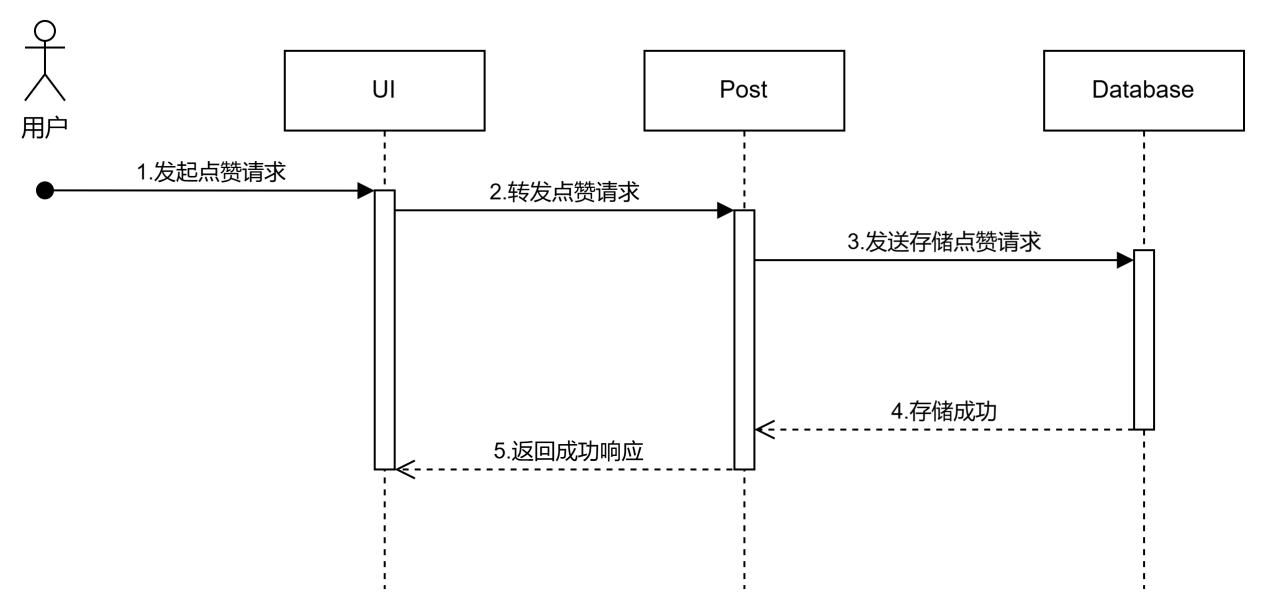
**b. 点赞**

iteration1：（旧版）

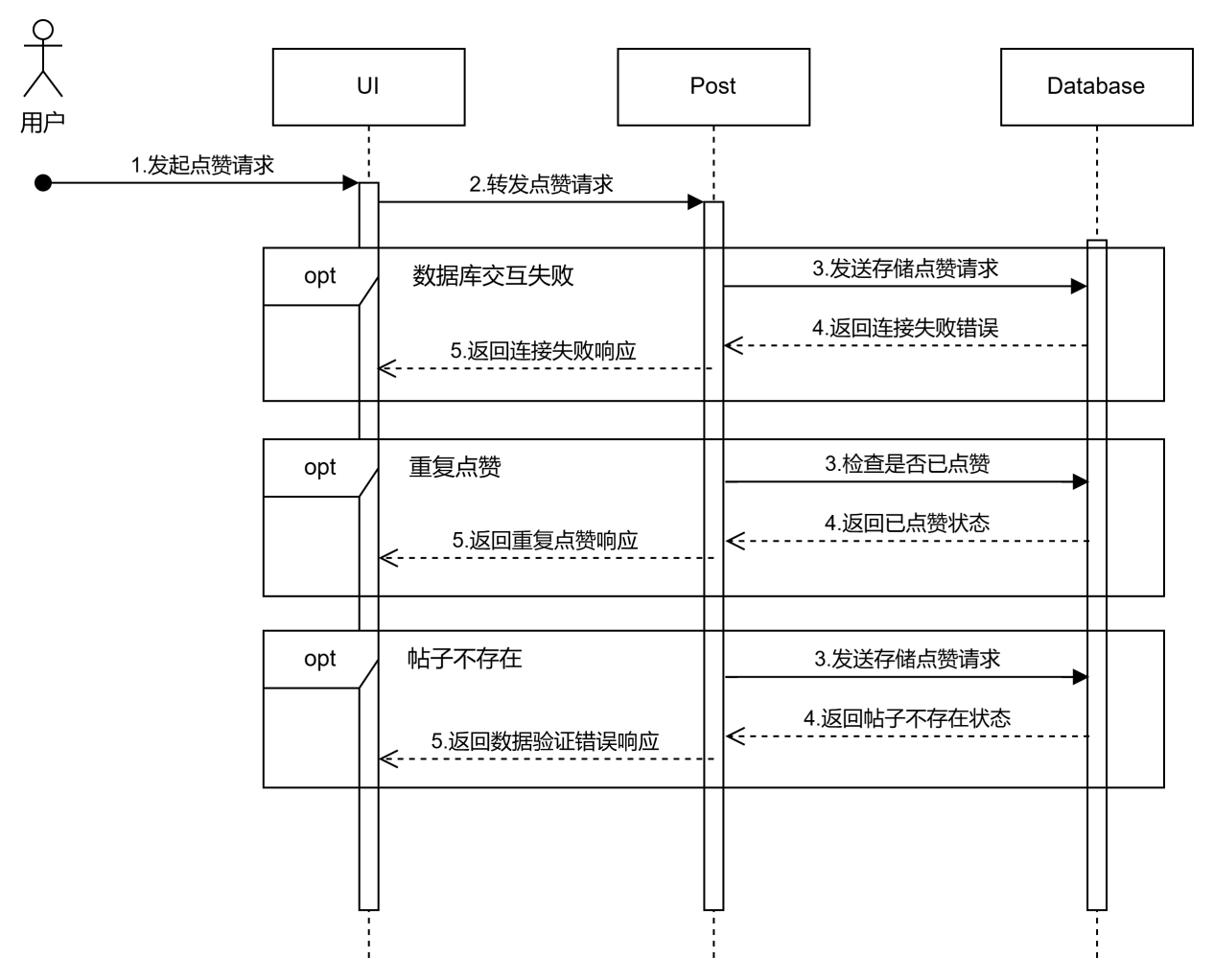


iteration2：

**正常流：**



**异常流：**



**参与者：**

* **用户（User）**：发起点赞请求的终端用户。
* **UI**：用户界面，负责接收用户输入并显示响应。
* **Post**：处理点赞请求的实体，负责业务逻辑处理。
* **Database**：存储点赞数据的数据库系统。

**流程描述：**

1. **正常流程（Normal Flow）**：

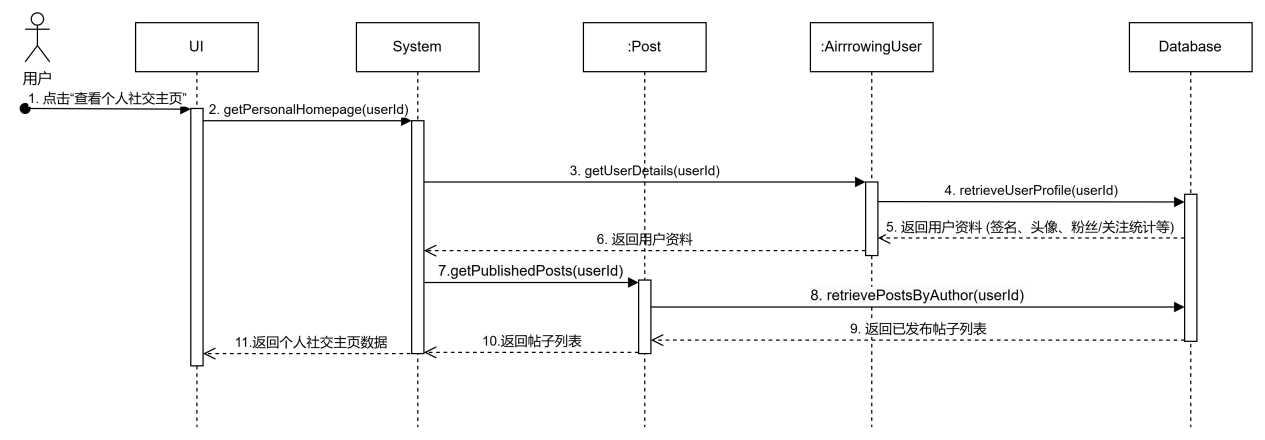
* 用户通过 **UI** 发起点赞请求。
* **UI** 将请求转发给 **Post**。
* **Post**处理请求并向 **Database** 发送存储请求。
* **Database** 返回存储成功的确认。
* **Post**返回成功响应给 **UI**。
* **UI** 将点赞成功的信息展示给用户。

1. **异常流程（Exception Flows）**：

* **异常情况 1**：数据库连接失败。
* **异常情况 2**：数据验证错误（如帖子不存在）。
* **异常情况 3**：重复点赞（用户已点赞过该帖子）。

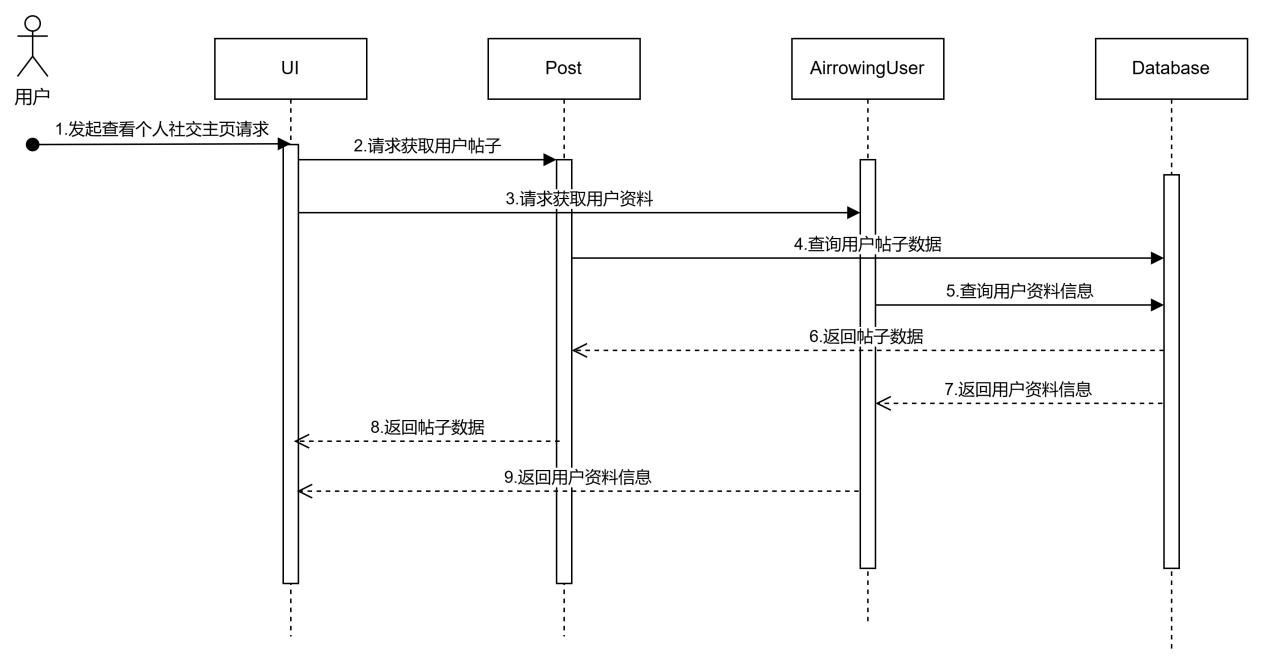
**2.4.3.2.6 查看个人社交主页**

iteration1：（旧版）

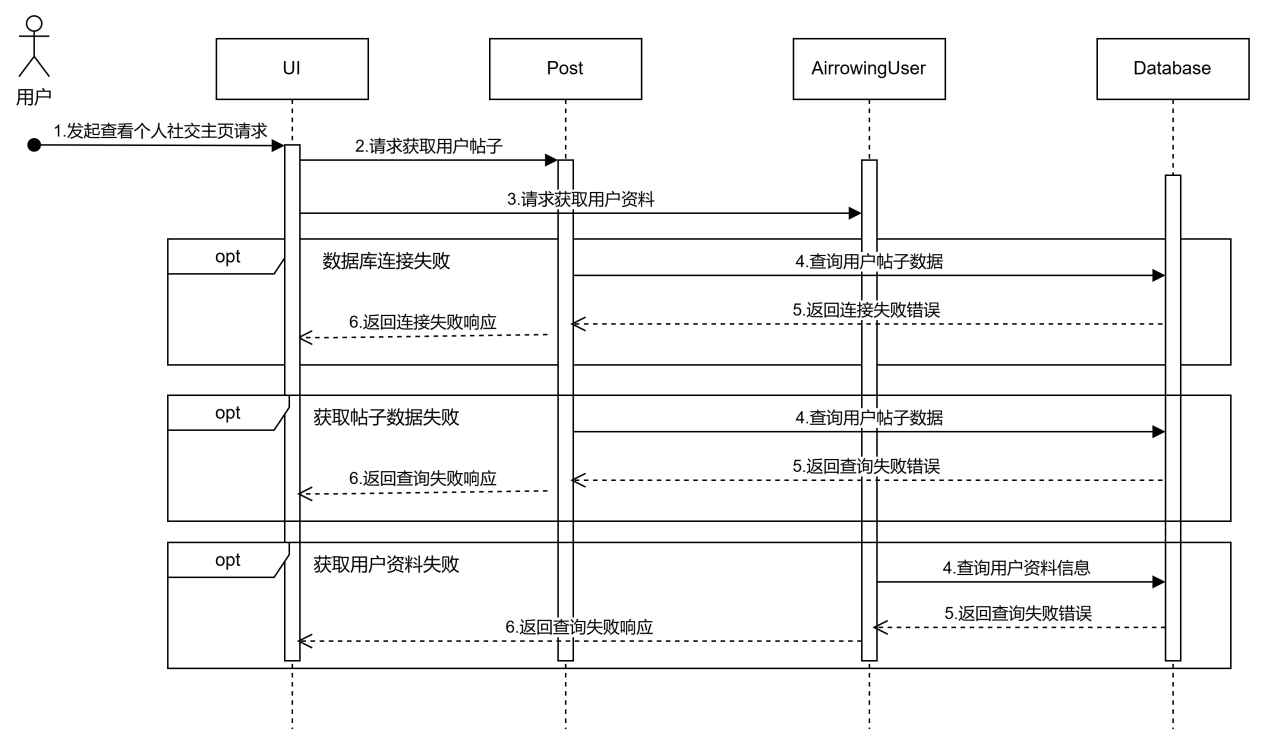


iteration2：

**正常流：**



**异常流：**



**参与者：**

* **用户（User）**：系统的终端用户，发起查看个人社交主页的请求。
* **UI**：用户界面，负责接收用户输入并显示响应。
* **Post**：处理与帖子相关的业务逻辑实体，负责获取用户的帖子。
* **AirrowingUser**：处理与用户资料相关的业务逻辑实体，负责获取用户的资料信息。
* **Database**：存储数据的数据库系统，负责提供所需的数据。

**流程描述：**

1. **正常流程（Normal Flow）**：

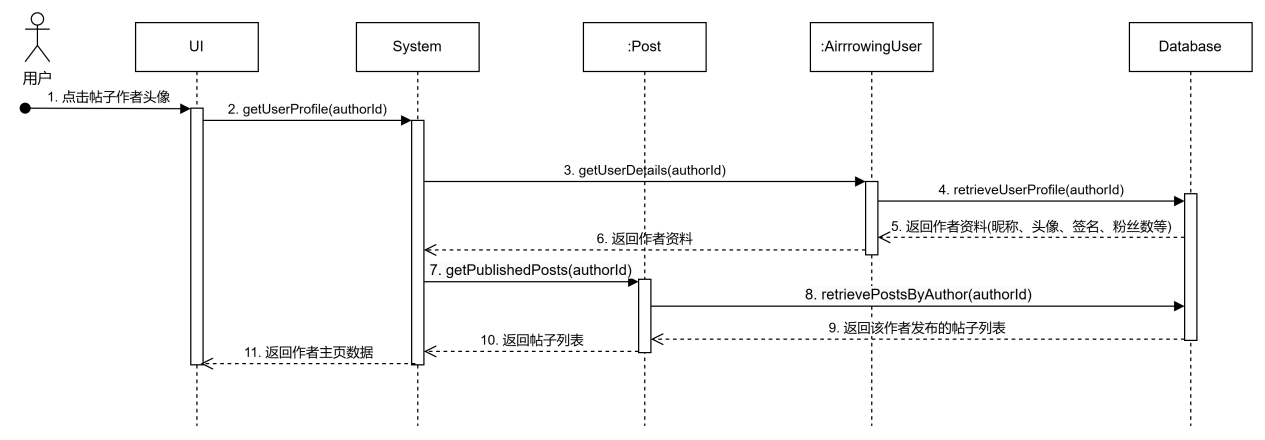
* 用户通过 **UI** 发起查看个人社交主页的请求。
* **UI** 将请求转发给 **Post** 和 **AirrowingUser**。
* **Post** 向 **Database** 请求获取用户的帖子数据。
* **AirrowingUser** 向 **Database** 请求获取用户的资料信息。
* **Database** 分别返回帖子数据和用户资料信息给 **Post** 和 **AirrowingUser**。
* **Post** 和 **AirrowingUser** 将获取的数据返回给 **UI**。
* **UI** 将个人社交主页的信息展示给用户。

1. **异常流程（Exception Flows）**：

* **异常情况 1**：数据库连接失败。
* **异常情况 2**：获取帖子数据失败。
* **异常情况 3**：获取用户资料失败。

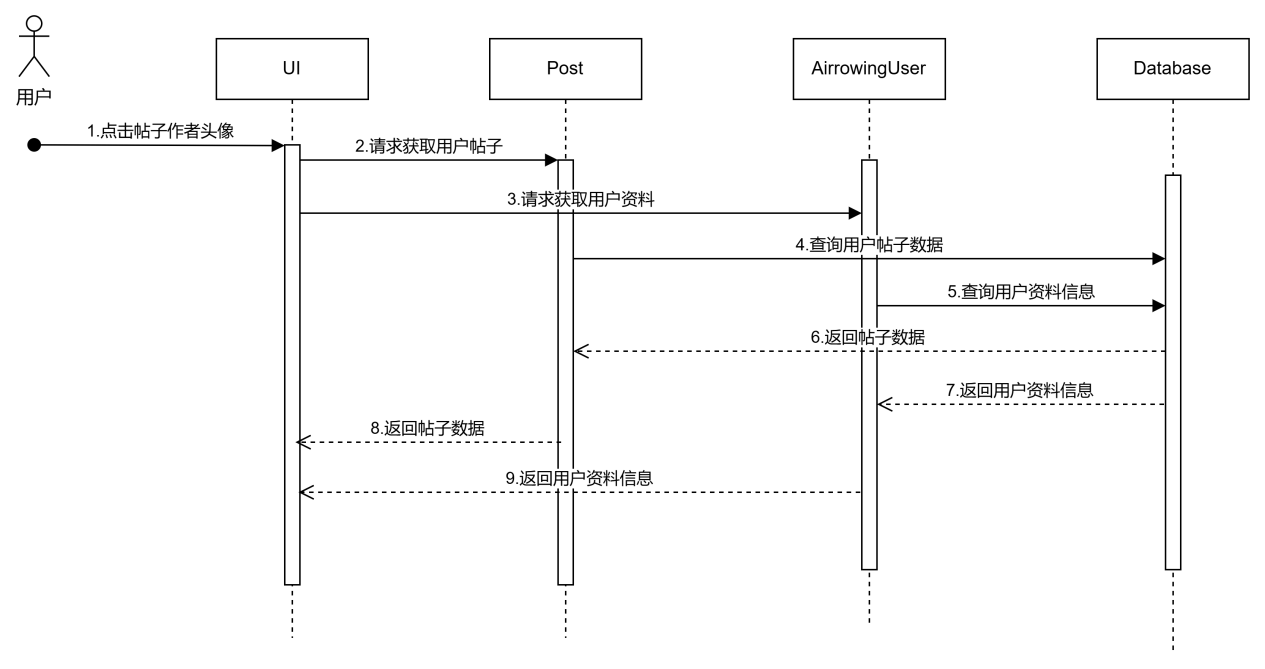
**2.4.3.2.7 查看其他用户的社交主页**

iteration1：（旧版）

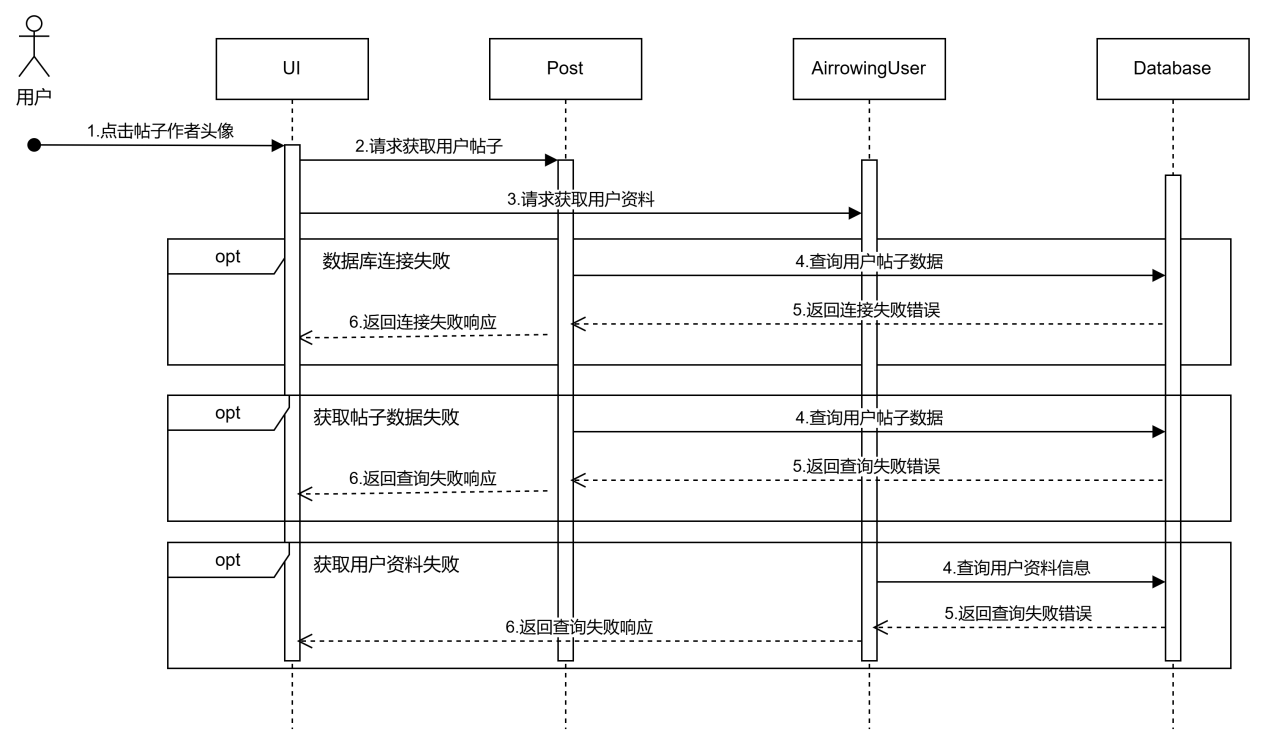


iteration2：

**正常流：**



**异常流：**



**参与者：**

* **用户（User）**：系统的终端用户，发起查看个人社交主页的请求。
* **UI**：用户界面，负责接收用户输入并显示响应。
* **Post**：处理与帖子相关的业务逻辑实体，负责获取用户的帖子。
* **AirrowingUser**：处理与用户资料相关的业务逻辑实体，负责获取用户的资料信息。
* **Database**：存储数据的数据库系统，负责提供所需的数据。

**流程描述：**

1. **正常流程（Normal Flow）**：

* 用户通过 **UI** 点击帖子作者的头像。
* **UI** 将请求转发给 **Post** 和 **AirrowingUser**。
* **Post** 向 **Database** 请求获取用户的帖子数据。
* **AirrowingUser** 向 **Database** 请求获取用户的资料信息。
* **Database** 分别返回帖子数据和用户资料信息给 **Post** 和 **AirrowingUser**。
* **Post** 和 **AirrowingUser** 将获取的数据返回给 **UI**。
* **UI** 将个人社交主页的信息展示给用户。

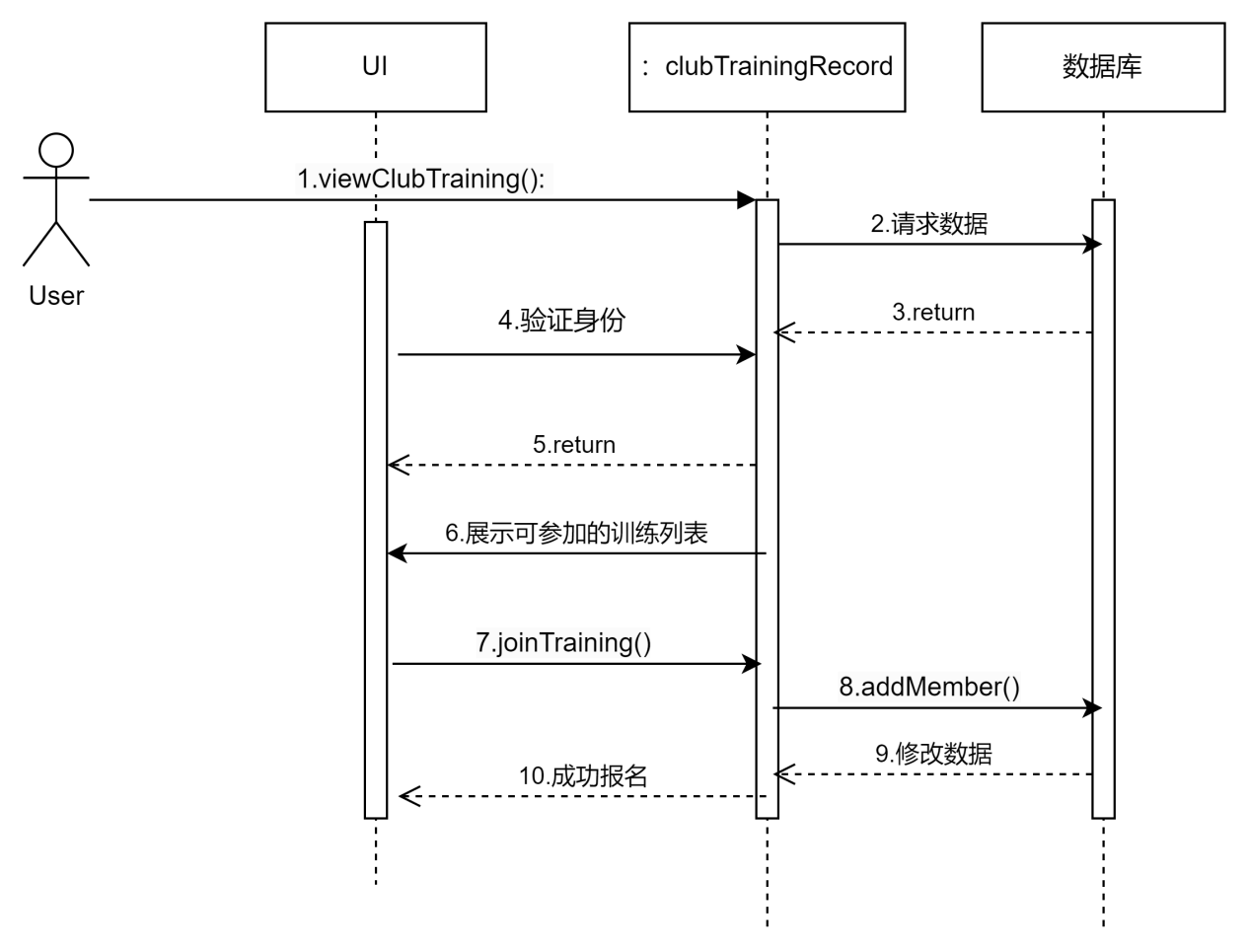
1. **异常流程（Exception Flows）**：

* **异常情况 1**：数据库连接失败。
* **异常情况 2**：获取帖子数据失败。
* **异常情况 3**：获取用户资料失败。

**2.4.3.3 俱乐部模块**

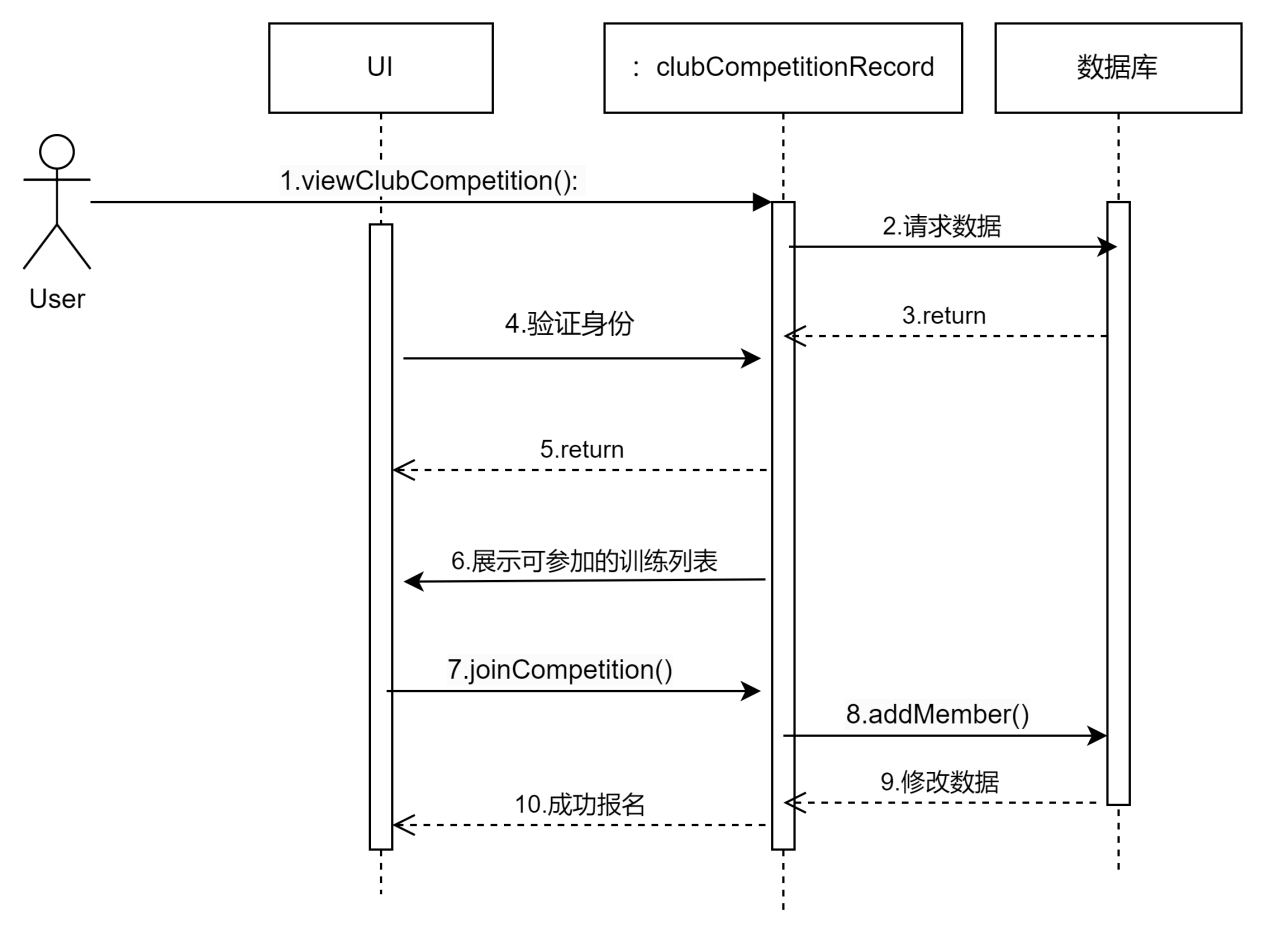
**2.4.3.3.1 比赛、训练报名**

**训练报名**



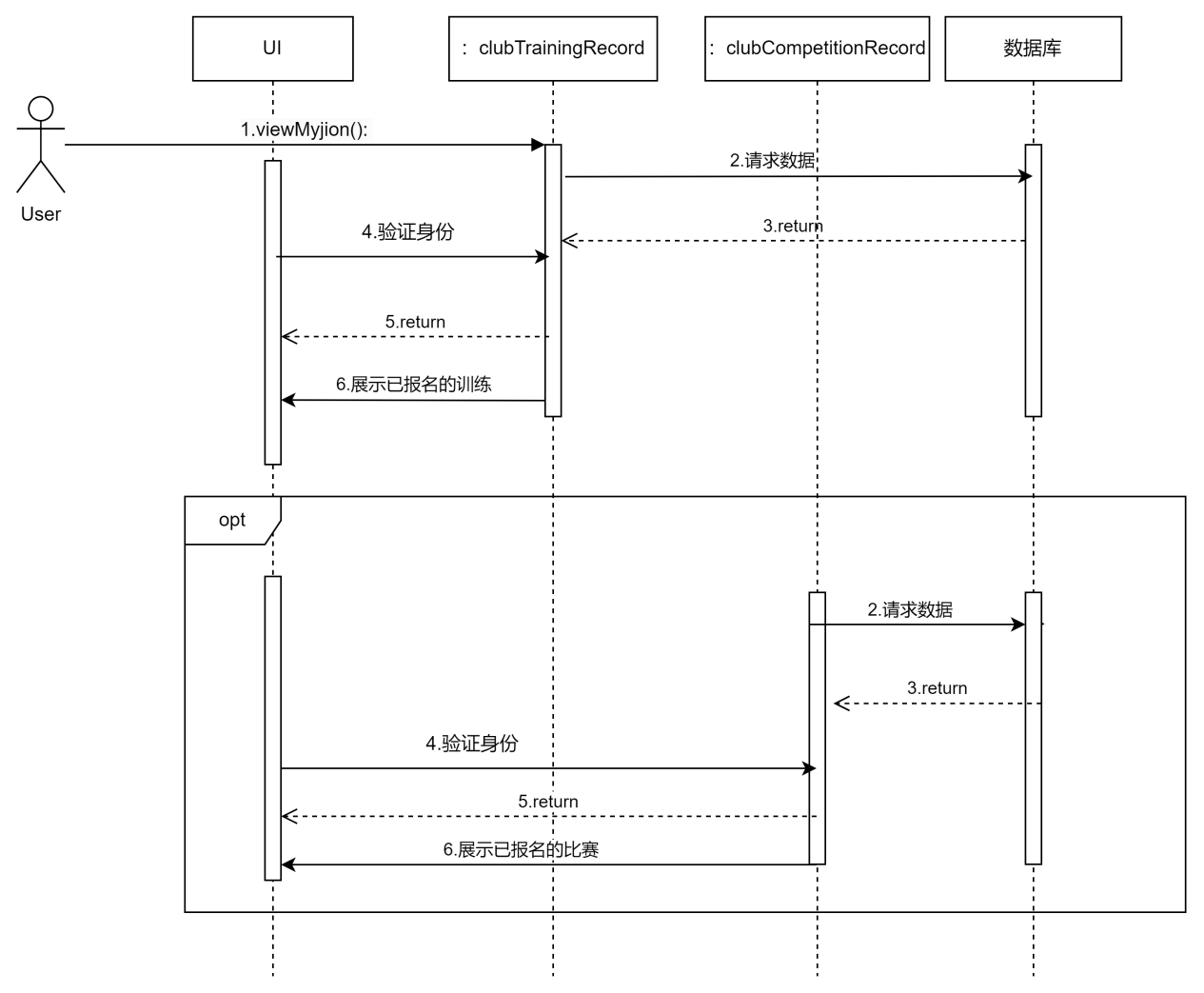
1. 用户通过 **UI** 调用 viewClubTraining() 方法，请求查看俱乐部的训练信息。
2. **clubTrainingRecord** 向 **数据库** 发送数据请求。
3. 数据库返回训练数据给 **clubTrainingRecord**。
4. **clubTrainingRecord** 将数据返回给 **UI**。
5. **UI** 展示用户可参与的训练列表。
6. 用户选择训练，调用 joinTraining() 方法报名。
7. **clubTrainingRecord** 调用 addMember() 方法，更新训练记录。
8. **clubTrainingRecord** 将更新信息同步到 **数据库**。
9. 数据库完成数据修改。

**比赛报名**



1. 用户通过 **UI** 调用 viewClubCompetition() 方法，请求查看俱乐部的训练信息。
2. **clubCompetitionRecord** 向 **数据库** 发送数据请求。
3. 数据库返回训练数据给 **clubCompetitionRecord**。
4. **clubCompetitionRecord** 将数据返回给 **UI**。
5. **UI** 展示用户可参与的训练列表。
6. 用户选择训练，调用 joinTraining() 方法报名。
7. **clubCompetitionRecord** 调用 addMember() 方法，更新训练记录。
8. **clubCompetitionRecord** 将更新信息同步到 **数据库**。
9. 数据库完成数据修改。

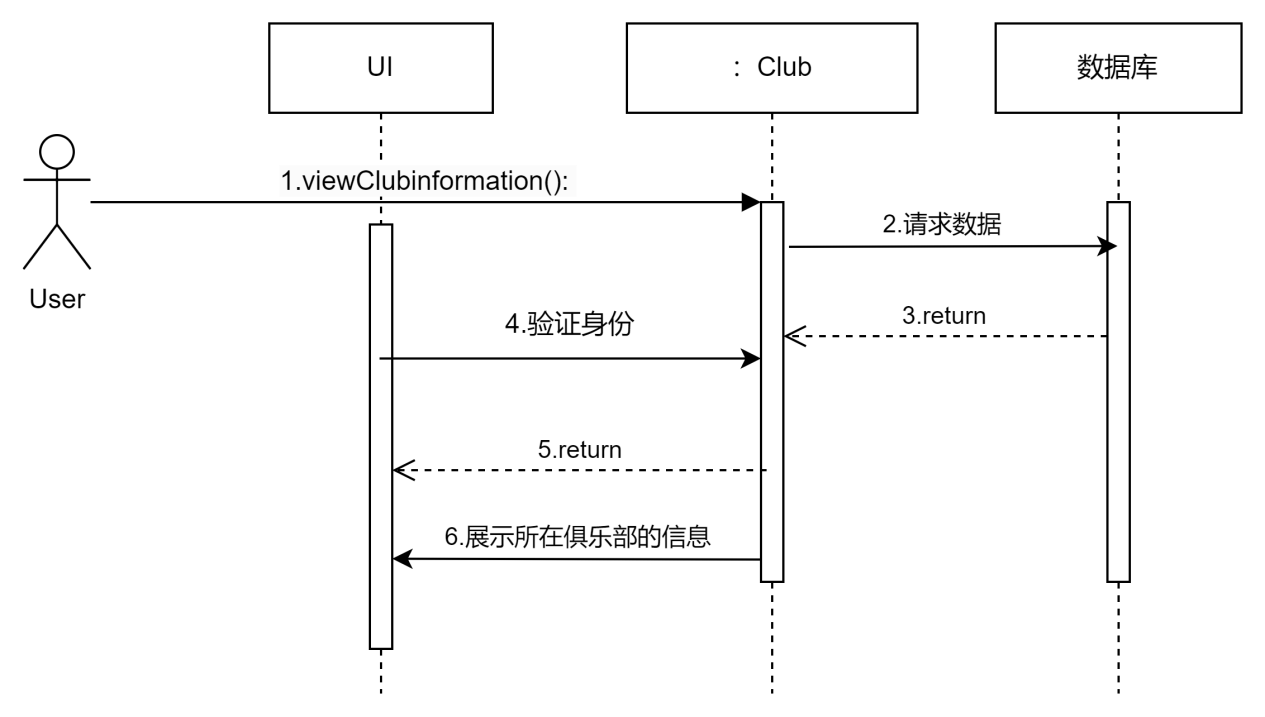
**查看我的报名**



1. 用户通过 **UI** 调用 viewMyjion() 方法，发起请求查看自己已报名的比赛和训练。
2. **clubCompetitionRecord或clubTrainingRecord** 接收到请求后，向 **数据库** 请求相应的数据。
3. 数据库返回查询到的报名数据给 **clubCompetitionRecord或clubTrainingRecord**。
4. **clubCompetitionRecord或clubTrainingRecord** 将整理后的数据返回给 **UI**。
5. **UI** 展示用户已报名的比赛和训练信息。

**2.4.3.3.2 查看俱乐部信息**

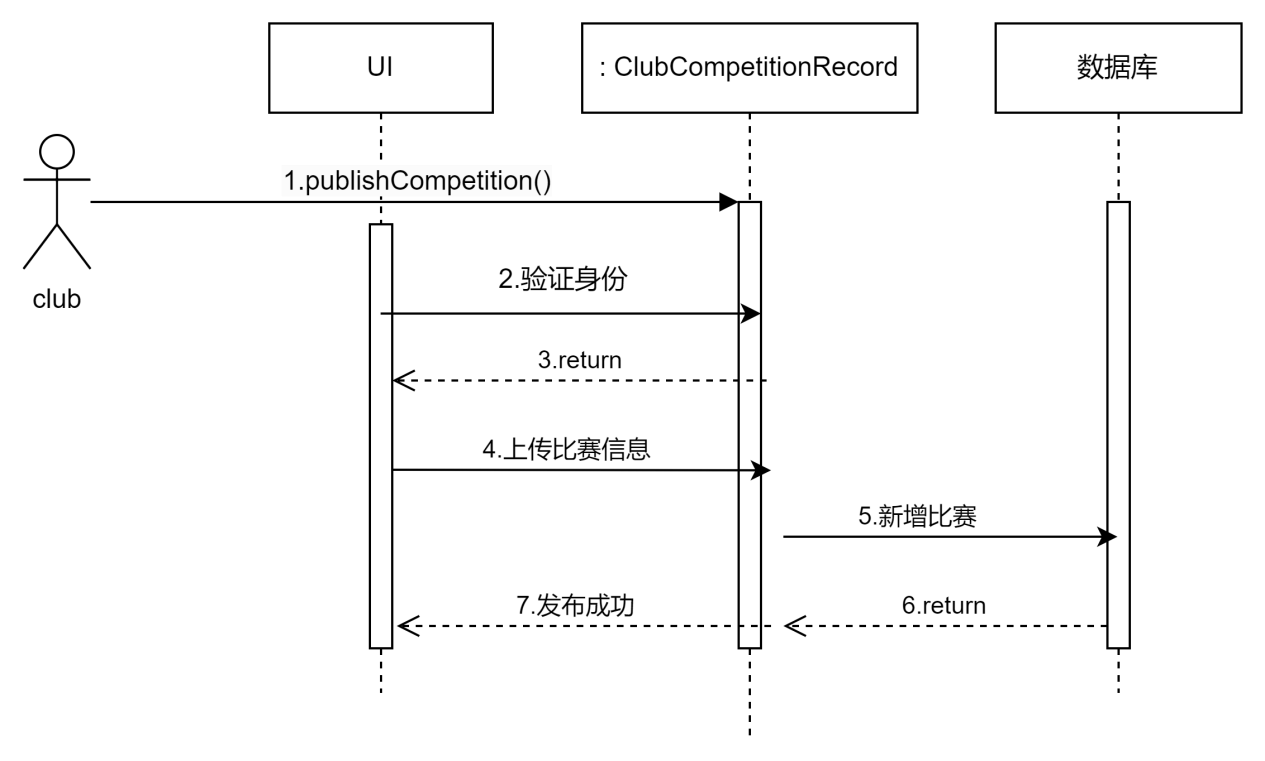
正常流



1. 用户通过 **UI** 调用 viewClubInformation() 方法，发起请求查看自己已报名的比赛和训练。
2. **ClubinformationService** 接收到请求后，向 **数据库** 请求相应的数据。
3. 数据库返回查询到的报名数据给 **ClubinformationService**。
4. **ClubinformationService** 将整理后的数据返回给 **UI**。
5. **UI** 展示用户已报名的比赛和训练信息。

**2.4.3.3.3 发布比赛信息**

正常流：

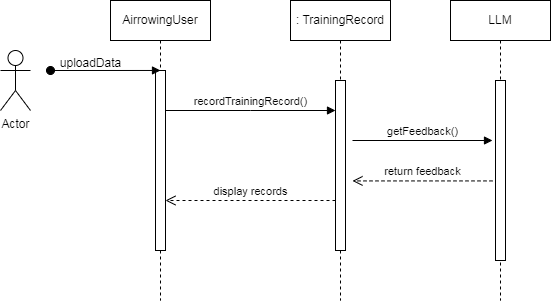


1. 用户通过 **UI** 调用 publishCompetition() 方法，发起发布赛事的请求。
2. **ClubCompetitionService** 接收请求后，进行相应的逻辑处理。
3. **ClubCompetitionService** 返回初步响应给 **UI**，表示可以上传赛事信息。
4. 用户通过 **UI** 上传具体的比赛信息。
5. **ClubCompetitionService** 将赛事信息新增到 **数据库**。
6. **数据库** 返回操作成功的确认信息给 **ClubCompetitionService**。
7. **ClubCompetitionService** 将发布成功的结果反馈给 **UI**，并提示用户发布成功。

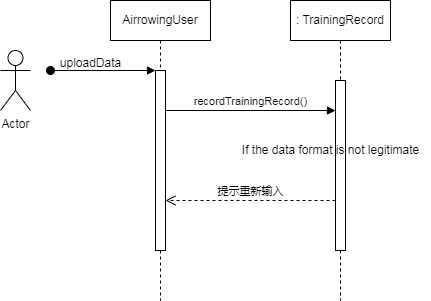
**2.4.3.4 训练分析模块**

**2.4.3.4.1 训练数据分析**

**正常流：**



**异常流：**



**参与者：**

* **AirrowingUser**：系统的终端用户，发起上传训练数据并获取分析建议的请求。
* **TrainingRecord**：负责存储训练数据和分析结果的业务逻辑实体。
* **LLM（大模型）**：负责进行训练数据分析并生成个性化建议的模型。

**流程描述：**

1. **正常流程（Normal Flow）**：

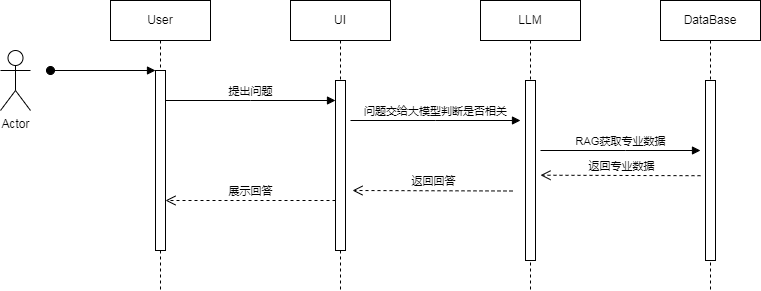
* AirrowingUser 提交训练数据（如训练时长、心率等）。
* 系统检查数据有效性：
* 如果数据有效，系统将数据提交给 LLM 进行分析。
* LLM 对训练数据进行分析，生成个性化的分析结果和建议。
* 系统将分析结果和建议存储为一个新的 TrainingRecord，并将其与 AirrowingUser 的训练历史相关联。
* 系统返回分析结果和个性化建议给 AirrowingUser，用户可以查看详细的训练数据和反馈。

1. **异常流程（Exception Flows）**：

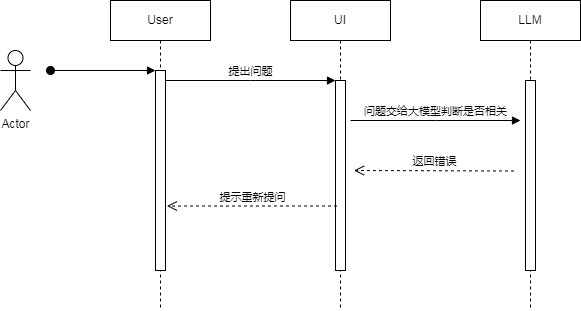
* **异常情况 1**：上传数据失败。
* 系统提示上传失败并要求用户重新上传数据。
* **异常情况 2**：数据格式无效。
* 系统提示数据格式错误并要求用户修改数据格式。

**2.4.3.4.2 智能指导**

**正常流：**



**异常流：**



**参与者：**

* **UI**：用户界面，负责接收用户输入并展示响应。
* **User**：系统的终端用户，发起查询并获取训练建议。
* **LLM（大模型）**：负责处理用户输入的问题，并生成专业的训练建议和回答。
* **Database**：存储和提供知识库及相关数据的数据库。

**流程描述：**

1. **正常流程（Normal Flow）**：

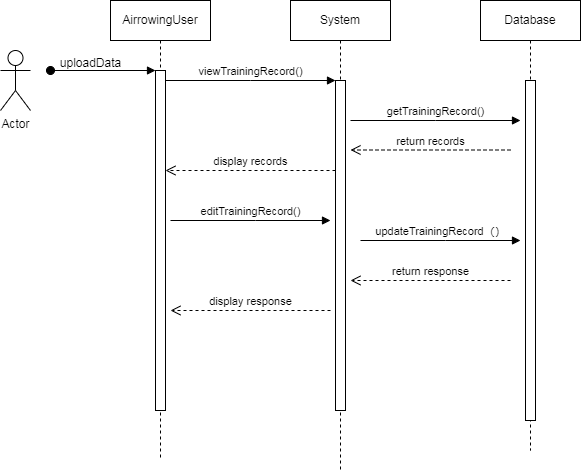
* **User** 通过 UI 在训练分析页面输入与训练相关的问题。
* **UI** 将问题传递给系统进行有效性检查，确保问题与训练主题相关。
* 如果问题有效，系统通过 RAG 技术向 **Database** 检索知识库中的相关信息。
* **Database** 返回相关的知识内容，系统将其传递给 **LLM**。
* **LLM** 根据知识库中的内容生成专业的回答，并提供相应的训练建议和指导。
* **UI** 展示 **LLM** 生成的答复和训练建议给 **User**。

1. **异常流程（Exception Flows）**：

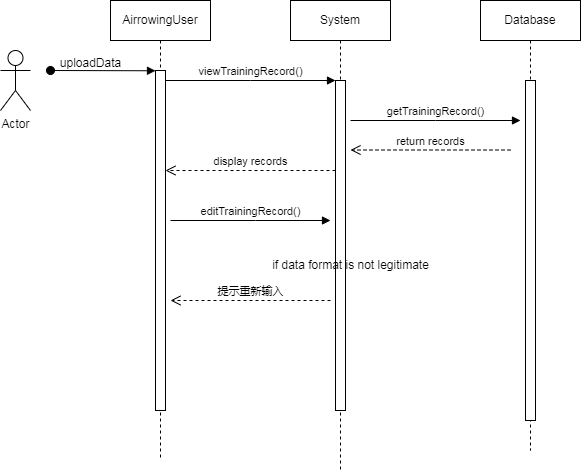
* **异常情况 1**：问题无效或与训练主题无关。
* 系统提示 **User** 调整问题并重新输入。

**2.4.3.4.5 查看与编辑历史训练记录**

**正常流：**



**异常流：**



**参与者：**

* **AirrowingUser**：系统的终端用户，发起查看和编辑历史训练记录的请求。
* **System**：负责管理和处理用户的训练记录，包括展示、编辑和保存操作。
* **Database**：存储和提供用户历史训练记录的数据库。

**流程描述：**

1. **正常流程（Normal Flow）**：

* **AirrowingUser** 进入训练记录管理页面，浏览自己历史上传的训练记录。
* **System** 展示所有历史记录供 **AirrowingUser** 浏览。
* **AirrowingUser** 选择一条记录进行查看。
* 如果 **AirrowingUser** 希望修改某条记录，点击“编辑”按钮。
* **System** 进入编辑模式，允许 **AirrowingUser** 修改相关训练数据（如训练时长、心率等）。
* **AirrowingUser** 在编辑模式下修改所需信息后，点击“保存”按钮。
* **System** 检查修改后的数据格式与内容的有效性：
* 如果数据有效，**System** 将修改后的信息保存至 **Database**。
* **System** 提示 **AirrowingUser** “更新成功”。

1. **异常流程（Exception Flows）**：

* **异常情况 1**：修改的数据无效（如格式错误）。
* **System** 提示 **AirrowingUser** 数据无效，并要求重新输入正确的数据。

3. **非功能需求**

**3.1 性能要求**

**3.1.1 精度**

* 数据输入输出精度：**小数点后 3 位**
* 数据传输精度：**小数点后 3 位**
* 计算精度：**小数点后 7 位**

|  |
| --- |
| **说明：**   * 输入输出精度主要针对系统对外提供的数据接口、报表以及前端呈现。 * 计算精度主要针对训练数据分析与姿态识别等复杂算法的内部处理过程。 |

**3.1.2 时间特性要求**

iteration1:（旧版）

* **响应时间：**
* 用户发起关键操作（如登录、打开训练分析页面）后，系统应在 3 秒内完成响应并呈现主要信息；
* 对于较大数据量的分析操作（如对长时间训练视频做姿态分析），应在合理硬件条件下尽可能控制在 5~10 秒内完成初步分析反馈。
* **更新处理时间：**
* 数据更新（如训练记录的新增、修改）应在 2 秒内完成并在前端界面同步显示；
* 后端大批量数据同步（如整合俱乐部统计报表）可在短暂后台任务中执行，一般不超过 30 秒。
* **数据转换和界面更新传输时间：**
* 当用户切换页面或执行需要动态刷新界面的操作时，界面更新应在 1~3 秒内完成（与网络带宽和服务器性能相关）。

iteration2:

* **高并发处理**：
* **读操作**：在高并发读场景下（如海量用户浏览笔记、点赞、收藏），系统应确保响应时间不超过 2 秒。
* **写操作**：在高并发写场景下（如用户发布笔记、点赞、关注），系统应确保响应时间不超过 3 秒，避免数据库和缓存的性能瓶颈。
* **分布式事务处理**：在跨服务操作（如用户关注与笔记发布）中，确保分布式事务的处理时间在 5 秒内完成，以保持数据一致性和系统响应速度。
* **消息队列处理时间**：RocketMQ 消息队列的处理延迟应控制在 500 毫秒以内，确保异步操作的实时性。

**3.1.3 输入输出要求**

* **输入数据类型：**

1. **用户注册、登录相关数据：**文本、JSON

* 媒体/格式：使用 HTTPS 传输的 JSON 或表单数据
* 数值范围与精度：用户名长度限制在 5~20 字符，密码长度限制至少 8 位；无特殊精度要求。

1. **训练数据：**数字、视频文件、图像文件

* 媒体/格式：支持常见视频格式（如 mp4、mov）与图像格式（jpg、png）；数值数据采用 JSON 结构传输
* 数值范围与精度：里程、心率、配速等数据小数点后 3 位；姿态识别算法内部处理精度达小数点后 7 位

1. **俱乐部信息数据：**文本、JSON
2. **社区帖子：**文本、图片、视频，存储方式同上

* **输出数据类型：**

1. **前端页面输出：**HTML、CSS、JavaScript 生成的可视化报表或图表（如训练数据趋势图、姿态对比图等）
2. **后端数据接口：**JSON 数据结构（对外或对移动端开放的 API）
3. **报表输出：**PDF/Excel/图形格式（训练周报、俱乐部成绩统计等）
4. **硬拷贝报告：**仅限管理员或俱乐部管理者可导出并打印，包括正常结果输出、状态输出及异常输出；报表内容主要含训练大数据分析结果、趋势图等

|  |
| --- |
| **举例：**  用户在训练分析界面提交姿态视频，系统会先返回“已接收”JSON 消息，并进入姿态识别算法处理。在处理完成后，通过 JSON 形式返回识别结果与建议，再在前端可视化显示。 |

**3.2 数据管理能力要求**

iteration1:（旧版）

1. **数据存储**

* 应支持高并发和海量数据存储（训练视频、历史训练记录、社区帖子等），并在负载增大时保持稳定性。
* 采用数据库（如 MySQL）+ 分布式文件系统（可选择对象存储）实现海量视频和图片的保存。

1. **数据访问**

* 后端需提供高效的数据检索服务（如索引、缓存）以保证训练数据、俱乐部统计数据的快速查询。
* 保证写入操作（新增训练记录、更新俱乐部信息等）在高并发下依旧迅速且数据不丢失。

1. **备份与恢复**

* 每日增量备份，保证训练和社区数据完备；
* 每周或每月进行全量备份，保存在异地；
* 在出现灾难性故障时，可通过备份在 2 小时内恢复关键业务数据。

|  |
| --- |
| **估算：**   * 用户规模（注册用户数）可能在 1k 到 5k 之间； * 视频资料按人均每周上传 2~3 个训练视频估算，1 年将达到十万级别视频存储量。 * 文字和社区帖子则相对数据量更大但单条占用存储小。 |

iteration2:

* **分布式架构支持**：
* **数据库分片**：采用 MySQL 分片策略，按用户 ID 或其他关键字段分片，确保数据库在海量数据下的高效读写。
* **分布式缓存**：使用 Redis 分布式集群，支持高并发读写和故障转移，确保缓存系统的高可用性。
* **数据一致性**：
* **强一致性与最终一致性**：关键数据（如用户关系、点赞记录）采用强一致性机制，其他数据（如笔记内容、评论）可采用最终一致性，平衡系统性能与数据一致性需求。
* **分布式锁**：在需要保证数据一致性的操作（如用户关注、点赞）中，采用分布式锁机制（如基于 Redis 的 Redisson）避免数据竞争和不一致。
* **数据冷热分层**：
* **冷热数据划分**：将访问频率高的热点数据（如热门笔记、活跃用户）存储在高速缓存中，冷数据则存储在分布式存储系统中，优化存储资源和访问性能。

**3.3 安全及保密性要求**

iteration1:（旧版）

* **用户身份验证：**
* 采用强密码策略，并结合 JWT 或 OAuth 进行认证。
* **数据加密：**
* 对敏感数据（如用户个人信息、隐私训练数据）进行加密存储；
* 传输中采用 HTTPS/TLS，避免明文中间人攻击。
* **访问控制：**
* 敏感操作（如删除训练记录、导出大量用户数据）仅向高级别权限用户开放。
* **用户训练数据隐私：**
* 训练数据仅所属俱乐部和用户本人可见；
* 允许用户自行设置是否对公众/社区部分公开个人训练结果。

iteration2:

* **访问控制：**
* **取消角色区分**，允许所有用户管理自己的俱乐部。
* 实现基于用户所有权的访问控制，确保用户只能访问和管理自己创建的俱乐部及相关数据。

**3.4. 灵活性要求**

iteration1:（旧版）

* **操作方式变化：**
* 前端 UI 需具备一定的可配置能力，使得不同俱乐部或不同类型用户可根据自身需求个性化配置。
* **运行环境变化：**
* 可在不同云平台、不同操作系统（Windows、Linux、Android）上部署和使用。
* **与其他软件接口变化：**
* 预留开放 API（RESTful 或 GraphQL），便于后续与第三方软件或设备对接（如可穿戴设备、智能桨等传感器）。
* **精度和有效时限变化：**
* 若后续对训练数据精度要求进一步提高，系统在架构层面可方便地进行算法替换或数据库字段扩容。
* **计划的变化或改进：**
* 如后续扩展到更大规模、更多项目类型（如龙舟），在系统模块化设计下可较容易地进行功能拓展。

|  |
| --- |
| **说明：** 对于在架构与设计层面为了应对上述灵活性而做出的特别设计（如插件式算法接口、可扩展的角色权限设计等），需要在详细设计文档中予以说明。 |

iteration2:

* **微服务扩展性**：
* **服务注册与发现**：利用 Nacos 进行服务注册与发现，支持动态扩展和缩减微服务实例，确保系统在高并发下的弹性扩展能力。
* **容器化部署**：支持使用 Docker 和 Kubernetes 进行容器化部署，简化微服务的管理和扩展。
* **持续集成与持续部署（CI/CD）**：
* **自动化部署**：建立 CI/CD 流水线，支持代码的自动化构建、测试和部署，提升开发效率和系统稳定性。
* **灰度发布与回滚**：支持灰度发布和快速回滚机制，确保新功能上线的平滑性和系统的持续可用性。

**3.5. 其他专门要求**

* **可维护性：**
* 采用模块化、可插拔式架构，保证后期功能升级与维护成本可控。
* **易用性：**
* 提供简洁直观的用户界面，减少用户的学习成本；
* 社区功能需便于发帖和检索，社交主页可快速查看他人动态或训练成果。
* **可靠性：**
* 关键功能（训练数据记录、俱乐部管理）出现异常时需及时报警并有快速恢复机制。
* **异常处理：**
* 需对用户输入异常、网络异常、数据格式错误等做充分的防范与提示。
* **运行环境可转换性：**
* 除 Windows 平台外，系统也能在常见 Linux 发行版、容器环境（Docker 等）中正常部署。

4. **运行环境规定**

**4.1. 设备**

* **处理器型号及内存容量：**
* 服务器建议使用至少 4 核以上 CPU、8GB 以上内存（根据用户规模可水平扩展或采用云服务）；
* 若需本地部署姿态识别算法并进行 GPU 加速，则至少需配备 NVIDIA GPU（如 Tesla 系列）或同级别的 AI 加速卡。
* **外存容量：**
* 由于视频存储规模大，需大容量硬盘或对象存储；建议启用云存储方案，初始至少 1TB 容量，后续按需扩容。
* **输入及输出设备：**
* 前端 PC 或移动端设备无特别限制，常见浏览器和智能手机即可。
* 教练/管理端可能需要高清显示器以查看视频细节和数据分析结果。
* **数据通信设备：**
* 无特殊要求，一般服务器与客户端均可通过互联网访问。若是内网部署，需要能访问外部网络以实现备份和社区互动。
* **功能键及其他专用硬件：**
* 无强制要求；可选配摄像头或智能传感器，以获取训练者的实时姿态数据。

**4.2. 支持软件**

* **操作系统平台：**
* 服务端：Windows 10 以上版本，或常见 Linux 发行版（Ubuntu, CentOS 等）；
* 移动端：Android、iOS；
* **数据库系统平台：**
* MySQL 5.7+/8.0+；如需进一步扩展大数据分析可考虑引入分布式数据库或数据仓库。
* **网络和硬件设备平台：**
* 无特殊网络需求，如带宽能支持视频上传/下载即可。
* **编译/开发/测试支持软件：**
* **Java 编译环境：** 使用 Java 作为主要编程语言，采用 JDK 11 或更高版本；
* **后端框架：** Spring Boot 或类似 Java 框架；
* **前端框架：** React、Vue.js 等现代前端框架；
* **开发工具：** 常用 IDE（如 IntelliJ IDEA）、构建工具（如 Maven、Gradle）、测试框架（如 JUnit）；
* **持续集成/持续部署（CI/CD）工具：** Jenkins, GitLab CI 等。

**4.3. 与其他系统的接口**

iteration1:

**姿态识别算法接口：**

* 提供标准化的 RESTful API 或 RPC 接口，用户上传训练视频时调用此接口返回姿态分析数据（如关节角度、身体位置、划桨频率等）。
* 保证算法接口的可扩展性，可集成新的姿态算法或其他运动识别算法。

iteration2:

**系统与大语言模型（LLM）的接口交互**

1. **接口背景与目的**

* **背景**  
   AirRowing 系统需要对用户上传的训练数据、赛艇姿态信息以及用户输入的问题进行智能分析与理解，并基于分析结果输出个性化的训练建议、效果评估和纠正意见。由于传统规则引擎或简单的算法模型无法满足复杂的语义理解与多模态分析需求，因此引入大语言模型（LLM）作为外部服务。
* **目的**
* 利用大语言模型强大的自然语言处理与多模态分析能力，实现对赛艇训练数据的综合解读。
* 在用户对训练问题、训练记录等提出疑问或希望获得指导时，大语言模型可根据上下文理解并反馈专业建议。
* 针对用户上传的赛艇姿态图片或视频，LLM 与图像/视频处理算法配合，给出更精准的姿态纠正和优化思路。

1. **接口形式**

* **接口协议**  
   系统与大语言模型之间通常通过 **RESTful API** 或 **RPC（如 gRPC）** 的方式进行通信。根据具体实现，可能使用 HTTPS 保证传输安全性，也可在内网环境下结合微服务网格进行安全管控。
* **数据格式**
* **请求数据** 通常采用 **JSON** 格式，包含训练数据、图像/视频资源链接或姿态关键点，以及用户的问题描述等。
* **响应数据** 同样为 **JSON** 格式，返回具体的分析结果、评估反馈或纠正建议。

1. **数据交互流程**
2. **请求阶段**

* **训练数据**：AirRowing 系统会将用户的训练记录（如距离、时长、心率、功率等）打包成 JSON，并附带用户 ID、训练场景描述等上下文信息后，通过接口请求发送给大语言模型服务。
* **姿态信息**：针对赛艇姿态分析，系统会先将用户上传的图片/视频进行预处理（如生成关键点数据），再将关键点数据（或文件链接）与用户场景描述一起发送给 LLM。
* **自然语言问题**：当用户提问（如“如何提高划船效率？”、“这个月训练效果如何？”等），系统会将问题文本连同用户的历史训练数据上下文一并发送给 LLM，以便模型进行语义理解与回答。

1. **大语言模型处理阶段**

* **多模态或语言理解**：LLM 接收到系统发来的训练数据、姿态数据或自然语言问题后，进行多模态或语义解析。
* **知识检索/上下文对齐**：若采用 RAG（Retrieval-Augmented Generation）等技术，则在此阶段会先进行知识库检索或上下文对齐，确保 LLM 的回答或评估基于已有的专业知识（如赛艇运动原理、人体力学等）。
* **模型推理**：模型进行推理生成答案或分析结果，可能包括用户训练不足点、建议的训练强度、姿态矫正的动作要点、个性化计划等。

1. **响应阶段**

* **结果返回**：LLM 处理完成后，会将分析结果打包成 JSON 格式返回给 AirRowing 系统。
* 对于训练数据分析，返回内容包括训练效果评估、关键不足点以及改进建议。
* 对于姿态数据分析，返回关节角度、错误动作说明及纠正提示。
* 对于智能问答，返回对用户问题的文本化回答或可视化建议。
* **系统整合**：AirRowing 系统接收响应后，进一步将该结果整合到用户界面、训练方案或社区帖子中，供用户查看和互动。

1. **主要接口字段示例**

以下为通过 RESTful 方式与大语言模型进行通信时，**请求** 与 **响应** 数据的示例结构：

1. **请求数据示例**

|  |
| --- |
| JSON {  "userId": "user123",  "context": {  "historyRecords": [  {"date": "2024-06-10", "distance": 5000, "time": 30, "power": 200},  {"date": "2024-06-15", "distance": 6000, "time": 35, "power": 210}  ],  "question": "我想提高划船效率，应该加强哪些方面的训练？"  },  "poseData": {  "fileUrl": "https://oss.example.com/user123/video\_001.mp4",  "keypoints": [ ... ]  } } |

1. **响应数据示例**

|  |
| --- |
| JSON {  "analysisResult": {  "trainingEvaluation": {  "strengths": ["有稳定的训练频率", "配合度较高"],  "weaknesses": ["核心力量不足", "腿部发力不均匀"]  },  "poseCorrection": {  "advice": "收腿时背部保持平直，躯干不应过度前倾。",  "angles": {  "hipAngle": 45,  "kneeAngle": 90  }  },  "recommendation": "建议增加核心稳定性训练，提高腿部推蹬力量。每周可安排2次高强度间歇训练。"  },  "answer": "可适当增加核心训练，如平板支撑和腿部爆发力练习，以提升划船效率。" } |

* **请求参数（简述）**：
* userId：用户标识
* context：包含训练历史记录和用户提问文本，用于提供模型上下文
* poseData：可选，用于姿态分析，包含多媒体文件链接或关键点数据
* **响应参数（简述）**：
* analysisResult.trainingEvaluation：对训练数据的定性评估
* analysisResult.poseCorrection：对姿态的动作纠正建议
* analysisResult.recommendation：对整体训练的个性化指导
* answer：对自然语言问题的直接回答

1. **安全与鉴权**

* **身份验证**：  
   系统调用大语言模型时，需要确保已在后台完成用户登录及权限验证。若大语言模型托管在第三方云服务上，则系统可能需要携带相应的 API Key 或访问令牌，避免滥用或无授权调用。
* **数据脱敏与隐私保护**：  
   在传输用户数据给大语言模型时，应尽可能对敏感信息进行脱敏处理，如屏蔽用户名真实信息或隐私字段，确保用户隐私。
* **流量控制与超时处理**：  
   对模型接口调用要设置合理的超时时间和重试机制，防止因模型计算时间较长或网络波动导致系统阻塞或崩溃。

1. **整体作用与价值**

* **精准指导**：通过大语言模型进行多模态或语义分析，可提供更精准、更具上下文关联度的训练和姿态指导。
* **用户体验提升**：用户提问可得到自然语言的详细回答，或获得针对自身训练历史的个性化方案，增强系统的交互感与服务深度。
* **可扩展性**：随着大语言模型能力的升级或知识库的扩充，系统也能不断更新完善，从而满足更广泛和深入的赛艇训练场景需求。

**4.4. 控制**

* **用户权限管理：**
* 采用可视化权限配置或静态配置文件方式，管理员可在管理后台为不同角色配置权限。
* **审计日志：**
* 系统应记录用户关键操作（登录、修改数据、删除数据等），包括时间、IP、操作类型、操作对象等信息；
* 日志保留期限不低于 6 个月，以便追溯问题。

5. **需求跟踪管理**

* **需求记录与追踪：**
* 使用专业需求管理工具（如 Jira、Azure DevOps等）记录和追踪所有需求；
* 在需求变更时，保持需求文档同步更新，并在对应小节中增加 iteration2, iteration3 等标记。
* **定期评估：**
* 每个阶段或每个 Sprint 结束后，由项目团队与相关干系人评估需求的实现情况，确保软件功能和性能与用户需求及项目目标一致。
* **文档模板：**
* 以 ProjectEverySprintEstimation.xls、ProjectStoryBacklog.xls、SprintBacklog.xls 三个文档模板为依据进行需求的估算、拆分与跟踪；
* 在需求管理系统或文档中使用超链接形式关联到上述 Excel 模板，以便快速查看。