《小福同学设计说明书》

目录

一、引言

（一）目的

本设计说明书旨在详细描述 AI 对话软件的设计架构、功能模块、流程和数据结构，为软件的开发、测试和维护提供全面的技术指导。

（二）背景

随着人工智能技术的发展，模拟角色和场景的对话软件越来越受到用户的欢迎。本软件旨在通过自然语言处理技术，让用户能够与 AI 角色进行逼真的对话，并通过评分和成就系统增加用户的参与度和乐趣。

二、系统总体设计

（一）软件架构

1. 分层架构
    1. **表现层**：包括用户界面，负责用户与软件的交互，如注册登录界面、对话选择界面、对话展示界面和成就排名查看界面等。
    2. **应用逻辑层**：包含 AI 角色模块（小福同学）和后端模块，处理业务逻辑，如对话流程控制、分值计算和数据处理等。

3. **数据访问层**：主要是数据库模块，负责数据的存储、查询和更新。

2. **模块间通信**

    1. 表现层通过调用应用逻辑层的接口来实现用户操作，如选择对话场景、提交用户对话等。
    2. 应用逻辑层与数据访问层通过数据库操作语句进行数据交互，如写入用户信息、查询分值排名等。

（二）技术选型

1. **编程语言**：选择 Python 作为主要编程语言，因其在自然语言处理和机器学习领域有丰富的库和框架支持。
2. **AI 框架**：采用 TensorFlow 或 PyTorch 等深度学习框架来实现 AI 角色的对话能力，利用预训练模型进行自然语言生成和理解。
3. **数据库**：使用 MySQL 或 SQLite 等关系型数据库来存储用户数据、对话记录和成就信息，确保数据的持久化和一致性。

三、功能模块设计

（一）用户模块

1. **注册与登录**

1. **功能描述**：用户通过输入用户名、密码和确认密码进行注册，注册成功后可使用用户名和密码登录系统。
2. **界面设计**：注册界面包含输入框、确认按钮和提示信息；登录界面有用户名和密码输入框以及登录按钮。
3. **数据存储**：用户注册信息存储在 User 表中，包括 user_id、username 和 password 字段。

2. **账号管理**

1. **功能描述**：用户可以修改密码、找回密码和更新个人信息。
2. **界面设计**：在用户主界面设置账号管理入口，进入后有修改密码、找回密码和更新信息的操作界面。
3. **数据操作**：修改密码时需验证旧密码，并更新 User 表中的 password 字段；找回密码可通过邮箱或密保问题实现。

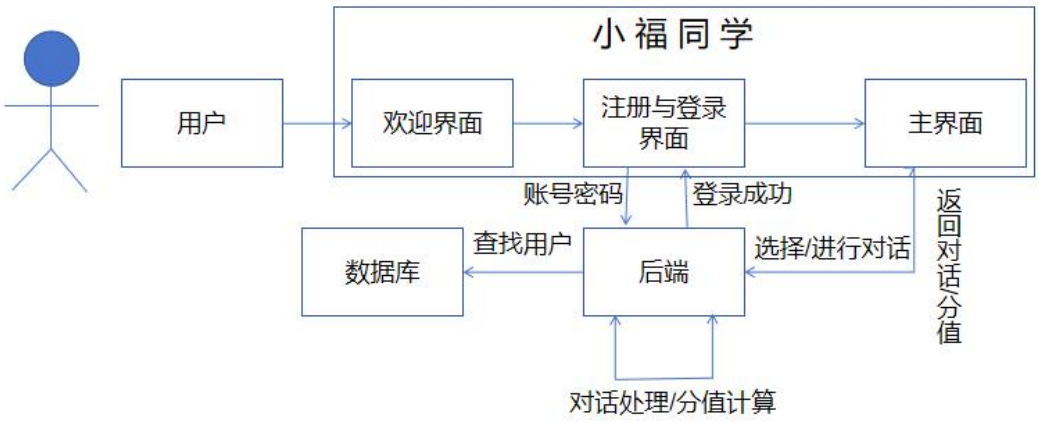

图 1-uml 协作图

（二）AI 角色模块（小福同学）

1. **对话场景选择**

    1. **功能描述**：用户在主界面可以从预设的多个对话场景中选择一个，AI 角色根据所选场景进行对话。
    2. **界面设计**：在主界面展示场景列表，用户点击选择，场景包括但不限于角色扮演、知识问答等。
    3. **数据传递**：将用户选择的场景信息传递给后端模块，触发对话流程。

2. **对话处理**

    1. **功能描述**：AI 角色根据场景发出开场白，接收用户对话内容，做出回应，并在对话结束时给出评价和建议。
    2. **对话逻辑**：利用自然语言处理技术，分析用户输入，生成合适的回复。回复的生成可以基于预训练的语言模型和对话策略。
    3. **分值计算**：根据预设的评分规则，判断用户发言的质量，给出 score 值，并更新 total_score。例如，语法正确、语义合理的发言得分较高。
    4. **评价与建议**：对话结束后，根据用户的总分和对话表现，给出针对性的评价和改进建议。

（三）后端模块

1. **请求处理**
    1. **功能描述**：接收来自用户和 AI 角色的请求，如用户的场景选择、AI 角色的分值查询等，并进行处理。
    2. **请求路由**：根据请求类型，将请求分发给相应的处理函数，例如将用户的场景选择请求转发给对话处理模块。
2. **数据处理与存储**
    1. **功能描述**：在对话过程中，记录相关数据，如对话开始时间、用户得分等，并将数据存储到数据库中。
    2. **数据记录**：在对话开始时，将用户选择的场景、开始时间等信息写入数据库；在对话过程中，实时更新用户的分值等数据。
    3. **数据查询与更新**：根据业务需求，查询数据库中的数据，如查询用户的最终分值排名，更新用户的成就状态等。

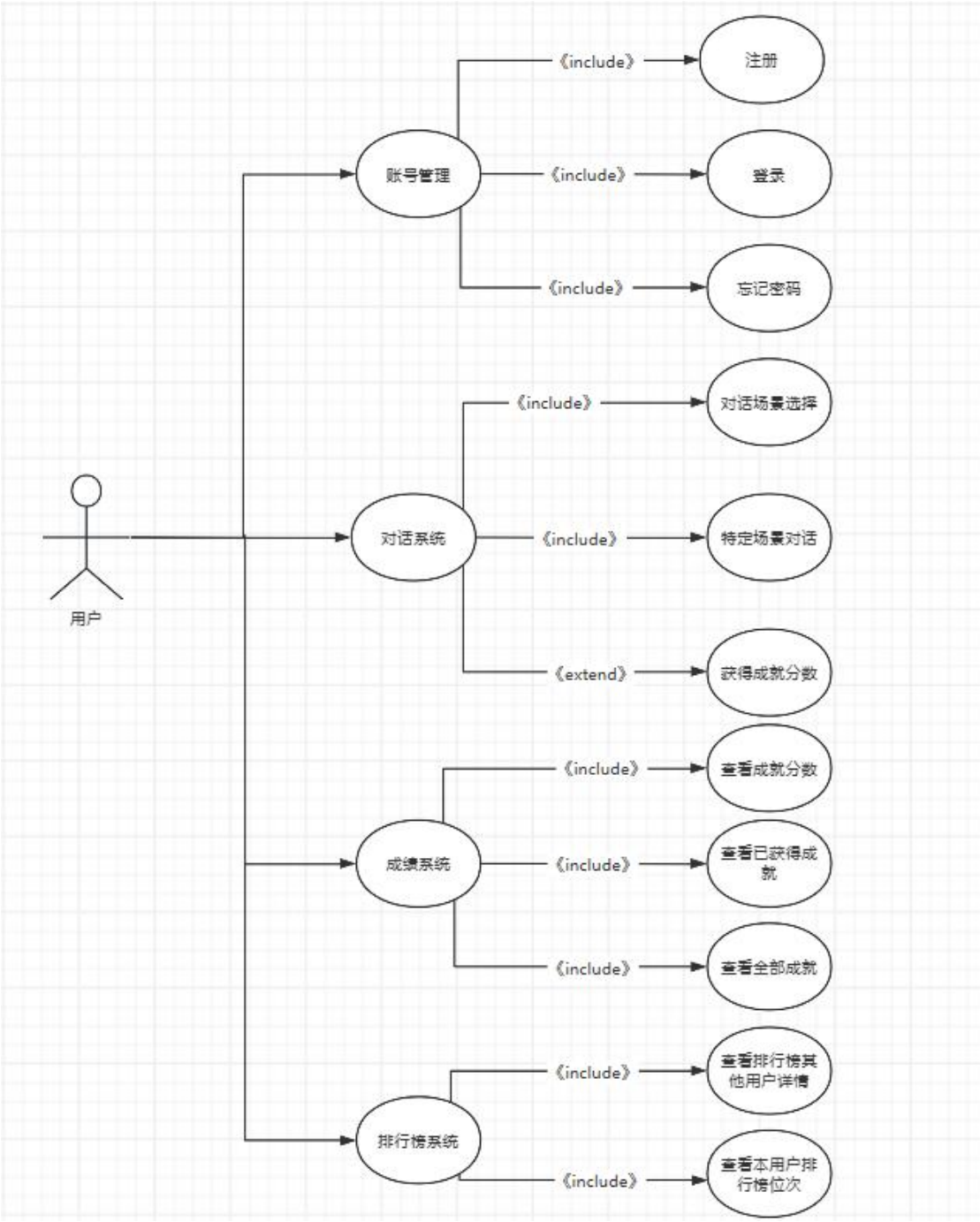

图 2-uml 用例图

（四）数据库模块

1. 表结构设计

1. **Ranking** 表

1. **ranking_id**：排名 ID，自增整数，作为主键。
2. **user_id**：用户 ID，与 User 表关联，用于确定排名对应的用户。
3. **user_score**：用户得分，整数类型，记录用户在对话中的总得分。
4. **game_project**：游戏项目，字符串类型，标识对话场景或项目名称。

2. **User 表**
    1. **user_id**：用户 ID，自增整数，主键。
    2. **username**：用户名，字符串类型，唯一标识用户。
    3. **password**：密码，字符串类型，存储加密后的用户密码。

3. **Achievement 表**
    1. **achievement_id**：成就 ID，自增整数，主键。
    2. **achievement_name**：成就名称，字符串类型，如“对话达人”等。
    3. **achievement_description**：成就描述，字符串类型，详细说明成就的获取条件。
    4. **achievement_status**：成就状态，字符串类型，如“已解锁”或“未解锁”，表示用户是否获得该成就。

5. **user_id**：用户 ID，与 User 表关联，确定成就所属用户。

4. **Access 表**

    1. **access_id**：访问 ID，自增整数，主键。
    2. **user_id**：用户 ID，与 User 表关联，确定访问记录所属用户。
    3. **timestamp**：时间戳，字符串类型，记录用户的访问时间。

2. **数据库操作**

    1. **数据插入**：在用户注册、对话开始和成就解锁等场景下，向相应表中插入数据。
    2. **数据查询**：根据用户 ID、场景名称等条件查询用户得分、成就状态和访问历史等数据。
    3. **数据更新**：在对话过程中更新用户得分，在成就解锁时更新成就状态等。

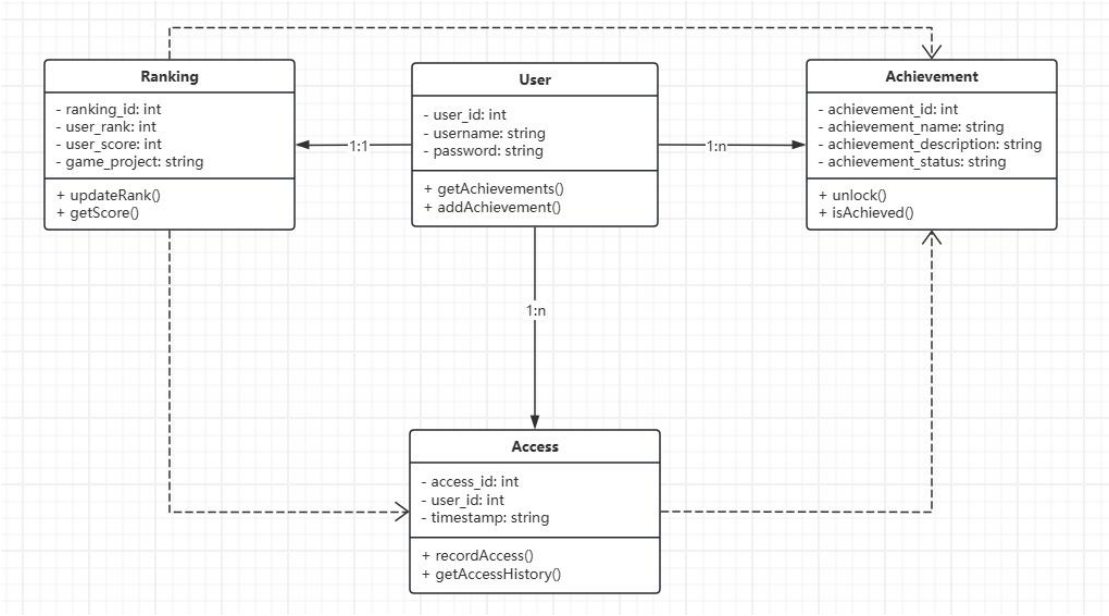

图 3-uml 类图

四、流程设计

（一）用户登录与场景选择流程

1. 用户打开软件，进入欢迎界面。
2. 用户选择登录或注册，如果选择注册，则输入用户名、密码进行注册；如果选择登录，则输入用户名和密码进行登录。
3. 登录成功后，用户进入主界面，展示对话场景列表。
4. 用户选择想要模拟的对话场景。

（二）对话流程

1. 系统初始化对话得分：total_score=n（n 为初始分数，可设为 0）。
2. AI 角色（小福同学）根据所选场景发出开场白。
3. 系统获取用户的对话输入。

4. 系统对用户发言进行分析，判断好坏程度，给出 score 值。
5. 更新 total_score：total_score = total_score + score。
6. AI 角色根据用户发言做出回复。
7. 判断 total_score 是否达到预设标准：
    1. 如果分数过低，AI 角色给出提示，对话结束，结算此次对话数据。
    2. 如果分数达标，继续对话，重复步骤 3 - 7，直到对话自然结束。
8. 对话结束后，AI 角色给出评价和建议，结算此次对话，包括更新用户得分、判断成就解锁等。

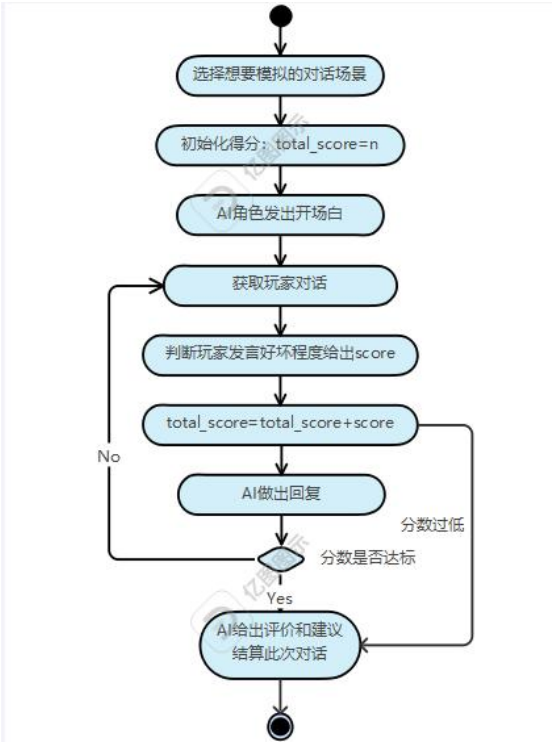

图 4-uml 时序图

（三）数据操作流程

1. 对话开始时
    1. 后端将用户选择的场景、对话开始时间等信息写入数据库的相关表中。
    2. 将用户的初始得分等数据记录到数据库。
2. 对话过程中
    1. 后端根据 AI 角色的请求，查询当前用户得分等数据，并返回给 AI 角色。
    2. 根据用户的对话表现，实时更新用户得分，并将更新后的数据写入数据库。
3. 对话结束后
    1. 后端将此次对话的最终得分、评价和建议等数据写入数据库。
    2. 根据用户的最终得分和表现，判断是否解锁成就，更新 Achievement 表中的成就状态。
    3. 查询用户的最终分值排名，并在界面上展示给用户。

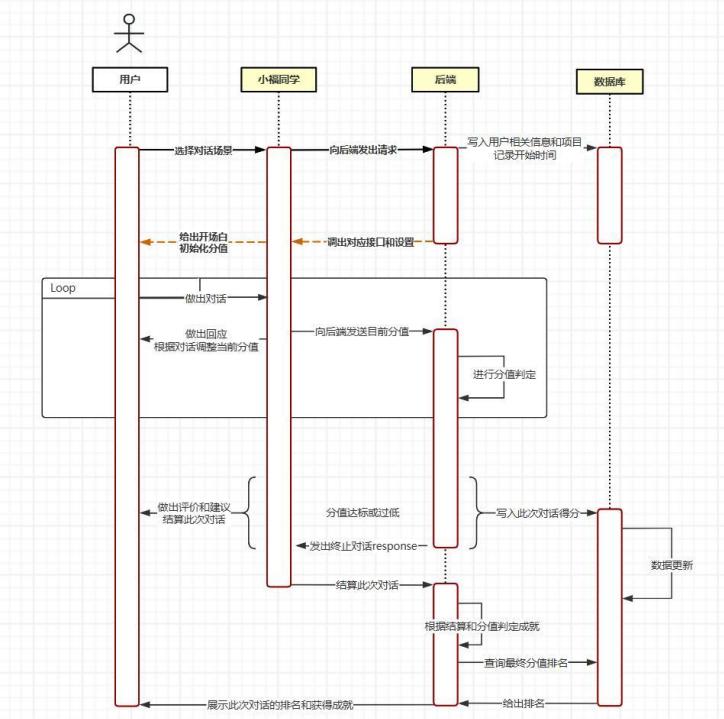

图 5-uml 活动图

五、系统性能与安全

（一）性能优化

1. 缓存机制

1. 在后端模块中，对于频繁查询的数据，如对话场景列表、用户基本信息等，采用缓存技术，减少数据库查询次数，提高系统响应速度。

2. 数据库优化

1. 对数据库表进行合理的索引设计，加速数据查询操作。例如，在 Ranking 表中对 user_id 和 user_score 字段建立索引，便于快速查询用户得分和排名。

2. 定期对数据库进行优化操作，如清理过期的对话记录和访问历史数据，减少数据库存储空间占用，提高查询效率。

（二）安全措施

1. **用户密码安全**
    1. 在用户注册和登录过程中，对用户密码进行加密处理，采用哈希算法（如 SHA - 256）将密码转换为不可逆的哈希值后存储在数据库中，防止密码泄露。
2. **输入验证**
    1. 在用户输入界面，对用户输入的内容进行合法性验证，如用户名长度限制、密码复杂度要求等，防止恶意用户输入非法数据导致系统故障或安全漏洞。
3. **数据访问控制**
    1. 在数据库操作中，对不同用户角色设置不同的访问权限。例如，普通用户只能访问和修改自己的数据，管理员可以对所有用户数据进行管理操作，确保数据的安全性和隐私性。

六、系统测试

（一）测试策略

1. **单元测试**
    1. 对各个功能模块进行单元测试，如测试用户注册登录功能、AI 角色的对话生成功能、后端的数据处理功能和数据库的操作功能等。
    2. 使用测试框架（如 Python 中的 unittest）编写测试用例，确保每个模块的功能正确性。
2. **集成测试**
    1. 在单元测试通过后，进行集成测试，测试各个模块之间的接口和数据交互是否正确。
    2. 模拟用户操作流程，从用户登录、选择场景、进行对话到查看成就排名，检查整个系统的集成功能是否正常。
3. **性能测试**
    1. 使用性能测试工具（如 JMeter）对系统进行性能测试，模拟多用户并发访问场景，测试系统的响应时间、吞吐量和资源利用率等性能指标。
    2. 根据性能测试结果，对系统进行优化，确保系统在高并发情况下能够稳定运行。

（二）测试用例示例

1. **用户注册测试用例**

1. **测试目标**：验证用户注册功能的正确性。
2. **输入数据**：用户名（testuser）、密码（testpassword）。
3. **预期结果**：注册成功，数据库 User 表中新增一条记录，用户名和密码正确存储。

2. **对话场景选择测试用例**

1. **测试目标**：检查用户能否正确选择对话场景并触发对话。
2. **输入数据**：用户选择“角色扮演”场景。
3. **预期结果**：AI 角色根据“角色扮演”场景发出开场白，对话流程正常启动。

七、总结

本设计说明书详细阐述了 AI 对话软件的架构、功能模块、流程和数据结构等方面的设计。通过合理的架构设计和功能实现，该软件将能够为用户提供丰富、有趣的对话体验，并通过评分和成就系统激励用户的参与。在开发过程中，需注重系统的性能优化和安全保障，确保软件的稳定运行和用户数据的安全。