第一周开发总结

在本周的开发过程中，我们主要实现了用户信息传递功能模块和用户问答模块以及用户关系模块

消息模块开发总结

在本周的开发过程中，我们实现了消息模块的API接口，主要包括获取会话列表、获取会话内容、发送消息、撤回消息、更新消息状态为已读、以及接收消息等功能。以下是详细的接口实现总结：

1. 获取会话列表

- **类名：** `UserMessagesController`
- **方法：** `GetConversations`
- **路由名称：** `/api/conversations`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `current_user_id` (Int): 当前用户ID
- **功能：** 获取当前用户的所有会话列表，包括会话ID、对方用户名、头像、最后一条消息内容及时间、未读消息数量等信息。

2. 获取会话内容

- **类名：** `UserMessagesController`
- **方法：** `GetConversationMessages`
- **路由名称：** `/api/conversations/{conversation_id}`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `conversation_id` (Int): 会话对象ID
  - `current_user_id` (Int): 当前用户ID
- **功能：** 获取指定会话的详细内容，包括所有消息列表（按时间排序）及每条消息的详细信息（如发送者、内容、时间、类型等）。

3. 发送消息

- **类名：** `UserMessagesController`
- **方法：** `SendMessage`
- **路由名称：** `/api/conversations/messages`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `conversation_id` (Int): 会话对象ID
  - `content` (String): 消息内容
  - `type` (String): 消息类型（如文本、图片等）
  - `current_user_id` (Int): 当前用户ID
  - `time` (DateTime): 发送时间
- **功能：** 向指定会话发送一条消息，并使用 `WebSocketService` 发送到指定客户端，同时在数据库中记录这条消息。

4. 撤回消息

- **类名：** `UserMessagesController`
- **方法：** `RetractMessage`
- **路由名称：** `/api/messages/{message_id}/retract`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `message_id` (Int): 消息ID
  - `current_user_id` (Int): 当前用户ID
  - `time` (DateTime): 撤回时间
- **功能：** 撤回一条已发送的消息，并更新数据库中的消息状态。同时使用 `WebSocketService` 通知指定客户端消息撤回。

5. 更新会话消息状态为已读

- **类名：** `UserMessagesController`
- **方法：** `UpdateConversationMessagesReadStatus`
- **路由名称：** `/api/conversations/{conversation_id}/update_read_status`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `conversation_id` (Int): 会话对象ID
  - `current_user_id` (Int): 当前用户ID
- **功能：** 将指定会话中的所有消息状态更新为已读。

6. 接收消息并更新阅读状态

- **类名：** `UserMessagesController`
- **方法：** `ReceiveMessage`
- **路由名称：** `/api/messages/receive`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `message_id` (Int): 消息ID
  - `receiver_in_window` (Boolean): 接收方是否在会话界面中
  - `sender_user_id` (Int): 发送方用户ID
- **功能：** 处理前端接收到的消息，更新消息状态为已读，并返回发送方的用户信息。

用户关系模块开发总结

在本周的开发过程中，我们实现了用户关系模块的API接口，主要包括管理用户关注关系、查询关注列表和粉丝列表等功能。以下是详细的接口实现总结：

1. 管理用户关注关系

- **类名：** `UserPostsController`
- **方法：** `ManageFollow`
- **路由名称：** `/api/user/follow`
- **HTTP 方法：** POST
- **传入参数及类型：**

    - `user_id` (Int): 被关注或取消关注的用户ID
    - `action` (String): 操作类型（"follow" 或 "unfollow"）
    - `current_user_id` (Int): 当前用户ID
- **功能：** 管理用户的关注关系，包括关注和取消关注。

2. 查询特定用户的关注列表

- **类名：** `UserPostsController`
- **方法：** `GetFollowingList`
- **路由名称：** `/api/user/following`
- **HTTP 方法：** POST
- **传入参数及类型：**
    - `user_id` (Int): 当前用户ID
- **功能：** 查询并返回特定用户的关注列表。

3. 查询对指定用户的关注状态

- **类名：** `UserPostsController`
- **方法：** `GetFollowStatus`
- **路由名称：** `/api/user/follow/status`
- **HTTP 方法：** POST
- **传入参数及类型：**
    - `target_id` (Int): 目标用户ID
    - `current_user_id` (Int): 当前用户ID
- **功能：** 获取当前用户对特定用户的关注状态。

4. 查询特定用户的粉丝列表

- **类名：** `UserPostsController`
- **方法：** `GetFollowersList`
- **路由名称：** `/api/user/followers`
- **HTTP 方法：** POST
- **传入参数及类型：**
    - `user_id` (Int): 当前用户ID
- **功能：** 查询并返回特定用户的粉丝列表。

用户问答模块开发总结

在本周的开发过程中，我们实现了用户问答模块的API接口，主要包括获取问题列表及回答列表、发布问题、撤回问题、回答问题、撤回回答等功能。以下是详细的接口实现总结：

1. 获取问题列表和回答列表

- **类名：** `QuestionAnswerController`
- **方法：** `GetQuestions`
- **路由名称：** `/api/questions`
- **HTTP 方法：** POST
- **传入参数及类型：**
    - `item_id` (String): 项目ID

- **功能：** 获取当前项目的所有问题列表，包括问题ID、提问者用户名、提问者头像、问题内容、提问时间、回答列表（包含回答ID、回答者用户名、回答者头像、回答内容及回答时间）等信息。

2. 获取指定问题的回答列表

- **类名：** `QuestionAnswerController`
- **方法：** `GetQuestionAnswers`
- **路由名称：** `/api/questions/{question_id}/answers`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `question_id` (Int): 问题ID
- **功能：** 获取指定问题的所有回答列表，包括回答ID、回答者用户名、回答者头像、回答内容及回答时间等信息。

3. 发布问题

- **类名：** `QuestionAnswerController`
- **方法：** `PostQuestion`
- **路由名称：** `/api/post_questions`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `content` (String): 问题内容
  - `item_id` (String): 项目ID
  - `current_user_id` (Int): 当前用户ID
  - `time` (DateTime): 提问时间
- **功能：** 向指定项目发布一个问题，并在数据库中记录提问信息。

4. 撤回问题

- **类名：** `QuestionAnswerController`
- **方法：** `WithdrawQuestion`
- **路由名称：** `/api/questions_retract/{question_id}`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `question_id` (Int): 问题ID
  - `current_user_id` (Int): 当前用户ID
- **功能：** 撤回指定问题，并在数据库中更新状态。

5. 回答问题

- **类名：** `QuestionAnswerController`
- **方法：** `PostAnswer`
- **路由名称：** `/api/questions/{question_id}/post_answers`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `question_id` (Int): 问题ID
  - `content` (String): 回答内容
  - `current_user_id` (Int): 当前用户ID

    - `time` (DateTime): 回答时间
- **功能：** 为指定问题发布一个回答，并在数据库中记录回答信息。

6. 撤回回答

- **类名：** `QuestionAnswerController`
- **方法：** `RetractAnswer`
- **路由名称：** `/api/answers_retract/{answer_id}`
- **HTTP 方法：** POST
- **传入参数及类型：**
    - `answer_id` (Int): 回答ID
    - `current_user_id` (Int): 当前用户ID
- **功能：** 撤回指定回答，并在数据库中更新状态。

第二周开发总结

在本周的开发过程中，我们主要实现了用户活跃度模块和用户评论模块

用户活跃度模块开发总结

在本周的开发过程中，我们实现了用户活跃度模块的API接口，主要包括查看近期活跃、新增活跃行为、更新活跃行为、删除活跃行为以及获取整体活跃度等功能。以下是详细的接口实现总结：

1. 查看近期活跃

- **类名：** `UserActivityController`
- **方法：** `GetRecentActivities`
- **路由名称：** `/api/activity/recent`
- **HTTP 方法：** POST
- **传入参数及类型：**
    - `current_user_id` (Int): 当前用户ID
- **功能：** 获取当前用户的近期活跃记录，包括活动ID、类型、分数和时间。

2. 新增活跃行为

- **类名：** `UserActivityController`
- **方法：** `AddActivity`
- **路由名称：** `/api/activity/add`
- **HTTP 方法：** POST
- **传入参数及类型：**
    - `user_id` (Int): 用户ID
    - `activity_type` (String): 活动类型
    - `score` (Int): 活动分数
    - `datetime` (DateTime): 活动时间
- **功能：** 新增一条活跃行为记录，并返回新增活动的ID。

3. 更新活跃行为

- **类名：** `UserActivityController`
- **方法：** `UpdateActivity`
- **路由名称：** `/api/activity/update`
- **HTTP 方法：** PUT
- **传入参数及类型：**
  - `activity_id` (Int): 活动ID
  - `activity_type` (String): 活动类型
  - `score` (Int): 活动分数
  - `datetime` (DateTime): 活动时间
- **功能：** 更新指定ID的活跃行为记录。

4. 删除活跃行为

- **类名：** `UserActivityController`
- **方法：** `DeleteActivity`
- **路由名称：** `/api/activity/delete`
- **HTTP 方法：** DELETE
- **传入参数及类型：**
  - `activity_id` (Int): 活动ID
- **功能：** 删除指定ID的活跃行为记录。

5. 获取整体活跃度

- **类名：** `OverallActivityController`
- **方法：** `GetOverallActivity`
- **路由名称：** `/api/overallactivity/overall`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `current_user_id` (Int): 当前用户ID
- **功能：** 获取指定用户的整体活跃度分数。

6. 获取所有用户的整体活跃度

- **类名：** `OverallActivityController`
- **方法：** `GetAllUserOverallActivity`
- **路由名称：** `/api/overallactivity/all`
- **HTTP 方法：** POST
- **功能：** 获取所有用户的整体活跃度，包括用户ID和对应的活跃度分数。

用户评论模块开发总结

在本周的开发过程中，我们实现了用户评论模块的API接口，主要包括添加评论、删除评论、以及查看评论等功能。以下是详细的接口实现总结：

1. 添加评论

- **类名：** `CommentsController`
- **方法：** `AddComment`
- **路由名称：** `/api/comments/add`

- **HTTP 方法：** POST
- **传入参数及类型：**
  - `item_id` (String): 评论所属项目ID
  - `user_id` (Int): 用户ID
  - `content` (String): 评论内容
  - `time` (DateTime): 评论时间
- **功能：** 添加一条新的评论，并记录用户的活动。使用事务确保数据一致性，若发生异常则回滚事务。

2. 删除评论

- **类名：** `CommentsController`
- **方法：** `DeleteComment`
- **路由名称：** `/api/comments/delete`
- **HTTP 方法：** DELETE
- **传入参数及类型：**
  - `comment_id` (Int): 评论ID
  - `user_id` (Int): 用户ID
- **功能：** 删除指定ID的评论，并根据删除结果返回成功或失败的状态。

3. 查看评论

- **类名：** `CommentsController`
- **方法：** `ViewItemComments`
- **路由名称：** `/api/comments/item`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `item_id` (String): 评论所属项目ID
- **功能：** 获取指定项目的所有评论，并返回评论列表。每条评论包含评论ID、用户信息（ID、名称、头像）、评论内容和时间。

第三周开发总结

在本周的开发过程中，我们主要实现了广告模块和VIP模块

广告模块开发总结

在本周的开发过程中，我们实现了广告模块的API接口，主要包括获取随机广告、记录广告点击、增加广告、更新广告、删除广告、获取所有广告列表以及获取广告具体细节等功能。以下是详细的接口实现总结：

1. 获取随机广告

- **类名：** `AdvertisementsController`
- **方法：** `GetRandomAd`
- **路由名称：** `/api/advertisement/GetRandomAd`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `user_id` (Int): 用户ID
- **功能：** 根据用户ID获取一个随机广告，并返回广告的详细信息（ID、内容、图片、URL、类型）。

2. 记录广告点击

- **类名：** `AdvertisementsController`
- **方法：** `ClickAd`
- **路由名称：** `/api/advertisement/ClickAd`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `user_id` (Int): 用户ID
  - `ad_id` (Int): 广告ID
  - `click_time` (DateTime): 点击时间
- **功能：** 记录用户对广告的点击，并记录点击时间和IP地址。

3. 增加广告

- **类名：** `AdvertisementsController`
- **方法：** `AddAdvertisement`
- **路由名称：** `/api/advertisement/AddAdvertisement`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `ad_content` (String): 广告内容
  - `ad_picture` (String): 广告图片
  - `ad_url` (String): 广告URL
  - `ad_type` (String): 广告类型
  - `start_time` (DateTime): 广告开始时间
  - `end_time` (DateTime): 广告结束时间
- **功能：** 添加一条新的广告，并返回广告ID。

4. 更新广告

- **类名：** `AdvertisementsController`
- **方法：** `UpdateAdvertisement`
- **路由名称：** `/api/advertisement/UpdateAdvertisement`
- **HTTP 方法：** PUT
- **传入参数及类型：**
  - `ad_id` (Int): 广告ID
  - `ad_content` (String, 可选): 广告内容
  - `ad_picture` (String, 可选): 广告图片
  - `ad_url` (String, 可选): 广告URL
  - `ad_type` (String, 可选): 广告类型
  - `start_time` (DateTime, 可选): 广告开始时间
  - `end_time` (DateTime, 可选): 广告结束时间
- **功能：** 更新指定广告的信息。只更新提供的字段，并根据更新结果返回成功或失败的状态。

5. 删除广告

- **类名：** `AdvertisementsController`
- **方法：** `DeleteAdvertisement`
- **路由名称：** `/api/advertisement/DeleteAdvertisement`

- **HTTP 方法：** DELETE
- **传入参数及类型：**
  - `ad_id` (Int): 广告ID
- **功能：** 删除指定ID的广告，并返回操作的成功与否。

6. 获取所有广告列表

- **类名：** `AdvertisementsController`
- **方法：** `GetAdvertisements`
- **路由名称：** `/api/advertisement/GetAdvertisements`
- **HTTP 方法：** POST
- **功能：** 获取所有广告的列表，包括广告的ID、内容、图片、URL、类型、开始时间和结束时间。

7. 获取广告具体细节

- **类名：** `AdvertisementsController`
- **方法：** `GetAdvertisementDetails`
- **路由名称：** `/api/advertisement/GetAdvertisementDetails`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `ad_id` (Int): 广告ID
- **功能：** 获取指定广告的详细信息，包括广告的内容、图片、URL、类型、开始时间、结束时间、点击次数和展示次数，以及广告点击的统计信息（用户ID、点击时间、IP地址）。

VIP模块开发总结

在本周的开发过程中，我们实现了VIP模块的API接口，涵盖了VIP会员的检查、充值、状态获取、添加、删除和更新等功能。以下是详细的接口实现总结：

1. 判断是否是VIP会员

- **类名：** `UserVIPController`
- **方法：** `IsVIPMember`
- **路由名称：** `/api/vip/isMember`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `user_id` (Int): 用户ID
- **功能：** 根据用户ID判断用户是否为VIP会员，并返回结果。

2. 用户充值VIP

- **类名：** `UserVIPController`
- **方法：** `RechargeVip`
- **路由名称：** `/api/vip/RechargeVIP`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `user_id` (Int): 用户ID
  - `recharge_time` (Int): 充值时长（月）
  - `total_amount` (Double): 充值金额

- **功能：** 处理用户VIP充值，生成充值订单，并返回订单ID、VIP开始时间、结束时间、充值金额及积分返回等信息。

3. 获取用户VIP状态

- **类名：** `UserVIPController`
- **方法：** `GetVIPstatus`
- **路由名称：** `/api/vip/GetVIPstatus`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `user_id` (Int): 用户ID
- **功能：** 获取用户的VIP状态，包括VIP会员ID、开始时间、结束时间和状态。

4. 添加VIP会员

- **类名：** `UserVIPController`
- **方法：** `AddVIPMember`
- **路由名称：** `/api/vip/AddVIPMember`
- **HTTP 方法：** POST
- **传入参数及类型：**
  - `user_id` (Int): 用户ID
  - `vip_start_time` (DateTime): VIP开始时间
  - `vip_end_time` (DateTime): VIP结束时间
  - `vip_status` (String): VIP状态
- **功能：** 为用户添加VIP会员记录，并返回VIP会员ID。

5. 删除VIP会员

- **类名：** `UserVIPController`
- **方法：** `DeleteVIPMember`
- **路由名称：** `/api/vip/DeleteVIPMember`
- **HTTP 方法：** DELETE
- **传入参数及类型：**
  - `vip_member_id` (Int): VIP会员ID
- **功能：** 删除指定的VIP会员记录，并返回操作结果。

6. 更新VIP会员

- **类名：** `UserVIPController`
- **方法：** `UpdateVIPMember`
- **路由名称：** `/api/vip/UpdateVIPMember`
- **HTTP 方法：** PUT
- **传入参数及类型：**
  - `vip_member_id` (Int): VIP会员ID
  - `user_id` (Int, 可选): 用户ID
  - `end_time` (DateTime, 可选): VIP结束时间
  - `vip_status` (String, 可选): VIP状态

- **功能：** 更新VIP会员的信息。支持更新用户ID、VIP结束时间和VIP状态，但不允许修改VIP开始时间。返回更新结果。

第四周开发总结

在本周的开发过程中，我们主要实现了实时聊天和图片传输，并且加强了代码的鲁棒性

实时聊天 WebSocket 连接开发总结

在本次开发中，我们实现了一个用于处理 WebSocket 连接的服务类 `WebSocketService`，支持实时聊天功能。以下是详细的实现总结：

1. 类及功能概述

类名： `WebSocketService`

- **功能：** 处理 WebSocket 连接、接收消息、发送消息，以及管理用户与 WebSocket 连接的映射。

2. 主要方法

2.1 处理 WebSocket 连接请求

- **方法名：** `HandleConnection`
- **功能：** 接受 WebSocket 请求，将用户 ID 与 WebSocket 连接存储到字典中，并启动异步接收消息的任务。
- **实现细节：**
  - 确保请求是 WebSocket 请求。
  - 从查询参数中获取用户 ID。
  - 将用户 ID 和 WebSocket 连接存储到 `_connections` 字典中。
  - 调用 `ReceiveMessages` 方法来接收和处理消息。

2.2 接收和处理消息

- **方法名：** `ReceiveMessages`
- **功能：** 异步接收消息，将字节数组转换为字符串，解析 JSON 格式的消息，处理接收到的消息，处理消息的过程中处理异常。
- **实现细节：**
  - 使用缓冲区接收消息。
  - 将接收到的字节数组转换为字符串，并尝试解析为 `User_Messages` 对象。
  - 处理消息内容（目前注释掉了具体处理代码）。
  - 移除关闭的 WebSocket 连接，并关闭 WebSocket。

2.3 发送消息

- **方法名：** `SendMessageAsync`
- **功能：** 向指定用户发送消息。
- **实现细节：**
  - 从 `_connections` 字典中获取 WebSocket 连接。

- 检查 WebSocket 状态是否为开放状态。
- 将消息对象序列化为 JSON 字符串，并将其转换为字节数组。
- 发送消息到 WebSocket 连接。