功能需求

系统角色

- 普通用户：可正常使用该软件功能的普通用户。
- 管理员：可对消息、图片、音视频及动态等进行审核、管理、删除。

用户故事

- 普通用户：用户进行注册登录后，可以管理自己的个人信息及密码，管理自己的联系人及群组，可以发送、接收与联系人及群组内的消息，可以浏览、发布动态，在动态中进行点赞及回复。
- 管理员：进行登录后，可以管理所有的用户、消息及动态等内容，包括对内容进行审核、删除等。

功能点清单

- 普通用户：
  - 用户与联系人
    - 用户可以注册登录、修改密码
    - 用户可以修改ID、个人信息、头像、状态等
    - 用户可以查找、添加、删除联系人
  - 聊天与会话
    - 可向联系人发起聊天
    - 可向通信对象实时推送消息并以通知形式提醒
    - 可以查看、删除历史消息
    - 会话中可发送、接收文字
    - 会话中可发送、接收、播放音频
    - 会话中可发送、接收、播放视频
    - 会话中可发送、接收并展示地理位置信息
  - 群聊与管理
    - 可指定若干位联系人、发起群聊
    - 可查看群聊成员
    - 可邀请联系人加入群聊或退出群聊
  - 动态发布
    - 按时间线浏览所有联系人发布的动态
    - 发布图文动态
    - 发布视频动态
    - 对动态进行点赞、展示单条动态的所有点赞信息
    - 对联系人及自己发布的动态进行回复、展示回复信息
- 管理员：
  - 对用户进行管理，审核、删除违规账号等
  - 对会话消息进行审核、管理，删除违规消息等
  - 对群组进行管理、审核、删除违规群组等
  - 对动态进行管理，审核、删除违规动态、评论回复等
- 系统及`UI`支持
  - `APP`正确签名打包，可在`Android`机正常运行
  - `UI`界面设计完整、合理，包括会话列表、会话界面、联系人列表、动态浏览列表、动态发布页面及用户信息浏览与设置界面

非功能需求

性能

在正常网络情况下，后端部分应支持500`QPS`以上的并发，100个用户同时在线，每秒钟应能处理至少200以上的文字消息；语音及视频消息上传下载速度在应在`100`KB以上。

`Android`应用程序应正常进入首页及过渡到其他页面、能正常浏览会话列表、联系人列表、动态列表等，不会出现`UI`的明显卡顿、页面应正常进行过渡及加载。

安全

后端部分应禁止数据库的对外直接访问，对用户密码进行不可逆的哈希加密处理，必要情况下应对会话消息、动态等也进行加密处理，应部署在`HTTPS`及`WSS`的环境下。

`Android`应用程序应禁止保存用户密码，对用户信息、会话、动态等信息的缓存应禁止其他应用程序的访问。

数据的备份恢复

应对`mongodb`进行定时备份或采用复制集进行部署，以保持数据库的安全及高可用性。

初步设计

后端

使用`Java`语言进行开发，使用`SprintBoot`进行依赖注入、切面等的管理，使用`Netty`进行`Http`及`Websocket`接口的监听并处理请求，数据库使用`mongodb`进行信息的持久化存储，使用`redis`作会话缓存；向前端提供用户信息、会话信息、联系人信息、群组信息、动态信息等`Http`接口的服务，及实时会话通知的`Websocket`接口服务；使用`TimeLine`模型及扩散写进行消息的同步及存储。

客户端

使用`Java`语言进行开发，主要使用`MVVM`的架构，使用`Activity/Fragment`进行`UI`展示，使用`LiveData`及`ViewModel`进行数据界面绑定及`Room`进行缓存，使用`Retrofit`进行`Http`请求，`Scarlet`进行`Websocket`连接，使用`Hilt`进行依赖注入。

数据库设计

User

> 存储用户信息

id	name	avatar	password_hash	friends	timeline
用户ID	用户名字	用户头像	密码	好友信息	同步timeline

```
friend: {
  nickname:"昵称",
  timeline:"会话ID",
  id:"好友ID"
}
```

Group

> 存储群组信息

id	name	members	timeline
群组ID	群组名字	群组成员id列表	会话ID

Message

id	content	content_type	message_type	time	from	to
消息ID	消息内容	内容类型	消息类型	消息时间	发送者ID	接受者ID

```
content_type: 'text' | 'audio' | 'video'
# `single`指单聊 `group`指群聊 `application`指好友申请 `invite`指邀请加入群聊 `confirm`指同意好友请求
type: 'single' | 'group' | 'application' | 'invite' | 'confirm'
```

TimelineSaved

> 消息存储库，对消息进行持久化存储

id	messages
会话ID	会话消息ID

TimelineSync

> 消息同步库，对每个用户进行消息同步

id	messages
会话ID	会话消息ID

Discover

> 存储用户发布的动态

id	sender	content	images	time	likes	replies
动态ID	发送者ID	文字内容	图片列表	时间	点赞用户ID	评论

接口设计

采用`Restful`风格

用户

> 初步使用Cookie进行鉴权

请求方法	意义	链接	请求体	响应体
GET	获取单个用户信息	`/user/:id`		Profile
POST	注册	`/signup`	SignUpBody	Profile
POST	登录	`/signin`	SignInBody	Profile
POST	登出	`/signout`		
POST	确认添加好友	`/addfriend`	AddFriendBody	

```
SignUpBody: {
  id:"微信号"
  name:"姓名",
  password:"密码"
}

SignInBody: {
  id:"微信号",
  password:"密码"
}

AddFriendBody {
  id:"好友ID"
}

Profile:{
  id:"微信号",
  name:"姓名"
}
```

群组

请求方法	意义	链接	请求体	响应体
POST	创建群组	`/group`	CreateGroupBody	GroupProfile
POST	邀请加入群聊	`/group/invite`	InviteInToGroupBody	
POST	退出群聊	`/group/exit`	ExitGroupBody	

```
CreateGroupBody:{
  name:"群组名字",
  members:[{
   weixinId:"联系人微信号"
  }]
}
```

```
InviteIntoGroup:{
  groupId:"群组ID"
  weixinId:"联系人微信号"
}

ExitGroupBody:{
  groupId:"群组ID"
}
```

消息

请求方法	意义	链接	请求体	响应体
GET	获取同步消息	`/timelinesync/:id`		Messages
GET	获取会话漫游消息	`/timelinesaved/:id`		Messages
POST	发送消息	`/message`	Message	

动态

请求方法	意义	链接	请求体	响应体
GET	获取动态	`/discover`		Discovers
POST	发布动态	`/discover`	CreateDiscoverBody	Discover

```
CreateDiscoverBody {
    content:"",
    images:[]
}
```

文件

请求方法	意义	链接	请求体	响应体
GET	获取文件	`/upload/:id`		FileContent
POST	上传文件	`/upload`	FileContent	

接口逻辑实现

用户

- 注册：在`User`中新增用户，对密码进行加密，新建`timeline`，保存在`User`,`TimeLineSync`中
- 登录：验证微信好、密码，返回加密cookie，并在`Redis`中加入缓存
- 登出：在`Redis`中清除该用户
- 确认添加好友：在`User`中双方`friends`中均新增好友信息，新建`timeline`，保存在`friend`，`TimeLineSaved`中

消息

- 获取同步消息：若用户在线，则在扩散写时通过`websocket`发送消息，发送成功后，若未在线，则登录后获取同步消息，同步消息使用`time`作为检验，在客户端保存最新同步`time`，每次获取同步消息时，仅获取`time`之后的消息，需要考虑设置`timeline`过期时间后删除
- 发送消息：消息类型包括单聊，群聊，好友申请，内容类型包括音频、视频、文本，消息发送成功后，扩散写进入接受者`timeline`，单聊和好友申请直接写入`friends`中即可，群聊需要写入所有群组成员`timeline`中，
- 消息漫游：跨端时需进行消息漫游，考虑消息会缓存在客户端，每次检查同步`time`，若首次不存在`time`，客户端调用获取消息存储库中所有消息，直接返回所有消息即可

动态

- 发布动态：在`Discover`中加入动态
- 获取动态：按时间排序获取动态

待定

- 地图实现

其他

目前管理员端为低功能需求