<开心论坛系统>

需求规格说明书

作　　者：周星宇 10175101110
胡一涵 10175101107

完成日期：2020.6.28

修订历史记录

日期	版本	说明	作者
2020.6.28	V1.0	初版	周星宇,胡一涵

目录

1. 引言

1.1 目的

该文档首先给出项目的整体结构和功能结构概貌，试图从总体架构上给出整个系统的轮廓。同时对功能需求、性能需求进行了详细的描述。便于用户、开发人员进行理解和交流，反映出用户问题的结构，可以作为软件开发工作的基础和依据以及确认测试和验收的依据。

本文档面向多种读者对象：

（1）项目经理：项目经理可以根据该文档了解预期产品的功能，并据此进行系统设计、项目管理。

（2）设计员：对需求进行分析，并设计出系统，包括数据库的设计。

（3）程序员：了解系统功能，编写《用户手册》。

（4）测试员：根据本文档编写测试用例，并对软件产品进行功能性测试和非功能性测试。

（5）用户：了解预期产品的功能和性能，并与分析人员一起对整个需求进行讨论和协商。

在阅读本文档时，首先要了解产品的功能概貌，然后可以根据自身的需要对每一功能进行适当的了解。

1.2 背景

本次待开发的软件为一个在线问答系统——开心论坛。

当代大学生在校内的缺少一个公共交流学习的平台，学习中的问题得不到及时的解决，有一些学术上的观点，也得不到互相交流。该软件在 Web 终端实现了一个可以供多用户讨论的公共论坛，让大家畅所欲言提问、回答和评论，可以用来解决上述问题。

1.3 定义

序号	缩写	定义
1	无	无

2. 项目概述

2.1 产品描述

用户通过使用该软件在 Web 终端实现的一个论坛，可以与他人一起提问、回答问题以及评论交友。

2.2 产品功能

当前高校内缺少一个内部的类似论坛的软件，我们的软件实现的功能，能够提高学生学习的兴趣，更好地解决学生现存在的问题，给大家提供互相学习，并交友的平台。

该项目需要满足以下要求：

1.非登陆用户可以看其他人的问答，按最后一个回答时间的倒序排列

2.登陆用户可以发布、结束自己的问题。

3.对于未终结的问题，登陆用户可以发布、修改、删除自己的回答和提问。

4.普通人可申请注册，管理员同意后算作注册成功；管理员可将现有用户拉黑，但管理员不能删除用户。

5.被拉黑的用户的问题、回答对其他人不可见。

6.管理员可以屏蔽掉某个问题或者回答，也可取消屏蔽。

7.被屏蔽的问题或回答对其他人不可见；提问人或回答人自己可见。

8.被屏蔽的问题或者回答，提问人或回答人不能修改、删除。

2.3 用户特点

本软件的最终用户为大学生和高校教师，该用户群体普遍接受互联网文化，学习及适应能力强。能快速适应该网页，并充分感受到在交流过程中的效能变化，提出合理改进意见。

本软件预期为用户提供长期的论坛服务。

2.4 一般约束

进行本软件开发工作的约束条件如下：

1. 疫情期间，学生学习环境变化，学习效率不一，导致论坛需求不明晰。
2. 所采用的方法与技术有限：项目团队成员的技术水平不够成熟，需要在开发中并发学习多种技术和能力。
3. 项目团队成员人数有限，只有两人，任务繁重。

2.5 假设与依据

本项目是否能够成功实施，主要取决于以下的条件：

（1）团队成员的积极合作配合，为了项目的开发和实施，对个人时间进行合理规划同时为团队做出合理牺牲，配合队友完成任务。

（2）教师和学生提供完整详细的功能和性能需求资料，以便于团队对其进行分析，从而形成完善的软件需求。

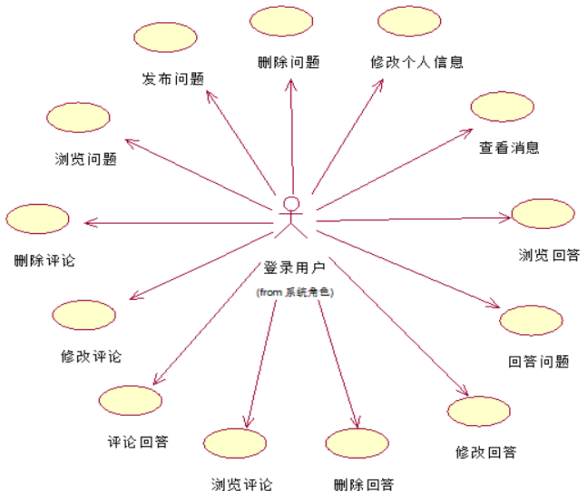
（3）团队掌握先进的能够适用于该项目的技术，这是系统的性能是否优化和项目能否成功的保证。

3. 功能需求

3.0 角色划分

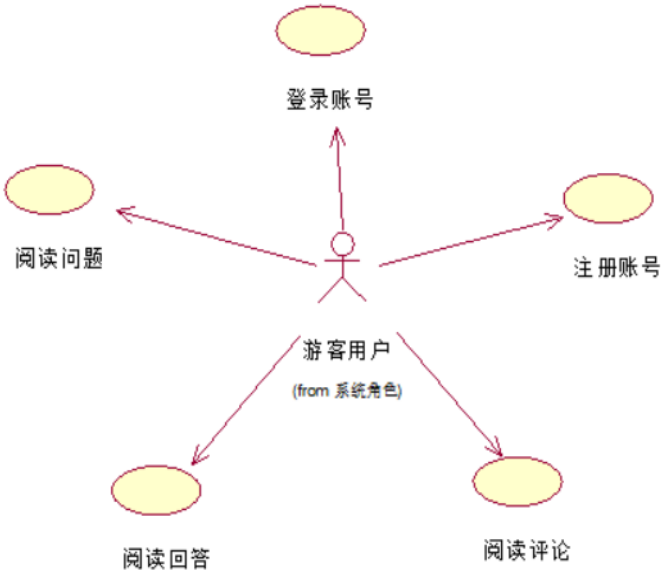
首先设计各角色用例图以明确该项目中的角色及其角色所要可以完成的操作，这里我们分成三类使用者：

3.0.1 登录用户

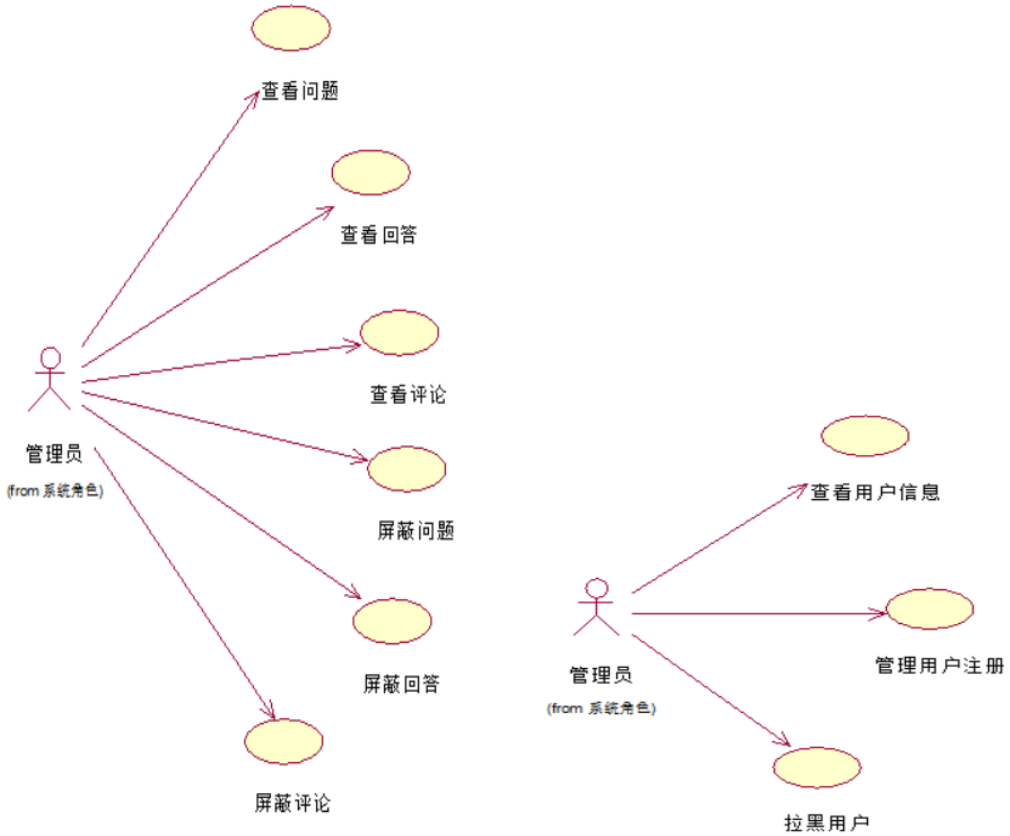


属性名	类型	长度	注释
icon	varchar	255	用户头像
phone	varchar	11	用户电话号码（主键）
name	varchar	255	用户名
password	varchar	255	用户密码
sex	int	1	用户性别
age	int	3	用户年龄
shield	int	1	拉黑状态，0 表示被拉黑，1 表示正常

3.0.2 游客



3.0.3 管理员



属性名	类型	长度	注释
name	varchar	127	管理员名称（主

			键）
password	varchar	255	管理员密码
status	int	1	管理员账号的登陆状态，0 表示未登录，1 表示已登录

3.1 系统角色及登陆

该系统共有三种角色：管理员，注册登录用户，游客。前两种角色都具有登陆功能，根据角色不同登陆后进入各个角色所对应的页面以及拥有不同的权限。

3.2 游客

3.2.1 账户注册

注册时，用户在填写用户名等信息后提交前进行手机号码验证，一个手机号只能注册一个账户且用户名不能重复，注册成功会收到邮件通知。

成员名	成员类型（字段、方法或属性）	成员含义说明
userRegister	int	用户申请注册
registerByCode	String	发送注册的验证码

[Happy_Forum-VisitorRegister-0001]

userRegister 用于用户申请注册，获取用户输入手机号、姓名、密码、性别、年龄等，并将数据加入数据库中。

```
registerUser(Register register) {
                User user = new User();
                user.setPhone();
                user.setName();
                user.setPassword(sha256 password);
                user.setSex();
                user.setAge();
                sendEmail();
                addUser();
        }
```

[END]

[Happy_Forum-VisitorRegister-0002]

registerByCode 用于调用外部接口，发送验证码。

```
    sendCode(User user){
        return verificationCode(user.getPhone());
    }
[END]
```

3.3 注册用户

3.3.1 登录

用户通过输入账号密码，点击登录；可通过账号密码登录和手机验证码登录两种方式。

成员名	成员类型（字段、方法或属性）	成员含义说明
loginUserByPassword	int	用户根据用户名和密码登录
loginUserByAuthCode	int	用户根据手机号码和验证码登录

[Happy_Forum-UserLogin-0001]

loginUserByPassword 用于用户通过密码登录。将用户输入用户名密码和数据库所存的进行比对，如以下为代码所示外，需进行 sha256 解密加密。

```
loginUserByPassword(User user) {
    if(isUserExistByName(user.getName())){
        return -1;
    }else if(findUserByName(user.getName()).getPassword().equals(user.getPassword())){
        return 1;
    }else{
        return 0;
    }
}
[END]
```

[Happy_Forum-UserLogin-0002]

loginUserByAuthCode 用于用户通过手机和验证码登录。将用户输入手机号和验证码与外部服务发送的验证码进行比对，如以下为代码所示外，需进行 sha256 解密加密。

```
loginUserByAuthCode(String phone) {
    if (userDao.isUserExistByPhone(phone)) {
        return verificationCode(phone);
    } else {
        return null;
    }
} [END]
```

3.3.2 查看未读消息以及消息提醒

主页面上右上角应有消息盒子模块提示用户查看信息，并且点击后可以进入相关页面进行进一步操作（阅读、回复等）。

成员名	成员类型（字段、方法或属性）	成员含义说明
findMessageByName	List<Message>	根据用户名查询该用户所接收的所有消息
countAvailableMessage	int	查询用户未读消息的数量
updateMessageStatus	int	修改消息的已读属性

[Happy_Forum-UserMessage-0001]

findMessageByName 用于通过用户名返回出所有信息。

```
findMessageByName(String name) {
List<Message> messageList = jdbcTemplate.query(查询 query);
if (messageList.size() > 0) {
    return messageList;
} else {
    return null;
}
}
```

[END]

[Happy_Forum-UserMessage-0002]

countAvailableMessage 用于通过用户名返回出所有信息的数量。

```
findMessageByName(String name) {
    List<Message> messageList = jdbcTemplate.query(查询 query 并 count);
    return messageList.size();
}
```

[END]

[Happy_Forum-UserMessage-0003]

updateMessageStatus 用于更改信息状态。

```
findMessageByName(String name) {
    int count = update(更改状态语句);
    return count;
}
```

[END]

3.3.3 查看问题详情以及回答

点击主页面任意问题后，可进入问题详情页。可以查看阅读，也可以再下方的输入框中输入并回复。

成员名	成员类型（字段、方法或属性）	成员含义说明
findAllWhiteAnswers	List<Answer>	查询所有未屏蔽回答
findAnswerById	Answer	根据回答编号查询回答
findWhiteAnswerByQuestionId	List<Answer>	根据问题编号查询该问题下未被屏蔽的所有回答
addAnswer	int	新增回答
deleteAnswerById	int	删除回答
findAnswersByKeyword	List<Answer>	模糊查询回答

[Happy_Forum-UserAnswer-0001]

findAllWhiteAnswers 用于查询所有未屏蔽回答。

```
List<Answer> findAllWhiteAnswers() {
    List<Answer> answerList = jdbcTemplate.query(查询语句);
    return answerList;
}
```

[END]

[Happy_Forum-UserAnswer-0002]

findAnswerById 用于根据回答编号查询回答。

[END]

[Happy_Forum-UserAnswer-0003]

findWhiteAnswerByQuestionId 用于根据问题编号查询该问题下未被屏蔽的所有回答。

[END]

[Happy_Forum-UserAnswer-0004]

addAnswer 用于增加回答。

[END]

[Happy_Forum-UserAnswer-0005]

deleteAnswerById 用于删除回答。

[END]

[Happy_Forum-UserAnswer-0006]

findAnswersByKeyword 用于模糊查询回答。

[END]

3.3.4 用户提问

用户可以在相关板块填写问题标签和问题内容后点击提交。问题会出现在问题列表中，可在首页和热度榜展示。

成员名	成员类型（字段、方法或属性）	成员含义说明
updateQuestion	int	更改问题
addQuestion	int	新增问题
deleteQuestionById	int	删除问题
endQuestion	int	终结问题

[Happy_Forum-UserQuestion-0001]

updateQuestion 用于更改问题。

```
updateQuestion(Question question) {
        if (判断问题是否存在) {
            if (判断问题是否被屏蔽) {
                return 0;
            } else {
                updateQuestion(question);
            }
        }else {
            return -1;
        }
    }
```

[END]

[Happy_Forum-UserQuestion-0002]

addQuestion 用于新增问题。

```
addQuestion(Question question) throws Exception {
    if (判断问题是否存在) {
        if (判断用户是否被屏蔽) {
            question.setShield(0);
        } else {
            question.setShield(1);
        }
        question.setHeat(0);
        question.setEnd(1);
        addQuestion(question);
    }else {
        return -1;
    }
```

```
    }
 [END]
```

[Happy_Forum-UserQuestion-0003]

deleteQuestionById 用于删除问题。

```
deleteQuestionById(int id) {
        if(用 id 判断问题是否存在){
            if (判断问题是否被屏蔽) {
                return 0;
            } else {
                deleteQuestionById(id);
            }
        }else {
            return -1;
        }
    }
 [END]
```

[Happy_Forum-UserQuestion-0004]

endQuestion 用于终结问题。

```
    endQuestion(Question question) {
        if (用 id 判断问题是否存在) {
            if (用 id 判断问题是否未被屏蔽) {
                question.setEnd(0);
                questionSignal(question);
            } else {
                return 0;
            }
        } else {
            return -1;
        }
    }
[END]
```

3.3.5 问题搜索

在页面中，可通过问题标签、问题内容和提问者模糊查询问题，再进行回答操作。

成员名	成员类型（字段、方法或属性）	成员含义说明
findQuestionById	Question	根据问题编号查询问题
showQuestionsByKeyword	List<Question>	根据关键词查询问题

findQuestionByUserName	List<Question>	根据用户名查找该用户发布的所有问题
findQuestionsByKeyword	List<Question>	模糊查询问题

[Happy_Forum-UserQuestionSearch-0001]

findQuestionById 用于根据问题编号查询问题。

[END]

[Happy_Forum-UserQuestionSearch-0002]

showQuestionsByKeyword 用于根据关键词查询问题。

[END]

[Happy_Forum-UserQuestionSearch-0003]

findQuestionByUserName 用于根据用户名查找该用户发布的所有问题。

[END]

[Happy_Forum-UserQuestionSearch-0004]

findQuestionsByKeyword 用于模糊查询问题。

[END]

以上均是通过数据库用语句来查询。

3.3.6 查看个人资料并修改

个人中心显示用户头像和用户名，在页面中展示用户发布的问题，并且可以对其进行修改，删除操作。此外可以查看到回答过的问题，以及调用上述的需求中的消息提醒。此外，页面左侧为头像修改。右侧为信息修改及密码找回，其中用户名不可修改，修改手机号码时需先进行号码验证；并且系统应允许用户在验证手机号或者邮箱后修改密码。

成员名	成员类型（字段、方法或属性）	成员含义说明
userAllQuestion	List<Question>	查询某一用户所发布的所有问题
userAllAnswer	List<Answer>	查询某一用户所发布的所有回答
userQuestionEnd	int	终结问题
findUserByPhone	User	根据号码查询用户
findUserByName	User	根据用户名查询用户
queryPhoneByName	String	根据用户名返回用户号码
updateUser	int	更改用户信息
addUser	int	新增用户信息
deleteUserByPhone	int	删除用户信息
saveIcon	boolean	上传用户头像
showIcon	String	查询用户头像
decodeToImage	BufferedImage	base 编码转化为图片

getImageStr	String	获取 base64 编码

[Happy_Forum-PersonalInfo-0001]

userAllQuestion 用于查询某一用户所发布的所有问题。

[END]

[Happy_Forum-PersonalInfo-0002]

userAllAnswer 用于查询某一用户所发布的所有回答。

[END]

[Happy_Forum-PersonalInfo-0003]

userQuestionEnd 用于终结问题。

[END]

[Happy_Forum-PersonalInfo-0004]

findUserByPhone 用于根据号码查询用户。

[END]

[Happy_Forum-PersonalInfo-0005]

findUserByName 用于根据用户名查询用户。

[END]

[Happy_Forum-PersonalInfo-0006]

queryPhoneByName 用于根据用户名返回用户号码。

[END]

[Happy_Forum-PersonalInfo-0007]

updateUser 用于更改用户信息。

[END]

[Happy_Forum-PersonalInfo-0008]

addUser 用于新增用户信息。

[END]

[Happy_Forum-PersonalInfo-0009]

deleteUserByPhone 用于删除用户信息。

[END]

[Happy_Forum-PersonalInfo-0010]

saveIcon 用于上传用户头像。

[END]

[Happy_Forum-PersonalInfo-0011]

showIcon 用于查询用户头像。

[END]

以上均是通过数据库用语句来查询与操作。

[Happy_Forum-PersonalInfo-0012]

decodeToImage 用于 base 编码转化为图片。

```
Return decodeToImage(getImageStr(findUserByPhone(user.getPhone()).getIcon()));
```

[END]

[Happy_Forum-PersonalInfo-0013]

getImageStr 用于获取 base64 编码。

```
Return decodeToImage(getImageStr(findUserByPhone(user.getPhone()).getIcon()));
```

[END]

以上两个可用 Base64Utils 包来完成。

3.4 管理员

3.4.1 管理员登录

管理员通过输入账号密码，点击登录。

成员名	成员类型（字段、方法或属性）	成员含义说明
findAdminByName	Admin	根据管理员账号查询管理员
updateAdmin	int	修改管理员信息
addAdmin	int	新增管理员
deleteAdminByName	int	删除管理员
loginAdmin	int	验证管理员登录
exitAdmin	int	管理员退出登录

[Happy_Forum-AdminLogin-0001]

findAdminByName 用于根据管理员账号查询管理员。

[END]

[Happy_Forum-AdminLogin-0002]

```
    public int updateAdmin(Admin admin) {
        int count = jdbcTemplate.update("update admin set password = ?,status
= ? where name = ?",admin.getPassword(),admin.getStatus(),admin.getName());
        return count;
    }
```

[END]

[Happy_Forum-AdminLogin-0003]

```
    public int addAdmin(Admin admin) {
        int count = jdbcTemplate.update("insert into admin(name,password,status)
values(?,?,?)", admin.getName(),admin.getPassword(),admin.getStatus());
        return count;
    }
```

[END]

[Happy_Forum-AdminLogin-0004]

```
    public int deleteAdminByName(String name) {
        int count = jdbcTemplate.update("delete from admin where name = ?",name);
        return count;
    }
```

[END]

[Happy_Forum-AdminLogin-0005]

loginAdmin 用于验证管理员登录。

[END]

[Happy_Forum-AdminLogin-0006]

exitAdmin 用于管理员退出登录。

[END]

3.4.2 注册用户管理以及审核

在管理员系统中，通过列表展示提交了注册申请并成功验证手机号的用户。管理员判断是否同意注册，管理员通过或拒绝用户注册系统都会给用户发送邮件通知；此外，管理员可以对未注册用户进行模糊查询。

成员名	成员类型（字段、方法或属性）	成员含义说明
findAllRegister	List<Register>	查询所有申请注册的用户
findRegisterByPhone	Register	根据号码查询申请注册的用户
updateRegister	int	更改申请注册用户的信息
addRegister	int	新增申请注册用户的信息
deleteRegisterByPhone	int	删除申请注册用户的信息
applyRegister	int	用户申请注册

findNotPassedRegister	List<Register>	查询所有未注册成功的用户信息
findRegistersByKeyword	List<Register>	模糊查询注册用户

[Happy_Forum-UserRegister-0001]

```
    public List<Register> findAllRegister(){
        List<Register> registerList = jdbcTemplate.query("select * from register",new Object[]{},new BeanPropertyRowMapper<>(Register.class));
        return registerList;
    }
```

[END]

[Happy_Forum-UserRegister-0002]

```
    public Register findRegisterByPhone(String phone){
        List<Register> registerList = jdbcTemplate.query("select * from register where phone = ?",new BeanPropertyRowMapper<>(Register.class),phone);
        if(registerList.size()>0){
            return registerList.get(0);
        }else{
            return null;
        }
    }
```

[END]

[Happy_Forum-UserRegister-0003]

```
    public int updateRegister(Register register){

        int count = jdbcTemplate.update("update register set name = ?,password = ?,sex = ?,age = ?,mail = ? where phone = ?",register.getName(),register.getPassword(),register.getSex(),register.getAge(),register.getMail(),register.getPhone());
        return count;
    }
```

[END]

[Happy_Forum-UserRegister-0004]

```
    public int addRegister(Register register){

        int count = jdbcTemplate.update("insert into register(phone,name,password,sex,age,mail)
values(?,?,?,?,?,?)",register.getPhone(),register.getName(),register.getPassword(),register.getSex(),register.getAge(),register.getMail());
        return count;
    }
```

[END]

[Happy_Forum-UserRegister-0005]

```
    public int deleteRegisterByPhone(String phone){

        int count = jdbcTemplate.update("delete from register where phone = ?",phone);
        return count;
    }
```

[END]

[Happy_Forum-UserRegister-0006]

applyRegister 用于用户申请注册。

[END]

[Happy_Forum-UserRegister-0007]

findNotPassedRegister 用于查询所有未注册成功的用户信息。

[END]

[Happy_Forum-UserRegister-0008]

findRegisterByKeyword 用于模糊查询注册用户。

[END]

3.4.3 已通过审核用户管理（拉黑操作）

在管理员系统中,通过列表展示所有用户的信息。管理员可进行拉黑或取消拉黑的操作，被拉黑用户的问题、回答对其他人不可见；可以对用户进行模糊查询；可以用选择器选择查看已拉黑用户或未拉黑用户。

成员名	成员类型（字段、方法或属性）	成员含义说明
findAllUser	List<User>	查询所有用户
findAllWhiteUsers	List<User>	查询所有未拉黑用户
findAllBlackUsers	List<User>	查询所有已拉黑用户
findUserByPhone	User	根据号码查询用户
findUserByName	User	根据用户名查询用户
queryPhoneByName	String	根据用户名返回用户号码
updateUser	int	更改用户信息
addUser	int	新增用户信息
deleteUserByPhone	int	删除用户信息
registerUser	int	注册用户
blacklistUser	int	拉黑用户
whitelistUser	int	取消用户拉黑

[Happy_Forum-PassedUser-0001]

```
    public List<User> findAllUser(){
        List<User> userList = jdbcTemplate.query("select * from user",new Object[]{},new BeanPropertyRowMapper<>(User.class));
        if(userList.size()>0){
```

```
            return userList;
        }else{
            return null;
        }
    }
[END]

[Happy_Forum-PassedUser-0002]
    public List<User> findAllWhiteUsers() {
        List<User> userList = jdbcTemplate.query("select * from user where shield
= ?",new BeanPropertyRowMapper<>(User.class),1);
        return userList;
    }
[END]

[Happy_Forum-PassedUser-0003]
    public List<User> findAllBlackUsers() {
        List<User> userList = jdbcTemplate.query("select * from user where shield
= ?",new BeanPropertyRowMapper<>(User.class),0);
        return userList;
    }
[END]

[Happy_Forum-PassedUser-0004]
    public User findUserByPhone(String phone){
        List<User> userList = jdbcTemplate.query("select * from user where phone
= ?",new BeanPropertyRowMapper<>(User.class),phone);
        if(userList.size()>0){
            return userList.get(0);
        }else{
            return null;
        }
    }
[END]

[Happy_Forum-PassedUser-0005]
    public User findUserByName(String name) {
        List<User> userList = jdbcTemplate.query("select * from user where name
= ?",new BeanPropertyRowMapper<>(User.class),name);
        if(userList.size()>0){
            return userList.get(0);
        }else{
            return null;
        }
    }
[END]
```

[Happy_Forum-PassedUser-0006]

queryPhoneByName 用于根据用户名返回用户号码。

[END]

[Happy_Forum-PassedUser-0007]

```
    public int updateUser(User user){
        int count = jdbcTemplate.update("update user set name = ?,sex = ?,age = ?
where phone = ?",user.getName(),user.getSex(),user.getAge(),user.getPhone());
        return count;
    }
```

[END]

[Happy_Forum-PassedUser-0008]

```
    public int addUser(User user){
        int count = jdbcTemplate.update("insert into
user(icon,phone,name,password,sex,age,shield)
values(?,?,?,?,?,?,?)","D:/icon/1591843981727.png",user.getPhone(),user.getName(),u
ser.getPassword(),user.getSex(),user.getAge(),user.getShield());
        return count;
    }
```

[END]

[Happy_Forum-PassedUser-0009]

```
    public int deleteUserByPhone(String phone){
        int count = jdbcTemplate.update("delete from user where phone = ?",phone);
        return count;
    }
```

[END]

[Happy_Forum-PassedUser-0010]

```
    public int registerUser(Register register) throws ClientException {
        if (registerDao.isRegisterExistByPhone(register.getPhone())) {
            User user = new User();
            user.setPhone(register.getPhone());
            user.setName(register.getName());
            user.setPassword(Sha256Utils.getSHA256Str(register.getPassword()));
            user.setSex(register.getSex());
            user.setAge(register.getAge());
     sendMailUtils.sendSuccessRegisterMail(register.getMail(),register.getName());
            return addUser(user);
        }else {
            return -1;
        }
    }
```

[END]

[Happy_Forum-PassedUser-0011]

```
    public int blacklistUser(User user) {
       if (userDao.isUserExistByPhone(user.getPhone())) {
           User user1 = userDao.findUserByPhone(user.getPhone());
           if (user1.getShield() == 1) {
               user1.setShield(0);
               userDao.userShield(user1);

questionService.blacklistQuestions(questionDao.findQuestionsByUserName(user1.getNam
e()));

answerService.blacklistAnswers(answerDao.findAnswersByUserName(user1.getName()));
               messageDao.userBlackMessage(user1.getName());
               return 1;
           } else {
               return 0;
           }
       } else {
           return -1;
       }

   }
```

[END]

[Happy_Forum-PassedUser-0012]

```
    public int whitelistUser(User user) {
       if (userDao.isUserExistByPhone(user.getPhone())) {
           User user1 = userDao.findUserByPhone(user.getPhone());
           if (user1.getShield() == 0) {
               user1.setShield(1);
               userDao.userShield(user1);

questionService.whitelistQuestions(questionDao.findQuestionsByUserName(user1.getNam
e()));

answerService.whitelistAnswers(answerDao.findAnswersByUserName(user1.getName()));
               messageDao.userWhiteMessage(user1.getName());
               return 1;
           } else {
               return 0;
           }
       } else {
           return -1;
       }
   }
```

[END]

3.4.4 问题管理（删除、屏蔽、模糊查询）

在管理员系统中,通过列表展示所有问题信息。管理员可通过折叠板查看问题具体内容；可对问题进行屏蔽或取消屏蔽操作，被屏蔽的问题对其他人不可见，提问人自己可见，提问人不能进行修改或删除；可以对问题进行模糊查询；可以用选择器选择查看已屏蔽问题或未屏蔽问题。

成员名	成员类型（字段、方法或属性）	成员含义说明
findAllQuestion	List<Question>	查询所有问题
findAllWhiteQuestions	List<Question>	查询所有未屏蔽问题
findAllBlackQuestions	List<Question>	查询所有已屏蔽问题
findWhiteQuestionByTimeOrder	List<Question>	将问题按最新回答的时间顺序展示
showQuestionsByKeyword	List<Question>	根据关键词查询问题
findQuestionByUserName	List<Question>	根据用户名查找该用户发布的所有问题
findQuestionById	Question	根据问题编号查询问题
updateQuestion	int	更改问题
addQuestion	int	新增问题
deleteQuestionById	int	删除问题
blacklistQuestion	int	屏蔽问题
whitelistQuestion	int	解除屏蔽问题
blacklistQuestions	int	屏蔽一串问题
whitelistQuestions	int	取消屏蔽一串问题
endQuestion	int	终结问题
sortQuestionByHeat	List<Question>	将所有问题按照热度降序排列
findQuestionsByKeyword	List<Question>	模糊查询问题

[Happy_Forum-UserQuestion-0001]

```
    public List<Question> findAllQuestion() {
        List<Question>  questionList  =  jdbcTemplate.query("select  *  from question",new Object[]{},new BeanPropertyRowMapper<>(Question.class));
        return questionList;
    }
```

[END]

[Happy_Forum-UserQuestion-0002]

```
    public List<Question> findAllWhiteQuestions() {
        List<Question> questionList = jdbcTemplate.query("select * from question where shield = ?",new BeanPropertyRowMapper<>(Question.class),1);
        return questionList;
```

```
    }
[END]

[Happy_Forum-UserQuestion-0003]

    public List<Question> findAllBlackQuestions() {
        List<Question> questionList = jdbcTemplate.query("select * from question where shield = ?",new BeanPropertyRowMapper<>(Question.class),0);
        return questionList;
    }
[END]

[Happy_Forum-UserQuestion-0004]

    public List<Question> findWhiteQuestionByTimeOrder() {
        List<Question> questionList = questionDao.findAllWhiteQuestions();
        TimeSort.questionListSort(questionList);
        return questionList;
    }
[END]

[Happy_Forum-UserQuestion-0005]

    public List<Question> showQuestionsByKeyword(String strWord) {
        List<Question> questionList = jdbcTemplate.query("select * from question where (details like ? or label like ? or questioner like ?) and shield = ?",new Object[]{"%"+strWord+"%","%"+strWord+"%","%"+strWord+"%",1},new BeanPropertyRowMapper<>(Question.class));
        return questionList;
    }
[END]

[Happy_Forum-UserQuestion-0006]

    public List<Question> findQuestionsByUserName(String name) {
        List<Question> questionList = jdbcTemplate.query("select * from question where questioner = ?",new BeanPropertyRowMapper<>(Question.class),name);
        return questionList;
    }
[END]

[Happy_Forum-UserQuestion-0007]

    public Question findQuestionById(int id) {
        List<Question> questionList = jdbcTemplate.query("select * from question where questionId = ?",new BeanPropertyRowMapper<>(Question.class),id);
        if(questionList.size()>0){
            return questionList.get(0);
        }else{
            return null;
        }
```

```
    }
[END]

[Happy_Forum-UserQuestion-0008]

    public int updateQuestion(Question question) {
        int count = jdbcTemplate.update("update question set details = ?,questioner = ?,shield = ?,heat = ?,label = ?,end = ?,lastTime = ? where questionId = ?",question.getDetails(),question.getQuestioner(),question.getShield(),question.getHeat(),question.getLabel(),question.getEnd(),FormatChange.dateTimeChange(new Date()),question.getQuestionId());
        return count;
    }
[END]

[Happy_Forum-UserQuestion-0009]

    public int addQuestion(Question question) {
        int count = jdbcTemplate.update("insert into question(questionId,details,questioner,shield,time,heat,label,end,lastTime) values(?,?,?,?,?,?,?,?,?)",maxQuestionId()+1,question.getDetails(),question.getQuestioner(),question.getShield(),FormatChange.dateTimeChange(new Date()),question.getHeat(),question.getLabel(),question.getEnd(),FormatChange.dateTimeChange(new Date()));
        return count;
    }
[END]

[Happy_Forum-UserQuestion-0010]

    public int deleteQuestionById(int id) {
        int count = jdbcTemplate.update("delete from question where questionId = ?",id);
        return count;
    }
[END]

[Happy_Forum-UserQuestion-0011]

    public int blacklistQuestion(Question question) {
        if (questionDao.isQuestionExist(question.getQuestionId())) {
            if (questionDao.findQuestionById(question.getQuestionId()).getShield() == 1) {
                question.setShield(0);
                questionDao.questionShield(question);
                Message message = new Message();

message.setRecipient(questionDao.findQuestionById(question.getQuestionId()).getQuestioner());
                message.setQuestionId(question.getQuestionId());
```

```
                messageDao.questionBlackMessage(message);
                return 1;
            } else {
                return 0;
            }
        } else {
            return -1;
        }
    }
[END]
```

[Happy_Forum-UserQuestion-0012]

```
    public int whitelistQuestion(Question question) {
        if (questionDao.isQuestionExist(question.getQuestionId())) {
            if  (questionDao.findQuestionById(question.getQuestionId()).getShield()
== 0) {
                question.setShield(1);
                questionDao.questionShield(question);
                Message message = new Message();

message.setRecipient(questionDao.findQuestionById(question.getQuestionId()).getQues
tioner());
                message.setQuestionId(question.getQuestionId());
                messageDao.questionWhiteMessage(message);
                return 1;
            } else {
                return 0;
            }
        } else {
            return -1;
        }
    }
[END]
```

[Happy_Forum-UserQuestion-0013]

```
    public int blacklistQuestions(List<Question> questionList) {
        for (int i = questionList.size(); i > 0; i--) {
            if          (questionDao.findQuestionById(questionList.get(i          -
1).getQuestionId()).getShield() == 1) {
                questionList.get(i - 1).setShield(0);
                questionDao.questionShield(questionList.get(i - 1));
            }
        }
        return questionList.size();
    }
```

[END]

[Happy_Forum-UserQuestion-0014]

```
    public int whitelistQuestions(List<Question> questionList) {
        for (int i = questionList.size(); i > 0; i--) {
            if          (questionDao.findQuestionById(questionList.get(i          -
1).getQuestionId()).getShield() == 0) {
                questionList.get(i - 1).setShield(1);
                questionDao.questionShield(questionList.get(i - 1));
            }
        }
        return questionList.size();
    }
```

[END]

[Happy_Forum-UserQuestion-0015]

```
    public int endQuestion(Question question) {
        if (questionDao.isQuestionExist(question.getQuestionId())) {
            if  (questionDao.findQuestionById(question.getQuestionId()).getShield()
== 1) {
                question.setEnd(0);
                return questionDao.questionSignal(question);
            } else {
                return 0;
            }
        } else {
            return -1;
        }
    }
```

[END]

[Happy_Forum-UserQuestion-0016]

```
    public List<Question> sortQuestionByHeat() {
        List<Question> questionList = jdbcTemplate.query("select * from question
where shield = ? order by heat desc",new BeanPropertyRowMapper<>(Question.class),1);
        return questionList;
    }
```

[END]

[Happy_Forum-UserQuestion-0017]

```
    public List<Question> findQuestionsByKeyword(String strWord) {
        List<Question> questionList = jdbcTemplate.query("select * from question
where questionId like ? or details like ? or questioner like ? or label like ?",new
Object[]{"%"+strWord+"%","%"+strWord+"%","%"+strWord+"%","%"+strWord+"%"},new
BeanPropertyRowMapper<>(Question.class));
        return questionList;
```

```
    }
[END]
```

3.4.5 回答管理（删除、屏蔽、模糊查询）

在管理员系统中,通过列表展示所有回答信息。管理员可通过折叠板查看回答具体内容；可对回答进行屏蔽或取消屏蔽操作，被屏蔽的回答对其他人不可见，回答人自己可见，回答人不能进行修改或删除；可以对回答进行模糊查询；可以用选择器选择查看已屏蔽回答或未屏蔽回答。

成员名	成员类型（字段、方法或属性）	成员含义说明
findAllAnswer	List<Answer>	查询所有回答
findAllWhiteAnswers	List<Answer>	查询所有未屏蔽回答
findAllBlackAnswers	List<Answer>	查询所有已屏蔽回答
findAnswerById	Answer	根据回答编号查询回答
findAnswerByUserName	List<Answer>	根据用户名查询该用户所发布的所有回答
findWhiteAnswerByQuestionId	List<Answer>	根据问题编号查询该问题下未被屏蔽的所有回答
findWhiteVoAnswersByQuestionId	List<VoAnswer>	根据问题查询面向对象的回答数组 voAnswer
findWhiteVoCommentsByAnswerId	List<VoComment>	根据回答查询面向对象的评论数组 voComment
updateAnswer	int	更改回答
addAnswer	int	新增回答
deleteAnswerById	int	删除回答
blacklistAnswer	int	屏蔽回答
whitelistAnswer	int	解除屏蔽回答
blacklistAnswers	int	屏蔽一组回答
whitelistAnswers	int	解除屏蔽一组回答
findAnswersByQuestionId	List<Answer>	根据问题编号查找该问题所有的回答
findAnswersByKeyword	List<Answer>	模糊查询回答

```
[Happy_Forum-UserAnswer-0001]
    public List<Answer> findAllAnswer() {
        List<Answer> answerList = jdbcTemplate.query("select * from answer",new Object[]{},new BeanPropertyRowMapper<>(Answer.class));
        return answerList;
    }
[END]

[Happy_Forum-UserAnswer-0002]
    public List<Answer> findAllWhiteAnswers() {
```

```
        List<Answer> answerList = jdbcTemplate.query("select * from answer where shield = ?",new BeanPropertyRowMapper<>(Answer.class),1);
        return answerList;
    }
[END]
```

[Happy_Forum-UserAnswer-0003]

```
    public List<Answer> findAllBlackAnswers() {
        List<Answer> answerList = jdbcTemplate.query("select * from answer where shield = ?",new BeanPropertyRowMapper<>(Answer.class),0);
        return answerList;
    }
[END]
```

[Happy_Forum-UserAnswer-0004]

```
    public Answer findAnswerById(int id) {
        List<Answer> answerList = jdbcTemplate.query("select * from answer where answerId = ?",new BeanPropertyRowMapper<>(Answer.class),id);
        if(answerList.size()>0){
            return answerList.get(0);
        }else{
            return null;
        }
    }
[END]
```

[Happy_Forum-UserAnswer-0005]

```
    public List<Answer> findAnswersByUserName(String name) {
        List<Answer> answerList = jdbcTemplate.query("select * from answer where answerer = ?",new BeanPropertyRowMapper<>(Answer.class),name);
        return answerList;
    }
[END]
```

[Happy_Forum-UserAnswer-0006]

```
    public List<Answer> findWhiteAnswersByQuestionId(int id) {
        List<Answer> answerList = jdbcTemplate.query("select * from answer where aQuestionId = ? and shield = ? ",new BeanPropertyRowMapper<>(Answer.class),id,1);
        return answerList;
    }
[END]
```

[Happy_Forum-UserAnswer-0007]

```
    public List<VoAnswer> findWhiteVoAnswersByQuestionId(int id) {
        List<VoAnswer> voanswerList = jdbcTemplate.query("select * from answer where aQuestionId = ? and aAnswerId = ? and shield = ? order by time desc ",new
```

```
BeanPropertyRowMapper<>(VoAnswer.class),id,0,1);
        return voanswerList;
    }
[END]
```

[Happy_Forum-UserAnswer-0008]

```
    public List<VoComment> findWhiteVoCommentsByAnswerId(int id) {
        List<VoComment> voCommentList = jdbcTemplate.query("select * from answer where aAnswerId = ? and shield = ? order by startTime asc ",new BeanPropertyRowMapper<>(VoComment.class),id,1);
        return voCommentList;
    }
[END]
```

[Happy_Forum-UserAnswer-0009]

```
    public int updateAnswer(Answer answer) {
        int count = jdbcTemplate.update("update answer set details = ?,time = ? where answerId = ?",answer.getDetails(),FormatChange.dateTimeChange(new Date()),answer.getAnswerId());
        return count;
    }
[END]
```

[Happy_Forum-UserAnswer-0010]

```
    public int addAnswer(Answer answer) {
        int count = jdbcTemplate.update("insert into answer(answerId,details,answerer,aQuestionId,aAnswerId,shield,time,startTime) values(?,?,?,?,?,?,?,?)",maxAnswerId()+1,answer.getDetails(),answer.getAnswerer(),answer.getaQuestionId(),answer.getaAnswerId(),answer.getShield(),FormatChange.dateTimeChange(new Date()),FormatChange.dateTimeChange(new Date()));
        return count;
    }
[END]
```

[Happy_Forum-UserAnswer-0011]

```
    public int deleteAnswerById(int id) {
        int count = jdbcTemplate.update("delete from answer where answerId = ?",id);
        return count;
    }
[END]
```

[Happy_Forum-UserAnswer-0012]

```
    public int blacklistAnswer(Answer answer) {
        if (answerDao.isAnswerExist(answer.getAnswerId())) {
            if (answerDao.findAnswerById(answer.getAnswerId()).getShield() == 1) {
                answer.setShield(0);
```

```
                answerDao.answerShield(answer);
                Message message = new Message();

message.setRecipient(answerDao.findAnswerById(answer.getAnswerId()).getAnswerer());
                message.setAnswerId(answer.getAnswerId());
                messageDao.answerBlackMessage(message);
                return 1;
            } else {
                return 0;
            }
        } else {
            return -1;
        }
    }
[END]
```

[Happy_Forum-UserAnswer-0013]

```
    public int whitelistAnswer(Answer answer) {
        if (answerDao.isAnswerExist(answer.getAnswerId())) {
            if (answerDao.findAnswerById(answer.getAnswerId()).getShield() == 0) {
                answer.setShield(1);
                answerDao.answerShield(answer);
                Message message = new Message();

message.setRecipient(answerDao.findAnswerById(answer.getAnswerId()).getAnswerer());
                message.setAnswerId(answer.getAnswerId());
                messageDao.answerWhiteMessage(message);
                return 1;
            } else {
                return 0;
            }
        } else {
            return -1;
        }
    }
[END]
```

[Happy_Forum-UserAnswer-0014]

```
    public int blacklistAnswers(List<Answer> answerList) {
        for (int i = answerList.size(); i > 0; i--) {
            if             (answerDao.findAnswerById(answerList.get(i             -
1).getAnswerId()).getShield() == 1) {
                answerList.get(i - 1).setShield(0);
                answerDao.answerShield(answerList.get(i - 1));
            }
```

```
        }
        return answerList.size();
    }
[END]
```

[Happy_Forum-UserAnswer-0015]

```
    public int whitelistAnswers(List<Answer> answerList) {
        for (int i = answerList.size(); i > 0; i--) {
            if (answerDao.findAnswerById(answerList.get(i - 1).getAnswerId()).getShield() == 0) {
                answerList.get(i - 1).setShield(1);
                answerDao.answerShield(answerList.get(i - 1));
            }
        }
        return answerList.size();
    }
[END]
```

[Happy_Forum-UserAnswer-0016]

```
    public List<Answer> findAnswersByQuestionId(int id) {
        List<Answer> answerList = jdbcTemplate.query("select * from answer where aQuestionId = ?",new BeanPropertyRowMapper<>(Answer.class),id);
        return answerList;
    }
[END]
```

[Happy_Forum-UserAnswer-0017]

```
    public List<Answer> findAnswersByKeyword(String strWord) {
        List<Answer> answerList = jdbcTemplate.query("select * from answer where answerId like ? or details like ? or answerer like ?",new Object[]{"%"+strWord+"%","%"+strWord+"%","%"+strWord+"%"},new BeanPropertyRowMapper<>(Answer.class));
        return answerList;
    }
[END]
```

3.4.6 热度问题管理

在管理员系统中，通过列表展示所有热度问题。管理员可通过查看用户浏览次数以及查看记录。此外还可以手动修改热度。

成员名	成员类型（字段、方法或属性）	成员含义说明
findAllHeat	List<Heat>	查询所有 Heat 类信息
isUserExistInQuestion	int	查询某用户是否浏览过该问题

updateHeat	int	修改 Heat 类信息
addHeat	int	新增 Heat 类信息
deleteHeat	int	删除 Heat 类信息
updateHeatOfQuestion	int	修改问题的热度属性

[Happy_Forum-Heat-0001]

```
    public List<Heat> findAllHeat(){
        List<Heat>  heatList  =  jdbcTemplate.query("select  *  from  heat",new Object[]{},new BeanPropertyRowMapper<>(Heat.class));
        if(heatList.size()>0){
            return heatList;
        }else{
            return null;
        }
    }
```

[END]

[Happy_Forum-Heat-0002]

```
    public int isUserExistInQuestion(String phone, int questionId) {
        return heatDao.isUserExistInQuestion(phone,questionId);
    }
```

[END]

[Happy_Forum-Heat-0003]

```
    public int updateHeat(Heat heat) {
        int count = jdbcTemplate.update("update heat set phone = ?,questionId = ? where id = ?",heat.getPhone(),heat.getQuestionId(),heat.getId());
        return count;
    }
```

[END]

[Happy_Forum-Heat-0004]

```
    public int addHeat(Heat heat) {
        int  count  =  jdbcTemplate.update("insert  into  heat(id,phone,questionId) values(?,?,?)",maxId()+1,heat.getPhone(),heat.getQuestionId());
        return count;
    }
```

[END]

[Happy_Forum-Heat-0005]

```
    public int deleteHeat(int id) {
        int count = jdbcTemplate.update("delete from heat where id = ?",id);
        return count;
    }
```

[END]

```
[Happy_Forum-Heat-0006]
    public int updateHeatOfQuestion(Heat heat) {
        if (heat.getPhone() == null || heatDao.isUserExistInQuestion(heat.getPhone(),
heat.getQuestionId()) == 1) {
            return -1;
        } else {
            heatDao.addHeat(heat);
            Question question = questionDao.findQuestionById(heat.getQuestionId());
            question.setHeat(question.getHeat()+1);
            return questionDao.updateHeatOfQuestion(question);
        }
    }
[END]
```

4. 非功能需求

4.1 外部接口需求

4.1.1 用户接口

无特殊需求。

4.1.2 硬件接口

无特殊需求。

4.1.3 软件接口

无特殊需求。

4.1.4 通信接口

无特殊需求。

4.2 性能需求

分类		性能要求	适用功能
静态页面类	静态页面类	响应时间≤1 秒	静态页面

事务处理类	普通响应类	响应时间≤5 秒	仅在网站系统完成事物处理的操作，例如：登录、注册等
	跨系统响应类	响应时间≤10 秒	需要在网站系统和其它业务系统交互共同完成的操作，例如：广告等（暂未加入）
查询类	小数据量简单查询类	响应时间≤2 秒	仅在网站系统完成数据查询的操作，例如：帐户信息等
	小数据量复杂多重查询类	响应时间≤5 秒	仅在网站系统完成数据查询的操作，例如：我的消息等
	批量数据简单查询类	响应时间≤10 秒	仅在网站系统完成数据查询的操作
	跨系统查询类	响应时间≤15 秒	需要网站系统和其它业务系统交互共同完成数据查询的操作
	复杂分析查询类	响应时间≤15 秒	需要在网站系统和其它业务系统交互共同完成数据查询，并做图表分析显示的操作
	模糊查询类	响应时间≤10 秒	仅在网站系统完成数据查询的操作，例如：问题搜索等
统计类	简单统计类	响应时间≤10 秒	如：帐户状态统计
	复杂统计类	响应时间≤30 秒	如：流量分析、用户行为分析

4.3 安全需求

4.3.1 权限控制

根据不同用户角色，设置相应权限，用户的重要操作都做相应的日志记录以备查看。

4.3.2 重要数据加密

对一些重要的数据按一定的算法（如 SHA256 等）进行加密，如用户密码等。

4.3.3 数据备份

允许后台管理人员对数据库及缓存中数据进行备份。

4.3.4 记录日志

本系统应该能够记录系统运行时所发生的所有错误，包括本机错误和网络错误。这些

错误记录便于查找错误的原因。日志同时记录用户的关键性操作信息。