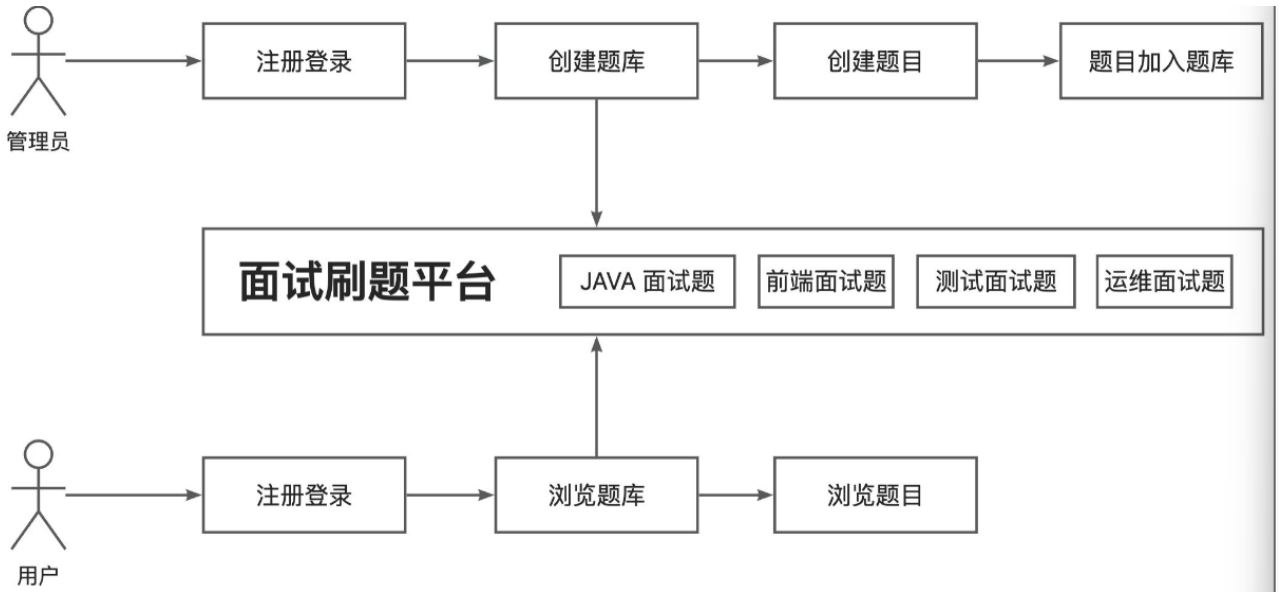
一、需求分析

需求优先级：

根据核心业务业务流程，明确需求开发优先级

- P0为核心功能，非做不可。
- P1为重点功能，尽量都去实现。
- P2为实用的拓展功能，能做就做。
- P3可做可不做，搞不搞全看心情和后续需求。

核心业务流程



项目功能梳理+需求优先级：

基础功能（P0）：

- 用户模块
  - 用户注册
  - 用户登录（账号密码）
  - 【管理员】管理用户 - 增删改查
- 题库模块
  - 查看题库列表
  - 查看题库详情（展示题库下的题目）
  - 【管理员】管理题库 - 增删改查1659491662109405186_0.19640093190871566
- 题目模块
  - 题目搜索
  - 查看题目详情（进入刷题页面）
  - 【管理员】管理题目 - 增删改查（比如按照题库查询题目、修改题目所属题库等）

高级功能（P1-P2）：

- 题目**批量**管理（提高管理效率）P1
    - 【管理员】批量向题库添加题目
    - 【管理员】批量从题库移除题目
    - 【管理员】批量删除题目1659491662109405186_0.09510289639229197
- 分词题目搜索 P1
- 用户刷题记录日历图 P1
- 自动缓存热门题目 P2
- 网站流量控制和熔断 P2
- 动态 IP 黑白名单过滤 P2
- 同端登录冲突检测 P2
- 分级题目反爬虫策略 P2

鱼皮的面试鸭上的一些真实的功能需求列表

	开始日期	项目模块	工作事项	负责人	优先级	类型	工作详情	工作进度
1	2024年5月9日	用户模块	注册	鱼皮	P0	新增	用户注册功能实现	0%
2	2024年5月9日	用户模块	登录	鱼皮	P0	新增	用户登录功能实现	0%
3	2024年5月9日	用户模块	管理	鱼皮	P1	新增	用户增删改查-仅管理员可用	0%
4	2024年5月9日	应用模块	创建应用	鱼皮	P0	新增	用户创建应用，输入应用名、描述等字段	0%
5	2024年5月9日	应用模块	修改应用	鱼皮	P1	新增	用户修改应用内容，可修改应用名、描述等字段	0%
6	2024年5月9日	应用模块	删除应用	鱼皮	P1	新增	用户删除自己创建的应用	0%
7	2024年5月9日	应用模块	查看应用列表	鱼皮	P0	新增	用户查看所有的应用	0%
8	2024年5月9日	应用模块	查看应用详情	鱼皮	P0	新增	用户点击某个应用，展示应用的详情	0%
9	2024年5月9日	应用模块	查看自己创建的应用	鱼皮	P1	新增		0%
10	2024年5月9日	应用模块	管理应用	鱼皮	P0	新增	增删改查，仅管理员可用	0%
11	2024年5月9日	应用模块	审核发布和下架应用	鱼皮	P0	新增	仅管理员可用	0%
12	2024年5月9日	应用模块	应用分享	鱼皮	P2	新增	根据当前应用生成一个二维码，用户扫码即可访问这个应用	0%
13	2024年5月9日	题目模块	创建题目	鱼皮	P0	新增	用户创建应用后需要创建题目	0%
14	2024年5月9日	题目模块	修改题目	鱼皮	P1	新增	用户修改题目标题或选项	0%
15	2024年5月9日	题目模块	删除题目	鱼皮	P1	新增	用户删除指定的题目	0%
16	2024年5月9日	题目模块	管理题目	鱼皮	P1	新增	增删改查，仅管理员可用	0%
17	2024年5月9日	题目模块	AI 生成题目	鱼皮	P1	新增	利用 AI 一键生成题目	0%
18	2024年5月9日	评分模块	删除评分结果	鱼皮	P1	新增		0%
19	2024年5月9日	评分模块	根据回答计算评分结果	鱼皮	P0	新增	含多个评分策略，包括自定义测评类评分规则、自定义打分类评...	0%
20	2024年5月9日	评分模块	创建评分结果	鱼皮	P0	新增		0%
21	2024年5月9日	评分模块	修改评分结果	鱼皮	P1	新增		0%
22	2024年5月9日	评分模块	管理评分结果	鱼皮	P1	新增	增删改查，仅管理员可用	0%

二、库表设计

根据需求中的三个模块，分为用户表、题库表、题目表、题库题目关联表共4张核心表。

库名：mianshiba

数据库初始化脚本（navicat创建直接创建即可）：

```
/*
 Navicat Premium Data Transfer
 Source Server Type    : MySQL
 Source Schema         : mianshiba
 Date: 28/08/2024 23:54:47
*/

SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

SET FOREIGN_KEY_CHECKS = 1;
```

1、用户表

核心设计

用户表的核心是用户登录凭证（账号密码）和个人信息，SQL 如下：

updateTime与editTime区别：updateTime是一个系统级别的控制，比如管理员强制把unionId修改后，updateTime会发生变化而editTime不应发生变化。

当**逻辑删除（isDelete == true）数据过多**时，可以把数据迁移至其他的额外的表中做备份与归纳，同时要考虑好恢复策略，对类似日志这种价值相对较低的内容给可以放到存储代价更低的介质中进行“冷存”。

```
-- 用户表
create table if not exists user
(
    id           bigint auto_increment comment 'id' primary key,
    userAccount  varchar(256)                           not null comment '账号',
    userPassword varchar(512)                           not null comment '密码',
    unionId      varchar(256)                           null comment '微信开放平台id',
    mpOpenId     varchar(256)                           null comment '公众号openId',
    userName     varchar(256)                           null comment '用户昵称',
    userAvatar   varchar(1024)                          null comment '用户头像',
    userProfile  varchar(512)                           null comment '用户简介',
    userRole     varchar(256) default 'user'            not null comment '用户角色:
user/admin/ban',
    editTime     datetime     default CURRENT_TIMESTAMP not null comment '编辑时间',
    createTime   datetime     default CURRENT_TIMESTAMP not null comment '创建时间',
    updateTime   datetime     default CURRENT_TIMESTAMP not null on update
CURRENT_TIMESTAMP comment '更新时间',
    isDelete     tinyint      default 0                 not null comment '是否删除',
    index idx_unionId (unionId)
) comment '用户' collate = utf8mb4_unicode_ci;
```

其中，unionId、mpOpenId 是为了实现公众号登录的，也可以省略。每个微信用户在同一家公司（主体）的 unionId 是唯一的，在**同一个公众号的 mpOpenId 是唯一的。**（同一家用户有多个产品时，可以同时在多个产品之间追踪同一用户）

拓展设计

（1）如果要实现会员功能，可以对表进行如下扩展：

1. 给 userRole 字段新增枚举值 `vip`，表示会员用户，可根据该值判断用户权限
2. 新增会员过期时间字段，可用于记录会员有效期
3. 新增会员兑换码字段，可用于记录会员的开通方式1659491662109405186_0.729305784461405
4. 新增会员编号字段，可便于定位用户并提供额外服务，并增加会员归属感

   对应sql：

```
vipExpireTime datetime     null comment '会员过期时间',
vipCode       varchar(128) null comment '会员兑换码',
vipNumber     bigint       null comment '会员编号'
```

（2）如果要实现用户邀请功能，可以对表进行如下扩展：

1. 新增 shareCode 分享码字段，用于记录每个用户的唯一邀请标识，可拼接到邀请网址后面，比如 https://mianshi8.com/?shareCode=xxx
2. 新增 inviteUser 字段，用于记录该用户被哪个用户邀请了，可通过这个字段查询某用户邀请的用户列表。
3. 形式可以为生成邀请二维码或者邀请海报之类的。

   对应sql：

```
shareCode     varchar(20)  DEFAULT NULL COMMENT '分享码',
inviteUser    bigint       DEFAULT NULL COMMENT '邀请用户 id'
```

2、题库表

核心设计

题库表的核心是题库信息（标题、描述、图片），SQL 如下：

```
-- 题库表
create table if not exists question_bank
(
    id          bigint auto_increment comment 'id' primary key,
    title       varchar(256)                       null comment '标题',
    description text                               null comment '描述',
    picture     varchar(2048)                      null comment '图片',
    userId      bigint                             not null comment '创建用户 id',
    editTime    datetime default CURRENT_TIMESTAMP not null comment '编辑时间',
    createTime  datetime default CURRENT_TIMESTAMP not null comment '创建时间',
    updateTime  datetime default CURRENT_TIMESTAMP not null on update CURRENT_TIMESTAMP
comment '更新时间',
    isDelete    tinyint  default 0                 not null comment '是否删除',
    index idx_title (title)
) comment '题库' collate = utf8mb4_unicode_ci;
```

当text不够使的时候可以用mediumText（而非longText，太大了容易遭攻击，如果校验没做好可能会被盗刷流量）

其中，picture 存储的是图片的 url 地址，而不是完整图片文件。通过 userId 和用户表关联，在本项目中只有管理员才能创建题库。

由于用户很可能按照标题搜索题库，所以给 title 字段增加索引。

拓展设计

（1）如果要实现题库审核功能，可以对表进行如下扩展（鱼皮的AI答题应用平台中有讲实现，课程链接：AI答题应用平台链接）：

1. 新增审核状态字段，用枚举值表示待审核、通过和拒绝
2. 新增审核信息字段，用于记录未过审的原因等
3. 新增审核人 id 字段，便于**审计**操作。比如出现了违规内容过审的情况，可以追责到审核人。
4. 新增审核时间字段，也是便于审计。

   对应sql如下：

```
reviewStatus  int      default 0  not null comment '状态：0-待审核，1-通过，2-拒绝',
reviewMessage varchar(512)        null comment '审核信息',
reviewerId    bigint              null comment '审核人 id',
reviewTime    datetime            null comment '审核时间'
```

（2）如果要实现题库排序功能，可以新增整型的优先级字段，并且根据该字段排序。

1. 该字段还可以用于快速实现题库精选和置顶功能，比如优先级 = 1000 的题库表示精选，优先级 = 10000 的题库表示置顶。

   对应的 SQL 如下：

```
priority  int  default 0  not null comment '优先级'
```

（3）如果要实现题库浏览功能，可以新增题库浏览数字段，每次进入题目详情页时该字段的值 +1，还可以根据该字段对题库进行排序。对应的 SQL 如下：

```
viewNum  int  default 0  not null comment '浏览量'
```

同时，访问量也可以为精选与排序提供一个分析维度。

如果要实现用户浏览数（同一个用户浏览数最多 +1），还需要额外的题库浏览记录表，可以用来防止同一用户刷浏览量。

3、题目表

核心设计

题目表的核心是题目信息（标题、详细内容、标签），SQL 如下：

```
-- 题目表
create table if not exists question
(
    id         bigint auto_increment comment 'id' primary key,
```

```
    title      varchar(256)                       null comment '标题',
    content    text                               null comment '内容',
    tags       varchar(1024)                      null comment '标签列表（json 数组）',
    answer     text                               null comment '推荐答案',
    userId     bigint                             not null comment '创建用户 id',
    editTime   datetime default CURRENT_TIMESTAMP not null comment '编辑时间',
    createTime datetime default CURRENT_TIMESTAMP not null comment '创建时间',
    updateTime datetime default CURRENT_TIMESTAMP not null on update CURRENT_TIMESTAMP
comment '更新时间',
    isDelete   tinyint  default 0                 not null comment '是否删除',
    index idx_title (title),
    index idx_userId (userId)
) comment '题目' collate = utf8mb4_unicode_ci;
```

注意事项：

1. 题目标题 title 和题目创建人 userId 是常用的题目搜索条件，所以添加索引提升查询性能。
2. 题目**可能有多个标签**，为了简化设计，没有采用关联表，而是以 JSON 数组字符串的方式存储，比如 `["Java", "Python"]`。
3. 题目内容（详情）和题目答案可能很长，所以使用 text 类型。

拓展设计：

（1）如果要实现题目审核功能，可以参考上述题库审核功能，新增 4 个字段即可：

```
reviewStatus  int      default 0  not null comment '状态：0-待审核，1-通过，2-拒绝',
reviewMessage varchar(512)        null comment '审核信息',
reviewerId    bigint              null comment '审核人 id',
reviewTime    datetime            null comment '审核时间'
```

（2）可能有很多评价题目的指标，比如浏览数、点赞数、收藏数，参考字段如下：

```
viewNum       int      default 0    not null comment '浏览量',
thumbNum      int      default 0    not null comment '点赞数',
favourNum     int      default 0    not null comment '收藏数'
```

（3）如果要实现题目排序、精选和置顶功能，可以参考上述题库表的设计，新增整型的优先级字段，并且根据该字段排序。对应的 SQL 如下：

```
priority  int  default 0  not null comment '优先级'
```

（4）如果题目是从其他网站或途径获取到的，担心有版权风险，可以增加题目来源字段。最简单的实现方式就是直接存来源名称：

```
source   varchar(512)  null comment '题目来源'
```

（5）如果想设置部分题目仅会员可见，可以给题目表加上一个“是否仅会员可见”的字段，本质上是个布尔类型，用 1 表示仅会员可见。参考 SQL 如下：

```
needVip  tinyint  default 0  not null comment '仅会员可见（1 表示仅会员可见）'
```

4、题库题目关系表

核心设计

由于一个题库可以有多个题目，一个题目可以属于多个题库，所以需要关联表来实现。

实现基础功能的 SQL 如下：

```
-- 题库题目表（硬删除）
create table if not exists question_bank_question
(
    id             bigint auto_increment comment 'id' primary key,
    questionBankId bigint                             not null comment '题库 id',
    questionId     bigint                             not null comment '题目 id',
    userId         bigint                             not null comment '创建用户 id',
    createTime     datetime default CURRENT_TIMESTAMP not null comment '创建时间',
    updateTime     datetime default CURRENT_TIMESTAMP not null on update CURRENT_TIMESTAMP
comment '更新时间',
    UNIQUE (questionBankId, questionId)
) comment '题库题目' collate = utf8mb4_unicode_ci;
```

几个重点：

1. 上述代码中的 userId 表示添加题目到题库的用户 id，仅管理员可操作
2. 由于关联表中的数据记录并没有那么重要（一般由管理员维护），所以直接采用硬删除的方式，如果将题目移出题库，直接删掉对应的数据即可。按照这种设计，createTime 就是题目加入到题库的时间。
3. 通过给题库 id 和题目 id 添加 **联合唯一索引**（可以出面试题，为啥给这两个字段索引？为啥用联合索引？为啥quertionBankId要放在questionId前面？），防止题目被重复添加到同一个题库中。而且要注意，将 questionBankId 放到前面，因为数据库中的查询大多是基于 `questionBankId` 进行的（大多数情况要用题库的id去查题目的id），比如查找某个题库中的所有问题，或者在一个题库中查找特定问题，将 `questionBankId` 放在前面符合查询模式，会使得这些查询更加高效（索引的**最左前缀原则**）。

拓展设计

（1）如果要对题库内的题目进行排序，可以增加题目顺序字段（整型）。对应的 SQL 如下：

```
questionOrder  int  default 0  not null comment '题目顺序（题号）'
```

需要注意，如果要实现**任意移动题目顺序**的功能，可能每次要更新多条记录的顺序，比较影响性能。如果追求更高性能的话，可以先在内存中计算出要变更的题目顺序，以减少更新的记录数。比如将第 100 题移动到第 98 题，只需要修改几条记录的顺序，不影响前面的题目（总结：只更新需要更新的部分）。