项目名称：tzlOJ

文档名称：概要设计说明书

*HUSTZL*

*侯皓斐 软件2003班 U202010851*

*刘铭宸 软件2003班 U202010783*

*甘凤轩 软件2003班 U202010822*

版本：V2.1

文 档 信 息

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√ ] 正式发布  [ ] 正在修改 | 文件标识： | HUSTZL\_005 |
| 当前版本： | V2.1 |
| 作者： | 侯皓斐 |
| 发布日期： | < 2023.4.23 > |

文 档 更 改 记 录

| 版本 | 更改日期 | 更改人 | 更改原因 | 说明 |
| --- | --- | --- | --- | --- |
| V1.0 | 2023.4.15 | 侯皓斐 |  | 初版 |
| V2.0 | 2023.4.20 | 甘凤轩 | 校对 | 勘误并调整行文语句 |
| V2.1 | 2023.4.23 | 甘凤轩 | 调整格式 | 完善文档大纲，调整段落格式，修改序列编号 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1.系统概述 1](#_Toc133158502)

[1.1系统任务 1](#_Toc133158503)

[1.1.1系统目标 1](#_Toc133158504)

[1.1.2运行环境 1](#_Toc133158505)

[1.1.3与其他系统关系 1](#_Toc133158506)

[2.架构设计 1](#_Toc133158507)

[2.1系统物理结构 1](#_Toc133158508)

[2.1.1系统部署图 1](#_Toc133158509)

[2.1.2 设备清单 2](#_Toc133158510)

[2.2系统软件结构 3](#_Toc133158511)

[2.2.1模块结构图 3](#_Toc133158512)

[2.2.2模块（子系统）清单 4](#_Toc133158513)

[3.前端设计 5](#_Toc133158514)

[3.1前端结构设计 5](#_Toc133158515)

[3.1.1需求规定 5](#_Toc133158516)

[3.1.2运行环境 6](#_Toc133158517)

[3.1.3基本设计思想和处理流程 7](#_Toc133158518)

[3.1.4质量设计 8](#_Toc133158519)

[3.1.5结构 9](#_Toc133158520)

[3.1.6功能与程序结构的关系 10](#_Toc133158521)

[3.2前端接口设计 10](#_Toc133158522)

[3.2.1用户接口 10](#_Toc133158523)

[3.2.2外部接口 17](#_Toc133158524)

[3.3前端数据结构设计 18](#_Toc133158525)

[3.3.1逻辑结构设计要点 18](#_Toc133158526)

[3.3.2物理结构设计要点 18](#_Toc133158527)

[3.3.3数据结构与程序的关系 18](#_Toc133158528)

[3.4质量与模块关系 18](#_Toc133158529)

[4.后端设计 19](#_Toc133158530)

[4.1后端结构设计 19](#_Toc133158531)

[4.1.1运行环境 19](#_Toc133158532)

[4.1.2基本设计思想和处理流程 20](#_Toc133158533)

[4.1.3质量设计 20](#_Toc133158534)

[4.1.4结构设计 22](#_Toc133158535)

[4.1.5功能与程序结构的关系 23](#_Toc133158536)

[4.2后端接口设计 24](#_Toc133158537)

[4.2.1用户接口 24](#_Toc133158538)

[4.2.2外部接口 24](#_Toc133158539)

[4.3后端数据结构设计 24](#_Toc133158540)

[4.3.1逻辑结构设计要点 25](#_Toc133158541)

[4.3.2物理结构设计要点 26](#_Toc133158542)

[4.4质量与模块关系 26](#_Toc133158543)

# 1.系统概述

## 1.1系统任务

### 1.1.1系统目标

tzlOJ系统是一个面向高校教学和培训的在线评测系统，其主要目标是为用户提供一个方便的平台来提交编程题目的代码并进行自动评测，同时也提供相关的竞赛和排名功能。

### 1.1.2运行环境

OnlineJudge系统需要在多个Linux或Unix的Web服务器上运行，以支持用户通过Web界面访问和使用系统。其中包含前后端服务器，数据库服务器，测评服务器（多个CPU性能较强的服务器），从而保证系统在高并发的环境下稳定运行。

### 1.1.3与其他系统关系

OnlineJudge系统需要与用户的本地编程环境、数据库系统和网络环境进行交互，以实现提交代码、编译运行、获取评测结果等功能。此外，系统也需要与用户的账户系统进行集成，以实现用户注册、登录、身份验证等功能。在竞赛模式下，系统还需要与排名系统进行集成，以计算用户的得分和排名。为了保证系统的安全性，系统还需要与防火墙、反恶意软件系统等进行集成。

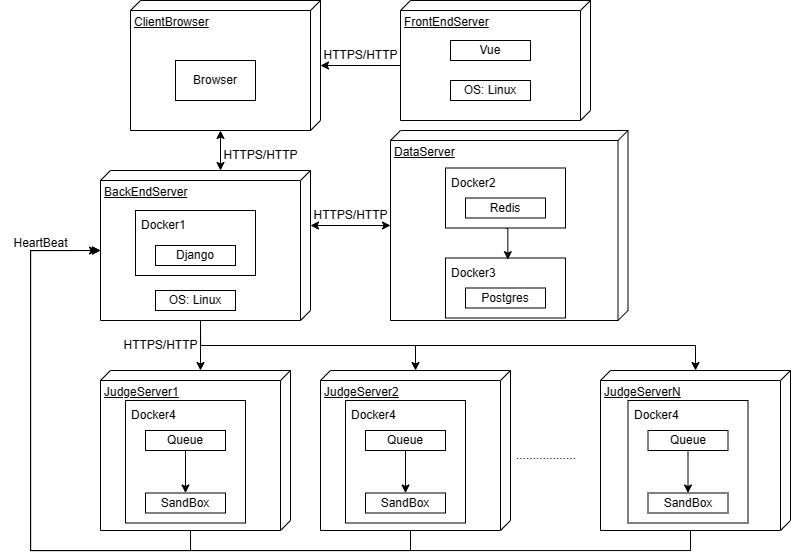
# 2.架构设计

## 2.1系统物理结构

### 2.1.1系统部署图

我们采用docker进行部署，同时可以使用多个物理机，提高我们测评的效率，我们知道，性能限制出现在测评环节，即测评机进程，而且测评机进程极容易被恶意程序卡崩，所以必须采取冗余的战术和心跳的技术，当测评机出现问题时，后端及时调整测评。

我们的物理部署图如下：



### 2.1.2 设备清单

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 设备名称 | 数 量 | 型号和规格 |
| 1 | FrontEndServer | 1 | Dell PowerEdge R740xd服务器，配置为：  CPU：2个Intel Xeon Gold 6254处理器，共54核心  内存：256GB DDR4 ECC内存  存储：6 x 1.92TB SAS SSD（RAID 10）  网络：4 x 10GbE网络接口 |
| 2 | BackEndServer | 1 | HP ProLiant DL360 Gen10服务器，配置为：  CPU：2个Intel Xeon Gold 6240处理器，共40核心  内存：512GB DDR4 ECC内存  存储：8 x 1.92TB SAS SSD（RAID 10）  网络：4 x 10GbE网络接口 |
| 3 | DataServer | 1 | Lenovo ThinkSystem SR950服务器，配置为：  CPU：4个Intel Xeon Platinum 8276处理器，共112核心  内存：4TB DDR4 ECC内存  存储：16 x 1.92TB SAS SSD（RAID 10）  网络：4 x 10GbE网络接口 |
| 4 | JudgeServer | 1 | HP ProLiant DL360 Gen10服务器，配置为：  CPU：2个Intel Xeon Gold 6240处理器，共40核心  内存：512GB DDR4 ECC内存  存储：8 x 1.92TB SAS SSD（RAID 10）  网络：4 x 10GbE网络接口 |

## 2.2系统软件结构

### 2.2.1模块结构图

采用三层B/S模式，MVC模式和分层架构模式的架构图如下：

图示

描述已自动生成

将系统分为三个层次：用户界面层、应用层、领域和基础设施层。

用户界面层：用户界面层是OnlineJudge系统的顶层，负责与用户进行交互。该层提供用户界面和交互逻辑，包括学生提交和管理员管理等功能。

应用层：应用层负责处理系统的业务逻辑和数据处理。该层包括自动评测系统、用户管理管理和比赛管理等模块。题目管理系统负责管理题目题面和测试数据等信息，测试管理负责利用SandBox自动编译和运行学生提交的代码，并生成评测报告，学生提交作业和查看提交历史等功能，用户管理负责管理账号提供用户信息，比赛管理负责比赛及其所属的题目的管理。排行榜子系统负责用户积分等信息，实现排行榜功能。

领域和基础设施层：领域和基础设施层负责系统的数据模型和数据访问。该层包括用户信息、成绩信息和提交历史等领域模型，以及数据访问接口和服务，而且是OnlineJudge系统的底层支撑，负责提供系统的底层服务和基础设施，特别是运行和编译程序的SandBox。我们由于约束条件按照设计我们采用了第三方API，使用docker部署和运行，我们redis，postgres均将使用官方提供的标准docker，SandBox将使用HUSTOJ的沙箱运行环境的docker，三者均直接从云端拉取。我们不再仔细开发领域与基础设施层，调用开源软件调用其API。

### 2.2.2模块（子系统）清单

前端子系统清单：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 模块名称 | 模块标识 | 输入 | 处理 | 输出 |
| 1 | 学生提交模块 | submission | 学生代码、题目ID、语言类型 | 检查提交格式、向后端发送评测请求 | 提交结果、评测报告 |
| 2 | 管理员管理模块 | admin | 管理员登录信息、题目信息、比赛信息 | 管理员身份验证、管理题目和比赛信息 | 成功或失败信息 |
| 3 | 用户界面模块 | user | 用户登录信息、题目信息、排行榜信息 | 显示用户信息、题目信息、排行榜信息 | 用户界面显示 |

后端子系统清单：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 模块名称 | 模块标识 | 输入 | 处理 | 输出 |
| 1 | 自动评测系统 | judge | 学生代码、题目信息、测试数据 | 编译和运行学生代码，生成评测报告 | 评测结果、评测报告 |
| 2 | 题目管理系统 | problem | 题目ID、题目信息、测试数据 | 管理题目和测试数据 | 题目和测试数据的增删改查 |
| 3 | 用户管理 | user | 用户信息、登录信息 | 管理用户信息 | 用户信息的增删改查 |
| 4 | 比赛管理 | contest | 比赛信息、题目信息 | 管理比赛和题目信息 | 比赛和题目信息的增删改查 |
| 5 | 排行榜子系统 | leaderboard | 用户积分信息、排行榜信息 | 管理用户积分信息和排行榜信息 | 排行榜信息的查询和更新 |
| 6 | 评测机管理模块 | JudgeManagement | 测评的代码，题目，数据 | 进行数据分析 | 合适的评测机 |

Redis子系统清单：（使用第三方现成库）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 模块名称 | 模块标识 | 输入 | 处理 | 输出 |
| 1 | Redis缓存模块 | redis | 数据库读写请求 | 查询缓存、更新缓存 | 查询结果或更新结果 |

Postgres子系统清单：（使用第三方现成库）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 模块名称 | 模块标识 | 输入 | 处理 | 输出 |
| 1 | 数据管理模块 | DataManagement | 信息 | 存储信息 | 存储结果 |
| 2 | 数据访问接口 | DataAccess | 数据库访问请求 | 查询和操作数据库 | 数据库查询和操作结果 |

沙箱子系统清单：（使用第三方现成库）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 模块名称 | 模块标识 | 输入 | 处理 | 输出 |
| 1 | 编译模块 | Compilation | 用户提交的代码和编译选项 | 编译用户提交的代码 | 编译结果 |
| 2 | 执行模块 | Execution | 用户提交的代码和测试数据 | 运行用户提交的代码 | 运行结果和输出 |
| 3 | 安全模块 | Security | 用户提交的代码和运行环境 | 隔离用户提交的代码并保证安全 | 隔离结果 |

# 3.前端设计

## 3.1前端结构设计

### 3.1.1需求规定

详细请看需求规约文档，现将关键点重点显示。

1. 用户登录与注册：用户应该可以通过输入用户名和密码来进行登录，如果没有账号，用户应该可以通过注册页面创建新的账号。
2. 题目列表：系统应该能够展示所有可用的编程题目列表，并能够根据用户输入的搜索条件来筛选和排序题目。
3. 题目详情页：用户应该可以查看题目的详情页，包括题目描述、输入输出样例、评测结果和题目提交历史等信息。用户应该可以通过提交代码来尝试解决该题目。
4. 代码编辑器：系统应该提供一个代码编辑器，用户可以在其中编辑和提交自己的代码。编辑器应该支持多种编程语言，并提供自动缩进、语法高亮和自动补全等功能。
5. 评测结果页面：用户应该可以查看自己提交的代码的评测结果，包括编译错误、运行错误、通过测试点的数量等信息。
6. 个人信息页面：用户应该可以查看和编辑自己的个人信息，包括用户名、密码、邮箱等信息。
7. 比赛页面：系统应该支持比赛功能，用户可以通过比赛页面查看和参加系统中的比赛。比赛页面应该展示比赛的基本信息、排名、比赛规则和题目列表等信息。
8. 管理员页面：管理员应该可以通过管理员页面来管理系统中的用户、题目和比赛等信息。管理员页面应该提供用户管理、题目管理和比赛管理等功能。
9. 积分系统和排行榜：系统应该支持积分系统和排行榜功能，用户可以通过积累积分和参加比赛等活动来提高自己的排名。排行榜页面应该展示用户的排名和积分等信息。

### 3.1.2运行环境

Ubuntu20.04， NodeJs v8.12.0， VueJs

Dell PowerEdge R740xd服务器，配置为：

CPU：2个Intel Xeon Gold 6254处理器，共54核心

内存：256GB DDR4 ECC内存

存储：6 x 1.92TB SAS SSD（RAID 10）

网络：4 x 10GbE网络接口

### 3.1.3基本设计思想和处理流程

OnlineJudge系统的基本设计思想是提供一个支持多种编程语言的自动评测系统，使学生可以在线提交代码，并获得自动评测结果。系统采用前后端分离的架构，前端负责提供用户界面和交互逻辑，后端负责处理系统的业务逻辑和数据处理。

前端模块的基本设计思想和处理流程如下：

1. 用户登录和注册模块：用户可以在前端界面输入用户名和密码进行登录，如果没有账号，用户可以通过注册页面创建新的账号。
2. 题目列表和题目详情模块：前端界面展示所有可用的编程题目列表，并支持用户输入搜索条件来筛选和排序题目。用户可以查看题目详情页，包括题目描述、输入输出样例、评测结果和提交历史等信息。
3. 代码编辑和提交模块：前端界面提供代码编辑器，用户可以在其中编辑和提交自己的代码。编辑器支持多种编程语言，并提供自动缩进、语法高亮和自动补全等功能。
4. 评测结果页面和排行榜模块：前端界面展示评测结果页面，用户可以查看自己提交的代码的评测结果，包括编译错误、运行错误、通过测试点的数量等信息。排行榜页面展示用户的排名和积分等信息。

系统的整体处理流程如下：

1. 用户登录：用户通过前端界面输入用户名和密码进行登录，系统通过后端接口验证用户身份。
2. 题目列表：用户可以在前端界面中查看所有可用的编程题目列表，并选择相应的题目进行解答。
3. 代码编辑：用户在前端界面中输入代码，并提交到后端进行评测。
4. 代码评测：后端评测机根据题目要求，自动编译和运行用户提交的代码，并根据测试数据生成评测报告。
5. 评测结果：后端评测机将评测结果返回给前端界面，用户可以在前端界面中查看评测结果，包括编译错误、运行错误和通过测试点的数量等信息。
6. 排行榜：系统支持积分系统和排行榜功能，用户可以通过积累积分和参加比赛等活动来提高自己的排名。排行榜页面展示用户的排名和积分等信息。

我们可以用时序图来描述OnlineJudge系统的核心的处理流程（题目测评），如下图所示：

图示

描述已自动生成

### 3.1.4质量设计

对于一个OnlineJudge系统最重要的质量属性是性能和安全。以下是前端质量设计的具体描述：

性能：性能是前端设计的一个重要方面，对于OnlineJudge系统而言，性能表现主要体现在响应时间和吞吐量方面。系统需要提供快速的响应时间，使用户可以快速地提交代码并获得评测结果。同时，系统需要支持高吞吐量，以便支持大量的用户并发访问与测评。

安全：安全是前端设计的另一个重要方面，对于OnlineJudge系统而言，安全包括以下方面：安全控制和物理保护措施：系统采用安全控制措施和物理保护措施来保护系统的安全，例如防火墙、加密传输、防病毒措施等。

1. 用户身份鉴别机制：系统采用身份鉴别机制来保证用户的身份安全，例如使用用户名和密码进行登录、使用双因素认证等。
2. 用户对系统的访问权限和范围：系统根据用户的角色和权限，限制用户对系统的访问范围和操作权限，以保证系统的安全性。
3. 病毒的防治措施：系统采用病毒的防治措施，以防止系统受到病毒或恶意软件的攻击。
4. 数据加密方法：系统采用数据加密方法，以保证用户数据的安全，例如使用SSL/TLS加密传输、使用加密算法对用户数据进行加密等。

### 3.1.5结构

前端主要采用调用后端的方法获取数据，而本身需要存储的数据十分少，所以应该对不同的页面进行不同的设计，实现各个页面的嵌套关系以及跳转关系即可。

|  |  |  |
| --- | --- | --- |
| **类结构及类** | **标识符** | **功能** |
| 用户登录模块 | UserLoginPage | 负责用户登录和验证 |
| 题目列表模块 | ProblemListPage | 负责展示所有可用的编程题目列表 |
| 题目详情模块 | ProblemDetailPage | 负责展示题目详情页，包括题目描述、输入输出样例、评测结果和提交历史等信息 |
| 代码编辑模块 | CodeEditorPage | 负责提供代码编辑器，支持多种编程语言，并提供自动缩进、语法高亮和自动补全等功能 |
| 代码提交模块 | CodeSubmitPage | 负责将用户提交的代码发送到后端评测机进行评测 |
| 评测结果页面模块 | SubmissionResultPage | 负责展示评测结果页面，用户可以查看自己提交的代码的评测结果，包括编译错误、运行错误、通过测试点的数量等信息 |
| 排行榜模块 | RankListPage | 负责展示用户的排名和积分等信息 |
| 后台模块 | AdminPage | 负责进行管理员操作，增加题目，修改题目，举办比赛等 |

### 3.1.6功能与程序结构的关系

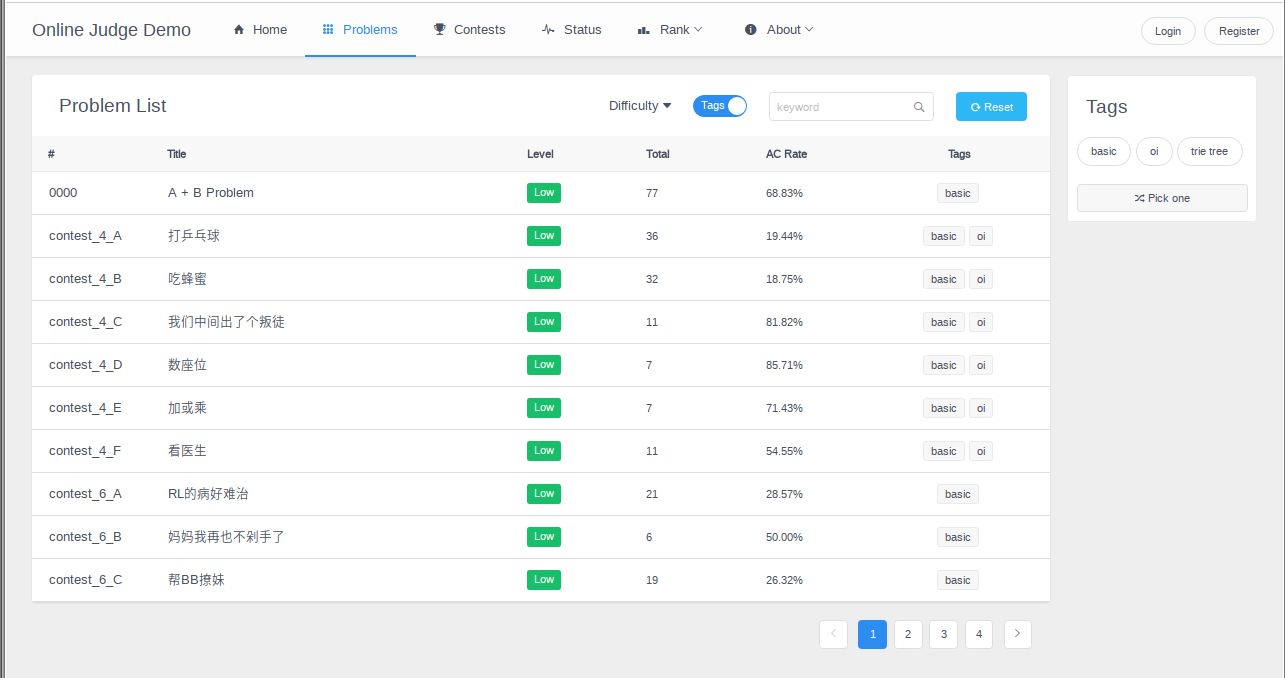
以下是各模块的具体功能和与程序模块的关系：

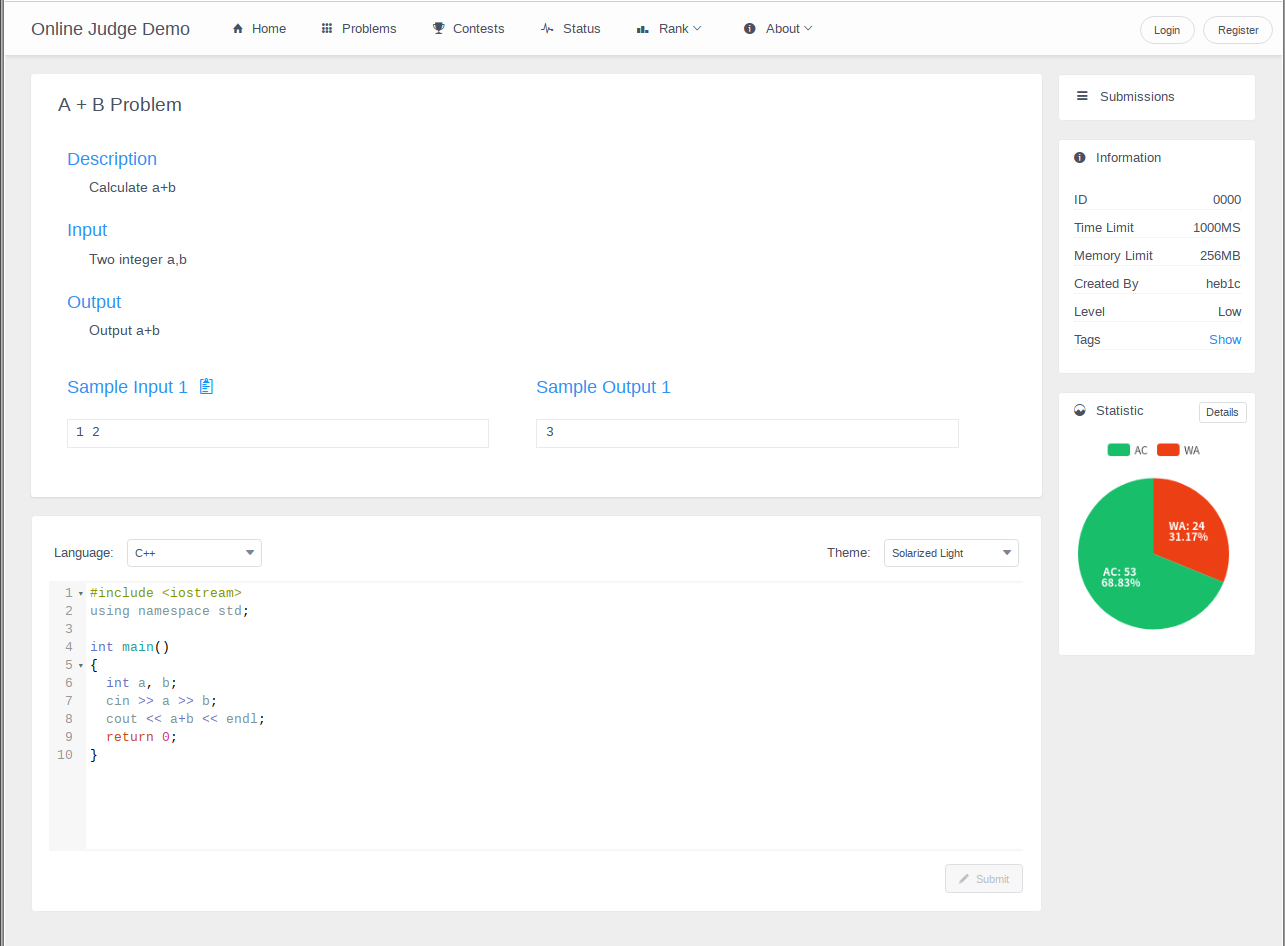
1. 用户登录模块：负责用户登录和注册功能，与用户信息相关的操作。
2. 题目列表模块：负责展示所有可用的编程题目列表，并支持用户输入搜索条件来筛选和排序题目。
3. 题目详情模块：负责展示题目详情页，包括题目描述、输入输出样例、评测结果和提交历史等信息。
4. 代码编辑模块：负责提供代码编辑器，用户可以在其中编辑和提交自己的代码。编辑器支持多种编程语言，并提供自动缩进、语法高亮和自动补全等功能。
5. 代码提交模块：负责将用户在代码编辑模块中编辑好的代码提交到后端进行评测。
6. 评测结果页面模块：负责展示评测结果页面，用户可以查看自己提交的代码的评测结果，包括编译错误、运行错误、通过测试点的数量等信息。
7. 排行榜模块：负责展示用户的排名和积分等信息，实现排行榜功能。
8. 后台模块：负责与后端进行数据交互，进行题目修改，增加，比赛的管理。

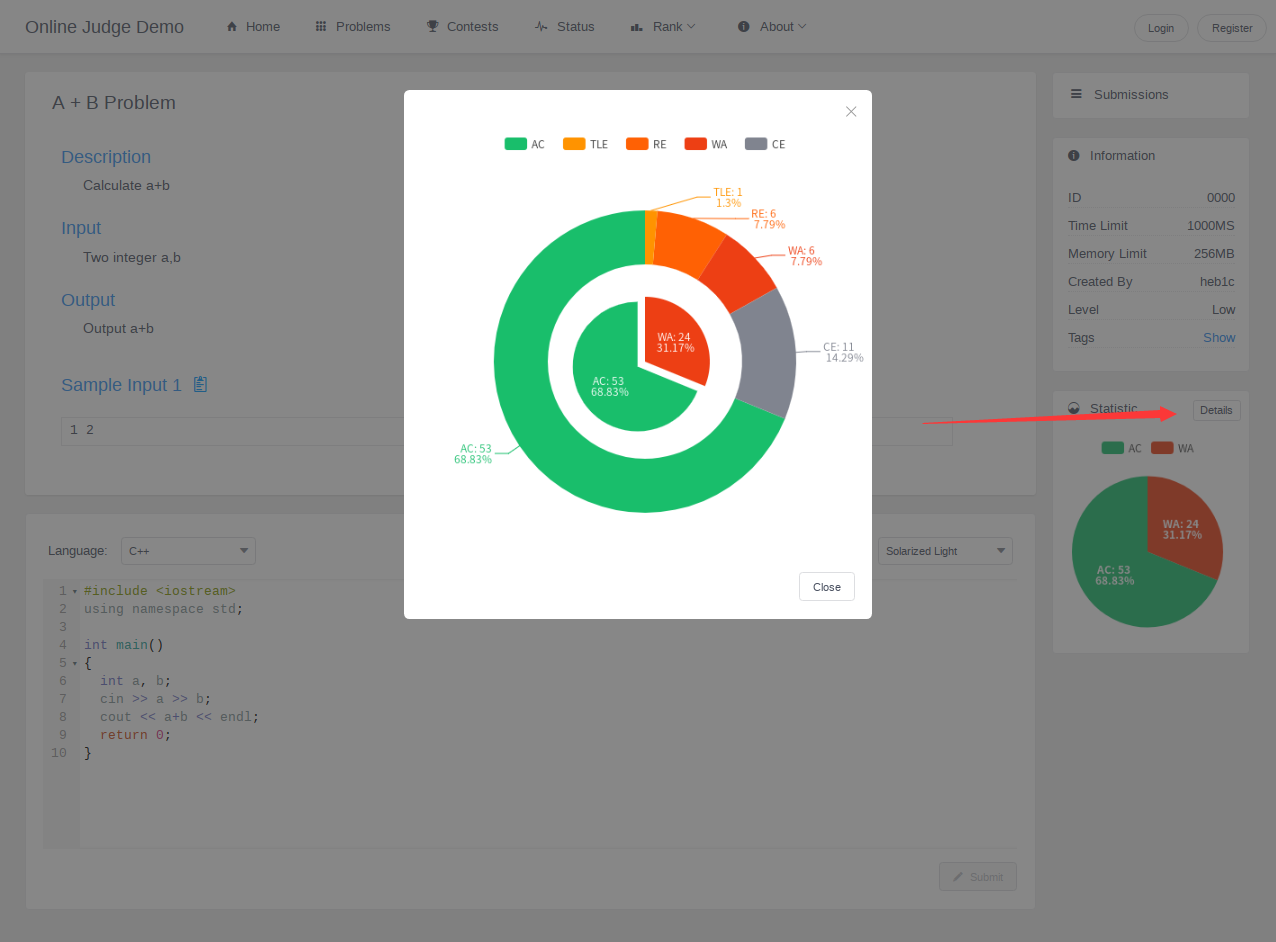
## 3.2前端接口设计

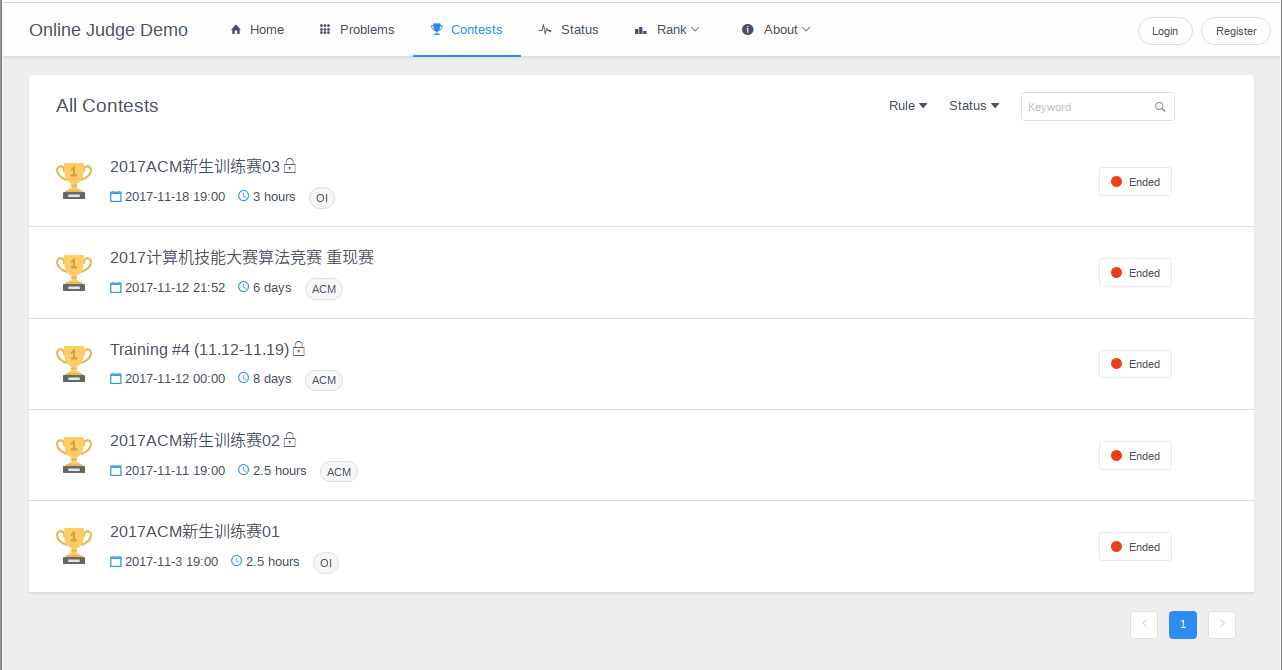
### 3.2.1用户接口

用户接口即为图形化界面，我们从各方面展示我们的原型系统如下：

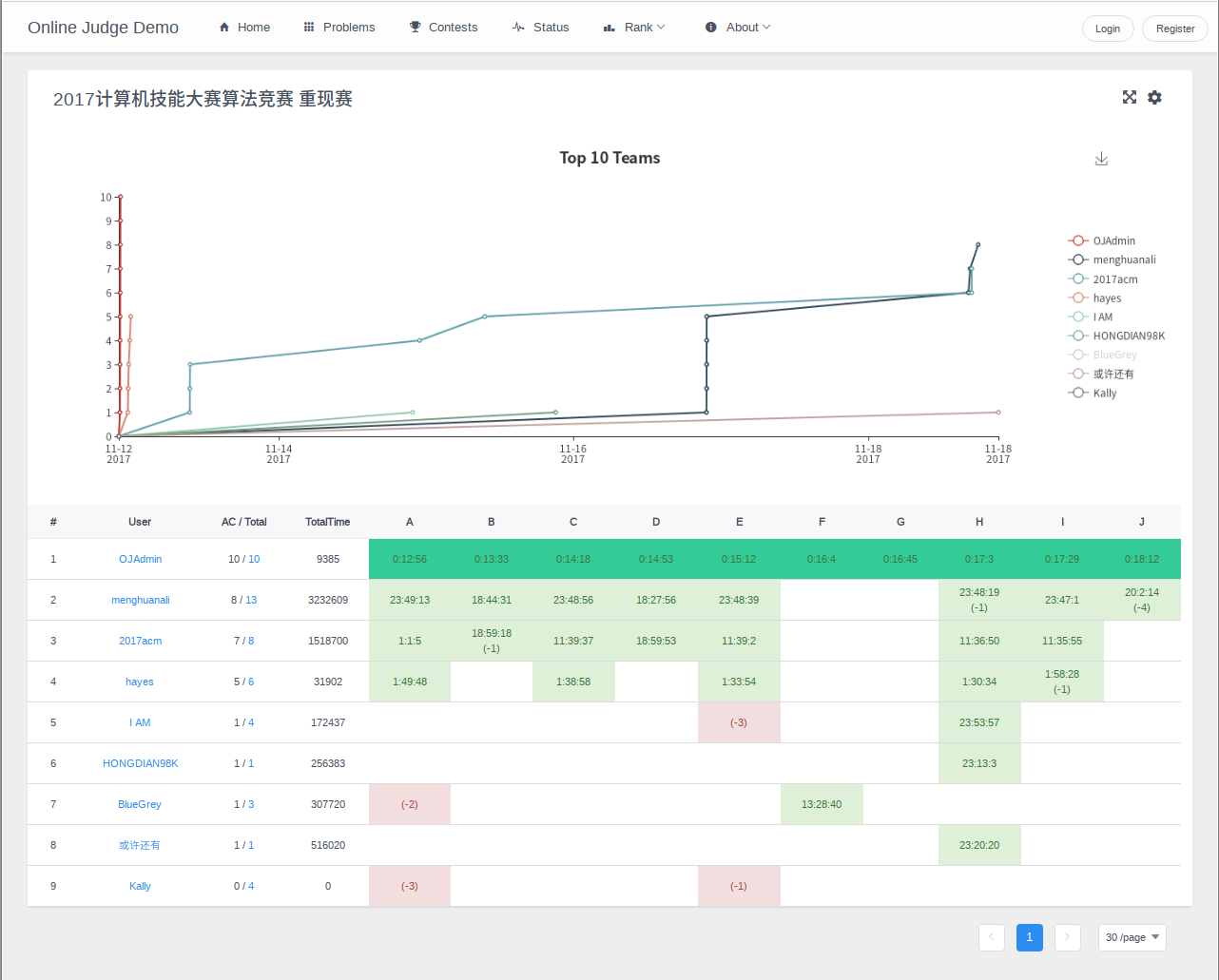
[](https://user-images.githubusercontent.com/20637881/33372506-402022e4-d539-11e7-8e64-6656f8ceb75a.png)

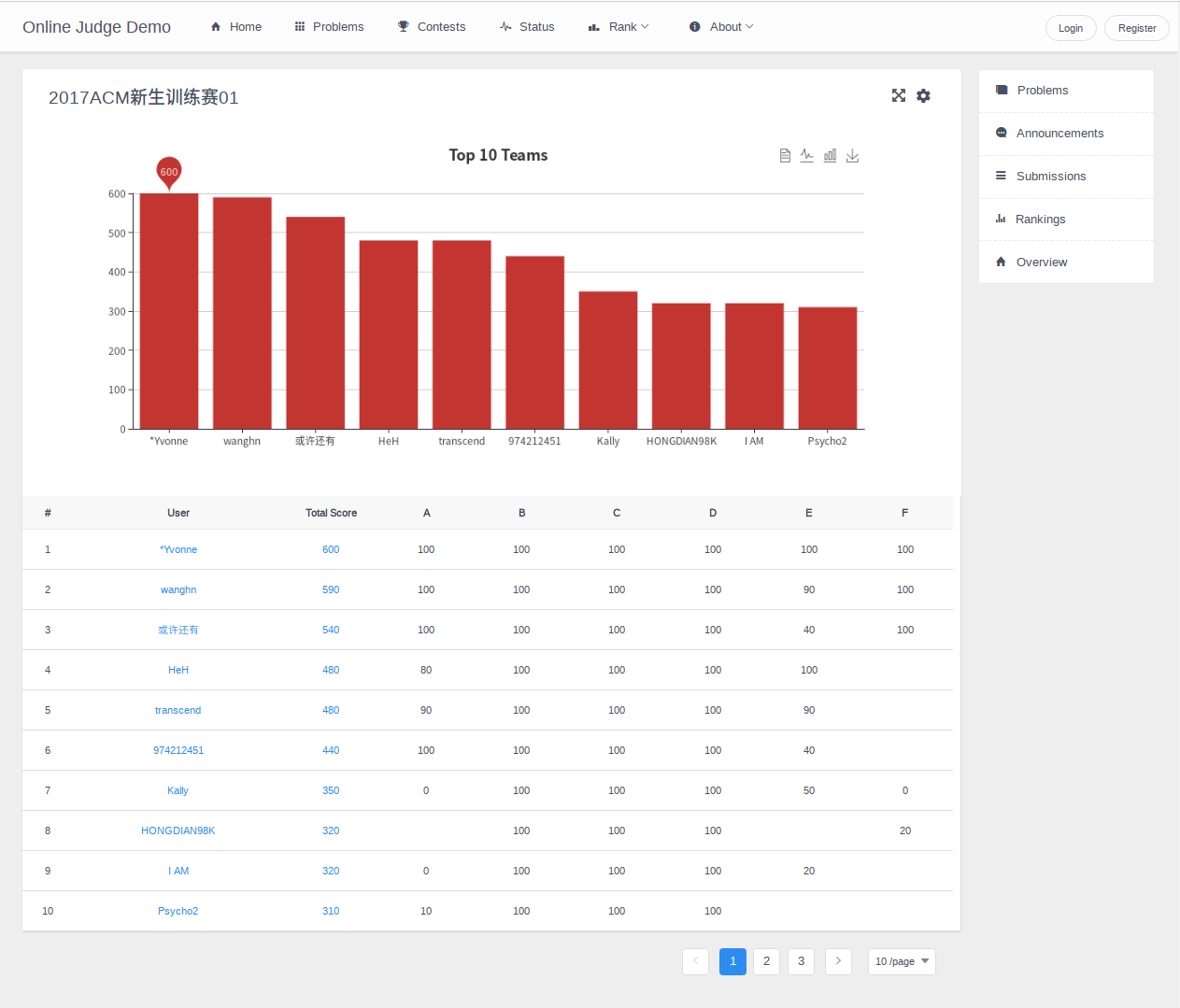
[](https://user-images.githubusercontent.com/20637881/33372507-4061a782-d539-11e7-8835-076ddae6b529.png)

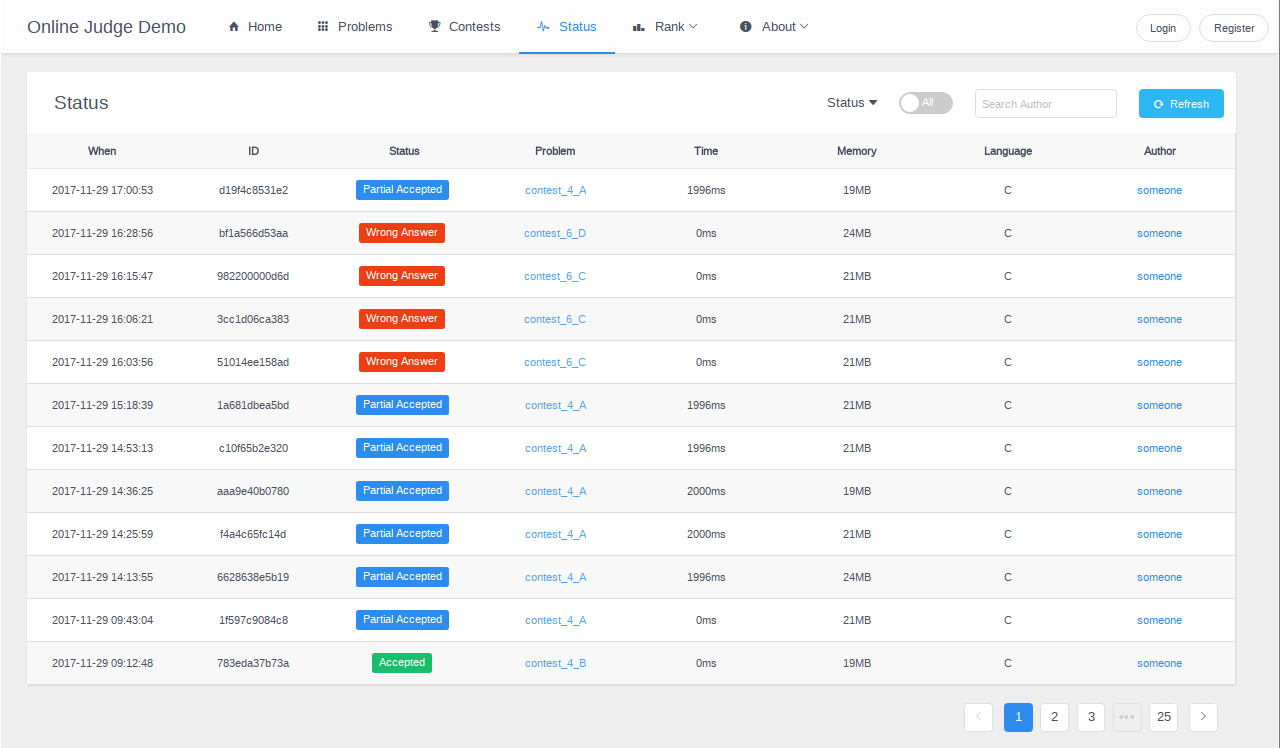
[](https://user-images.githubusercontent.com/20637881/33372508-40a0c6ce-d539-11e7-8d5e-024541b76750.png)

[](https://user-images.githubusercontent.com/20637881/33372509-40d880dc-d539-11e7-9eba-1f08dcb6b9a0.png)

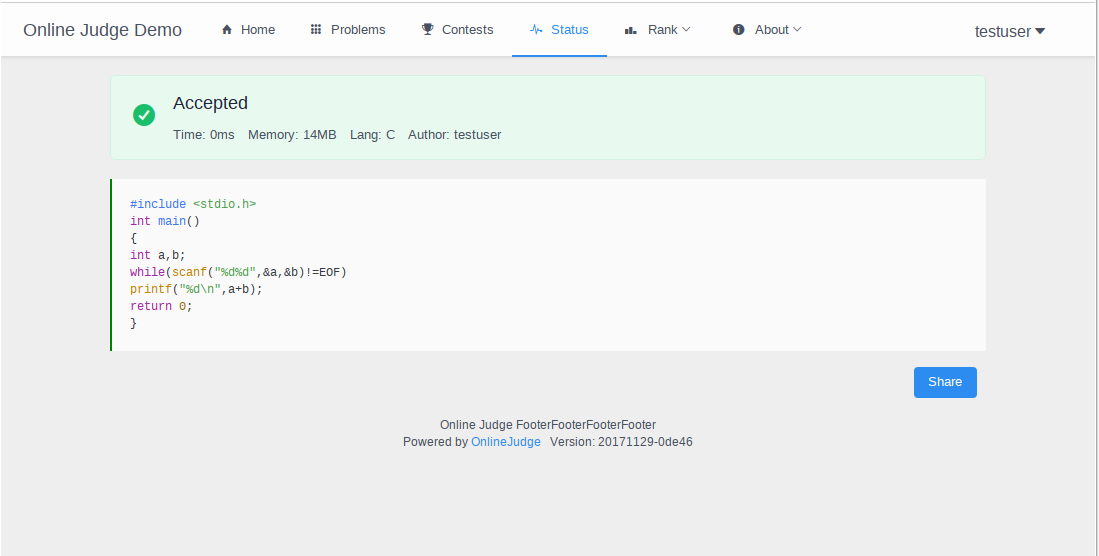
排行榜模块：

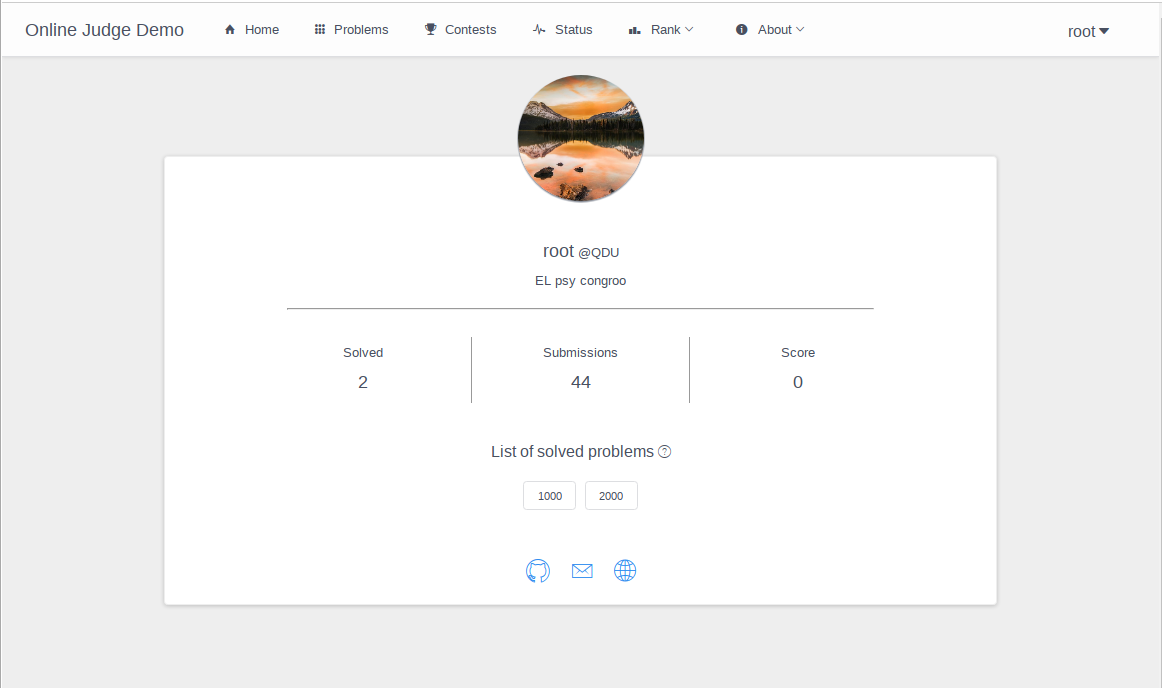
[](https://user-images.githubusercontent.com/20637881/33372510-41117f68-d539-11e7-9947-70e60bad3cf2.png)

[](https://user-images.githubusercontent.com/20637881/33372511-41d406fa-d539-11e7-9947-7a2a088785b0.png)

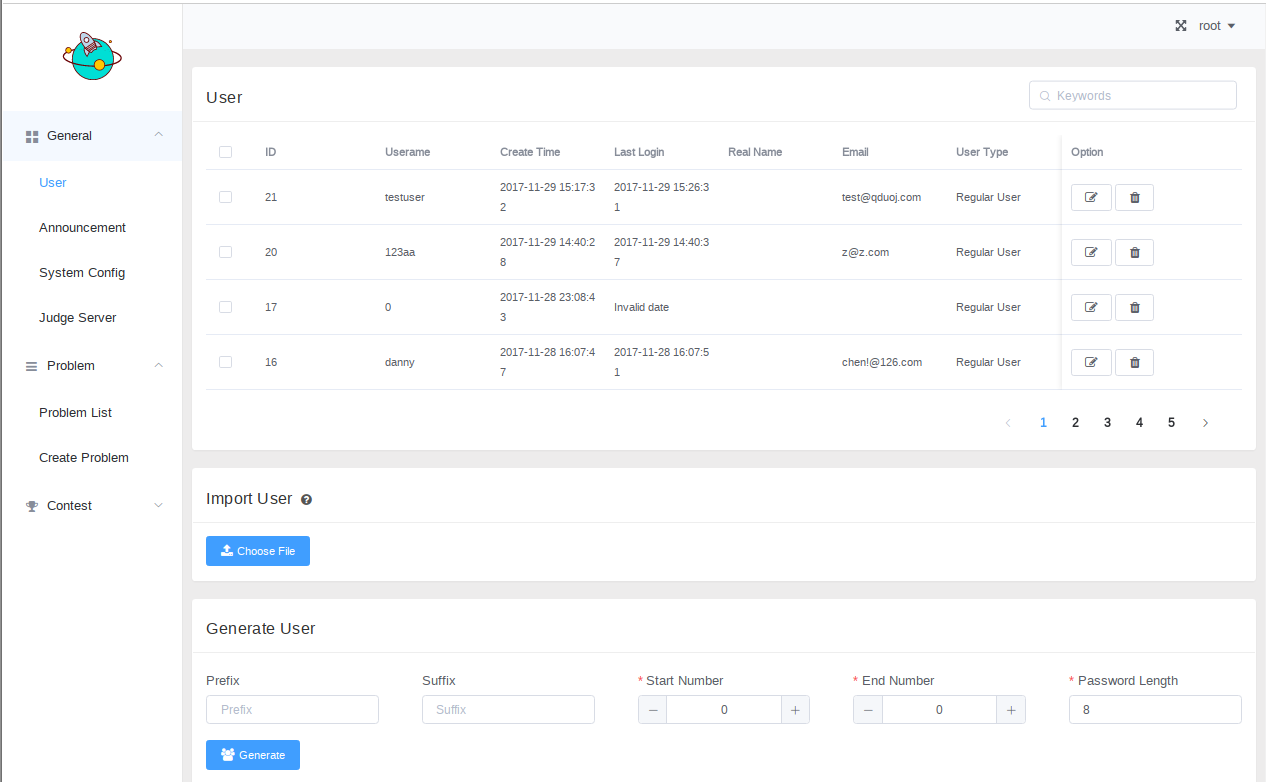
[](https://user-images.githubusercontent.com/20637881/33372512-420ba240-d539-11e7-8645-594cac4a0b78.png)

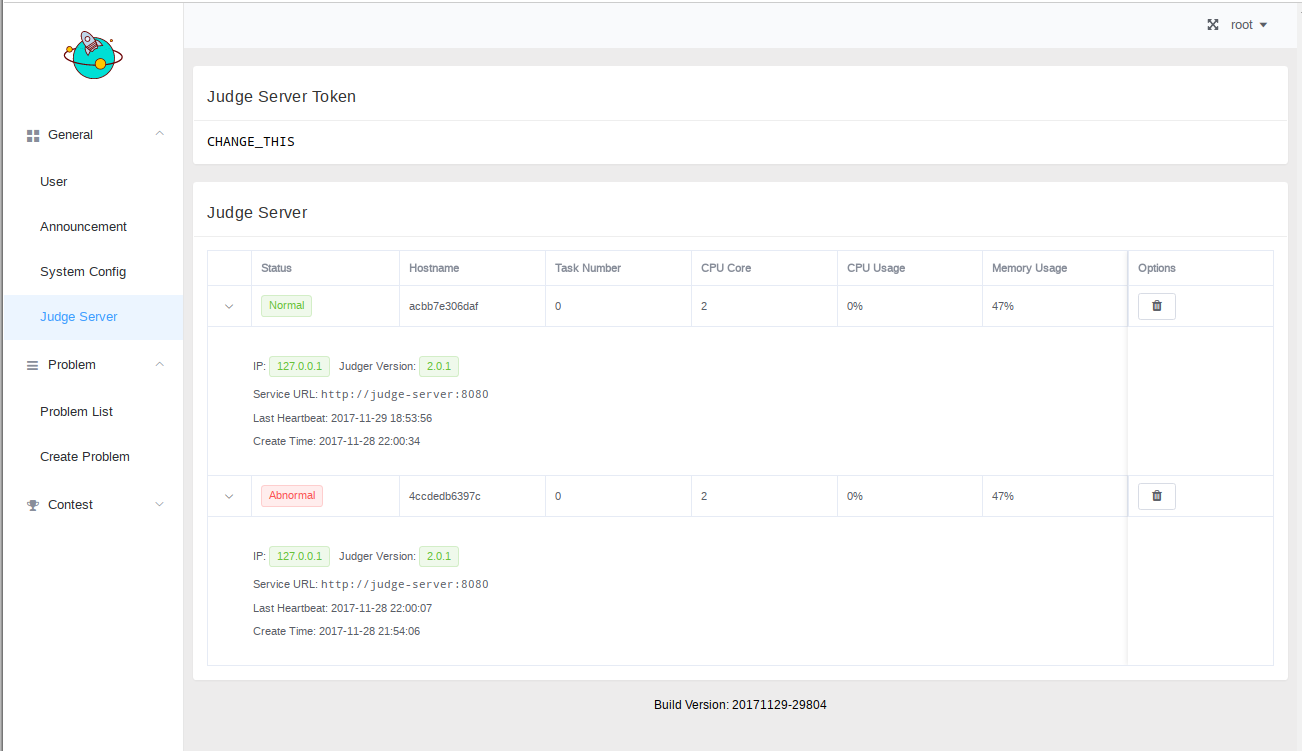
测评模块：

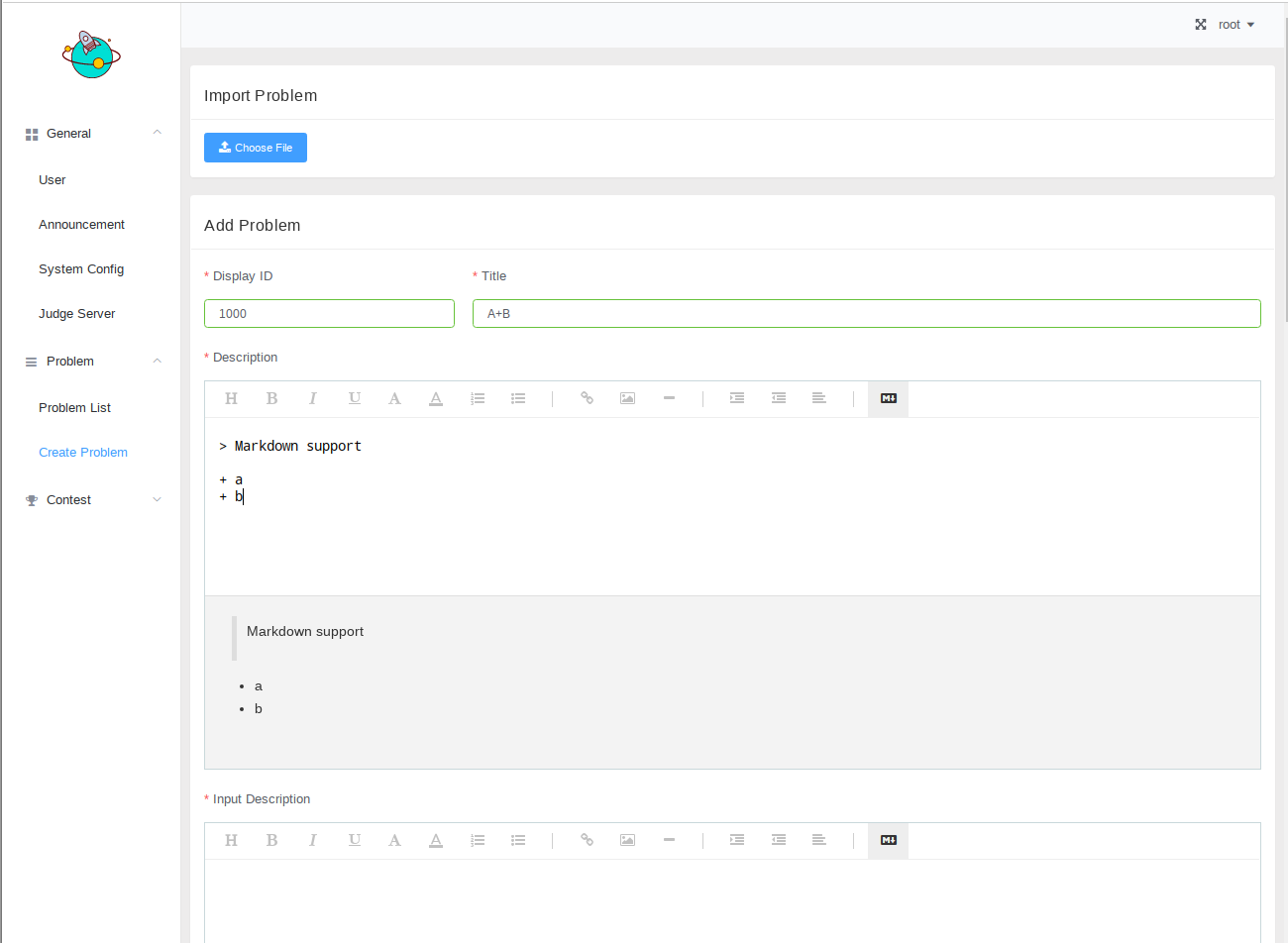
[](https://user-images.githubusercontent.com/20637881/33365523-787bd0ea-d523-11e7-953f-dacbf7a506df.png)

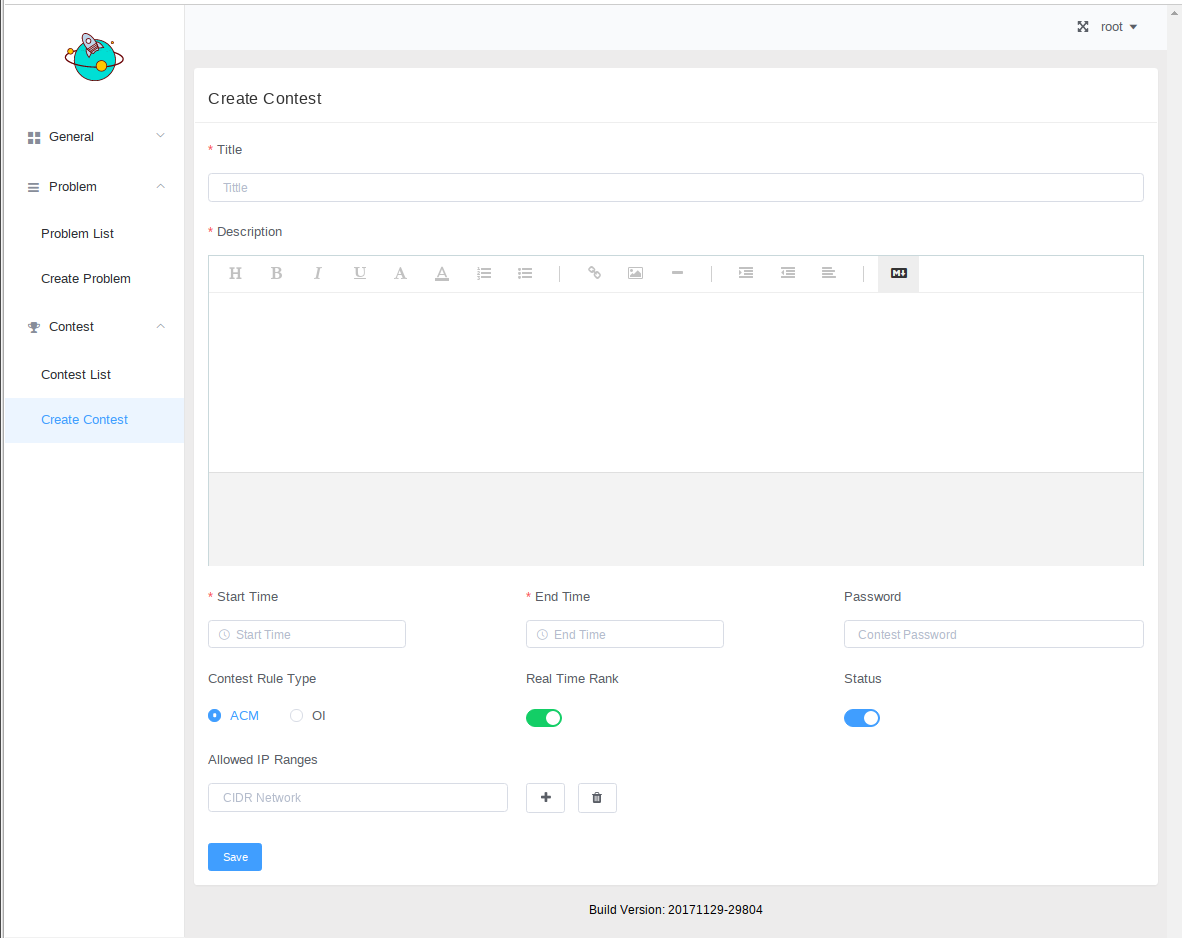
[](https://user-images.githubusercontent.com/20637881/33365521-7842d808-d523-11e7-84c1-2e2aa0079f32.png)

后台管理模块：

[](https://user-images.githubusercontent.com/20637881/33372516-42c34fda-d539-11e7-9f4e-5109477f83be.png)

[](https://user-images.githubusercontent.com/20637881/33372517-42faef9e-d539-11e7-9f17-df9be3583900.png)

[](https://user-images.githubusercontent.com/20637881/33372513-42472162-d539-11e7-8659-5497bf52dbea.png)

[](https://user-images.githubusercontent.com/20637881/33372514-428ab922-d539-11e7-8f68-da55dedf3ad3.png)

### 3.2.2外部接口

主要是前端与后端交互的接口，交互过程使用HTTPS网络传输。OnlineJudge系统前端与后端交互调用的主要url的接口如下，具体的接口规范仍需要精确设计。

1. 用户登录和注册接口
   1. 登录接口：/api/user/login
   2. 注册接口：/api/user/register
2. 题目相关接口
   1. 获取题目列表接口：/api/problem/list
   2. 获取题目详情接口：/api/problem/detail
   3. 获取题目提交历史接口：/api/submission/list
3. 代码提交和评测相关接口
   1. 代码提交接口：/api/submission/submit
   2. 获取评测结果接口：/api/submission/status
4. 排行榜和比赛相关接口
   1. 获取排行榜接口：/api/user/rank
   2. 获取比赛列表接口：/api/contest/list
   3. 获取比赛详情接口：/api/contest/detail
5. 管理员管理题目相关接口
   1. 添加题目接口：/api/admin/problem/add
   2. 编辑题目接口：/api/admin/problem/edit
   3. 删除题目接口：/api/admin/problem/delete
6. 管理员管理比赛相关接口
   1. 添加比赛接口：/api/admin/contest/add
   2. 编辑比赛接口：/api/admin/contest/edit
   3. 删除比赛接口：/api/admin/contest/delete

## 3.3前端数据结构设计

### 3.3.1逻辑结构设计要点

前端主要采用调用后端的方法获取数据，而本身需要存储的数据十分少，所以应该对不同的页面进行不同的设计，主要存储TOKEN和当前状态下提交的代码（以字符串形式）就能满足所有的数据存储需要，所以我们不再仔细设计逻辑上的数据结构。

### 3.3.2物理结构设计要点

存储到浏览器存储池即可。

### 3.3.3数据结构与程序的关系

通常情况下，前端数据结构与后端程序实现之间的数据交互采用统一的接口格式，比如JSON格式。前端通过调用后端的API接口，将前端页面中的数据提交给后端进行处理，并将后端处理结果返回到前端进行展示。因此，前端数据结构与程序实现之间的关系主要体现在数据交互的接口规范上，而不是具体的数据结构实现。

## 3.4质量与模块关系

对于一个OnlineJudge系统最重要的质量属性是性能和安全。以下是前端质量设计的具体描述：

1. 性能：性能是前端设计的一个重要方面，对于OnlineJudge系统而言，性能表现主要体现在响应时间和吞吐量方面。系统需要提供快速的响应时间，使用户可以快速地提交代码并获得评测结果。而这个过程的瓶颈不会出现在前端部分，前端即使使用Vue这种速度比较慢的框架，也足够满足多人同时并发的场景，而性能也足够满足。
2. 安全：安全是前端设计的另一个重要方面，对于OnlineJudge系统而言。
3. 用户登录模块：安全性要求较高，需要保证用户身份的合法性和隐私数据的安全性。应该采用加密协议，比如HTTPS协议，以保证登录信息的安全传输。同时，还需要进行身份认证和授权控制，比如密码强度检测、验证码校验等措施。
4. 代码编辑模块和代码提交模块：安全性要求较高，需要防止用户提交恶意代码或攻击性代码，以保护系统的稳定性和安全性。应该采用代码审查和代码过滤等措施，防止恶意代码的提交和执行。
5. 后台模块：安全性要求较高，需要保证后端系统的稳定性和安全性。应该采用身份认证和授权控制等措施，防止未授权访问和恶意攻击。同时，还需要进行数据加密和数据校验等措施，保护后端数据的安全性和完整性。

而对于题目列表模块和题目详情模块，评测结果页面模块和排行榜模块,安全性要求相对较低。

# 4.后端设计

## 4.1后端结构设计

### 4.1.1运行环境

Ubuntu20.04， Docker， Django

HP ProLiant DL360 Gen10服务器，配置为：

CPU：2个Intel Xeon Gold 6240处理器，共40核心

内存：512GB DDR4 ECC内存

存储：8 x 1.92TB SAS SSD（RAID 10）

网络：4 x 10GbE网络接口OnlineJudge

### 4.1.2基本设计思想和处理流程

OnlineJudge后端的基本设计思想是实现一个高效稳定、安全可靠的在线评测系统，主要包括三个方面的设计思想：

1. 模块化设计：将整个后端系统分解为多个独立的子系统，每个子系统负责不同的功能和业务逻辑，以实现系统的高内聚低耦合。
2. 面向对象设计：采用面向对象的设计方法，将系统中的各个对象进行抽象和封装，以实现系统的可扩展性和维护性。
3. 高并发处理：针对在线评测系统高并发的特点，采用分布式架构和异步处理等技术手段，以实现系统的高并发处理和高性能。

后端处理模块大致如下：

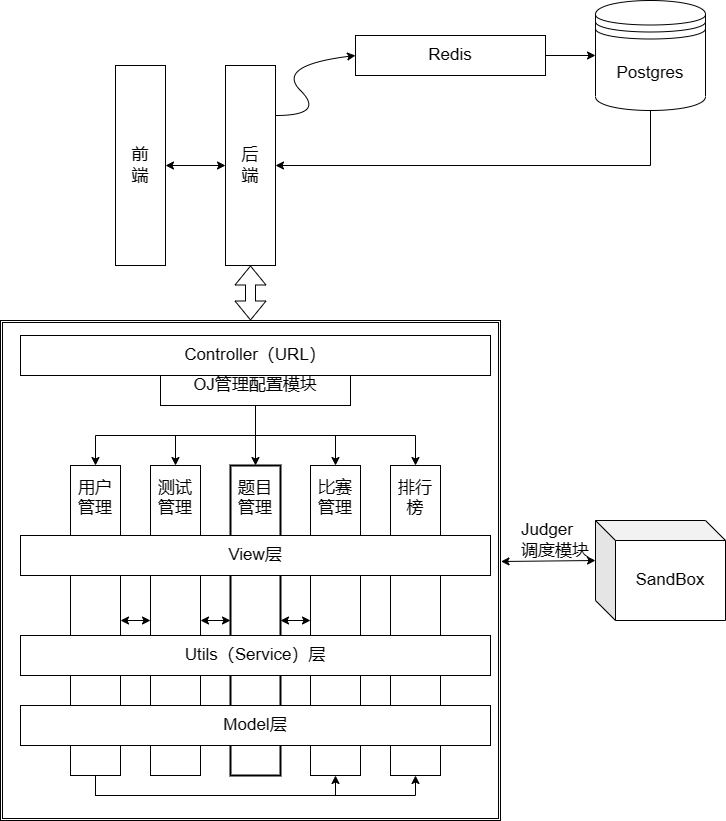
1. 用户管理模块：负责用户身份认证和授权管理，包括用户登录、注册、身份认证、权限管理等功能。
2. 题目管理模块：负责管理系统中的所有编程题目，包括题目添加、编辑、删除、搜索、排序等功能。
3. 提交管理模块：负责管理用户提交的编程代码，包括代码编译、运行、测试、评测等功能。
4. 排名管理模块：负责管理用户的排名和积分等信息，以实现排行榜功能。
5. 比赛管理模块：负责管理系统中的比赛，包括比赛添加、编辑、删除、管理等功能。
6. 数据库管理模块：负责管理系统中所有数据的存储和访问，包括用户数据、题目数据、提交数据、答案数据等。
7. 评测机管理模块：负责管理系统中的评测机，包括评测机的添加、编辑、删除、维护等功能。

以上各个模块之间相互关联，通过数据交换和接口调用等方式进行协作和协同工作，最终实现整个OnlineJudge系统的功能和业务逻辑。

### 4.1.3质量设计

1. 性能：
   1. 吞吐量：OnlineJudge系统的吞吐量要求高，需要能够支持同时多个用户的访问和请求。为此，系统采用分布式架构和负载均衡等技术手段，以实现系统的高并发处理和高吞吐量。
   2. 并发速度：OnlineJudge系统需要能够快速响应用户的请求，包括题目查询、提交、评测等操作。为此，系统采用异步处理、缓存优化、数据库索引等技术手段，以实现系统的高并发处理和高性能。
2. 安全：
   1. 安全控制和物理保护措施：OnlineJudge系统采用多层安全措施，包括网络安全、主机安全、应用程序安全等方面的措施，以保障系统的安全性。同时，系统还采用物理保护措施，包括数据备份、灾备恢复、设备监控等方面的措施，以防止硬件故障和数据丢失等问题。
   2. 用户身份鉴别机制：OnlineJudge系统采用严格的用户身份鉴别机制，包括密码强度检测、双因素身份认证、访问控制等措施，以确保用户身份的合法性和数据的安全性。
   3. 用户对系统的访问权限和范围：OnlineJudge系统采用细粒度的访问控制策略，以确保用户只能访问其有权限访问的数据和功能。同时，系统还采用审计和日志记录等措施，以追踪和监控用户的操作行为。
   4. 病毒的防治措施：OnlineJudge系统采用病毒扫描、漏洞扫描等安全措施，以防止系统受到病毒、木马等恶意软件的攻击和侵害。
3. 可靠性：OnlineJudge系统需要保证高可靠性，即在面对各种异常情况下，系统仍能够保持正常的运行和服务。为此，系统采用备份和容灾等措施，以确保系统的可用性和可靠性。
4. 可扩展性：OnlineJudge系统需要具备良好的可扩展性，即在需要增加或减少系统的规模时，能够快速、便捷地进行系统扩展或缩减。为此，系统采用分布式架构、容器化技术、自动化部署等措施，以提高系统的可扩展性和灵活性。
5. 可维护性：OnlineJudge系统需要具备良好的可维护性，即在系统出现故障或需要进行更新和维护时，能够快速、便捷地进行修复和升级。为此，系统采用模块化设计等措施，以提高系统的可维护性和可扩展性。

### 4.1.4结构设计



Django后端采用了MVC（Model-View-Controller）架构模式，将系统的各个功能分成了不同的层次和模块。下面是Django后端的结构分析：

1. 模型层（Model）：该层负责处理系统的数据模型和数据库访问操作。系统中的每一个数据实体都对应着一个模型类，该类负责定义实体的属性和操作方法。在Django中，该层对应着models模块。
2. 视图层（View）：该层负责处理系统的业务逻辑和数据处理。系统中的每一个业务实体都对应着一个视图函数，该函数负责处理实体的相关逻辑和操作。在Django中，该层对应着views模块。
3. 控制器层（Controller）：该层负责处理系统的请求和响应，以及控制系统的流程和状态。系统中的每一个请求都对应着一个控制器类或函数，该类或函数负责处理请求并生成相应的响应。在Django中，该层对应着urls模块。

同时为了应对各个模块对应的请求，我们同时采用了模块化的架构，通过新加入的Service层减少混淆，使得代码可维护性和可扩展性大大提升。

而且我们为了提高测评效率，应该加入评测机管理模块，该模块主要负责管理系统中的评测机，包括评测机的添加、编辑、删除、维护等功能。我们采用心跳机制，测评机定期向后端服务器指定端口发送心跳信号，后端回应。第一次收到信号则将测评机加入列表，若3分钟后没有成功接收到心跳信号，则将其移除列表，采用合适的调度算法，将测评分布到合适的测评机上。

### 4.1.5功能与程序结构的关系

1. 用户管理模块：该模块主要负责用户的身份认证和授权管理。其主要功能包括用户登录、注册、身份认证、权限管理等。在程序结构上，该模块与数据库模块密切相关，通过ORM框架和数据库适配器实现对用户数据的增删改查等操作。同时，该模块还与视图层和控制器层有一定的交互，以实现用户身份验证和授权等操作。
2. 题目管理模块：该模块主要负责管理系统中的所有编程题目，包括题目添加、编辑、删除、搜索、排序等功能。在程序结构上，该模块与数据库模块密切相关，通过ORM框架和数据库适配器实现对题目数据的增删改查等操作。同时，该模块还与视图层和控制器层有一定的交互，以实现题目的展示和管理等操作。
3. 提交管理模块：该模块主要负责管理用户提交的编程代码，包括代码编译、运行、测试、评测等功能。在程序结构上，该模块与评测机管理模块密切相关，通过评测机管理模块实现对用户提交代码的编译、运行、测试、评测等操作。同时，该模块还与数据库模块和视图层有一定的交互，以实现用户提交记录的存储和展示等操作。
4. 答案管理模块：该模块主要负责管理每个题目的标准答案和测试数据，以供评测使用。在程序结构上，该模块与数据库模块密切相关，通过ORM框架和数据库适配器实现对答案和测试数据的增删改查等操作。同时，该模块还与评测机管理模块有一定的交互，以实现测试数据的调用和使用等操作。
5. 排名管理模块：该模块主要负责管理用户的排名和积分等信息，以实现排行榜功能。在程序结构上，该模块与数据库模块密切相关，通过ORM框架和数据库适配器实现对用户积分和排名等信息的增删改查等操作。同时，该模块还与视图层和控制器层有一定的交互，以实现排行榜的展示和更新等操作。
6. 比赛管理模块：该模块主要负责管理系统中的比赛，包括比赛添加、编辑、删除、管理等功能。在程序结构上，该模块与数据库模块密切相关，通过ORM框架和数据库适配器实现对比赛数据的增删改查等操作。同时，该模块还与视图层和控制器层有一定的交互，以实现比赛信息的展示和管理等操作。
7. 评测机管理模块：该模块主要负责管理系统中的评测机，包括评测机的添加、编辑、删除、维护等功能。然后根据算法和实时情况进行调度，选择合适的评测机进行评测。对于测评功能具有重要作用。

## 4.2后端接口设计

### 4.2.1用户接口

主要的用户接口即为和前端交互的接口（URL形式）可见前文3.2.2节。

### 4.2.2外部接口

而查询Postgres的数据库已经被Django框架封装好，我们无需使用SQL等额外的形式进行数据查询。所以我们主要的外部接口为以网络形式调用测评机API。可见<https://opensource.qduoj.com/#/judgeserver/api>以查看详细API使用情况，而每一个API都将被使用到（因为都十分重要）。

## 4.3后端数据结构设计

### 4.3.1逻辑结构设计要点

OnlineJudge后端数据结构主要包括用户数据、题目数据、提交数据、答案数据和比赛数据等。

1. 用户数据结构：用户数据结构包括用户ID、用户名、密码、邮箱、角色等信息。其中，用户ID是唯一的标识符，用户名和密码用于身份认证和授权，邮箱用于找回密码和接收通知等，角色用于区分不同用户的权限和功能等。
2. 题目数据结构：题目数据结构包括题目ID、题目标题、题目描述、输入格式、输出格式、样例输入、样例输出、时间限制、空间限制、难度等信息。其中，题目ID是唯一的标识符，题目标题和描述用于展示题目的基本信息和要求，输入格式和输出格式用于指导用户编写代码和进行测试，样例输入和样例输出用于验证用户代码的正确性，时间限制和空间限制用于评测用户代码的性能，难度用于指导用户选择适当的题目进行练习和挑战。
3. 提交数据结构：提交数据结构包括提交ID、用户ID、题目ID、提交时间、提交代码、提交状态等信息。其中，提交ID是唯一的标识符，用户ID和题目ID用于标识用户和题目，提交时间用于记录用户提交代码的时间，提交代码用于保存用户提交的代码内容，提交状态用于记录代码的评测状态和结果等。
4. 答案数据结构：答案数据结构包括题目ID、测试数据、标准输出等信息。其中，题目ID用于标识题目，测试数据用于进行代码测试和评测，标准输出用于验证用户代码的正确性。
5. 比赛数据结构：比赛数据结构包括比赛ID、比赛标题、比赛描述、开始时间、结束时间、题目列表等信息。其中，比赛ID是唯一的标识符，比赛标题和描述用于展示比赛的基本信息和要求，开始时间和结束时间用于控制比赛的开始和结束时间，题目列表用于指导用户选择比赛题目和进行比赛。

以上数据结构在OnlineJudge后端中起到重要的作用，通过各种机制和接口实现了系统的各种功能和特性，为用户提供了高效、可靠、可扩展和易维护的服务和支持。

### 4.3.2物理结构设计要点

以上数据结构在OnlineJudge后端中又Django Model层自动实现物理结构的分配，简化了我们的设计。

## 4.4质量与模块关系

对于可靠性，可维护性等质量属性，其更依赖于开发者的自觉和系统冗余的整体设计。

对于OnlineJudge系统极为关键的性能质量要素来说。主要就是以下两个模块成为整个系统的瓶颈：

提交管理模块：该模块负责管理用户提交的编程代码，包括代码编译、运行、测试、评测等功能，对系统性能的影响比较大，需要保证代码的编译、运行、测试、评测等操作的速度和效率。

评测机管理模块：该模块负责管理系统中的评测机，对系统性能的影响主要取决于评测机的数量和规模，管理操作的复杂度也会影响系统性能。

我们为了提高测评效率，应该加入评测机管理模块，该模块主要负责管理系统中的评测机，包括评测机的添加、编辑、删除、维护等功能。我们采用心跳机制，测评机定期向后端服务器指定端口发送心跳信号，后端回应。第一次收到信号则将测评机加入列表，若3分钟后没有成功接收到心跳信号，则将其移除列表，采用合适的调度算法，将测评分布到合适的测评机上。

OnlineJudge后端的各模块都涉及到不同的安全问题，需要采取一系列安全措施来保障系统的安全性。

1. 用户管理模块：用户管理模块需要进行用户身份认证和授权管理，因此需要采用安全的身份认证机制，如使用密码哈希等方式存储用户密码，防止用户密码泄露。同时，需要限制用户访问权限和范围，只有经过授权的用户才能进行相应操作，如管理员才能进行用户权限的管理等。
2. 题目管理模块：题目管理模块需要进行题目的添加、编辑、删除、搜索、排序等操作，因此需要对题目的数据进行安全保护，如对题目的敏感信息进行加密和权限控制等。同时，需要限制题目的访问权限和范围，只有经过授权的用户才能访问相关的题目信息，防止敏感信息泄露。
3. 提交管理模块：提交管理模块涉及到用户提交的编程代码，需要采用安全的代码运行和评测机制，如使用沙箱等技术来隔离用户代码的运行环境，防止恶意代码的执行。同时，需要对用户提交的代码进行检查和审核，防止代码中包含安全漏洞等问题。
4. 答案管理模块：答案管理模块涉及到题目的标准答案和测试数据，需要进行保护和限制访问权限，只有经过授权的用户才能访问和修改相关信息。
5. 比赛管理模块：比赛管理模块涉及到比赛的添加、编辑、删除、管理等操作，需要对比赛的数据进行安全保护和限制访问权限，只有经过授权的用户才能进行相关操作。
6. 评测机管理模块：评测机管理模块涉及到评测机的添加、编辑、删除、维护等操作，需要采用安全的身份认证和访问控制机制，防止恶意攻击和非法操作。同时，需要对评测机的运行环境进行安全保护，如采用沙箱技术隔离评测机的运行环境，防止评测机被利用进行攻击等行为。