功能需求

系统服务端

（1） 账号管理功能

系统的使用者能够在系统注册并通过邮箱验证后登陆账号使用该系统。

（2） 比赛管理功能

A. 通过上传参赛人员csv表和提交结果标准格式包等竞赛信息，创建起一场比赛；

B. 历史比赛信息的修改维护和导出；

C. 考场布局信息维护和管理。

（3） 提交结果管理功能

A. 向各个选手客户端发送收集请求；

B. 接收各个参赛选手客户端提交过来的结果，并根据创建比赛时的标准提交格式包进行过滤其中的无关文件；并且提供查看详情和下载的功能；

C. 能够打包下载整理好的该场比赛所有选手提交。

（4） 提交结果统计与分析功能

A. 总览图：能够显示出考场布局图并在上面标识各个选手的状况；

B. 出席完整性统计报表：统计哪些同学没来参加比赛，并提供csv导出功能；

C. 竞赛完成度统计报表：统计参加同学是否有哪些题目是空的没有完成，并提供csv导出功能。

（5） 代码查重功能

系统能够对所收集到的代码进行选手间的代码查重比对。并可以在重复率达到一定阈值后的予以警告显示。

系统的客户端

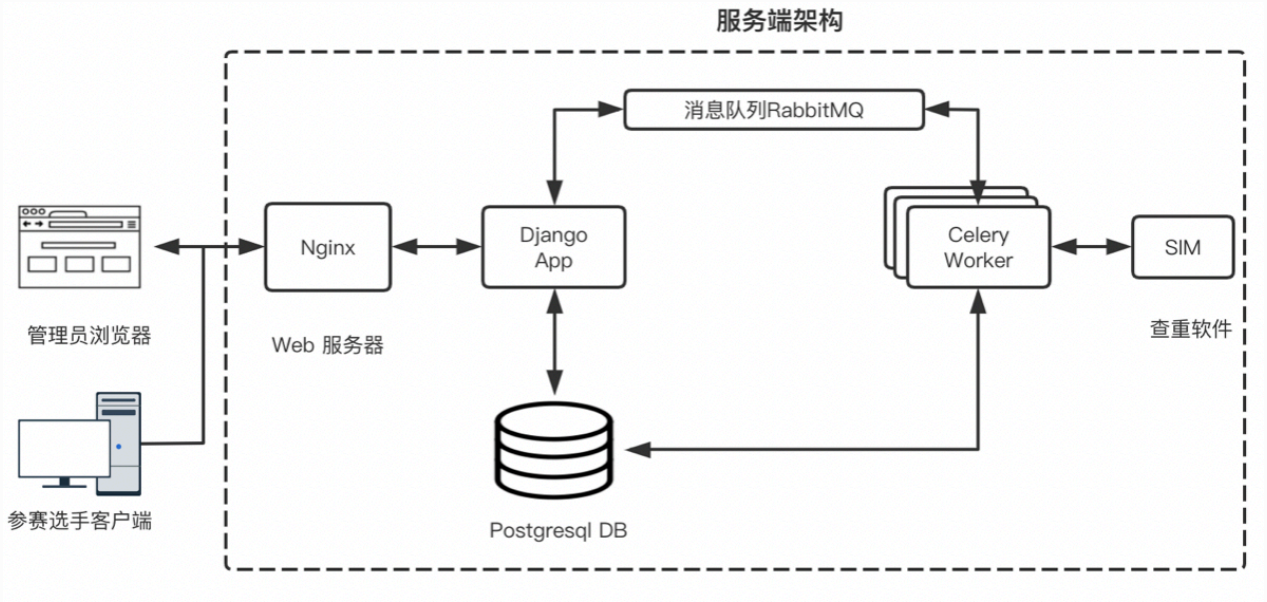
（1） 能够开机自动启动并后台运行，监听服务器发来的收集请求；

（2） 在收到服务端的请求后，打包代码文件所在文件夹成压缩包后，发送到服务端。

系统架构

总体架构：基于B/S与C/S相结合的系统总体架构为基础，设计并实现了切合编程竞赛需求的代码收集与检测系统。选手的代码赛后将通过C/S模式自动化地通过参赛选手机器上的客户端提交到服务端，避免选手在提交过程中的参与，从而杜绝作弊行为的风险。随后系统会进行选手代码的格式校验、无关文件过滤和统计分析等工作，管理员可以通过B/S模式使用浏览器对赛事的各方面情况进行把控管理。

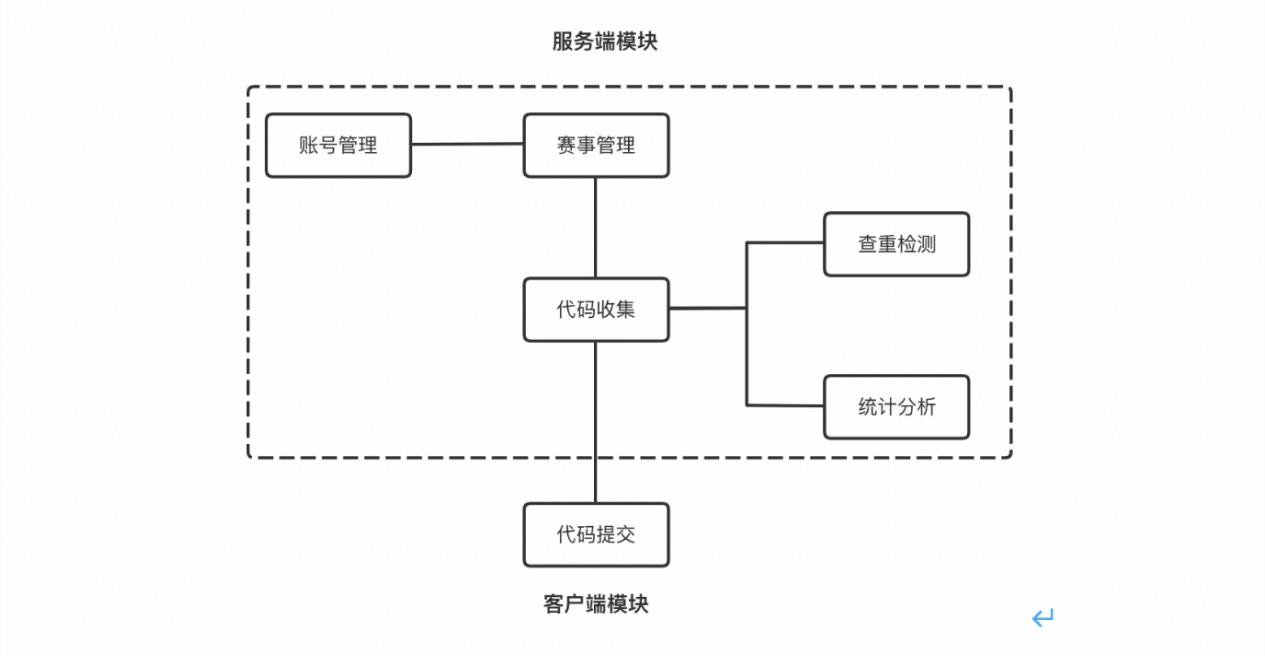
服务端架构：



其中 Nginx是异步的Web服务器，在系统中主要的作用就是当用户需要请求系统各个功能的时候，将请求反向代理到Django App去；Django App是服务端的核心部分，负责本系统各个功能需求；PostgreSQL是本系统使用的数据库，负责包括账号信息、比赛信息、选手提交等众多信息数据的存取；RabbitMQ作为消息队列，在查重时候将查重任务封装成task并将其送往消息队列RabbitMQ中，而消费者再后续从消息队列中去消费处理各个task；CeleryWorker是采用django-celery库实现的worker容器，作为消息队列的消费者，不断监听RabbitMQ，一旦有task便从队列上取下来处理，并可以将处理完的task结果存入DB中持久化；Sim是系统所集成的本地执行的查重软件。CeleryWorker在获取了查重task后，便会调用Sim进行必要的代码文件的相似度检测。最后将服务端上述组件都是用Docker来容器化，采用docker-compose进行容器编排。

客户端设计：客户端一直监听本机与服务端约定好端口，当收到服务端的特定收集请求时，从请求中解析出请求的代码文件夹路径、服务端的回调接口、该机器的选手信息等，将所请求的代码文件夹打包，并附上计算MD5校验值后和该机器的选手信息，传输到服务端的回调接口。由于为了使得客户端能够满足选手Linux和Windows操作系统的需求，客户端统一采用Python脚本实现后用pyinstaller库将脚本分别包装成Linux下的二进制可执行程序和Windows下的.exe可执行程序。

模块设计



根据功能需求分析，本系统的服务端主要分为五个功能模块：账号管理、赛事管理、代码收集、查重检测、统计分析。而客户端只有代码提交一个模块作为主模块。

其中账号管理模块负责系统账号的管理：例如用户的注册、验证、登录、注销等；赛事管理模块则负责赛事的各种信息的创建和维护，以及其比赛成员管理；代码收集模块负责收集选手从客户端提交的代码，收集完后对所收集的代码进行文件夹结构的校验，以及对进行对其中无关文件进行过滤，并管理员提供查看和下载功能；查重检测模块负责对一场竞赛选手的提交进行查重比对检测；统计分析模块负责进行比赛的选手代码情况进行统计分析。

各个模块的主要逻辑关系如下：管理员老师通过账号管理模块进行注册、登陆从而访问系统的Web管理端。进入系统后，管理员可以通过赛事管理模块进行赛事的创建和配置和历史赛事的查看与维护。完成了赛事的创建和配置后，可以进入当场比赛，利用代码收集模块提供的接口向客户端的代码提交模块发送收集请求，在收集代码后，查重检测模块会对所提交的代码文件进行必要的代码查重，管理员可以通过统计分析模块所提供的功能查看本场比赛的选手到场情况、各个选手的完成情况及比赛的状况总览图等统计分析信息。

存储&接口设计

注：存储使用Django的ORM功能，只需要在Django定义相应class并对相应的对象进行操作，便可以自动同步到数据库中，无需数据库建表和增删改查操作。

1. **账号管理模块（authenz）**
    - 存储

```
# 系统用户
class User(AbstractUser):
    # 已从AbstractUser继承name / password / email等字段
    nick_name = models.CharField(max_length=50, blank=True)
    isactivate = models.BooleanField(default=False) # 是否已邮箱激活
```

- 接口

```
url前缀（prefix）：r'authenz/'

urlpatterns = [
    url(r'register', user_register),  # 注册
    url(r'login', user_login),  # 登陆
    url(r'logout', user_logout),  # 注销
    url(r'activate/', activate),    # 激活账号
]
```

2. **赛事管理模块（competition）**

- 存储

```
# 考场
class Room(models.Model):
    name = models.CharField(max_length=64, unique=True)  # 考场名
    layout_matrix = models.CharField(max_length=512)  # 布局矩阵

# 赛事
class Competition(models.Model):
    name = models.CharField(max_length=64, unique=True)  # 竞赛名
    description = models.TextField(blank=True, null=True)  # 描述
    created_time = models.DateTimeField(auto_now_add=True)  # 创建时间
    submission_standard = JSONField()  # 标准提交格式
    submission_path = models.CharField(max_length=64)  # 选手存放路径
    room_layout = models.ForeignKey(to=Room)  # 使用考场

# 参赛者
class Participant(models.Model):
    name = models.CharField(max_length=64)  # 参赛者名
    pno = models.CharField(max_length=64)   # 考号
    province = models.CharField(max_length=32)  # 省份
    school = models.CharField(max_length=32)    # 学校
    grade = models.CharField(max_length=32)     # 年级
    id_num = models.CharField(max_length=32)    # 身份证号
```

```
    competition = models.ForeignKey(to=Competition)  # 所属比
赛
    host = models.CharField(max_length=64)   # 选手主机
    position = models.CharField(max_length=16)   # 选手座位号
```

- 接口

```
url前缀（prefix）：r'competition/'

urlpatterns = [
  # 参赛者相关接口
  url(r'(?P<cid>[0-9]{1,})/participants-list'), # 列出所有参赛
者
  url(r'(?P<cid>[0-9]{1,})/participants-create'), # 创建参赛者
  url(r'(?P<cid>[0-9]{1,})/participant-(?P<pid>[0-9]{1,})-
delete'), # 删除参赛者
  url(r'(?P<cid>[0-9]{1,})/participant-(?P<pid>[0-9]{1,})-
update'), # 更新参赛者
  url(r'(?P<cid>[0-9]{1,})/participants-csv'), # 下载参赛者名单
csv

  # 考场相关接口
  url(r'room-list'), # 列出所有考场
  url(r'room-(?P<rid>[0-9]{1,})-delete'), # 删除考场信息
  url(r'room-(?P<rid>[0-9]{1,})-update'), # 更新考场信息
  url(r'room-create'), # 创建考场信息

  # 赛事相关接口
  url(r'create'), # 创建比赛
  url(r'list-admin'), # 列出所有比赛供维护
  url(r'list-submission'), # 列出所有比赛供选择查看提交情况
  url(r'(?P<cid>[0-9]{1,})/delete'), # 删除比赛
  url(r'(?P<cid>[0-9]{1,})/detail'), # 某个比赛的提交详情
  url(r'(?P<cid>[0-9]{1,})/update'), # 更新某个比赛信息
]
```

3. **代码收集模块（submission）**

- 存储

```
# 选手的提交
class Submission(models.Model):
    bundle = models.FileField() # 选手提交压缩包的相对路径
    filtered_bundle = models.FileField()  # 已过滤的包相对路径
    participant = models.ForeignKey(to=Participant)   # 提交
者
    status = models.CharField(max_length=64)   # 状态
    bundle_structure = JSONField(blank=True, null=True)  # 包
内文件结构
    missing_files = JSONField(blank=True, null=True)    # 相
对于标准所缺文件
```

- 接口

```
url前缀（prefix）：r'submission/'

urlpatterns = [
  url(r'create'), # submission的接收与创建
  url(r'download-(?P<cid>[0-9]{1,})-all-submissions'), # 打包
下载所有提交
  url(r'request-(?P<cid>[0-9]{1,})-all-submissions'),  # 向所
有参赛选手发送收集请求
  url(r'request-(?P<cid>[0-9]{1,})-(?P<pid>[0-9]{1,})-
single-submissions', )  # 向某个参赛选手发送收集请求
]
```

4. **统计分析模块（statistics）**

- 存储

  无

- 接口

```
url前缀（prefix）：r'statistics/'

urlpatterns = [
  url(r'list-statistics'), # 列出所有比赛，供选择查看统计详情
  url(r'(?P<cid>[0-9]{1,})/summary_graph'),  # 总览图
  url(r'(?P<cid>[0-9]{1,})/attendance-statistics'), # 出席状况
统计
  url(r'(?P<cid>[0-9]{1,})/completion-statistics'), # 完成度统
计
  url(r'(?P<cid>[0-9]{1,})/attended-participant-csv'), # 下载
出席选手csv名单
  url(r'(?P<cid>[0-9]{1,})/unattended-participant-csv'), # 下
载缺席csv名单
  url(r'(?P<cid>[0-9]{1,})/compeltion-csv') # 下载完成度统计csv
]
```

5. **查重检测模块（detector）**

   - 存储

```
# 相似度记录
class Similarity(models.Model):
    competition = models.ForeignKey(to=Competition) # 所在比赛

    src_submission = models.ForeignKey(to=Submission) # 源选手提交
    src_file = models.CharField(max_length=64)  # 源文件

    dest_submission = models.ForeignKey(to=Submission) # 被比对的选手提交
    dest_file = models.CharField(max_length=64) # 被比对的文件

    percentage = models.IntegerField() # 相似百分比
```

   - 接口

```
url前缀（prefix）：r'detector/'

urlpatterns = [
    url(r'list-detector'), # 列出所有比赛供选择查看查重情况
    url(r'(?P<cid>[0-9]{1,})/plagiarism_detail'), # 某个比赛的查重详情报表
]
```