QAQ_OJ 需求规格说明书

学号	姓名	分工
15030199012	朱仁博	用例图、文档整理
15030199014	方小龙	用例图、对象图
15030199019	菅宁	对象图、文档编写
15030199025	张炳婷	用例图、类图

1. 范围

1.1. 标识

完整标识: QAQ_OJ (QAQ_Online Judge System) 1.0
版本号: 1.0
发行号: 1.0

1.2. 系统概述

本 OJ 可以为算法爱好者、普通学生和各高校的老师等群体提供方便的服务。最主要的功能为题目练习与比赛承办。

管理员可以开设一场比赛，普通用户可以选择参与比赛，在规定的时间内，尽可能完成相应的题目。管理员可以在比赛过程中和比赛结束后获得用户的答题统计情况和一系列排名数据。

项目开发过程主要靠四位成员协力合作，根据各自的擅长的方向完成各自的分工，项目前期包括整体逻辑结构的分析、主要功能的而实现(比如账号登录,上传题目与判题等功能)。中期完成后端数据库建设与维护、前端页面的设计与修改、安全性的考虑。后期则完成诸多细节的优化与用户体验的完善。

1.3. 文档概述

本文档为 QAQ_OJ 的需求规格说明书，其中包括本系统的用例图、对象图、类图、功能需求、非功能需求等。由于这只是一个课程项目，不涉及相关公司和相关部门的权利问题，因此没有特定的保密性和私密性要求。

1.4. 基线

需求基线，小组成员已讨论完成项目的各项需求，即将完成这份需求规格说明书。

2. 引用文件

《软件工程导论（第五版)》　张海藩编著 清华大学出版社出版

3. 需求

3.1. 所需的状态和方式

① 用户的状态:
   1) 未登录
   2) 已登陆

② 公告/留言/问题/比赛的状态:
   1) 隐藏
   2) 可见

③ 比赛的状态:
   1) 未开始
   2) 进行中
   3) 已结束

④ 比赛中用户的状态:

1) 未报名
2) 已报名

⑤ 提交/Hack 的状态图

1) 等待评测
2) 评测中
3) 评测结束

3.2. 需求概述

3.2.1. 目标

OJ 拥有完整的功能，安全高效的评测机制，良好的人机交互界面，让绝大部分使用的用户满意，其中，每种用户的权限如下:

① 超级管理员

1) 发布/管理主页的公告;
2) 管理所有用户的帐号密码等信息;
3) 管理讨论板块的数据;
4) 管理员权限的所有操作;

② 管理员

1) 新建问题，管理问题;
2) 新建比赛，管理自己创建的比赛;
3) 查看练习模块中的所有提交记录;
4) 查看联系模块中的所有 Hack 记录;
5) 普通用户的所有操作;

③ 普通用户

1) 管理个人帐号信息;
2) 查看公告信息;
3) 在练习模块中提交代码，查看返回结果;
4) 查看自己的提交记录;
5) 使用 Hack 功能，查看自己的 Hack 记录;
6) 在讨论模块中参与讨论;
7) 查看 Rank 榜单;
8) 参加比赛;

数据流图、数据字典可见可行性分析报告。

3.2.2. 运行环境

服务器端: Ubuntu16.04 LTS。

用户端: 拥有浏览器的系统均可。

3.2.3. 用户的特点

有以下 4 种类型用户:

① 算法爱好者: 希望提升算法水平。

② 普通学生: 希望提升编程水平。

③ 教师: 追求教学考核等的方便。

④ 公司或某些导师: 希望选拔人才。

3.2.4. 关键点

关键功能：判题功能、Hack 功能。

关键技术：django 框架、生产者消费者模型、沙箱技术。

3.2.5. 约束条件

经费限制：普通学生可以负担的开支。

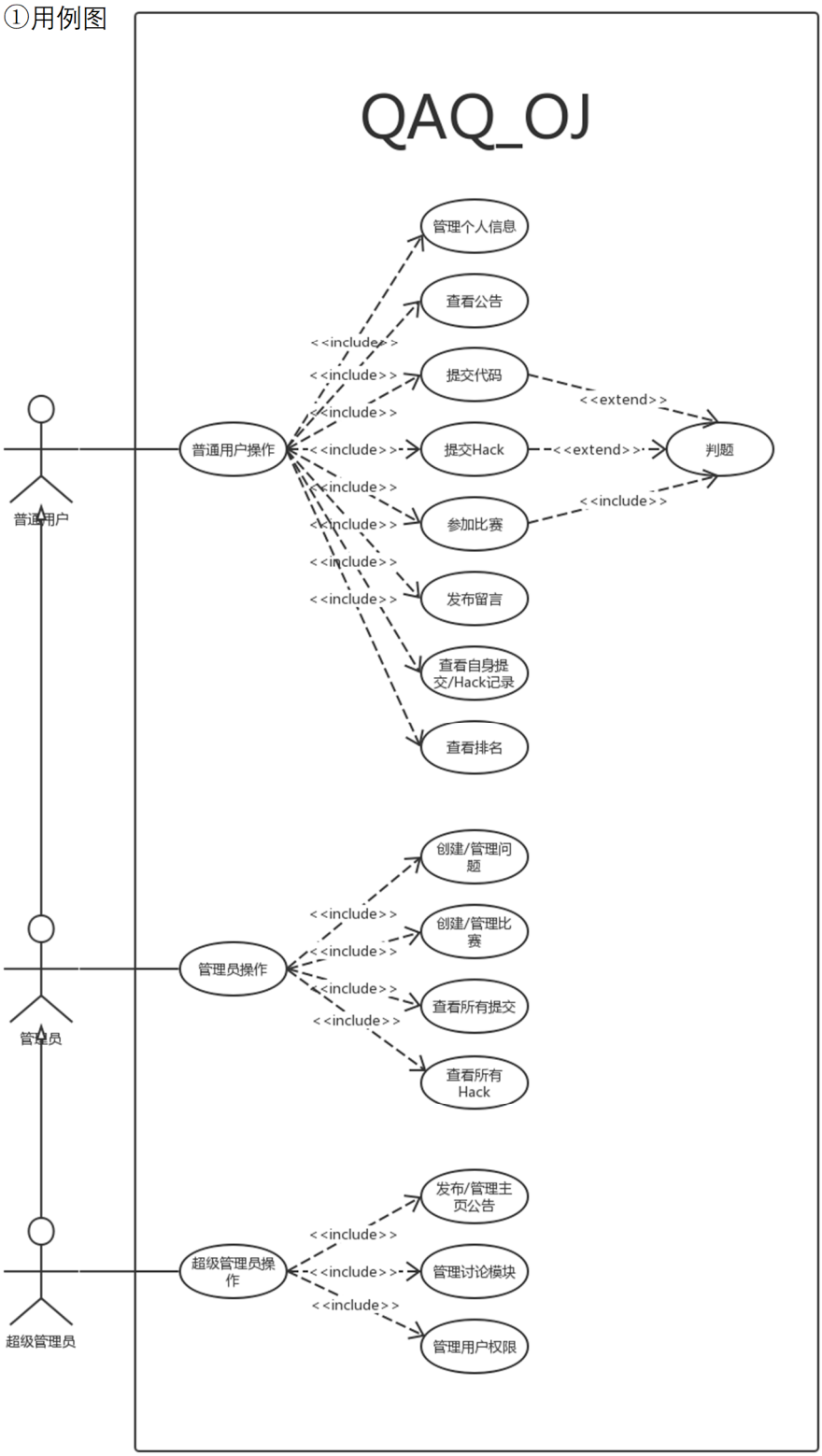
开发期限：3 个月。

无特殊的政治、社会、文化、法律约束条件。

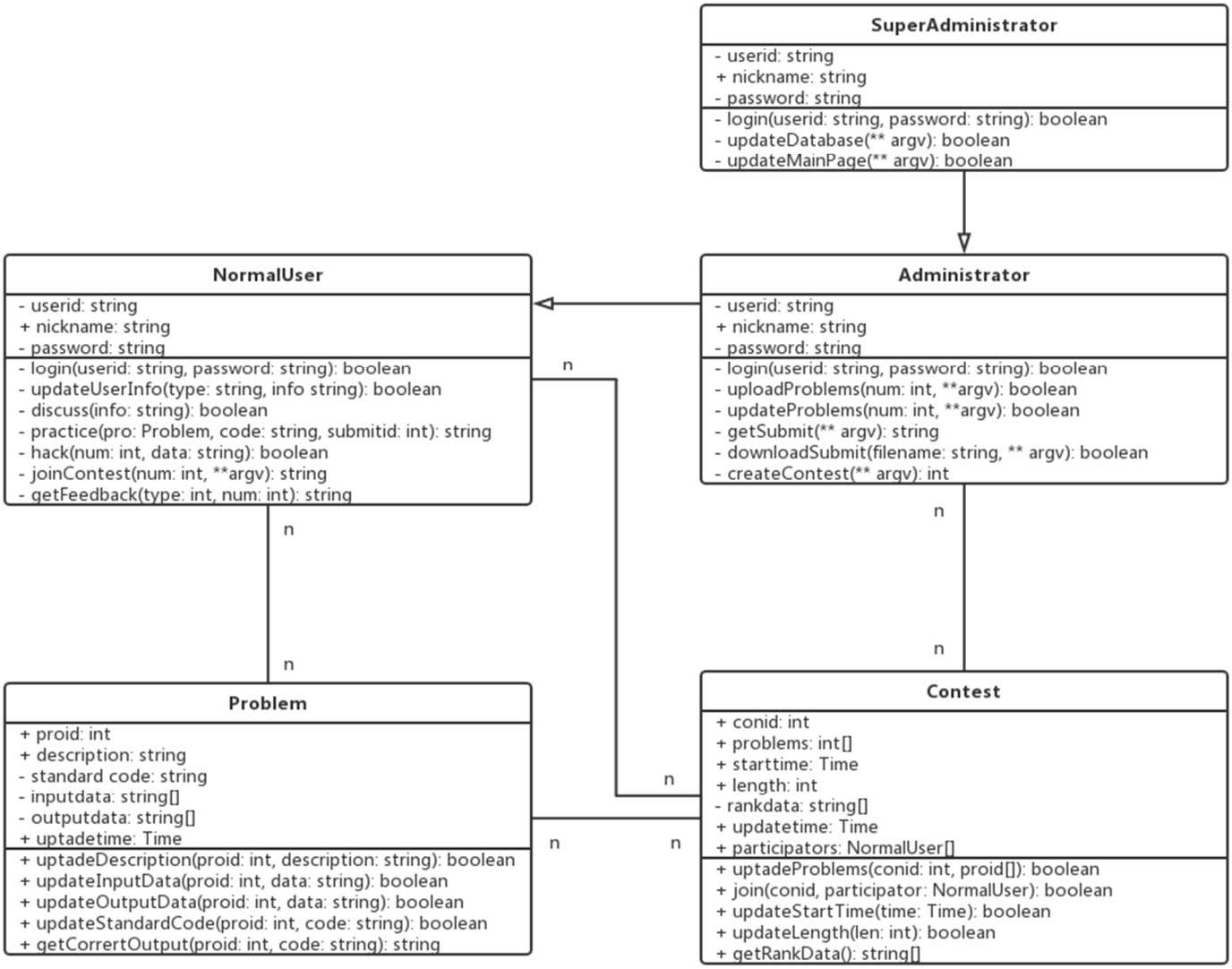
3.3. 需求规格

3.3.1. 软件系统总体功能/对象结构

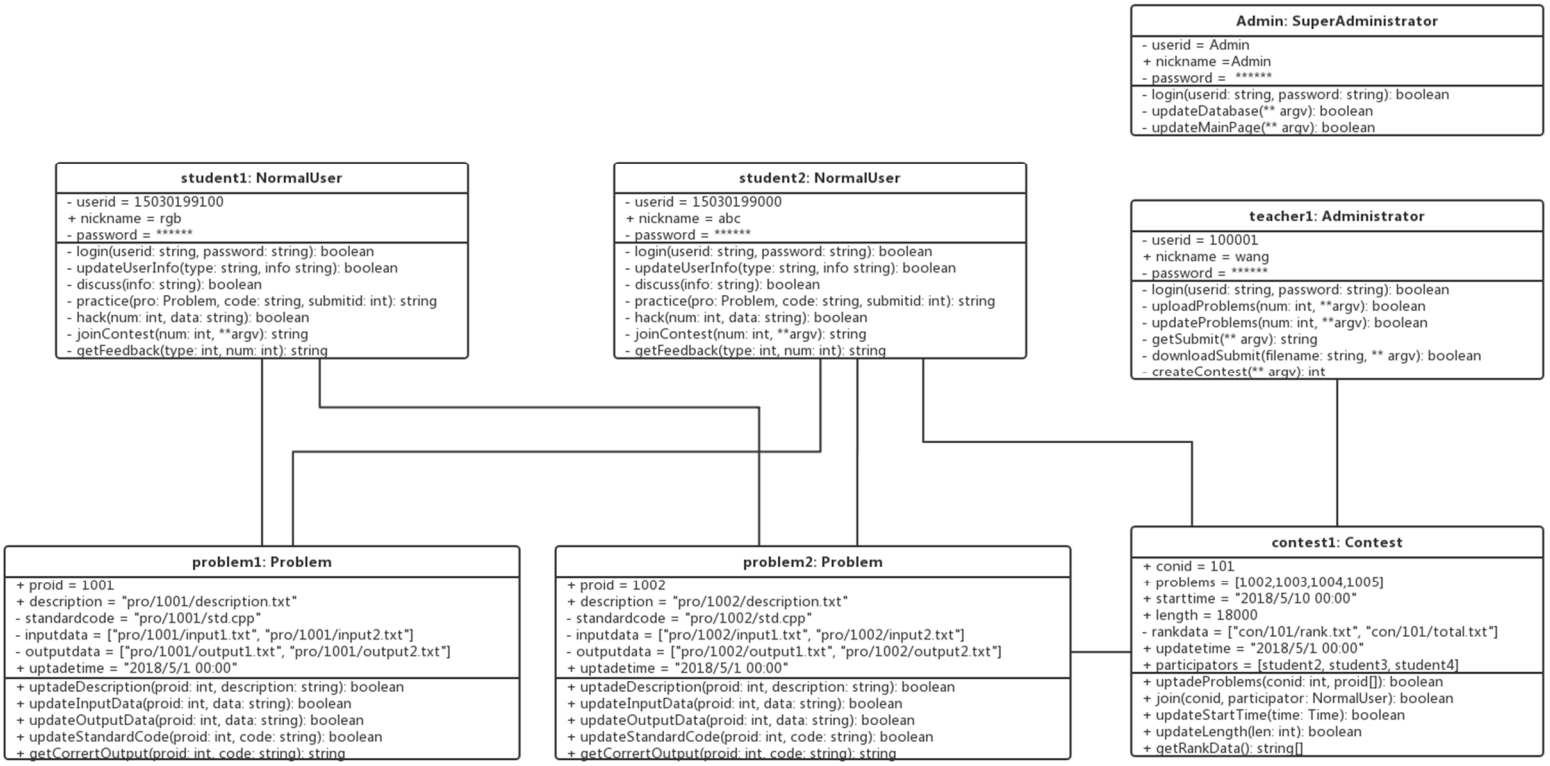
①用例图



②类图


SuperAdministrator
- userid: string
+ nickname: string
- password: string
- login(userid: string, password: string): boolean
- updateDatabase(** argv): boolean
- updateMainPage(** argv): boolean
NormalUser
- userid: string
+ nickname: string
- password: string
- login(userid: string, password: string): boolean
- updateUserInfo(type: string, info string): boolean
- discuss(info: string): boolean
- practice(pro: Problem, code: string, submitid: int): string
- hack(num: int, data: string): boolean
- joinContest(num: int, **argv): string
- getFeedback(type: int, num: int): string
Administrator
- userid: string
+ nickname: string
- password: string
- login(userid: string, password: string): boolean
- uploadProblems(num: int, **argv): boolean
- updateProblems(num: int, **argv): boolean
- getSubmit(** argv): string
- downloadSubmit(filename: string, ** argv): boolean
- createContest(** argv): int
Problem
+ proid: int
+ description: string
- standard code: string
- inputdata: string[]
- outputdata: string[]
+ uptadetime: Time
+ uptadeDescription(proid: int, description: string): boolean
+ updateInputData(proid: int, data: string): boolean
+ updateOutputData(proid: int, data: string): boolean
+ updateStandardCode(proid: int, code: string): boolean
+ getCorrertOutput(proid: int, code: string): string
Contest
+ conid: int
+ problems: int[]
+ starttime: Time
+ length: int
- rankdata: string[]
+ updatetime: Time
+ participators: NormalUser[]
+ uptadeProblems(conid: int, proid[]): boolean
+ join(conid, participator: NormalUser): boolean
+ updateStartTime(time: Time): boolean
+ updateLength(len: int): boolean
+ getRankData(): string[]
n
n
n
n
n
n
n
n

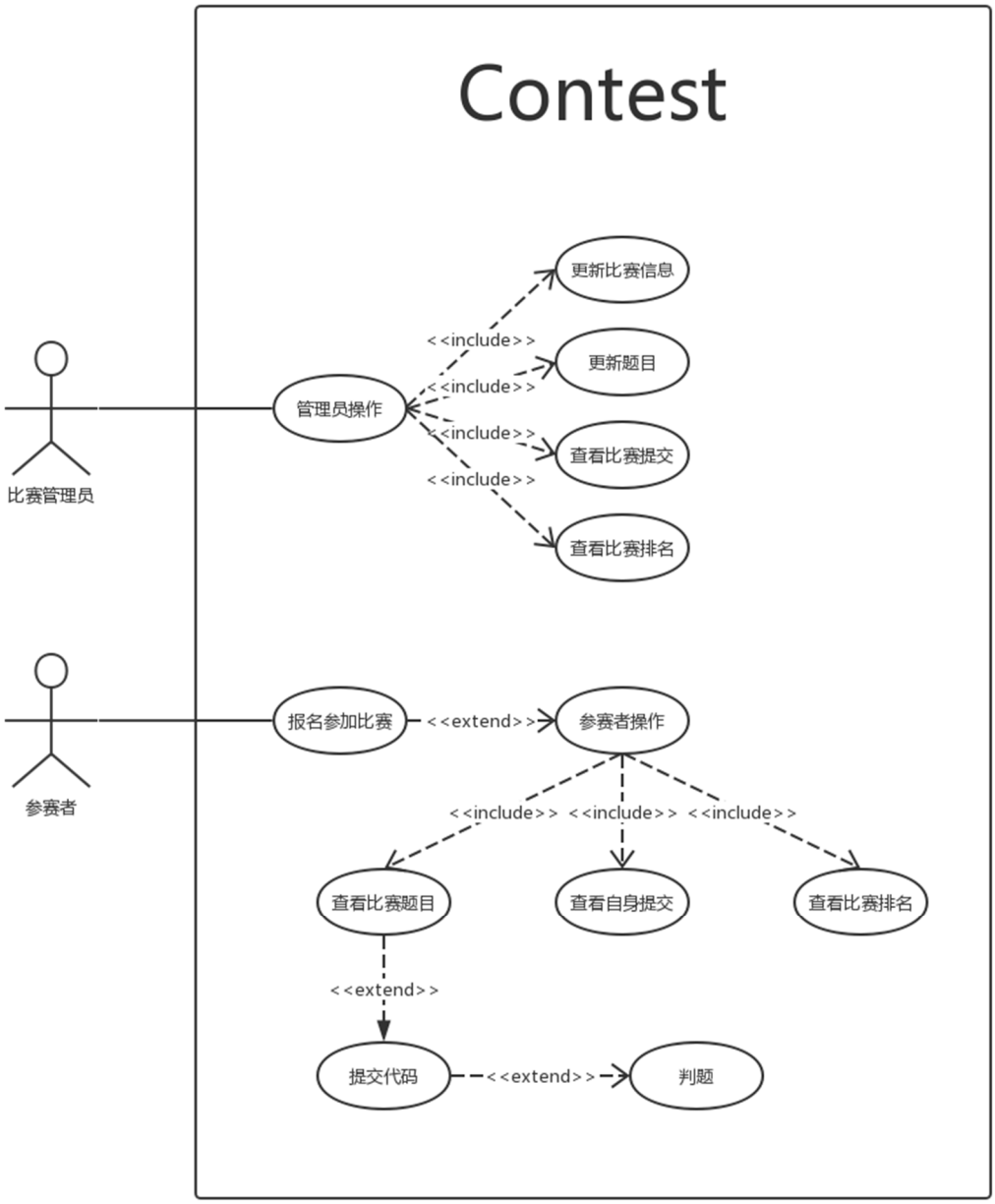
③对象图


Admin: SuperAdministrator
- userid = Admin
+ nickname =Admin
- password = ******
- login(userid: string, password: string): boolean
- updateDatabase(** argv): boolean
- updateMainPage(** argv): boolean
student1: NormalUser
- userid = 15030199100
+ nickname = rgb
- password = ******
- login(userid: string, password: string): boolean
- updateUserInfo(type: string, info string): boolean
- discuss(info: string): boolean
- practice(pro: Problem, code: string, submitid: int): string
- hack(num: int, data: string): boolean
- joinContest(num: int, **argv): string
- getFeedback(type: int, num: int): string
student2: NormalUser
- userid = 15030199000
+ nickname = abc
- password = ******
- login(userid: string, password: string): boolean
- updateUserInfo(type: string, info string): boolean
- discuss(info: string): boolean
- practice(pro: Problem, code: string, submitid: int): string
- hack(num: int, data: string): boolean
- joinContest(num: int, **argv): string
- getFeedback(type: int, num: int): string
teacher1: Administrator
- userid = 100001
+ nickname = wang
- password = ******
- login(userid: string, password: string): boolean
- uploadProblems(num: int, **argv): boolean
- updateProblems(num: int, **argv): boolean
- getSubmit(** argv): string
- downloadSubmit(filename: string, ** argv): boolean
- createContest(** argv): int
problem1: Problem
+ proid = 1001
+ description = "pro/1001/description.txt"
- standardcode = "pro/1001/std.cpp"
- inputdata = ["pro/1001/input1.txt", "pro/1001/input2.txt"]
- outputdata = ["pro/1001/output1.txt", "pro/1001/output2.txt"]
+ uptadetime = "2018/5/1 00:00"
+ uptadeDescription(proid: int, description: string): boolean
+ updateInputData(proid: int, data: string): boolean
+ updateOutputData(proid: int, data: string): boolean
+ updateStandardCode(proid: int, code: string): boolean
+ getCorrertOutput(proid: int, code: string): string
problem2: Problem
+ proid = 1002
+ description = "pro/1002/description.txt"
- standardcode = "pro/1002/std.cpp"
- inputdata = ["pro/1002/input1.txt", "pro/1002/input2.txt"]
- outputdata = ["pro/1002/output1.txt", "pro/1002/output2.txt"]
+ uptadetime = "2018/5/1 00:00"
+ uptadeDescription(proid: int, description: string): boolean
+ updateInputData(proid: int, data: string): boolean
+ updateOutputData(proid: int, data: string): boolean
+ updateStandardCode(proid: int, code: string): boolean
+ getCorrertOutput(proid: int, code: string): string
contest1: Contest
+ conid = 101
+ problems = [1002,1003,1004,1005]
+ starttime = "2018/5/10 00:00"
+ length = 18000
- rankdata = ["con/101/rank.txt", "con/101/total.txt"]
+ updatetime = "2018/5/1 00:00"
+ participators = [student2, student3, student4]
+ uptadeProblems(conid: int, proid[]): boolean
+ join(conid, participator: NormalUser): boolean
+ updateStartTime(time: Time): boolean
+ updateLength(len: int): boolean
+ getRankData(): string[]

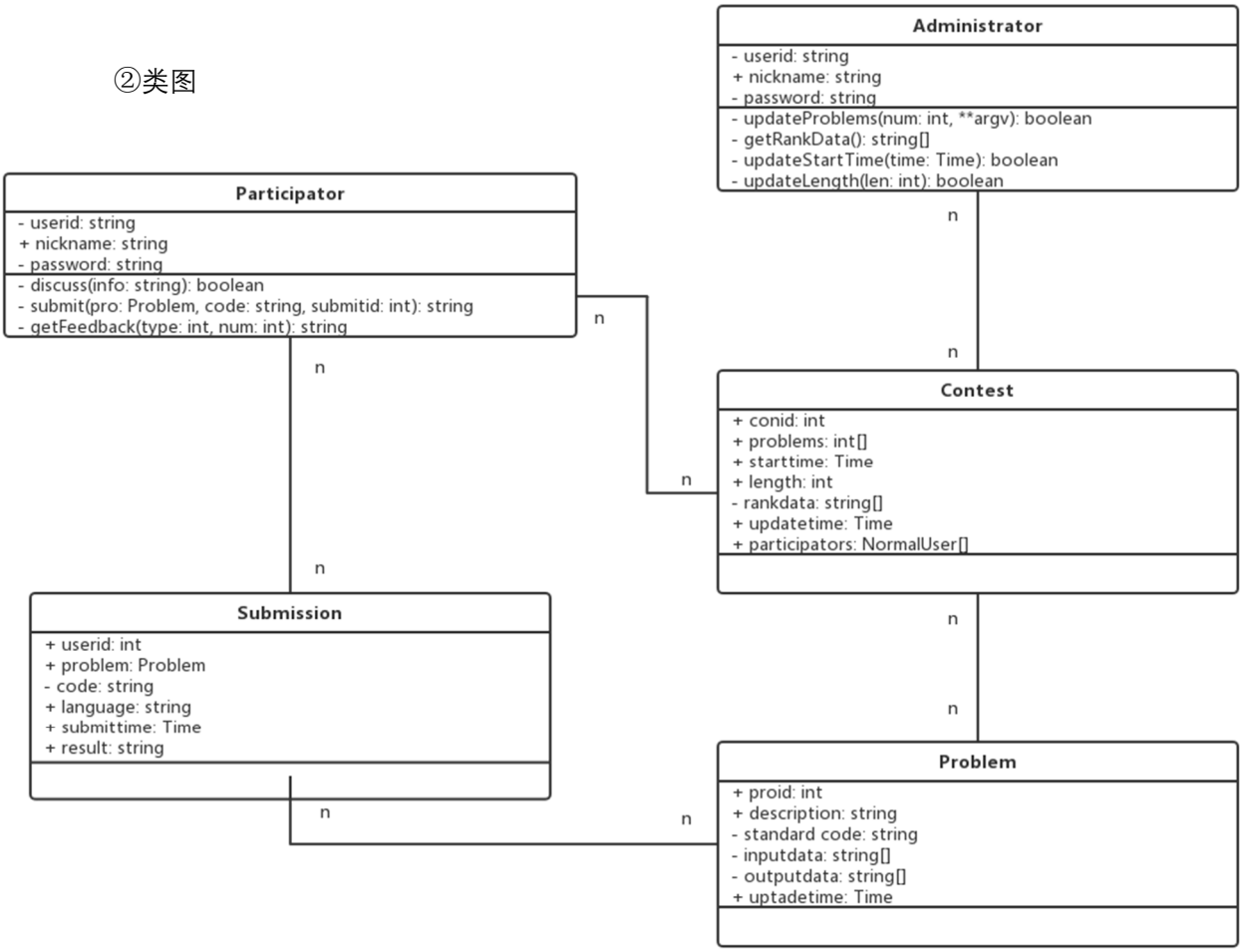
3.3.2. 软件子系统功能/对象结构

3.3.2.1. Contest

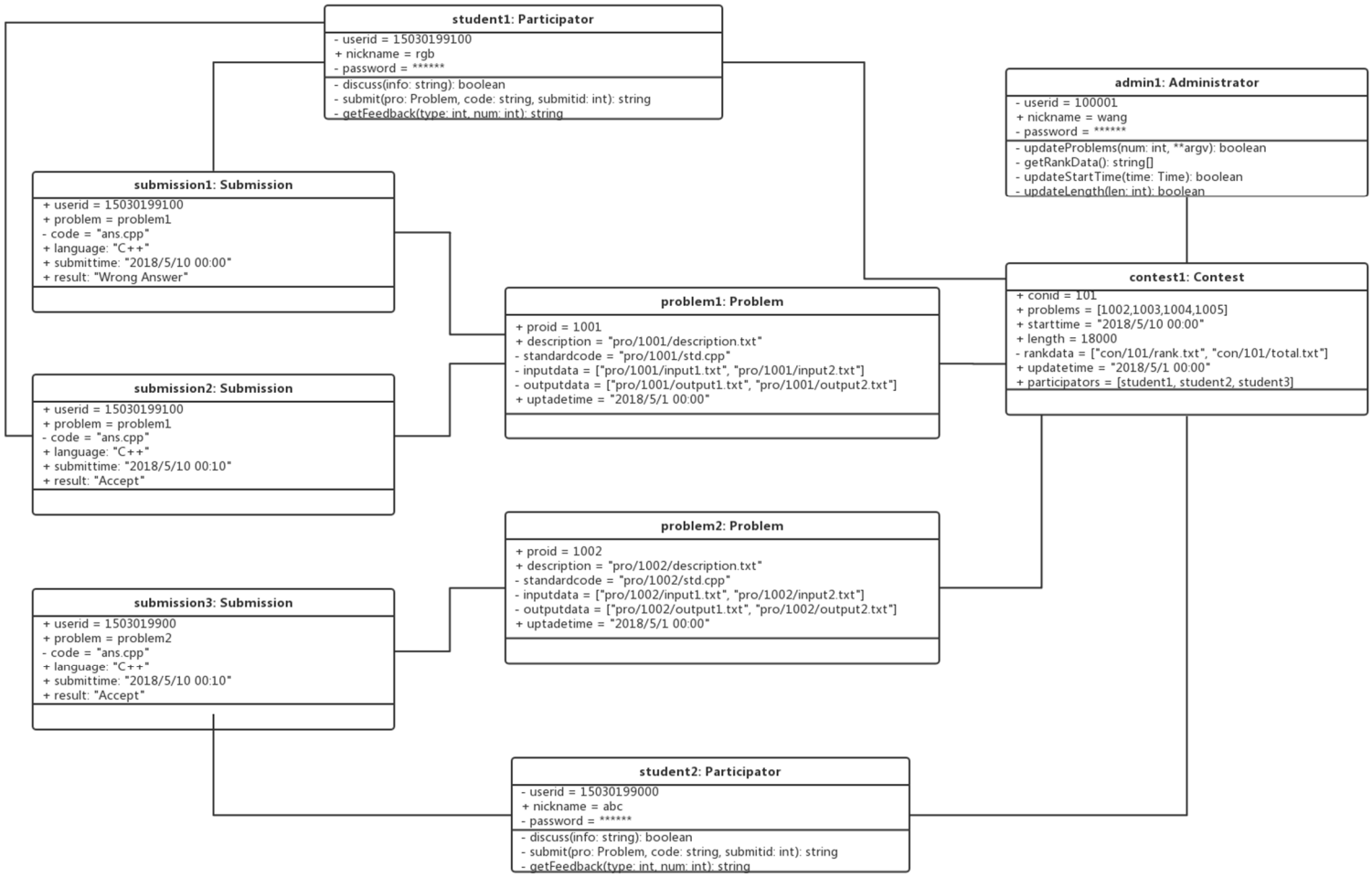
①用例图



②类图

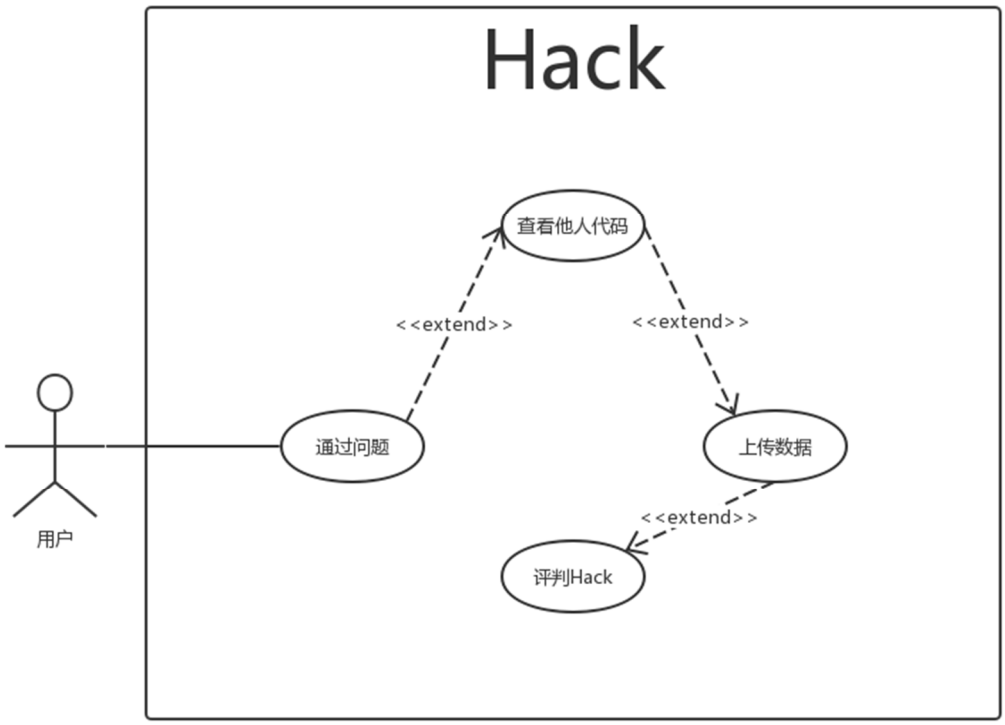

Administrator
- userid: string
+ nickname: string
- password: string
- updateProblems(num: int, **argv): boolean
- getRankData(): string[]
- updateStartTime(time: Time): boolean
- updateLength(len: int): boolean
Participator
- userid: string
+ nickname: string
- password: string
- discuss(info: string): boolean
- submit(pro: Problem, code: string, submitid: int): string
- getFeedback(type: int, num: int): string
Contest
+ conid: int
+ problems: int[]
+ starttime: Time
+ length: int
- rankdata: string[]
+ updatetime: Time
+ participators: NormalUser[]
Submission
+ userid: int
+ problem: Problem
- code: string
+ language: string
+ submittime: Time
+ result: string
Problem
+ proid: int
+ description: string
- standard code: string
- inputdata: string[]
- outputdata: string[]
+ uptadetime: Time
n

③对象图

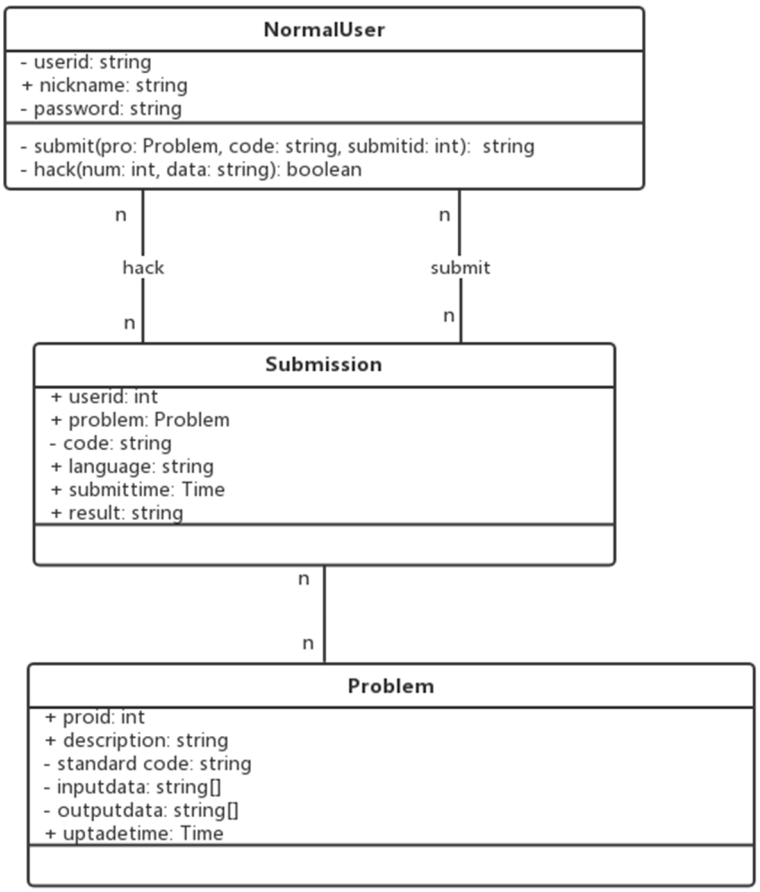

student1: Participator
- userid = 15030199100
+ nickname = rgb
- password = ******
- discuss(info: string): boolean
- submit(pro: Problem, code: string, submitid: int): string
- getFeedback(type: int, num: int): string
admin1: Administrator
- userid = 100001
+ nickname = wang
- password = ******
- updateProblems(num: int, **argv): boolean
- getRankData(): string[]
- updateStartTime(time: Time): boolean
- updateLength(len: int): boolean
submission1: Submission
+ userid = 15030199100
+ problem = problem1
- code = "ans.cpp"
+ language: "C++"
+ submittime: "2018/5/10 00:00"
+ result: "Wrong Answer"
problem1: Problem
+ proid = 1001
+ description = "pro/1001/description.txt"
- standardcode = "pro/1001/std.cpp"
- inputdata = ["pro/1001/input1.txt", "pro/1001/input2.txt"]
- outputdata = ["pro/1001/output1.txt", "pro/1001/output2.txt"]
+ uptadetime = "2018/5/1 00:00"
contest1: Contest
+ conid = 101
+ problems = [1002,1003,1004,1005]
+ starttime = "2018/5/10 00:00"
+ length = 18000
- rankdata = ["con/101/rank.txt", "con/101/total.txt"]
+ updatetime = "2018/5/1 00:00"
+ participators = [student1, student2, student3]
submission2: Submission
+ userid = 15030199100
+ problem = problem1
- code = "ans.cpp"
+ language: "C++"
+ submittime: "2018/5/10 00:10"
+ result: "Accept"
problem2: Problem
+ proid = 1002
+ description = "pro/1002/description.txt"
- standardcode = "pro/1002/std.cpp"
- inputdata = ["pro/1002/input1.txt", "pro/1002/input2.txt"]
- outputdata = ["pro/1002/output1.txt", "pro/1002/output2.txt"]
+ uptadetime = "2018/5/1 00:00"
submission3: Submission
+ userid = 1503019900
+ problem = problem2
- code = "ans.cpp"
+ language: "C++"
+ submittime: "2018/5/10 00:10"
+ result: "Accept"
student2: Participator
- userid = 15030199000
+ nickname = abc
- password = ******
- discuss(info: string): boolean
- submit(pro: Problem, code: string, submitid: int): string
- getFeedback(type: int, num: int): string

3.3.2.2. Hack

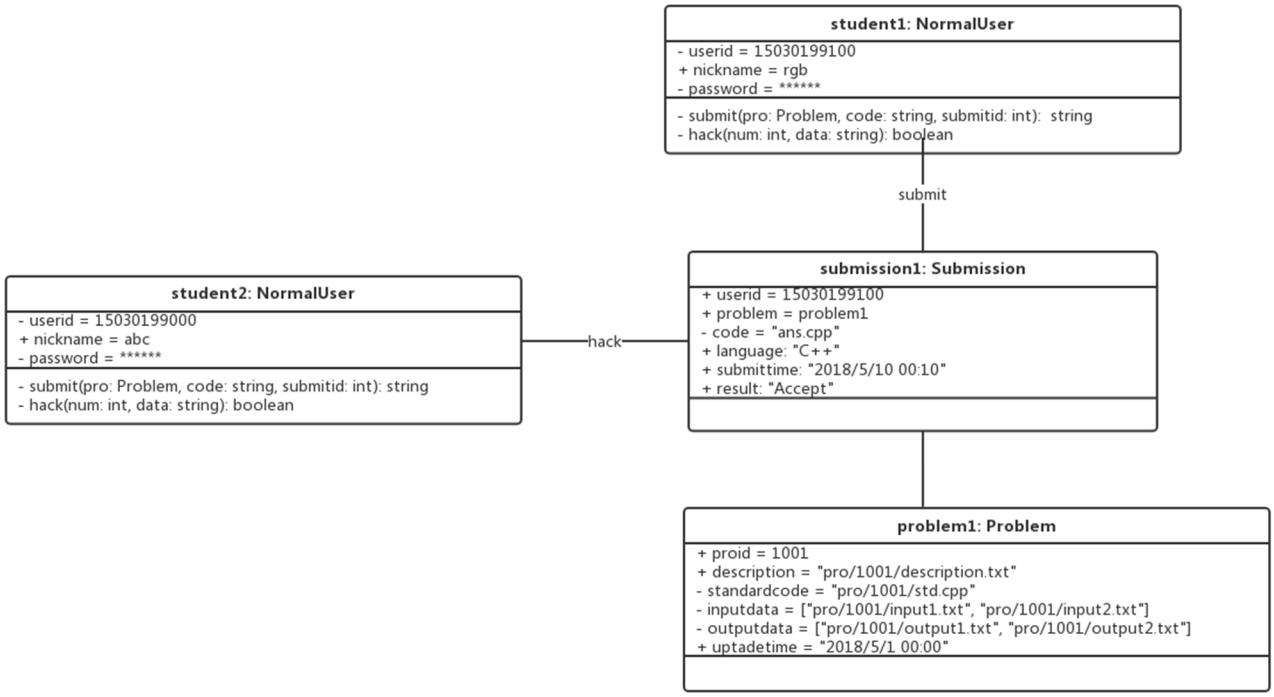
①用例图



②类图



③对象图



3.3.3. 描述约定

无。

3.4. CSCI 能力需求

无。

3.5. CSCI 外部接口需求

① 用户接口：采用 B/S 架构，所有界面使用 WEB 风格，用户界面的具体细节将在概要设计文档中描述。

② 硬件接口：可以链接到网络的硬件设备。

③ 软件接口：mysql 端口、redis 端口。

④ 通信接口：可以连接到网络的接口。

3.6. CSCI 内部接口需求

① 判题模块的接口：

输入：数据的路径，提交的代码，问题的限制。

输出：判题结果。

② Hack 模块的接口：

输入：提交记录，Hack 数据。

输出：Hack 结果。

3.7. CSCI 内部数据需求

数据库采用 mysql，由 django 登录 mysql 的 root 帐号管理。

3.8. 适应性需求

无。

3.9. 保密性需求

记录日志：本系统能够记录系统运行时所发生的所有错误，包括本机错误和网路错误。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。

3.10. 保密性和私密性需求

① 不同用户的权限设有相关限制，可以识别部分非法访问和恶意访问，数据库的信息只有 mysql 的 root 账户或 django 的超级管理员可以查看。

② 用户密码 Hash 加密储存。

3.11. CSCI 环境需求

服务器系统：Ubuntu16.04 LTS。

3.12. 计算机资源需求

3.12.1. 计算机硬件需求

服务器：CPU 2.3GHZ、RAM 4GB、硬盘 50G。

客户端：可浏览网页的硬件条件即可。

3.12.2. 计算机硬件资源利用需求

无。

3.12.3. 计算机软件需求

开发语言：python3。

网站架构：django。

数据库：mysql。

其它：gcc、g++、jdk、celery、redis、浏览器。

3.12.4. 计算机通信需求

可以连接到 Internet。

3.13. 软件质量因素

易于用户使用、服务器稳定、反馈迅速、易于维护。

3.14. 设计和实现的约束

无。

3.15. 数据

① 在线用户：服务器可承受的同时在线用户数大于 500。

② 判题效率：每秒至少处理 2 个提交（具体视数据量及机器配置而定）。

3.16. 操作

① 常规操作：用户注册，提交代码，提交 Hack，参加比赛等操作均无特殊要求。

② 特殊操作：管理员新建题目，要求上传题目数据；管理员举办比赛，对比赛的进行时间有限制，超级管理员管理帐号信息库，可设置用户权限。

③ 初始化操作：服务器第一次使用时，应先创建超级管理员用户。

④ 恢复操作：恢复数据时，要求不修改数据库表单。

3.17. 故障处理

① 数据库故障：回滚到某一备份的时刻。

② 网络错误：根据 HTTP 状态码分情况解决。

③ 服务器故障：重启服务器。

3.18. 算法说明

无。

3.19. 有关人员需求

无。

3.20. 有关培训需求

无。

3.21. 有关后勤需求

无。

3.22. 其它需求

题库需求：为了系统完成后可以投入运行，我们可能需要提前收集一些题目，编写标准程序以及输入数据和输出数据，以保证前期运行的用户体验。

3.23. 包装需求

无。

3.24. 需求的有限次序和关键程度

环境需求 > 硬件需求 > 性能需求 > 其它。

4. 合格性规定

仅规定系统的可用性，采用测试的方法，分别让管理员和普通用户执行所有相关的操作，查看后台数据库的更新情况和前端反馈的结果，若一切均正常，则合格。

5. 需求可追踪性

无。

6. 尚未解决的问题

后台判题核心的具体实现。

7. 注解

① 生产者消费者模型

有两个进程：一组生产者进程和一组消费者进程共享一个初始为空、固定大小为 n 的缓存（缓冲区）。生产者的工作是制造一段数据，只有缓冲区没满时，生产者才能把消息放入到缓冲区，否则必须等待，如此反复；同时，只有缓冲区不空时，消费者才能从中取出消息，一次消费一段数据（即将其从缓存中移出），否则必须等待。由于缓冲区是临界资源，它只允许一个生产者放入消息，或者一个消费者从中取出消息。问题的核心是：

1.要保证不让生产者在缓存还是满的时候仍然要向内写数据;

2.不让消费者试图从空的缓存中取出数据。

生产者和消费者对缓冲区互斥访问是互斥关系，同时生产者和消费者又是一个相互协作的关系，只有生产者生产之后，消费者才能消费，他们也是同步关系。

② 沙箱技术

Sandboxie(又叫沙箱、沙盘)即是一个虚拟系统程序，允许你在沙盘环境中运行浏览器或其他程序，因此运行所产生的变化可以随后删除。它创造了一个类似沙盒的独立作业环境，在其内部运行的程序并不能对硬盘产生永久性的影响。其为一个独立的虚拟环境，可以用来测试不受信任的应用程序或上网行为。

沙箱是一种按照安全策略限制程序行为的执行环境。早期主要用于测试可疑软件等，比如黑客们为了试用某种病毒或者不安全产品，往往可以将它们在沙箱环境中运行。

经典的沙箱系统的实现途径一般是通过拦截系统调用，监视程序行为，然后依据用户定义的策略来控制和限制程序对计算机资源的使用，比如改写注册表，读写磁盘等。