Code Challenge Specification

Team 142 [Coding Challenge!]
Hailei Yu - haileiy@andrew.cmu.edu
JiaXi Xiong - jiaxix@andrew.cmu.edu
Jing Li - jli3@andrew.cmu.edu
Status: draft
Last modified: 10/10/2015

Functionality

User service module

- **Registration**: any user, with a valid email address, can register an account.
- **Login**: a registered user can log in using his/her credentials. Most of the functionalities are available to loggedin users only.
- **Logout**: a user currently logged in can log out.
- Reset password: a registered user can reset password when logged in.
- **Group**: there are three groups with different access: root, admin and normal user. A registered user must belong to one of the groups.

Problem service module

- **Add new problem**: admin and root can add new problem (description, test cases).
- **Update existing problem**: admin and root can update the description and test cases of problems.

- **Solve a problem**: any registered user can code and submit for evaluation. There are four possible outcomes: accepted, wrong answer, time limit exceeded, memory limit exceeded. Only the first outcome is considered successful.

Search service module

- **Basic**: users, when logged in, can search by keywords for problems and discussions.
- **Category**: users can choose a category (problem or discussion) to search.

Forum service module

- **Create posts**: users, when logged in, can create posts to discuss. Every post must belong to a specific problem.
- **View posts related to a problem**: users, when logged in, can view the posts related to a specific problem.
- **Follow up**: users, when logged in, can follow up with a post by creating followup_posts.
- Vote: users, when logged in, can vote up or vote down posts. The user may update his/her votes, and only the latest vote is counted.

Data Models

Below is a skeleton of data models. Please refer to the models.py file for more details.

UserProfile: id, username, password, email, group

Submission: id, createdAt, result, language_type, **user**, **problem**

Problem: id, description, tests(fileUrl), createdAt, **author**, **admin**

DiscussionPost: id, **problem**, title, body, **author**, createdAt

FollowUp: id, **discussionPost**, body, **author**, createdAt

Vote: **user**, **problem**, option(boolean => 1 : upvote; -1 : downvote)

Tag: id, name, problems(MangToManyField),

UI Design

The complete set of wireframes for the non-trivial views withthe application have been uploaded to the specification folder.

Security

In online judge, users can write and run their code. It is possible that malicious users try to break down the service. The solutions to each malicious behaviors are as follows:
Timeout: run a code with predefined time limit. If the code doesn't stop, kill it and throw an exceptipn.
Memory limit: run a code with predefined memory limit. If the code uses up the allocated memory, kill it and throw an exception.
Malicious code: prohibit imported module; run the code in sandbox.

Product Backlog:

- User registration
- User login
- User logout
- Password reset
- User profile change (including upload Avatar)
- Problem online judge
- Problem discussion posts - Follow up posts
- Problem history
- Problem add/edit
- Tags for the problems
- Post Discussion on a problem
- Comment on a discussion
- upvotes and downvotes a discussion

Main points from feeback:

1. PrismJS to do syntax highlighting: will do
2. APIs like Google Hangout to do coordiate live coding for mock interviews: considering
3. Safe => run in Sandbox: will do
4. AWS s3 -> file storage: will do
5. shift the focus less from an online judge to more like ProjectEuler and CodeGolf: considering