CMPE 275 Section 1 Spring 2019
Term Project - OpenHack

Last updated: 05/18/2019 (status: published)

In this project, you will build an online hackathon management service to create and organize hackathon events. For sake for simplicity, we call this service OpenHack(OH), but you can name it whatever you want. OH must have its server hosted in the cloud, allowing clients to connect from anywhere. The client app can be a web app, mobile app, or you can provide both. There are no language or technology requirements for UI implementation. The primary language you use for server implementation, however, *must* be Java. You do not have to use Spring, but you need to exercise the principles, patterns, and methodologies that you have learned in the class, such as DI, AOP, MVC, ORM, and transactions. You *must* use a relational database, e.g., MySQL, Amazon RDS (supports many popular relational databases), MSFT SQL Server, or Google Cloud SQL.

If any feature described below is unclear or ambiguous, and you fail to get a clear answer from the instructor or TA, you can use your best judgment to interpret it and add the missing details, provided that you clearly document and explain your reasoning in your project report.

Users and Authentication

Anyone with a valid email address can register as a user for OH. There are the following roles that a user can possibly assume.

- **Admin** An admin manages hackathons, including creation, committee appointment, registration, operation, and expenses. For simplicity, any one with the email address from the @sjsu.edu domain *automatically* assumes the role of an admin, and *none* of the users from other domains can become an admin. To avoid conflict of interest, a user with the role of an admin *cannot* be assigned other roles listed below, unless explicitly stated otherwise.
- **Hacker** A hacker is a potential hackathon participant, who can register for any open hackathon events. Any OH user automatically assumes the role of a hacker, *unless* he is an admin.
- **Judge** A judge can grade hackathon submissions. This role is appointed on a per hackathon basis by an admin user, i.e., Alice can be appointed as a judge for some hackathons, but function as a regular hacker for others. To avoid conflict of interest, it goes without saying that hacker *cannot* be *both* a judge and participant of the same hackathon event.

You can use Oauth/OpenID for user sign-up and authentication, but you *must* also support the option of letting the user set up a “local” password in OH with his email as the user name and his own password for OH. No matter which registration option is used, local account with own

password, or signing in with Facebook or Google, OH must send an email to the user (for both admin and customers) with a verification code or link. The user needs to use that verification code or link to complete his account registration.

A registered user cannot really use features in the system until his account is verified. A confirmation email must be sent to the user after completion of account verification. It is OK to use libraries like Firebase Authentication to assist user auth.

User Profile

Profile Attributes

Besides a valid email address, every user needs to have a unique alpha-numeric (at least 3 characters, no white spaces) screen name. Besides, at least the following *optional* attributes for users must be supported.

- Name
- Portrait URL // If present, you are highly encouraged to render the user's portrait in the UI where appropriate.
- Business Title
- Organization // A user can belong to maximum one organization at any given time.
- AboutMe // Free text to describe the user.
- Address

Screen name and email must be provided by the user as part of the sign up process. In the case of Oauth/OpenID, you can request access to the email. Screen names and emails cannot be changed after signing up.

Profile Page

1. You must have a profile page in the UI to render the above profile file info, and link to it as you see appropriate.
2. In the profile page, a user must be able to change *any* of his attributes, except email and screen name.

Organization Creation and Management

Any hacker can create an organization. Once an organization is created by a user, the user becomes the owner of an organization. Other hackers can join the organization.

Organization Attributes

An organization should have at least the following attributes, where *only* the name and owner attributes are required.

- Name // Must be unique across the system
- Owner // For simplicity, we do not allow change of ownership, hence this effectively the creator of the organization.
- Description
- Address

Organizations Membership

A user can join and leave organizations through his profile page. You must prompt the user to confirm this intention.

Joint An Organization

A user can join an organization by specifying the target organization's name in his profile page.

1. All organizations' names are public, and hence searchable. When a hacker specifies an organization name in his profile page, you are highly encouraged to provide auto completion as the user types in the name, so that he can select from one of the suggested results.
2. Right after a user specifies a new or different organization name, a join request is automatically sent to the organization's owner by the system.
3. While a hacker can attempt to join any organization, the membership is not effective until the owner approves the join request.
4. Rejoining after leaving needs approval as well.
5. Unapproved organization members cannot use organizational discount.

Leave An Organization

A user can leave an organization by changing his current organization or empty or a valid new organization.

1. There is no need to get approval for leaving an organization.
2. Once the intention to change/leave is confirmed, the user loses his current membership *immediately*, without waiting for the new organization owner (if applicable) to approve the joining request.

Hackathon Creation and Operation

Attributes and Creation

An admin can create a hackathon, which includes at least the following attributes. All are *required* unless explicitly specified otherwise.

- Event Name
- Starting and end date
- Description // text; at least 10 alphabetic characters.
- Registration Fee (per person)// in USD
- Judges // at least one.
- Minimal Team Size // inclusive
- Maximum Team Size // inclusive
- Sponsors (organizations) // ***Optional***.
- Sponsor discount // optional, up to 50%.

Registration

Hackers have to register as a team to participate in any hackathon.

1. A hacker should be able to browse all future or ongoing hackathons.
2. Registration can only take place before the intended hackathon's start date.
3. The hacker who initiates the registration is treated as the team lead.
4. A team lead must specify team members and other team attributes.
    a. Must conform to the range of required team size.
    b. Must provide a team name, unique across the hackathon.
    c. Each team member must be assigned with a role
        i. ProductManger
        ii. Engineer
        iii. FullStack
        iv. Designer
        v. Other
5. Registration confirmation is automatically sent to each team member, with detail information about the hackathon, and a URL link back to pay for each team member to pay the registration fee.
6. Again, a hacker cannot participate any hackathon where he serves as a judge.
7. The registration fee will apply sponsor's discount if the participant belongs to one of the sponsor organizations, base on the membership at the time of payment.
8. Actual payment UI can be just symbolic/fake.
9. Payment confirmation/invoice must be sent through email.
10. Team lead should get an email notice after all team members have paid.

Operation

An admin can *open* and *close* a hackathon for code submission, and can also *finalize* a closed hackathon. Evaluation by judges can take place after submission is closed and before an admin *finalizes* the hackathon.

1. Ideally, submission can open only after the starting date. For ease of testing, we relax the constraint by allowing opening of code submission even before the starting date, in which case, the hackathon's starting date is automatically changed to the opening date.
2. An admin can close the submission after it is open, even before the end date. Once a hackathon is closed ***and*** any grade has been given by any judge to any submission, the hackathon ***cannot*** be re-opened for submission again.
3. An admin can finalize a hackathon closed for submission ***only*** after all submissions have been assigned a grade.

Hackathon Code Submission

Any team member can submit code. For simplicity, the submission should at least accept URL (e.g., Github repo). You can simply keep all the submitted content (e.g., URLs) and present them to the

Hackathon Evaluation

Judges will evaluate and grade submissions on a team basis.

1. Evaluation cannot take place until the hackathon is closed by an admin.

2. The grades should be a float number between 0-10, inclusive.
3. One judge's grading for any team overrides previous grading for the same team, no matter it was by the same or a different judge; i.e., only the latest grading counts.
4. No evaluation or grading changes can take place after a hackathon is finalized.

Result Report and Notification

Grading results should be available for all hackathon that have been closed or finalized. For a given hackathon, the results should include scores, team names, and their participants, and ordered in descending order by scores. The rendering of the results should make the top three winning teams visually distinguishable from others. This should be made available to every user.

All participants and judges need to be notified through email about the results once a hackathon is finalized.

- The email needs to contain a link to the details results mentioned above.
- Email for winners must have congratulatory phrases in the subject and message body.

Financial report

Registration Fee Payment Report

A screen must be provided for an admin to view the current registration payment status of the any given hackathon event. This needs to include information like which teams have (or have not) paid, and which participant has (or has not) paid. For any payment, there needs to be a way to find out the amount and payment time.

Hackathon Earning Report

For a given hackathon, a report screen must be provided to provide the following content:

I. Total revenue from paid registration fee.
II. Total unpaid registration feed.
III. Total revenue from sponsors. You can assume it's $1000 from each sponsoring organization.
IV. Total expenses (if there is any. Should be zero, unless you do the bonus feature below).
V. Profit (I+III-IV)

Bonus Features

- Per hackathon expense reporting. And admin should be able to add expenses for a hackathon before it is finalized. An expense should be in the format of
    - Expense title. E.g., event venue rental, judge compensation.
    - Expense description. Text explanation of the expense.
    - Expense time. Must be date before or up to the time of the reporting.
    - Expense amount. Float number. Always in USD.

- Invite non-members to participate in hackathons. Whoever gets the invitation still needs to go through the registration process.

Please note this is the end of functional specification. Any requirements are about your term project team, and not hackathon participants or teams.

Grouping

This term project is group based, with group size up to four people. Once the project plan is submitted, group membership cannot be changed.

Source Code Management

You are recommended to use a Source Control Management (SCM) system to manage your team's source code. This can be a private Bitbucket repository or your local git. During the grading of the term project, you may be asked to provide commit history or any other document to help evaluate each team member's contribution.

Cheating Policy

Your app must be built by yourself, and cannot be based on the code base of any existing app. If you used any code not written by yourself, it must be clearly documented in your README.TXT file, unless it is part of publicly available libraries. *If your app is already used to serve the requirements of any other class, it will not be accepted by this class*. In the case any form of cheating is confirmed, you will get an F grade for this class.

Deliverables and Grading

The project is worth 25 points in total. The actual *due dates* of the deliverables will be specified in Canvas.

Team Formation and Project Plan (1 point)

One pager to cover team formation, technological choices. Refer to Canvas for details.

Project Presentation and Demo (5 points, Due 05/08)

To be presented in class.

- The presentation should cover introduction, high-level design, and major features with screenshots. Time limit: 3 minutes.
- You must also do a live demo. The guideline for how to do demos is to be announced. Time limit: 3 minutes.
- Grading will be based on successfulness of the demo, the content and clarity of the slides, and the delivery of the presentation

The presentation slides must be submitted through Canvas as a PDF file.

Project Report (5 points, Due 05/21)

The report needs to cover the following topics.

1. Motivation and introduction of your app

2. High level and component level design
3. Technology choices
4. Description of features with final screenshots
5. Testing plan executed and results
6. Lessons learned and possible future work

You are recommended not to exceed 20 pages, but you will not be penalized just because the report is too long or too short, as long as the level of coverage for the required topics is reasonable and clear. The report must be submitted through Canvas as a PDF file.

Project App (15 regular points + 1.5 bonus points, Due 05/21)

Note: the instruction for submission is still *subject to change*. Please refer to submission instruction in Canvas.

1. You must submit all your source code / resource files through Canvas
2. Features correctness, stability, performance, choice of technology and implementation are worth 13 points; **all** grading will be done through your UI - **no** partial grades will be given for any feature not exposed or untestable in UI.
3. User experience is worth 2 points
4. The bonus features are worth 1.5 points
5. You need to keep your app live for at least a week before we finish grading
6. README.TXT, including
    a. The names, email IDs, and students IDs of the members
    b. The URL to access your app
    c. Any other instruction necessary for the TA to grade the app
    d. Build instructions
    e. Description of bonus features you implemented, if any.