6.170 Project 3: Crowd-approved Comments

PLAZA, Eddie Francis (efcplaza)

nQoot: http://nqoot.francis.ph

Design Analysis

Key design challenges

- One of the first design challenges for this project is to define what a crowd-approved comment system is. There are a lot of variety in existence right now. Many of them vary by features or provide entirely different systems. For example, Quora's approach is different than that of reddit although the idea of crowd-approved comment system is similar. In this project, the challenge is which approach to take or whether to implement and entirely new one.

- Another design issue that comes up is the security, i.e. how do we ensure the integrity of the users that use the system? That is, if unverified users start saturating the system, it defeats the purpose of a knowledge based platform as this system requires the integrity of all questions and answers (or posts and comments). To identify how to verify the credibility of the users is another design challenge.

- To get more specific to the features, another design challenge is the answering and voting features. We need to answer the questions, should the users be able to answer their own questions? Should they be able to vote their own answers? How many times should they be able to vote on a specific answer?

Data representation

In my system, I have *Questions* and *Answers*. A Question can have zero or more Answers. Each Answer can have zero or more votes. For every vote that user makes on a specific answer, a new instance of a Vote object is created containing the answer_id and user's uid to record the event. That is, once user has "upvoted" it can only do a "downvote" (i.e. remove the vote). This design will ensure that answers do not get less than zero votes and users can only add at most one vote to a specific answer.

The representation of the object model in the database is pretty straightforward. Since our User is basically a user inherited from the use of Facebook authentication, its attributes are generally tied to the ones used in Facebook authentication. In addition, there is also a staff user which is determined through the boolean attribute *is_staff* in the User model. As of now, in order to make

a user staff, one can simply do a remote procedure call through a GET method using a specified URL.

Users

provider: string
: a string that represents the name of the provider, in this case facebook

name: string
: a string that represents the name of the user based off their Facebook account

oauth_token: string
: a string that represents the token used in FB oauth

oauth_expires_at: datetime
: a datetime representation of the expiration time of the oauth_token

uid: string
: a string representation of the UID of the user based off their Facebook account

is_staff: boolean
: a boolean to determine whether the user is a staff of the system

Questions

question: string
: a string that represents the content of the question, the question itself

description: text
: a string that represents the description of a given question

slug: string
: a string slug that is used for unique URL of the question page

anonymity: boolean
: a boolean representing whether the user who created the question is anonymous or not

user_id: string
: a string that represents the Facebook ID of the user that created the question

has_many answer

Answers

answer: text
: a text that represents the content of the answer, the answer itself

anonymity: boolean
: a boolean representing whether the user who created the answer is anonymous or not

vote: integer
an integer count of the number of votes received by the question
is_staff_endorsed: boolean
a boolean indicating whether an answer is endorsed by the staff of the system
question_id: integer
an integer id of the question where the answer belongs
user_id: string
a string that represents the Facebook ID of the user that created the answer

belongs_to question

Key design decisions

- The first issue I identified above was on addressing what kind of format does this system follow, i.e. will it be like Quora, reddit, or StackOverflow? In thinking through the problem, I decided that implementing a familiar yet simple system will be the best way to go both in technical and user-centered perspectives. That is, a simple system will make development and deployment very rapidly and a familiar system will provide users an easy and smooth experience using the system.

  This system implements a Quora-like knowledge based platform where users add questions and answer existing ones. Users can vote on existing answers to questions. The primary motivation of this approach is to have a simple knowledge base system where all posts are classified as questions and comments as answers to these questions. This makes it easy and simple to classify things both in the backend as well as in the user interface.

- Another issue that I identified is ensuring the credibility of the system. That is, a successful knowledge base system is dependent upon the credibility of its users. Only users that trust the integrity of other users will actually engage in exchanging questions and answers. Now, the question is how can we ensure the integrity of the users of the system?

  When it comes to security and ensuring the credibility of all registered users, it is hard to find a single foolproof verification method. Many different systems implement different approaches. One simple way that this system decided to take is to use Facebook accounts to login. While using Facebook accounts does not totally ensure the credibility of the logging user, it at least increases our confidence that the user is who they say they are given current Facebook's approach of ensuring the integrity of their registered users.

- Another question that came to mind was, for a question-and-answer type system, are users who created the question allowed to answer their own questions? Are they also allowed to vote their own answers?

  The system will allow users to answer their own questions. The major consideration for this decision is that, while they might not be specific answers to their questions, it can be follow up comments to existing answers. However, users may not vote for their own answers.

Design Critique

Summary assessment from user's perspective

The UI was designed to be simple and intuitive for the user. It uses a familiar interface, such as the dashboard and the menu bar. There are visual cues for endorsed answers as well. Overall, it works as expected and the user interface is well thought out.

Summary assessment from developer's perspective

I think the code base is well thought out. The functions that are specific to the model were written in the model as opposed to the controllers and assets were all clearly grouped together (i.e. stylesheets, javascript files). The naming of functions and variables were readable and concise.

Most and least successful decision

The most successful decision I think is the simple definition of a user, question, and an answer. For example, instead of defining a separate model for the user, the staff account is simply another user with an is_staff attribute set to true. The simplicity of the model made the whole design more straightforward.

The least successful decision would probably be not thinking of other features ahead. This led for other necessary features such as edits partially or not implemented up to this version. It would have been much easier to implement them if the idea of edits and deletes were thought out ahead of time.

Analysis of design faults in terms of design principles

A more concise and clear separation between the model and the controller would be better. For example, putting functions that are model specific to the model instead of letting the controller handle it.

Priorities for improvement

For improving the application in future, I think implementing other necessary features such as edits and deletes would be the top priority and representing them in the UI in an clear and intuitive way. The next priority would be the code base such as improving on a clearer separation between the models and controllers and adhering to the DRY principle.