Design Analysis
MIT Job Talk
Josh Duncavage Mari Miyachi Timmy Galvin
Created 11/14/2012
Last Modified 11/16/2012

Key Design Challenges

We have a few major design challenges. I think our largest one when analyzing our priorities is how to protect anonymity/identify users (MVP). A simpler one we face is how we are going to support users searching through inputted data (MVP). And a final one that is reminiscent of project 3 is how to design our upvote system (Final Product).

Data Representation

Interviews, offers, and questions will jointly be identified by a companies and users (need either to identify). Answers will be jointly identified by questions and users (need either to identify). This sharing will obviously raise the question in which way to store them.

Key Design Decisions

Anonymity

The question here is how to protect a user's identity. If a user chooses to post something anonymously, will that content still be tied to their account in the back-end with that relation just being covered upfront? Or would it be better to have an "anonymous" account that anonymous content is posted under? We will probably implement the first option. While it requires us to be significantly more careful about not leaking the user's identity when displaying data, it will allow the user to go back and edit/delete that content because they still "own" it.

Content Search

There are a lot of ways to implement a content search: regex, closest to, begins with, etc. We will most likely implement a begins with system because our user base will be MIT students who we assume have some technical/computer knowledge and also know what company they are looking for. A begins with search is also the simplest and most resource-light approach.

Voting System

The question here is how to treat the vote system including factors like the uniqueness of votes, how upvotes and downvotes interact, etc. The first option would be to have two vote columns for every comment in the user table to minimize the number of cross-table mappings. The second option would be having a table that just stored a vote as a user, comment, and positive or negative one value (or two columns). The first option doesn't seem to have many positives,

it would require dynamicity in the table and it would be extremely space inefficient. I chose the second option because it seemed simpler in its overall construction, though I had to do a little designing to ensure a user only had either an upvote or downvote for a single element.