6.170 Project 2 Phase 3 Design Document

Michael Wee – October 15, 2013

**New feature descriptions**

* Ability to allow and deny access of notes to certain users
* Users with access can read, edit, or delete notes
* AJAX update of a user’s feed for everyone who has access (through the users view) upon creation or edit of a note

**Data Model**

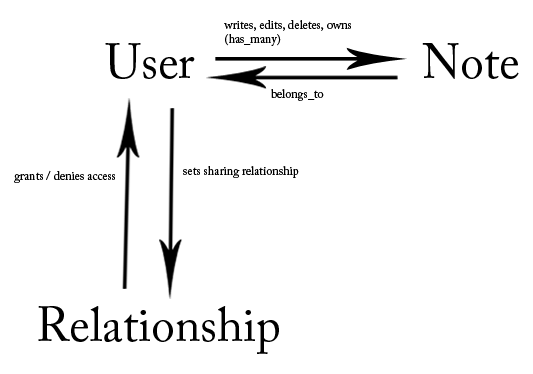
* I have shared\_id and a sharer\_id fields that create the relationship between the person granting access to the note (sharer) and the person with whom it is granted (shared). This is implemented through the relationships model and controller to toggle relationships.
* A naïve implementation for the data model would have a shared\_id and sharer\_id in a shared\_users table where we establish the relationship but also repeat many of the fields such as name and email. This is unnecessarily redundant because each row in the shared\_users table has data already in the users table.
* Instead, create a relationships table (model and controller) that relates shared\_id and sharer\_id through the relationship. To make a shared\_users array, I would pull out an array of shared\_id attributes and find each user. To make this procedure more convenient I use the Rails “has\_many through” relationship. So thus a user is sharing with many users through the relationships table sourced by the shared\_id.

Database table schema:

User.id 🡪 through 🡪 Relationships table (sharer\_id | shared\_id)

🡪 has\_many 🡪 user.shared\_users (lookup individual

users in the users table by id)



**Design Challenges**

* Allow and deny access to notes to users
  + Created a relationships permissions table in the data model with the has\_many through paradigm
  + Created a relationships model and controller to instantiate these
  + Added a share and unshared button on the users page to trigger the modification of these relationships
* Users with access can read, edit, or delete notes
  + Update partials, models, and controllers so that “show”, “edit”, and “delete” buttons appear on every note for every user who has sufficient permissions to view the note
* AJAX update of a user’s feed on the users view
  + One option is to instantiate and pass JSON objects around the backend containing information about the change or creation of notes
  + The option I implemented is to use jQuery’s getScript function to remotely call a script to update the user’s feed page for which the update is being pushed. This is simpler and changes fewer things in the code, necessitating just a snippet of code called periodically in application.js, a script to update the view, and an index method in the notes controller.
  + Works locally, but breaks on Heroku for unknown reasons