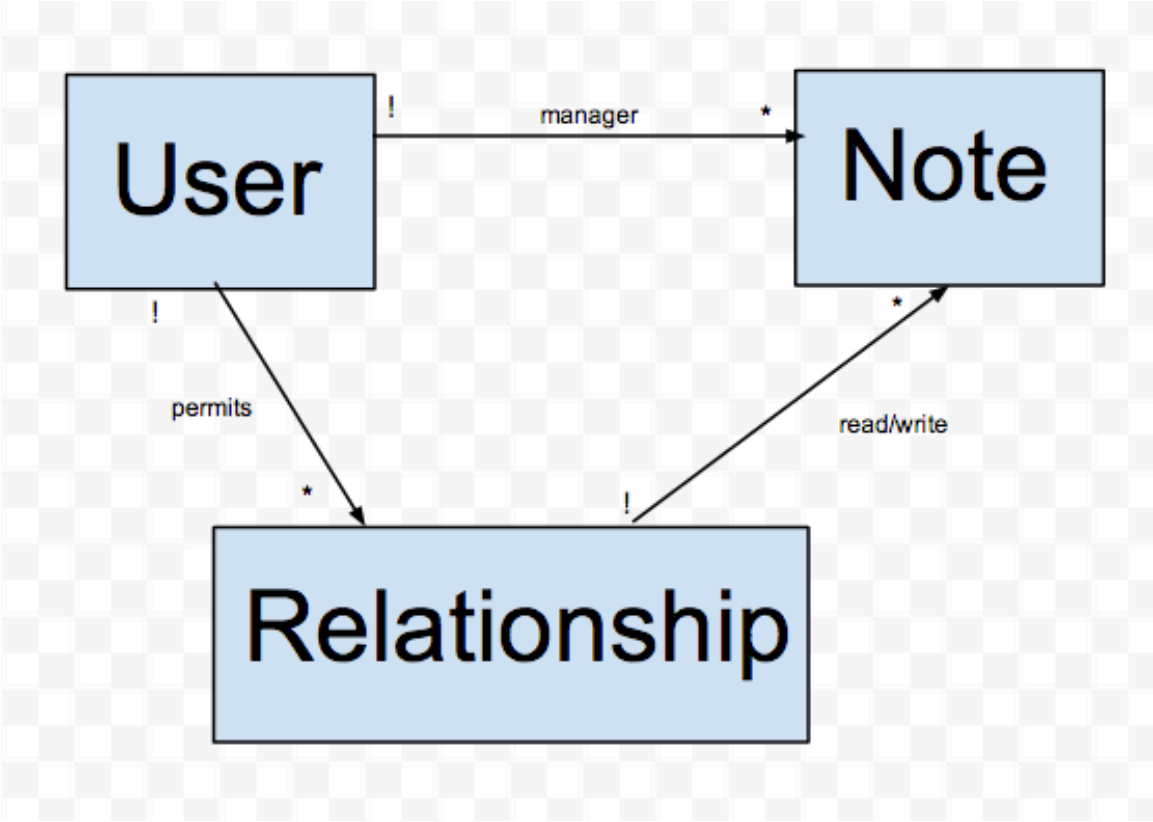
6.170 Project 2 Phase 3 Design Document
Michael Wee – October 15, 2013

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



Design Challenges

- Allow and deny access to notes to users – schema challenge
    - 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.

    - I chose the second option to avoid redundancy in the schema and have it have a more compact design
- Allow and deny access to notes to users – displaying the view conceptually
    - Added a share and unshared button on the users page to trigger the modification of these relationships
    - Could have also put a "share with" button on each individual user's page, but this would require extra clicks to reach a user's page, and I have not implemented individual user profiles yet, so I went with the first option.
- Users with access can read, edit, or delete notes – designing the view
    - Once the schema decisions were made, this portion only had one clear way to be implemented – restricting or allowing access by looking up permissions in the relationships table
- Users with access can read, edit, or delete notes – designing the view
    - 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
    - Another option is to render whole different pages for users with access and without, but re-renders more than you need to with the existence of Rails's partials feature, which I ended up using
- 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 and use AJAX to process these objects and update the page accordingly
    - 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. I chose this option because it 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