

Project 2: TeamNote

Design Analysis

Ido Efrati

6.170 Fall 2013

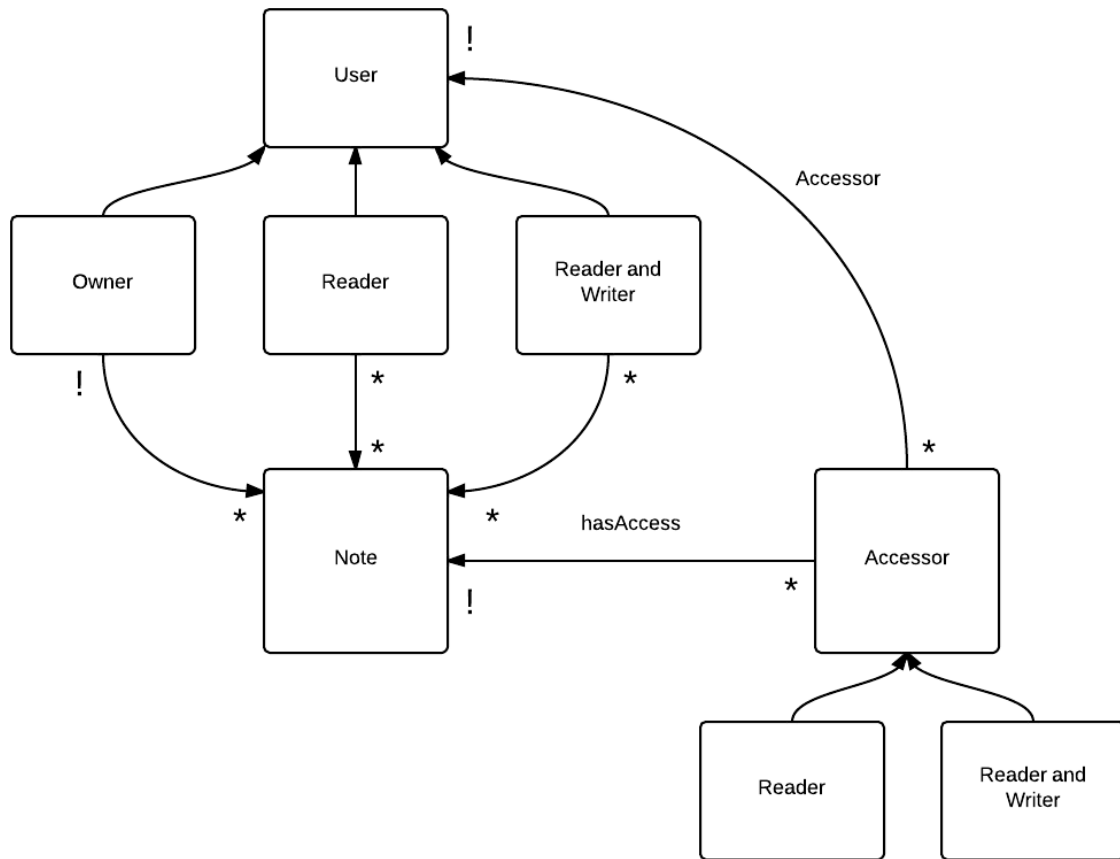
October 15, 2013

1. Overview:

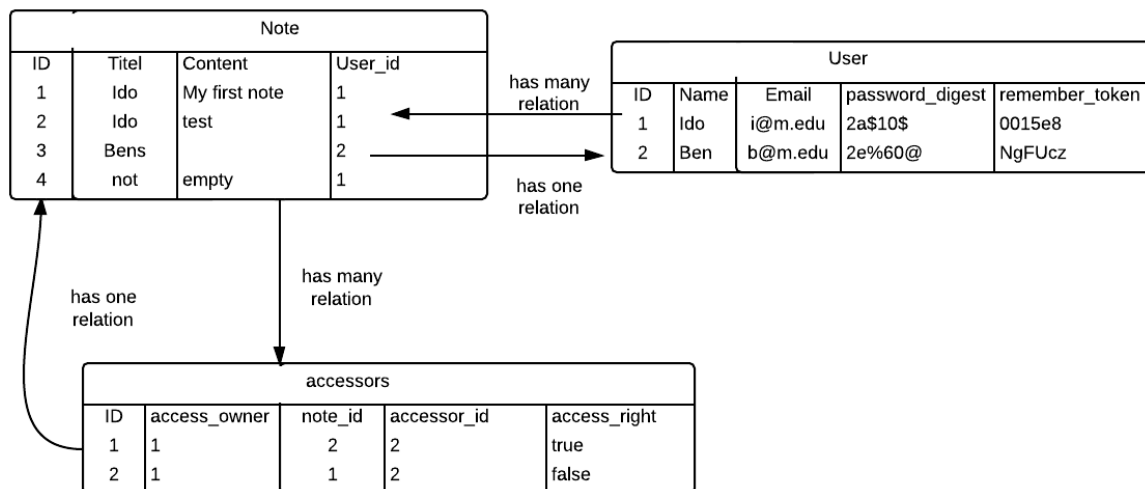
TeamNote phase 3 is a browser-based note-taking application that allows users to create and organize textual notes, save them and access them. Phase 3 introduces additional functionality to the project, permissions and Ajax support (render part of the page periodically). Users can grant *read only* and *read and write* permissions to other users. Meaning an owner of the note can now allow other users to access their notes and edit them (if the required permission was granted).

2. Concepts

2.1 Data Model



2.2 Database design and relations



As seen in the above diagram TeamNote consists of three tables, a note table, a user table, and an accessor table.

A user has a name, an email, and encrypted password to protect from attacks, and an encrypted session token to prevent from sessions attacks. Most of the validation is performed only at the model level (presence of required field, and

correct format), but uniqueness validation is performed both on the model level and on the database level. The reason behind the double uniqueness validation is to prevent a duplicated creation of an entry, if the user clicks the sign up button twice with a slow Internet connection.

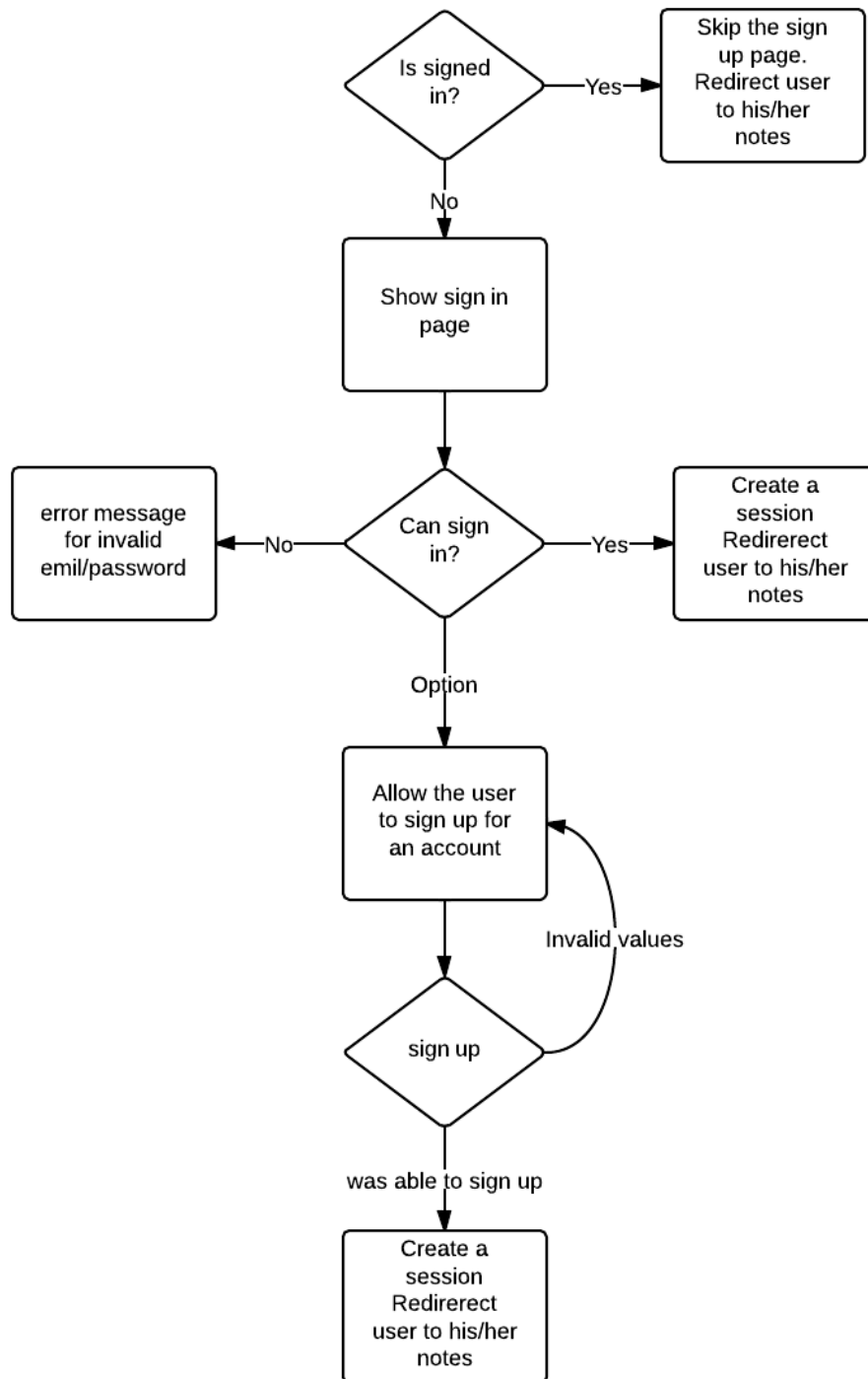
A note has a title, content and an owner (the user who created the note). A user can edit the title and the content of the note but cannot change an owner of a note. The owner field stored as an integer that corresponded to the primary key of a user in the user table. Finally, as demonstrated by the above diagram a user has a *has many* relation to notes, and a note has a *has one* relation to a user.

An *accessor* has an *access_owner* (the owner of the note who granted the access), a *note_id* stored as an integer that corresponded to the primary key of a note in the note table, an *accessor_id* stored as an integer that corresponded to the primary key of a user in the user table, and an *access_right* field (Boolean). A False *access_right* indicates read only permission, and a True *access_right* indicated read and write permission. I choose to use a Boolean for the above instead of two different columns (one for read and one for write) for two main reasons. First, simplify the query. I only need to check one variable. Second, to minimize the size table. It is important to note, that if a user does not have any access at all to another user's note, there will be not entry in the table. Finally, as demonstrated by the above diagram a note has a *has many* relation to *accessors*, and an *accessor* has a *has one* relation to a note.

2.2.1 dependencies

There are two dependencies in the above scheme. The first, between notes and a user, if a user was deleted from the database all of its notes will be deleted as well. The second dependency is between accessors and a note. If an owner deleted a note all of the note's permissions will be deleted as well

2.3 Users and session



In phase 2 users can create their own accounts, sign in to TeamNotes, and maintain a session. When a user first accesses the server, the server will check if the user is already signed in. If the user has a working session (stored in the user's table under remember_token), the user will be redirected to his/her notes.

Otherwise, the user will have to either sign in or sign up. If the user has an account he or she can use the sign in option:



Email

Password

Sign in

New user? [Sign up now!](#)

Otherwise, if the user wishes to create a new account he or she can use the sign up form:



Name

Email

Password

Confirmation

Create my account

Finally, when the user sign out of the system, the session will be cleared the remember_token value will be removed from the table.

2.4 Grant permission

An owner of a note can grant read only or read and write permissions to another user, by accessing the share option on a note he or she owns.



The following will lead the user to a grant permission form in which the user will be able to grant permission to a user. The form is set to prevent duplications in the *accessor* database. Meaning, before a creation of an entry in the *accessor* database, I check if such permission exists (i.e. if there exist a permission with the same *note_id* and *accessor_id*). If such permission exist I will update the *access_right* accordingly, else I will create a new entry.

👁 Grant Permission

Grant Permission To:

ido.p.efrati@gmail.com

Read only ☐

Read and write ☒

Grant Permission




2.4.1 dropdown vs. input field:

I decided to go with a dropdown vs. an input field for two reasons. The first reason is safety. A user will only be able to grant permission to existing users vs. trying to grant permission by free typing which may lead to human errors. The second is the scale of the application. I view TeamNote as a small application that is only being used by a small number of friends. If TeamNote were to be scaled to support hundreds of users I will have to reevaluate the grant permission page in one of two ways.

- 1) Replace the dropdown with an input field.
- 2) Introduce the concepts of friends or groups. In which TeamNote is similar to a social network, and users only share notes with their friends.

2.5 Permission control

Users can manage the permission they granted, as well as the permission granted to them by other users. They can view or delete them via edit permission screen.

Granted to:			
Note	Granted to	Permissions	
dffdfd	kellyz@mit.edu		Delete
Granted by:			
Note	Granted by	Permissions	
rerer	kellyz@mit.edu		Delete
note 2	kellyz@mit.edu		Delete

I decided to allow users to delete permissions that granted to them by another user for two main reasons. First, a user might find the content of the note irrelevant. For example, the user addressed the note already. Second, the content of the note may be offensive and the users may wish to remove it from their view. Finally, deleting permission will not affect the note in any way. Thus the note will persist after a permission deletion.


2.6 Notes

A note consists of three fields:

Title - the note title stored as a string

User- the name of the owner, the owner's value is being set during the creation of the note, and is based on the currently logged in user.


Content – the content of the note. A text area, since a string is limited in its length and a user might want to write a longer note.

 **New Note**

Title

Content


Create Note


 Back


3. Functionality and features


3.1 Features:


TeamNote currently supports the following features:

Create  – allows the user to create a new note as long as the user specifies the required fields (see section 3.2)

Share  – allows the user to share their note.

Edit  – allows the user to edit an existing note. Currently the user may change the name of the user who created the note. A user will not be able to save an edited note if required fields are left empty.

Delete  – allows the user to delete an existing note. Will prompt the user with a verification window.

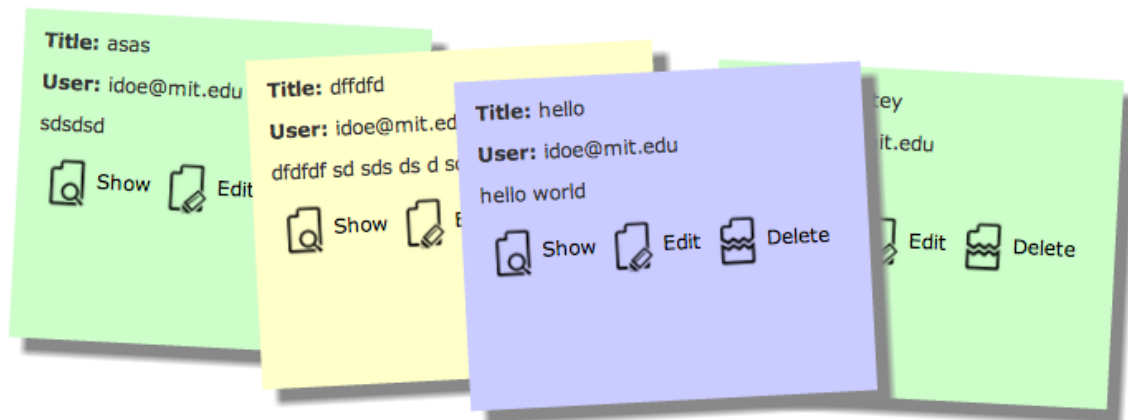
Back  - sends the user to the main page, where he or she can see all the existing notes

Sign up – users can create an account on TeamNote.

Sign in – users can sign in to TeamNote if they have an account.

Sign out – users can clear their session. After a sign out a user will be require a new sign in.

Dragging notes – notes can be dragged around. This feature was added in an attempt to learn how to integrating jquery/javascript into rails, and how to overcome issues with turbolinks.



4. Security

4.1 validations:

In order to create a note, a user must specify a title. If a user tries to create a new note without specifying the above fields, the application will prevent the user from creating the note. The reasoning behind this design decision is that every note has to have a creator, and a note cannot be empty (there is no point in adding empty entries to the table). From a better user interface perspective, the content field is optional. A user might want to create a short note that can be summarized only by its title (for example, “submit 6.170”). In this case the content will be redundant if the user does not wish to add anything else.

In order to create a new account a user must specify a username, email address, password and password confirmation. As mentioned earlier email address and username have to be unique. Any attempt to create a user without the above fields or with values that are not unique will prompt the user with an error message.

4.2 authentications

4.2.1 user authentication

When users tries to log in the server will try to authenticate their email address and password with the email address and the encrypted password that are stored in the database. Only if the two match the user will granted access and will be assigned a session.

4.2.1 session authentication

When the server verifies if a user is already signed in, it will compare the encrypted session that is stored in the user’s table under remember_token to an encrypted version of the local cookie. Only if the two match a user will be granted

access to his or her notes. By doing so the server guarantees that the cookie was not tempered with to allow unauthorized access.

4.2.3 access control:

Users cannot temper with the URL in an attempt to gain access to a note that they do not own. If a user tries to access a link before a session was establish he or she would not be able to access the site. If the user is already logged in and tries to change the URL path to access a different note (i.e. changing the id value in the URL) he or she will be redirected to a *page cannot be found*.

5. Testing

The following manual tests were performed on the application:

Positive tests:

Category: valid use of the application. Expected result: the user will be able to perform the require action.

- 1) Creation of a new note with values in all fields.
- 2) Creation of a note with values only in the title and user fields.
- 3) Creation of a note with the same title or with the same user.
- 4) Editing of a note to change its title/user/content.
- 5) Showing an existing note.
- 6) Deletion of an existing note.
- 7) Going back from every page to the main page.
- 8) Going back in the middle of an edit, verify that the content was not changed.

Category: valid use of the application. Expected result: the user will be able to sign in, sign up, and sign out.

- 1) Sign in with a valid email and password
- 2) Sign up with all required values
- 3) Viewing only notes that were created by a user
- 4) Accessing TeamNote after a session was establish
- 5) sign out and verify that the session was removed from the DB.

Category: valid use of permissions. Expected result: the user will be able to grant permissions, change permissions, delete permissions.

- 1) Grant a read only permission
- 2) Grant a read and write permission
- 3) Resubmit a form with a different permission; check that the value was updated.
- 4) Check permissions (granted to and granted by) in edit permissions

- 5) Delete permissions from granted to and granted by tables.

Category: AJAX testing. Expected result: the user will be able to see live updates of shared notes.

- 1) Grant a read only permission, check for a live update.
- 2) Grant a read and write permission, check for a live update.
- 3) Update permission, check for a live update.
- 4) Remove permission, check for a live update.
- 5) Delete a note with permission, check for a live update.

Negative tests:

Category: Invalid creation/editing of a note. Expected result: the user will not be able to create a note or save an edit of a note.

- 1) Creation of a note without a value in the title field.
- 2) Creation of a note without a value in the user field.
- 3) Creation of a note without values in both a title field and user fields.
- 4) Editing and saving of a note without a value in the title field.
- 5) Editing and saving of a note without a value in the user field.
- 6) Editing and saving of a note without values in the user and title fields.

Category: Invalid creation of a user, invalid sign in, and invalid access control. Expected result: the user will not be able to access TeamNote

- 1) Trying to sign in with invalid email/password.
- 2) Trying to sign up with an existing user.
- 3) Trying to sign up without specifying all required fields.
- 4) Open two windows in an attempt to access the system with two different users.
- 5) Changing URL values after a successful login , to gain access to notes that the user does not own.
- 6) Changing the URL values to bypass the sign in process.

Category: Invalid permissions. Expected result: the user will not be able to grant permission.

- 1) Change the URL of the permission page to gain access to a grant permission form.

5. Deprecated method

In phase 3 I decided to deprecate the *show* method. The main reasoning behind that move was that show did not give the user any additional information or any additional capabilities. Users can still view all of their notes at the main page, and

upon creation of a note they will be redirected to the main page; where they can view all of their notes including the new note they just created.

6. Attribution

Images are attributed to Vytautas Ajechnavicius from the Noun Project.

<http://thenounproject.com/vytautas.alechnavicius/>

Notes image - evernote.com/market/feature/3m?sku=MMMP00102