Project 2: Design Documentation

Features

Users need to create an account and log in before they are able to access the note application.

The application provides users the ability to Create, Edit, and Delete:

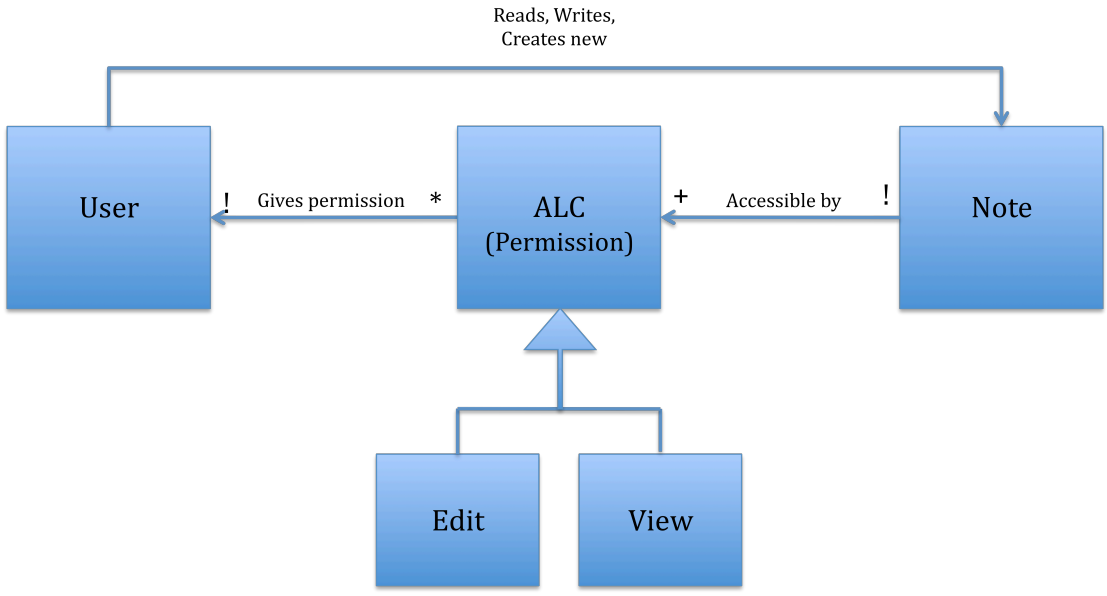
Create: A new note is created with an optional subject heading, and can be then saved.
Edit: The contents of existing notes can be edited, as well as their subject heading.
Delete: Existing notes can be permanently deleted.

Users can also grant permission to other users to edit or view created notes. Users that are allowed to edit a note are given this ability.

After log in, users are only allowed to view notes that they have created or have permissions for.

Data Model

There are two objects, User and Note. They have a many-many relationship, and a third table ALC supports this relationship. Each entry in the ALC has a user_id, note_id, and two Booleans that correspond to a user's read and write privileges to the respective note. The following diagram shows this relationship:



In this model, a user is given permission to a note by an ALC record. Each ALC record is either an 'edit' permission, or a 'view' permission. Each note is accessible by all the users that have permissions to that note. All the users who have an 'edit' permission, can read and write to that note. All the users who have view permission can only read the note. The creator of a note automatically has

an ‘edit’ permission. It is also worth noting that a user may not have created any notes, and as such has no permissions.

The permissions are differentiated by the Boolean fields for each ALC record. A record with both read and write access is an ‘edit’ permission, where as a record with merely read access is a ‘view’ permission. It is assumed that a record with write access automatically implies that it also has read access.

Design Challenges

The most important design challenge in this application was choosing how to represent the relationship between users and notes. I viewed users having many notes (notes that they can edit, notes that they can view), and notes belonging to many users (all the users that can view it, all the users that can edit it). In order to implement this relationship, a third table was introduced named ALC. Each record belongs to one user and one note, and through this record a user gains permissions to a specific note.

This behaviour could have been implemented by using the Rails has_and_belongs_to_many association, but this could potentially introduce some issues – how to differentiate between users that can only view notes and those who can read them. A solution could have been introducing two columns in the user table, where each column respectively contains a list of notes that can be viewed and a list that can be edited. However, I feel although this is valid in rails, does not agree with the spirit of relational databases.

Another challenge is to choose what features the permissions grant. Viewing on the surface implies that a user can only view the note, and editing on the other hand implies that a user can edit the contents of the note. However, a choice can also be made that allows users who have edit permissions to further grant permissions to other users.

A further design challenge concerns the data model chosen. In this model, users are all the same type, and their access to a note is determined by their permissions. However, it is easy to envision a design where perhaps users are split into more categories: authors, editors and viewers. These different categories would thus have different types of access to notes. For this application, I chose to have only one type of user in order to minimize the amount of models required.

Bugs

After creating an account, users still need to log in first before moving to the actual Note application page.