Design

Overview

Purpose & Goals

The purpose of this assignment is to create a web-based ‘sticky note’ system that allows a user to create and a view his/her personal collection of notes. These notes are kept in a centralized server, and the page on which the notes are displayed should make asynchronous calls when notes are added or updated.

The main goal is to understand how to implement client-side heavy applications in a clean way, and to understand how to efficiently use javascript. There are currently many implementations of web-based sticky note applications, with varying features and extensions. Because the use of the application is largely client-based, care must be taken to ensure that the user interface is extremely easy and convenient to use.

Context diagram

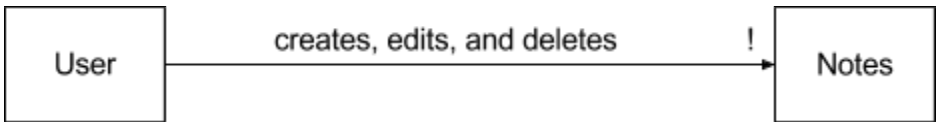


Figure 1. Context Diagram for note application

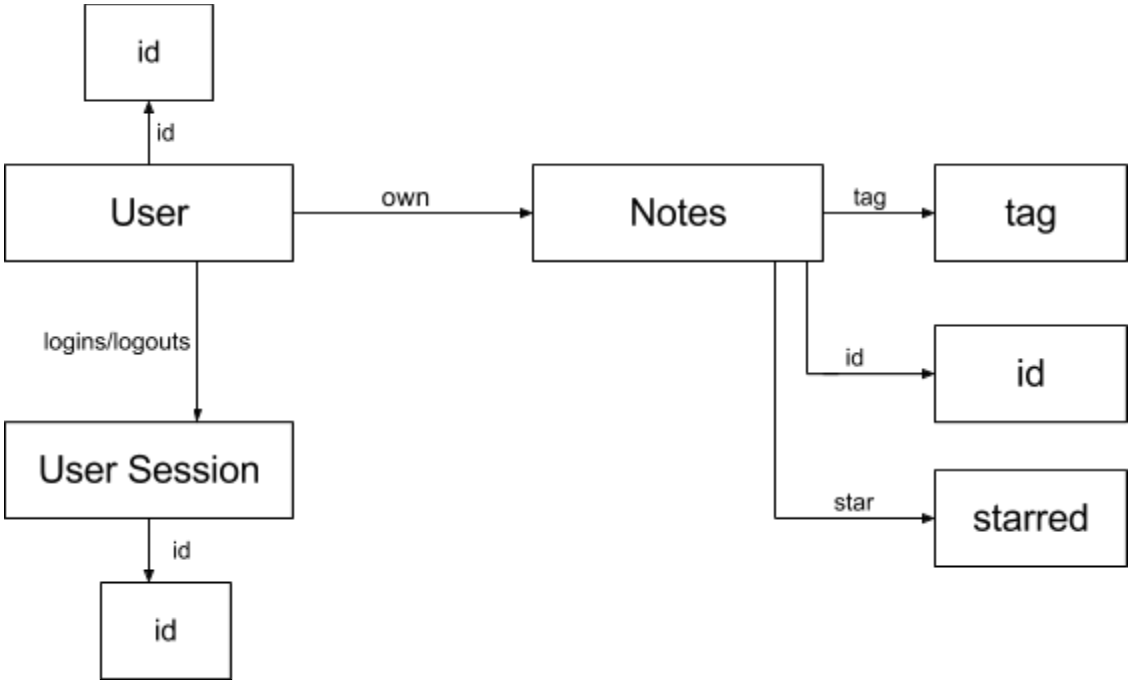
Figure 1 above shows a simple context diagram for the application. As seen, there are only two main components, a user and his/her notes. A user can create, edit and delete notes, and these notes will persist throughout.

Concepts

Key concepts

1. A *note* is a text string typed by a user.
2. A *user* is someone that uses the application to create and save notes.
3. A *tag* is a string used to specify categories (if any) that a note belongs in

Object model



Behavior

Feature descriptions

Minimal viable product:

1. Creating and maintaining notes

Allows users to create and edit notes, and having these notes persist on the server. The users should be able to view their notes after logging in, and be able to change these notes as well as delete them.

2. User accounts and sessions

Users of the application should be able to login to a personal account that remembers their notes; their notes should be private and should not be visible to other users.

Additional Features:

3. Custom prioritizing

All notes are ordered on the main page in order of date created, from earliest to latest. However, certain notes can be “starred” and these starred notes will be placed first.

4. Adding counter

Users are reminded of the number of remaining tasks they have left as motivation for task completion.

Security concerns

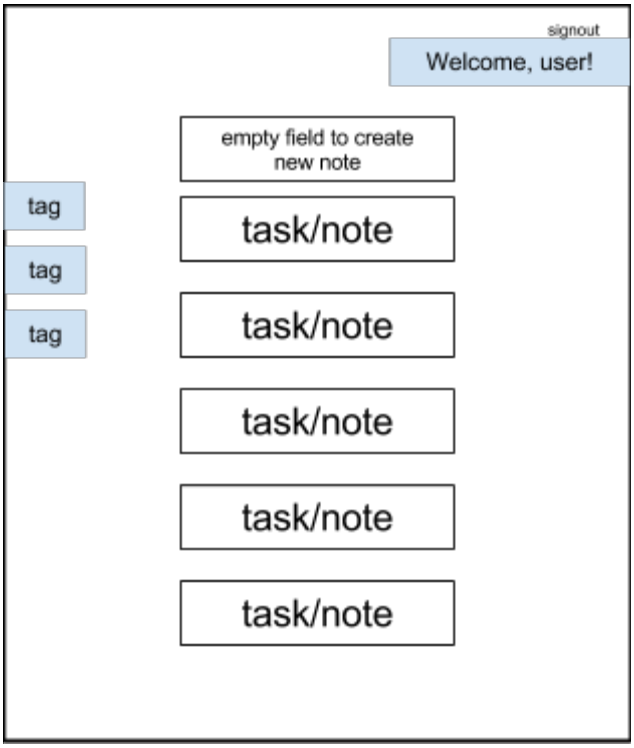
Security requirements:

1. Users should only have access to their own notes, controlled via the use of user sessions
2. When updating/creating notes, user id should be passed in the AJAX request to ensure that notes are updated under the correct user.

Possible attacks:

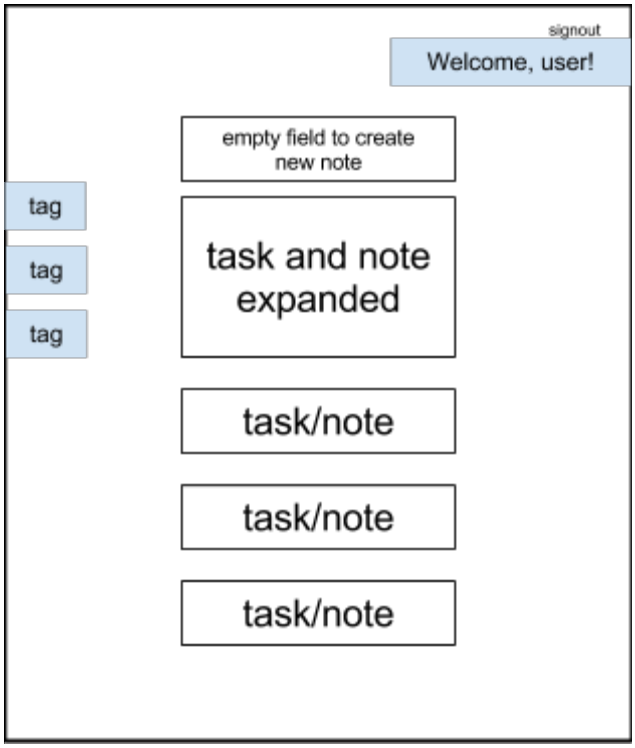
1. Use of AJAX means increased interactivity within a web application, which leads to an increase of XML, text and general HTML network traffic. This may lead to the exposure of back-end applications that were not vulnerable before, and if server isn't being well protected, it may also lead to unauthenticated users manipulating their privilege configurations.
2. XML HTTP requests function by using the same protocol (HTTP), which means that AJAX applications are vulnerable to the same hacking methods as normal applications.
3. Hackers may gain access to the hidden URLS needed for AJAX requests to be processed, which in turn will allow them to manipulate the requests being sent.
4. AJAX uses javascript to capture the user commands, which means that function calls are sent in plain visible text to the server. This easily reveals any database table fields, which can be manipulated.
5. Maliciously injected scripts (cross-site scripting) can use the AJAX-provided functionalities to trick the user and redirect the browsing session and/or monitoring the traffic.
6. Javascript can be used to map domestic/corporate networks, making any connected device on the network also vulnerable to attacks.

User Interface

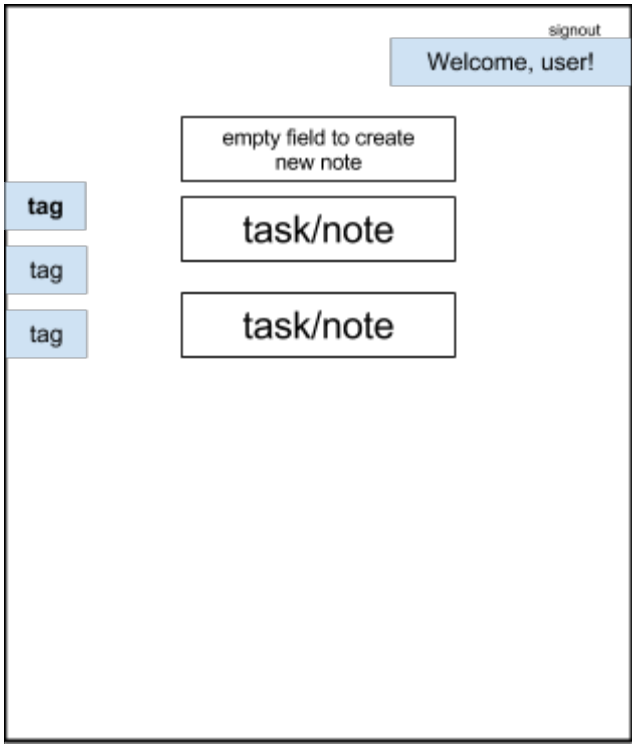


Main Screen

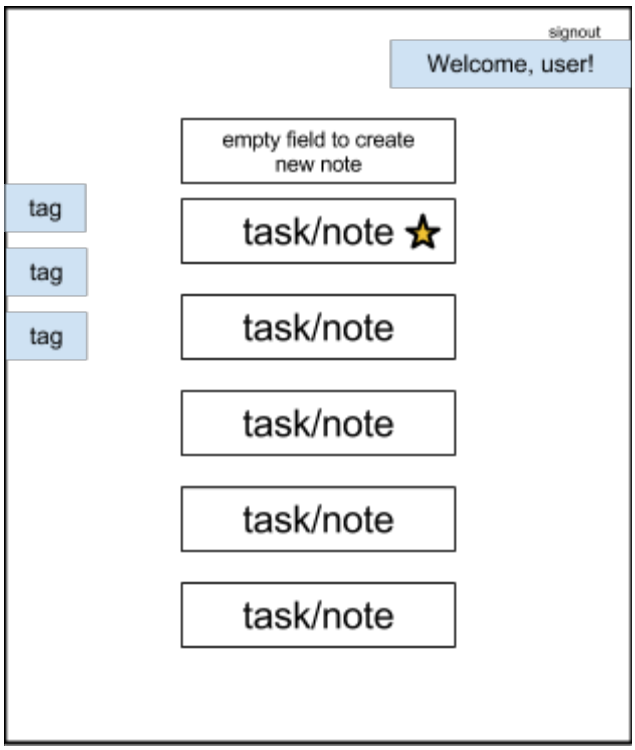
All notes are displayed, arranged in order created. Tags are displayed on the left, and the navigation bar is located top right. User logs in first to access notes.



Clicking on a task/note expands the box.



Clicking on a tag filters the notes via an AJAX call, and these filtered notes are then displayed.



Starring a note sends an AJAX request to the server that changes a database value.

Design challenges

Challenge 1: User authentication
The notes must persist on the server, and be viewable only to the owner of these notes. These means that in order to display any notes, a user must be authenticated.

This is implemented using the devise gem, which handles the user database as well as user sessions.

Challenge 2: Asynchronous calls
In order to ensure a smooth user experience, there should be no reloading of pages. All calls to the server must be asynchronous, done via an AJAX call and the use of partial templates in rails.

Most of the AJAX calls are done as recommended by rails. However, several costume methods (star and staroff) use JQuery ajax calls as this seemed to be the easiest way to implement a simple attribute update.

Challenge 3: Easy UI
As this is a note taking application, the user interface should be well designed, easy to use, and require minimal thinking on the user's part.

In order to provide an easy UI, the notes could directly be created from the index page, without additional button clicks. Notes can also be edited by double clicking text. The prioritizing functionality also allowed users to quickly identify important tasks with a simple click, and the rearranging of tasks to feature starred tasks first is convenient.
The application also displayed the number of unfinished tasks, which allowed the user to quickly access the amount of work that needs to be done.

Challenge 4: Starring
The user should be able to star and unstar a note, which may be hard to implement asynchronously. While there are multiple plugins that provide some sort of rating system, all of these were slightly too heavy for what was needed for the application.

Starring was finally implemented via simple javascript that switched between two images, and performed a corresponding AJAX call. The AJAX call was able to update the attribute through the passed note_id. The note body was then re-rendered so the starred notes will be displayed first.

Code Design

Main
Main page of the site, this displays the background and renders all relevant partial templates

User
id:integer (auto generated)
username:string (validate -> longer than 6 characters)
email:string (validate -> correct email address)
password:string (validate -> longer than 6, upper and lower case letters)

Session (Used for logging in and logging out)
session_id

Notes
id:integer (auto generated)
name:string
description:text (optional)
date_created:text
starred:boolean (default => false)
owner_id:integer (generated when note is created)
due_date:text (optional)
tag:string