6.170 Project 3.1 Design Document

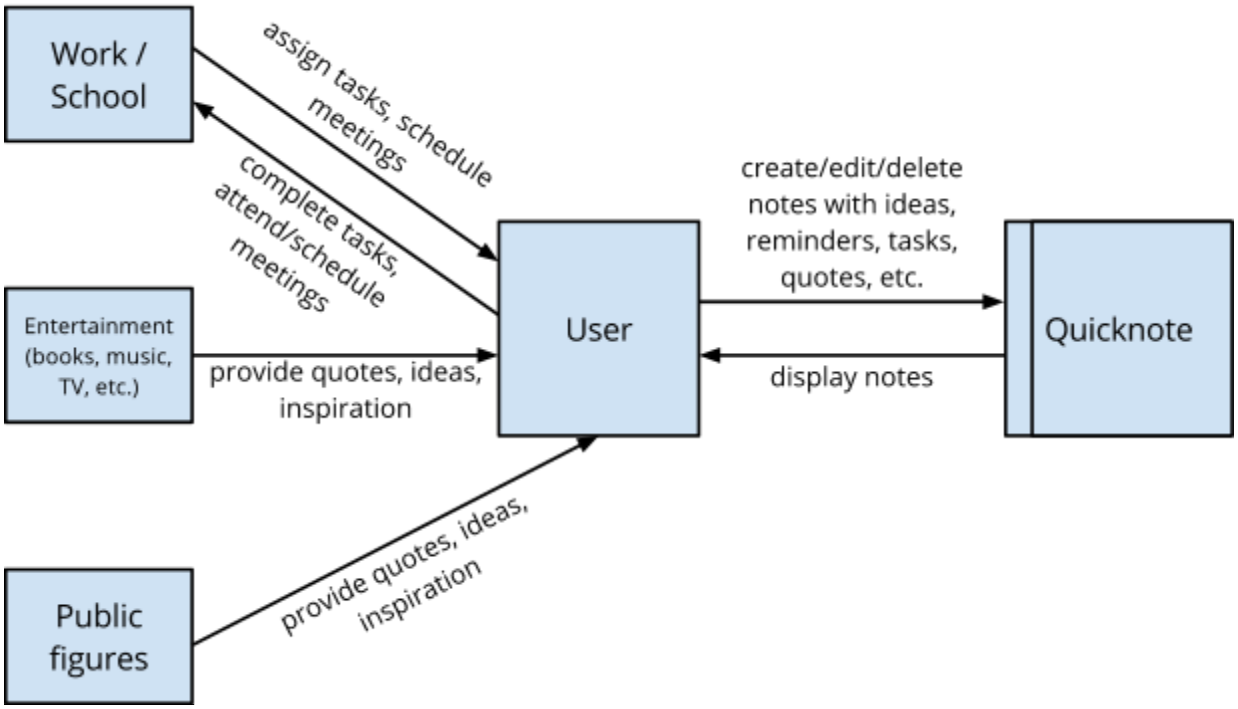
Carolyn Zhang (carolynz)

Overview

Purpose and goals

With this project, I would like to create a simple web-based interpretation of sticky notes, which I frequently use in my daily life. The appeal of physical sticky notes lies in their portability and modularity; as a result, they're the go-to solution for jotting down quick notes and reminders. Direct-to-digital translations of sticky notes, such as Mac OS X's Stickies app, are hindered by their efforts to replicate the physical product as much as possible. Moving notes around on a screen is cumbersome, and the implementations lack basic features people have come to expect from digital apps, such as search and tagging. This project, Quicknote, aims to provide a lightweight digital note-taking application that more closely resembles a basic text file than fuller-featured apps like Evernote. The minimum viable product only lets users view, create, edit, and delete fixed-position notes. In Phase 3, I hope to implement search, tagging, and note pinning.

Context diagram

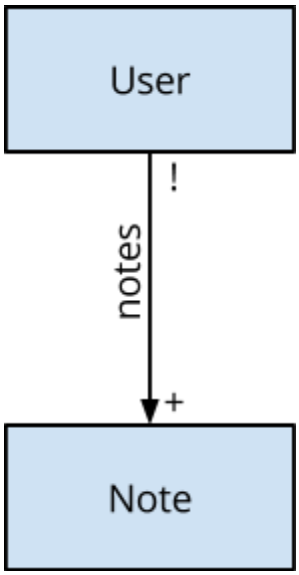


Concepts

Key concepts

- **User**: anyone with a Quicknote account. Users can only view and modify their own notes.
- **Note**: any bit of text the user wishes to record.

Object model



Behavior

Feature descriptions

All of these actions can only be done when a user is logged in.

- **Create a new note**: Users can create a new note by entering text into the text entry field at the top of the screen. Users cannot create blank notes.
- **Edit notes**: Users can click on a note to edit its contents. If users delete the contents of a note, the note will simply appear blank.
- **Delete notes**: Users can delete notes by clicking the “X” that appears in the upper right-hand corner of the note on mouse hover.
- **View all notes**: Notes are listed in order of creation date (newest to oldest).

Security concerns

Users must be able to view and edit only the notes associated with their own account. People are only able to access the main part of the website after logging in; anyone not

logged in is redirected to the login splash page.

Password exposure

Password fields (rather than text fields) are used on login and signup forms to minimize the chance that users expose their password to others. On the databases, passwords are not stored in plaintext but rather in its hashed form, so that accidental database exposure does not put user passwords at risk.

Authorization & Fake HTTP Requests

The app will use the Cancan gem for authorization. Since any requests associated with viewing, creating, editing, and deleting notes will be asynchronous, the application will verify two things before fulfilling the request:

1. that the request was indeed asynchronous (`request.xhr?`)
2. that the user ID stored in the session's cookies matches the user ID in the request URL (through Cancan's `load_and_authorize_resource` call in controllers

This prevents users of malicious intent from attempting to tamper with other users' notes through methods like visiting the URL "http://myapp.com/deletenote?user=3&note=all". Rails' built-in cross-site request forgery protection will also prevent attackers from sending fake asynchronous requests from other websites.

SQL & Javascript Injection

Attackers may attempt to include SQL and Javascript code in any text field on the website. To prevent these injection attacks, the app will sanitize all inputs before processing them.

Packet sniffing

Attackers may attempt to collect users' login details or cookies via packet sniffing. To prevent this, the app will use SSL for requests (HTTPS).

User Interface

Please refer to the wireframes at the end of the document.

Challenges

Design challenges

1) How should the notes be visually represented?

Option 1: The notes look like traditional rectangular Post-it notes.

Using this metaphor would help users understand the intention of the app (i.e., "use this app as you would use sticky notes"). However, depending on the note size, this may not be

the most efficient way for users to view their notes. Users typing in long paragraphs may also have difficulty reading over their words if the notes are too narrow.

> ***Option 1A: They are resizable to any size and dimension.***
> This would give users greater flexibility and control over the amount of content they want to put in each note, but it may be more difficult to display the notes in an easy-to-scan layout if the notes are all of different sizes.
>
> ***Option 1B: They are of fixed size.***
> This option can help ensure that notes can stay organized with less effort on the part of the user. A series of fixed-size post-its also introduces the possibility of a grid layout, which allows for efficient presentation and reading. However, users may want to write notes longer than what the note dimensions allow.
>
> ***Option 1C: Users have a few fixed-size options to choose from (e.g., 200x200 pixels, 400x200 pixels, and 400x400 pixels).***
> Giving users several options that work well together in a grid can be a solution to the content space vs. reading efficiency issue.

Option 2: The notes look more like strips of paper (i.e., lines of text in a text editor).
This option can be space-efficient and scannable, and works best when the user types long sentences and few line breaks. However, should a user attempt to type many short lines, this presentation may be less efficient than the Post-it rectangle solution.

Decision
Ultimately, I chose Option 2 because it has the best readability and space efficiency and is most likely to *guarantee* a good user experience because it is a more familiar pattern. Option 1C is also very attractive, but because it is not a commonly-used pattern, ironing out the kinks (e.g., dealing with resizing during editing notes) to create a satisfactory user experience may take longer than my time budget allows.

2) Should notes be rearrangeable in the default view?

Option 1: Yes. The notes will have an associated rank that indicates what order to display the notes in. A note's rank will default value based on its creation date, and can be changed with drag-and-drop.
This would mirror the stick-anywhere quality of physical Post-its. However, if a user drags an old note *N* to the top (intending it to be permanently displayed at the top of the note list), new notes would still appear at the top of the list, pushing down note *N*. Additionally, moving a note from the bottom of the list to the top, or vice-versa, would be an

unpleasant experience in long lists of, say, 300 notes.

Option 2: No. The default view for notes will be by creation date (newest to oldest).
Though this interface does not require any user effort to maintain or organize, it will be difficult to keep important notes persistently visible. Browsing notes may also be difficult when a user has many notes. Furthermore, this may be too inflexible for users, depending on their workflow and personal preferences

Option 3: Yes and no. Certain notes can be pinned to the top of the list, but unpinned notes will be displayed by creation date (newest to oldest).
This allows users some degrees of freedom in customizing their note display and echoes the "moveability" of real Post-its. When pinning or unpinning an old note in a long list, users may be confused by the "disappearance" of the note from the view. Power-user types may also be disappointed with the lack of options that pinning provides.

Decision
For my minimum viable product, I was forced to choose Option 2 due to time constraints. However, I anticipate implementing Option 3 for Phase 3, because it provides user-specified ordering features while still maintaining a simple and streamlined user experience. Adding "pinning tiers" or more user options would most likely create a confusing or cluttered experience that provides too many unused features.

3) Should notes have content length limits?

Option 1: Yes. A note should be limited to, say, 1000 characters.
This approach would make notes easier to browse, but it may be restricting for users who want to paste in longer snippets of text. However, users may work around this by splitting up notes into two segments - in which case, notes would be just as inconvenient to browse as if there were no content length restrictions.

Option 2: No.
This provides more flexibility for users. If they are already opting to use a service that doesn't let them rearrange notes, why restrict them even further?

Decision
I chose Option 2. The only advantage to Option 1 is that notes may be more browseable, but persistent users will simply split the contents up into two notes anyway.