Revised Design - CoCompose

6.170 Final Project

Stuart Finney, Lisandro Jimenez Leon, Kimberly Leon, Jessica Li

Overview

Description:

Our system is a real time collaborative music document editor, with a different notation and input mechanism when compared to traditional sheet music: piano keys will be displayed at the top of the document, and users can input their notes onto the area below. Multiple users can work on the same music document at the same time, in real time.

Key Purposes:

Allow real time collaboration on music documents:

- *Summary*: Users should be able to edit the same document at the same time and see live updates of other users' actions.
- *Explanation*: We aim to implement a purpose similar to what Google Docs does for real time collaborative document editing, but for music composition. This will allow two or more users to work on the same piece of music at the same time, while being able to see the music changes made by other users in real time.
- *Deficiencies of current solutions*: Current audio workstations are mostly native apps that don't have a collaboration feature. Only one user can work on a music document at a time

Allow ease of sharing music projects:

- *Summary*: Users should be able to send music documents to other users for their perusal with minimal hassle.
- *Explanation*: A user should be able to get his or her friend to look at what they have been working on in an intuitive way. Additionally, having one centralized copy of the music project is superior than having different versions lying around on your hard drive. This ensures everyone is looking at the latest copy of the music.
- *Deficiencies of current solutions*: With today's audio workstations, you have to email back and forth the project file, or have it on dropbox or another shared storage provider

Allow easy notation of music:

- *Summary*: The user can specify the pitch, timing, and duration of musical notation when writing music with an intuitive interface, different from traditional sheet music notation.
- *Explanation*: Our input method editor will consist of piano keys being displayed at the top of the document, and vertical rows corresponding to these keys. The user can then click on these rows to add a new note to the document. The vertical bars act as a timeline, with time increasing downwards (think DDR for piano).
- *Deficiencies of other solutions*: Certain music notation apps (such as Finale, Sibelius, or Flat) have their primary input mode in sheet music notation. While this is certainly the preferred input method of a traditionally trained composer, it is not very beginner-friendly.

Our interface aims to be more intuitive for anyone who has banged on a piano, without the requirement of knowing how to read sheet music (similar to why tabs are more popular with guitar players than sheet music)

Allow listening of composition:

- *Summary*: A user can listen to the music he/she has written.
- *Explanation:* It is important to be able to hear what you are creating, so a user should be able to play back their composition. Additionally, it is important to be able to hear what a collaborator has written.
- *Deficiencies of other solutions*: None (all solutions implement this to some extent)

Allow view of sheet music:

- *Summary:* A user can view the sheet music for the composition he or she has written.
- *Explanation:* A user should have the option of viewing their composition in sheet music notation if they prefer. Since most performers are more likely to know how to read sheet music than our unique notation, it is important to have this purpose for performers.
- *Deficiencies of other solutions:* If you aren't using a program with sheet music notation as its primary input method, then it is very difficult to convert your composition to sheet music.

Design Essence

Concepts:

Concept 1

Name: Note

Purpose: To allow the user to specify the pitch, timing, and duration of musical notation when writing music.

Operational Principle: When the user clicks on a column, a note of a default length is recorded at the specified point in the music with the corresponding pitch for that column. The user can then modify that note by extending or shortening its length, dragging it to another column to change its pitch, or moving it to another point in time of the music.

Concept 2

Name: Sheet

Purpose: To organize notes in a meaningful way.

Operational Principle: When a user creates a note, it is added to the current sheet. When a user clicks the "Play" button, the notes are played in the order specified in the sheet.

Concept 3

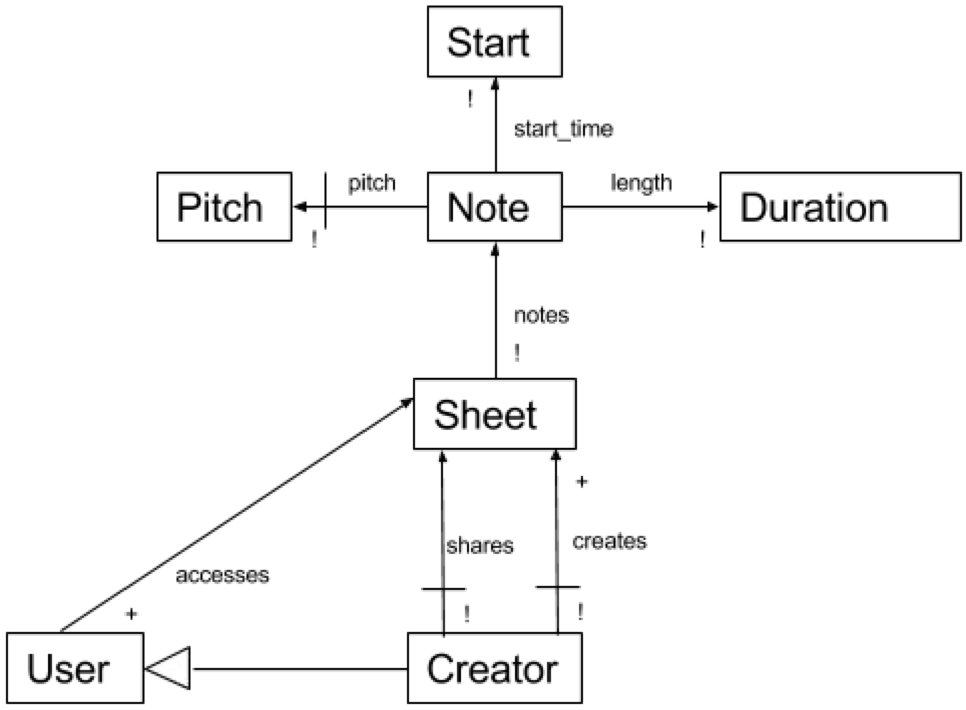
Name: Locking

Purpose: To allow collaborative editing while avoiding conflicts in simultaneous editing.

Operational Principle: When a user clicks on a note, it becomes locked so that other users can't click on or drag it.

Concept 4
Name: Collaborator
Purpose: To allow collaborative editing.
Operational Principle: When the owner of a sheet shares it with other users, everyone who received the sheet can view/edit.

Data Model:



Textual constraints:
- A user can only edit a sheet if it has been shared with him/her.
- A user cannot share a sheet with himself/herself.

Explanations:
- Accesses means that a user can create, edit, and delete notes.

Insights:
- To maintain permissions for a sheet, every sheet must have a creator, the only user who is allowed to share the sheet with other users. This is to mitigate the risk of sheets being edited by users who should not have been given access to the sheet.
- Two of a note's three properties (its start and duration) are mutable in order to allow ease of editing for the user. For example, once a user creates a note on the sheet, he/she can extend the duration of the note by dragging either end of the note away from the center of the note. If the user drags the top end of a note, this both changes the duration of the note and changes its start time. Ideally, users would also be able to drag a note to a different pitch/time. We may choose to add this feature if there is time to do so before the final demo.

- A note is unique to a sheet, i.e. different sheets have different note instances, even if the notes have the same pitch, start, and duration

Security Concerns:

Summary of key security requirements and how they will be addressed:

- Usernames are unique, preventing multiple people from accessing the same owned/shared sheets lists.
- Passwords required for users to log in, preventing people from logging into another user's account.
- Owners of a note have read, write, delete access to their own sheets
- Only the sheet owner can control who has access to the sheet, decreasing the number of users who can alter/attack the sheet.
- Other than the owner, only shared users can view or edit sheets.

How standard web attacks are mitigated:

- XSS:
    - HTML-escape values using the syntax "<%=" for EJS and "{{" for Handlebars
    - Use helmet -- https://github.com/helmetjs/helmet
- CSRF:
    - Use csurf module -- http://maximilianschmitt.me/posts/tutorial-csrf-express-4/
- SQL Injection:
    - Sanitize inputs before processing/saving/rendering with validator.js -- https://github.com/chriso/validator.js
- Brute-force attack against login:
    - Use npm package ratelimiter -- https://www.npmjs.com/package/ratelimiter
- Evil Regexes:
    - Use a node.js tool safe-regex -- https://www.npmjs.com/package/safe-regex

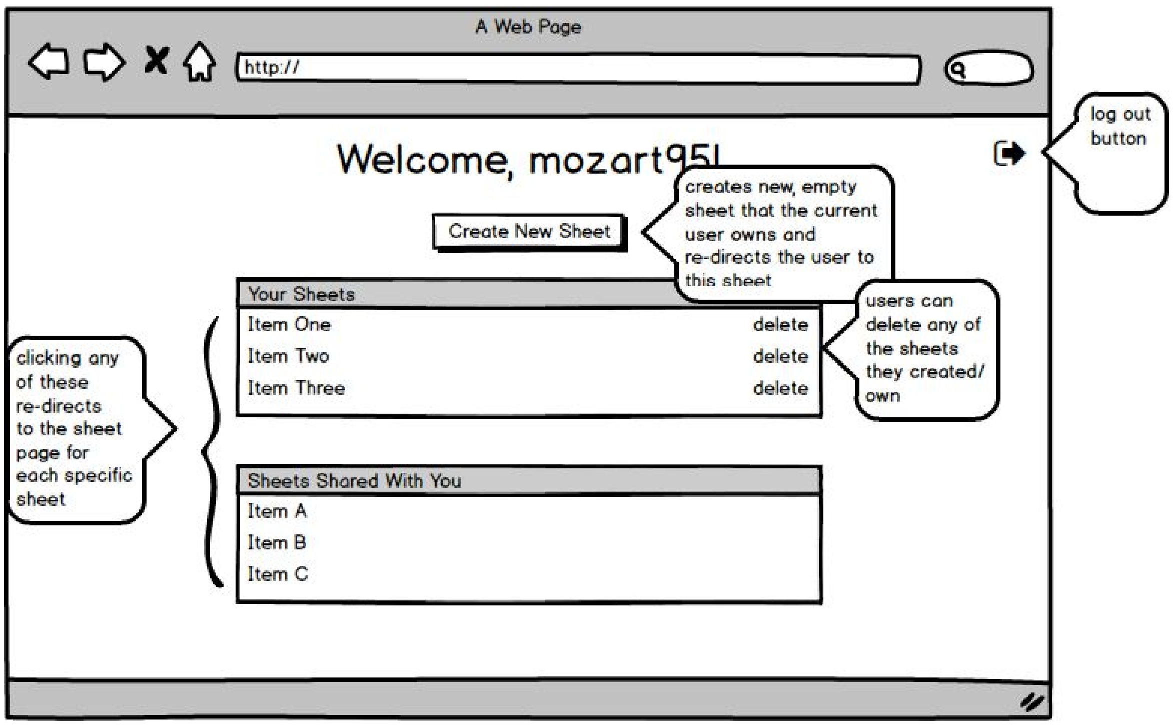
Threat model:

- Unauthenticated user can construct requests (curl, Postman) to eavesdrop/tamper with other users' sheets.
- User with basic credentials can construct requests or make requests via forms.
- We are only storing basic sheet music on our app, so the likelihood of attracting interest from highly sophisticated attackers isn't high.

User Interface:

Wireframes:

- User Homepage



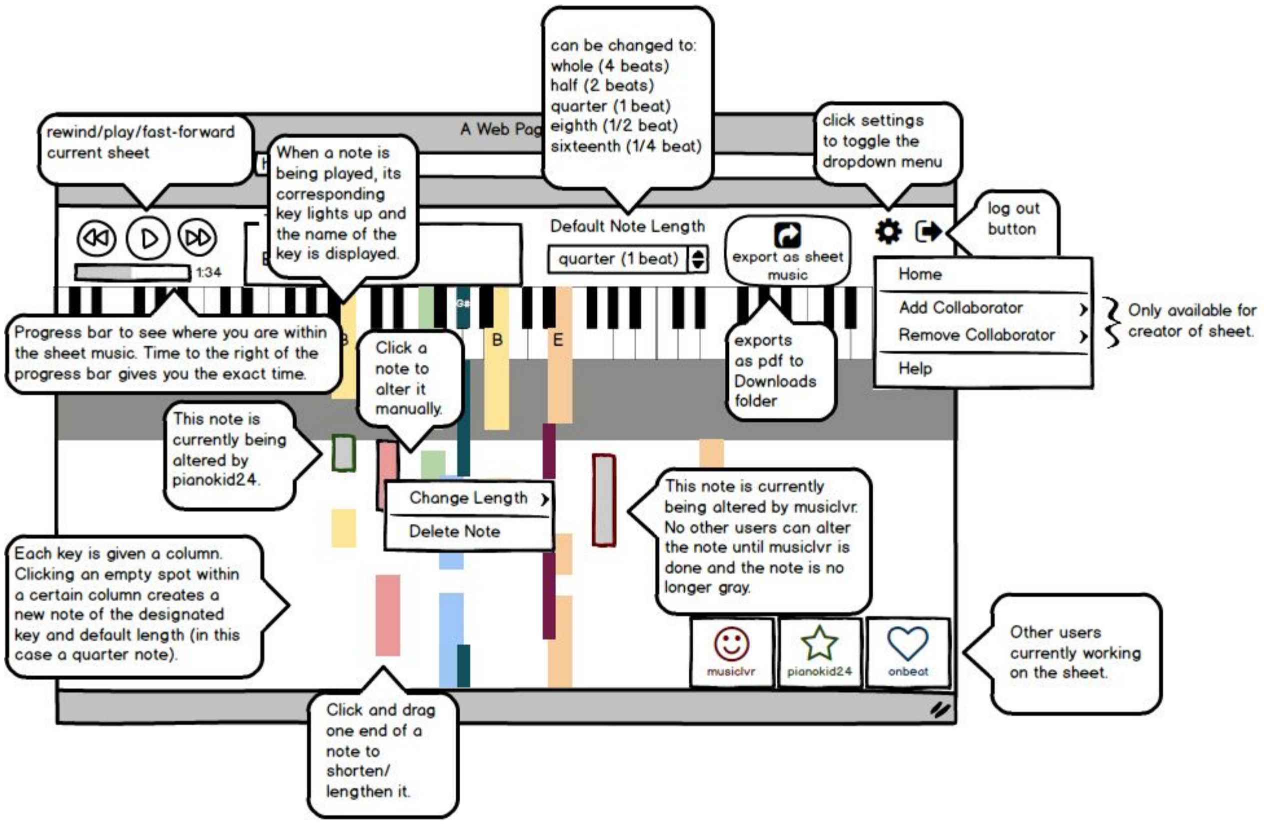
log out button

creates new, empty sheet that the current user owns and re-directs the user to this sheet

users can delete any of the sheets they created/ own

clicking any of these re-directs to the sheet page for each specific sheet

- Sheet Page



can be changed to: whole (4 beats) half (2 beats) quarter (1 beat) eighth (1/2 beat) sixteenth (1/4 beat)

rewind/play/fast-forward current sheet

When a note is being played, its corresponding key lights up and the name of the key is displayed.

click settings to toggle the dropdown menu

log out button

Only available for creator of sheet.

Progress bar to see where you are within the sheet music. Time to the right of the progress bar gives you the exact time.

Click a note to alter it manually.

exports as pdf to Downloads folder

This note is currently being altered by pianokid24.

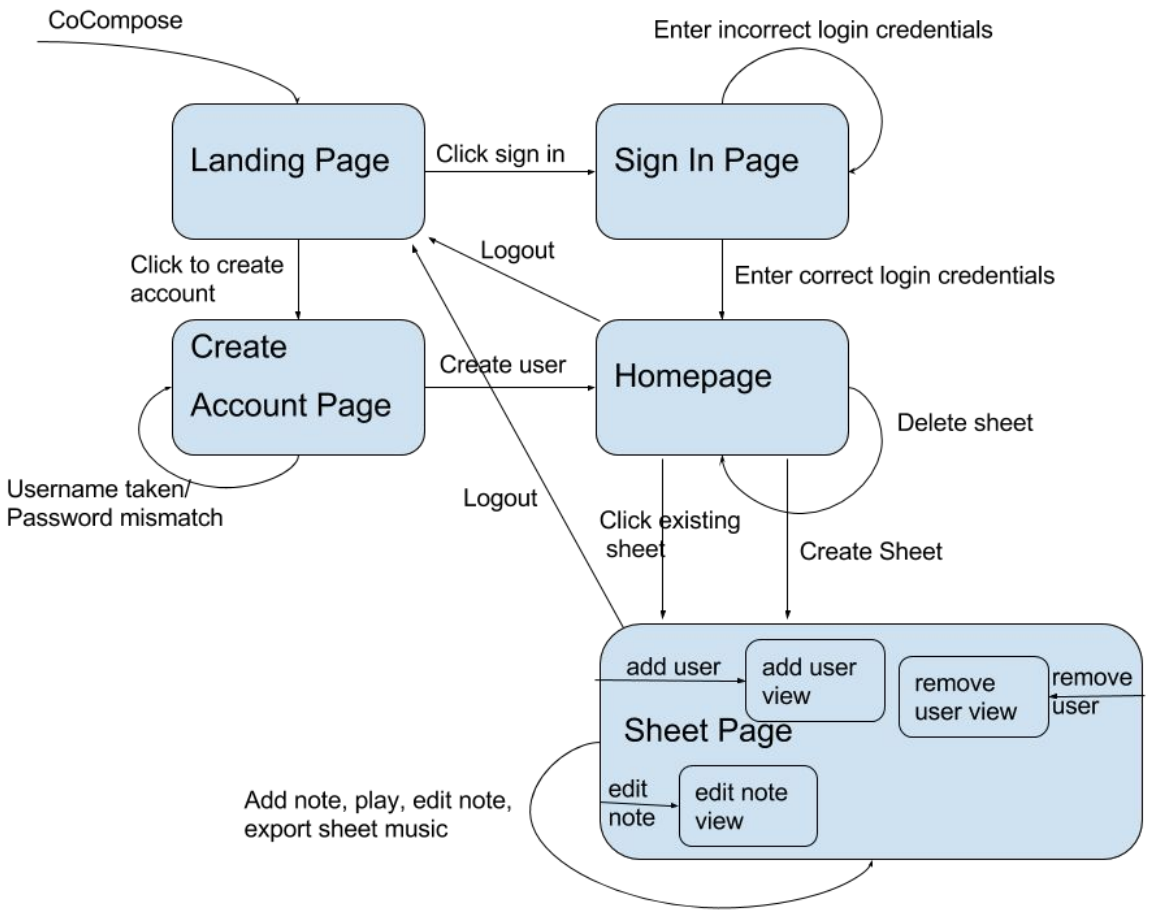
This note is currently being altered by musiclvr. No other users can alter the note until musiclvr is done and the note is no longer gray.

Each key is given a column. Clicking an empty spot within a certain column creates a new note of the designated key and default length (in this case a quarter note).

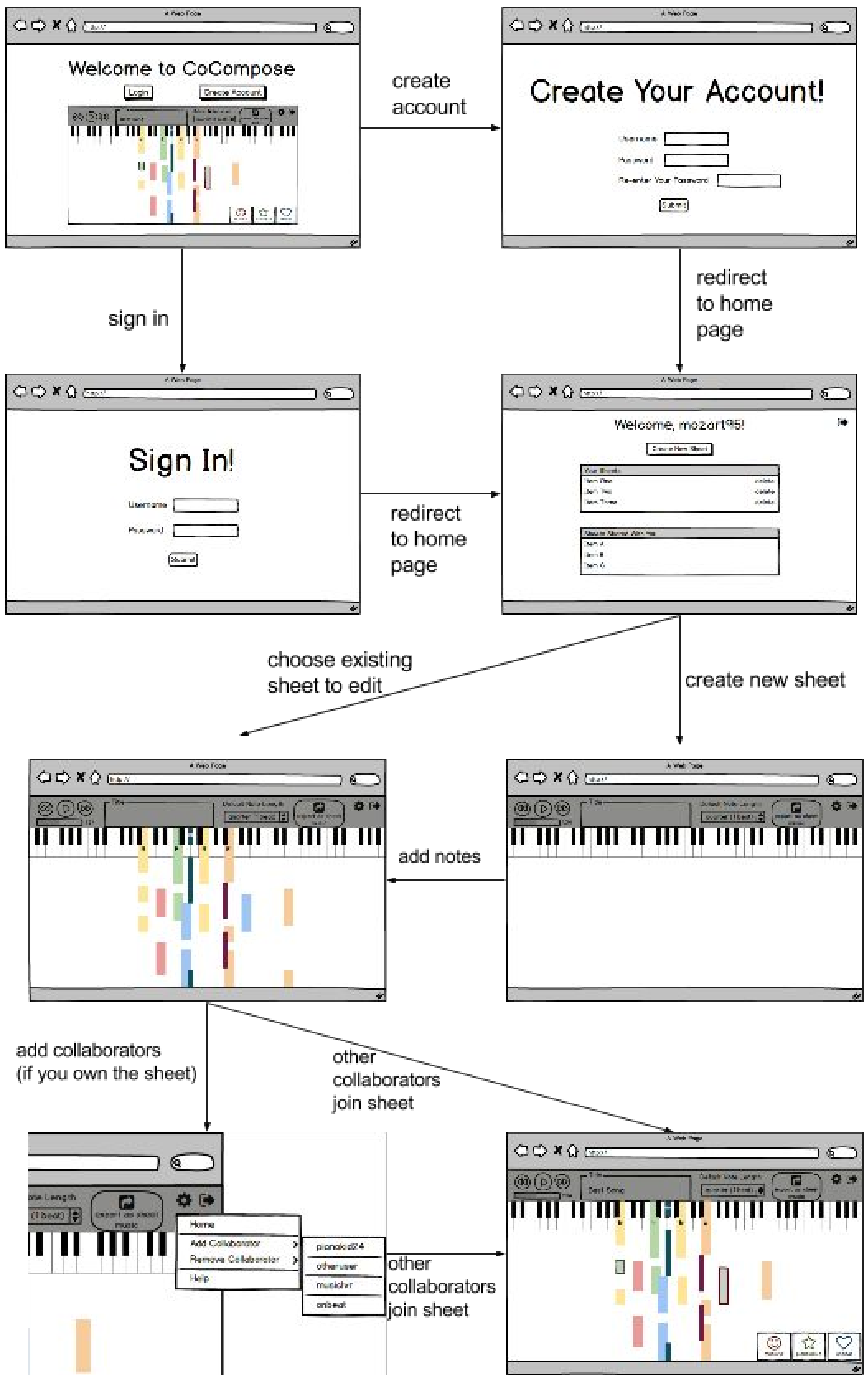
Other users currently working on the sheet.

Click and drag one end of a note to shorten/ lengthen it.

Wireframe Flow Diagram:

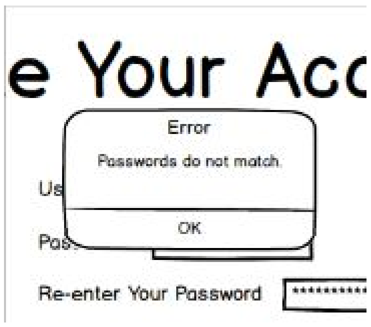

CoCompose
Landing Page
Click sign in
Sign In Page
Enter incorrect login credentials
Click to create account
Logout
Enter correct login credentials
Create Account Page
Create user
Homepage
Delete sheet
Username taken/ Password mismatch
Logout
Click existing sheet
Create Sheet
add user
add user view
remove user view
remove user
Sheet Page
Add note, play, edit note, export sheet music
edit note
edit note view

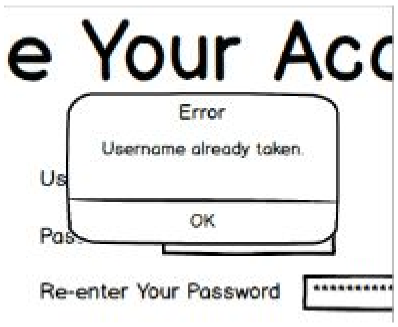
Transition diagram:



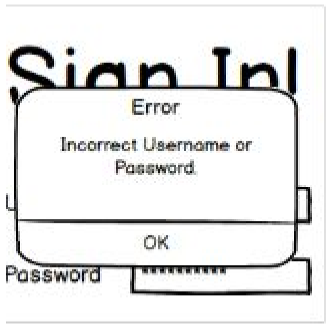
Error handling:

Create Account Errors

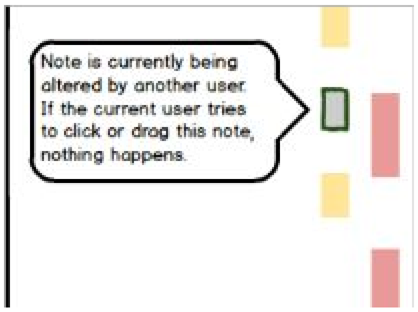




Sign In Error



Two Users Try to Alter the Same Note


Note is currently being altered by another user. If the current user tries to click or drag this note, nothing happens.

Challenges

Design Challenges:

Problem 1 : How to decide when the play function begins playing music

Options and Evaluations:

- Always start from the beginning of the music
  - Always starting from the beginning would set a strict standard that would not be confusing and would be easy to implement
- Always start from where the user is currently editing
  - Always starting from where the user is currently editing would give the user more options, but be more difficult to implement
- Allow user to set a “start point”:
  - A user can specify a location from where to start, so if they want to start listening from a certain landmark in the music, they don’t have to listen to the entire song/listen only right at the point they are editing. Would give the most flexibility, but would be more difficult to implement.

Choice and Justification: Always start from where the beginning of the music

- While the user would optimally not have to start from the beginning every time, the midi.js playback module plays from the beginning of the music file each time. Thus, due to constraints with midi, we have decided it is beyond the scope of the project to let the user when to start the play function and only to play from the beginning of the music file every time.

Problem 2: How to decide who can share a sheet of music

Options and Evaluations:

- Anyone who has access to the sheet
    - Giving any user who has access to the sheet sharing permissions would make sharing more streamlined and less hierarchical
- Only the creator of the sheet
    - Giving only the creator of the sheet sharing permissions would give the creator more control over his/her music

Choice and Justification: Give only the creator of the sheet sharing permissions

- The creator is generally more invested in his/her own music and wants more control over who is able to see it due to security reasons such as plagiarism

Problem 3: How to deal with simultaneous editing of the same note by multiple users

Options and Evaluations:

- Have a visual cue to show users when a note is being edited
    - This should make users aware of the danger of editing the note, but does not stop them from doing so outright
- Lock a note while it is being edited so multiple users cannot edit it at the same time
    - Completely prevents users from editing the note when it is being edited by someone else. However, a user could click a note and then leave their laptop/the browser CoCompose is running on for hours without un-clicking, leaving the note gray and unattended for the duration.

Choice and Justification: Lock a note so multiple users cannot edit it at the same time

- Locking a note is a safer precaution to take to deal with the problem of simultaneous editing which should not happen at all.
- Although this could leave a note grayed out for hours, it is easier for users to deal with and easier for us to handle on the backend.

Problem 4: User presses play while other users are making alterations

Options and Evaluations:

- While a user is editing a note, track this and play the current version of the music.
    - While this would play the current version of the music, tracking all of the changes currently being made to the sheet could be difficult if many are happening at the same time and the music that plays would not be what the user expects since they are unaware of the exact changes being made to each note.
- Every time a user finishes making an alteration, the sheet music saves. When a user presses play, the last saved copy is played.
    - Users will hear the music they expect, and the sheet is saved less often.

<u>Choice and Justification:</u> Every time a user finishes making an alteration, the sheet music saves. When a user presses play, the last saved copy is played.

- This choice means users will hear the notes that are currently shown on their screen. In addition, the sheet will be saved to our database fewer times than it would in the first option.

Problem 5: How do we plan to update the UI as users modify the backend
<u>Options and Evaluations:</u>

- Continuously pull all data from the server
    - While this would be easier to implement and would avoid edge cases, it could overload the server if there are many requests made
- Only push updated changes from the server to the UI
    - This option would have more edge cases to account for, but only handles requests that are necessary each time. Thus, there is less danger of the server loading slowly.

<u>Choice and Justification:</u> Only push updated changes from the server to the UI

- This choice means the server is unlikely to behave slowly as it does not have to load all data continuously. Thus, we will be able to process and update the UI quickly and seamlessly without disrupting the users.