Inseong Cho
App Development & Design

Frances Guo
Design, Website & Documentation

Clifford Huang
App Development & User Testing

Jared Wolens
Design & Prototype Development

MUSIT

Break the mold of SMS and hyperlinks and truly share your music.

PROBLEM AND SOLUTION OVERVIEW

Fig. 1. Overview of Musit application features

Musit aims to create a sharing community that fosters user connectivity and music discovery through proximity and tailored suggestions. Current methods of sharing music over music clients such as HypeMachine and SoundCloud involves copying and pasting bland hyperlinks into texts or emails to the recipient. These types of system are fragmented and reduce the share function to an auxiliary feature; they lack the aesthetic appeal and organized functions that encourage social exchange. To solve this problem, we designed a novel mobile app that is devoted to users find new music with the help of the people around them.

REPRESENTATIVE TASKS & FINAL INTERFACE SCENARIOS

1. *Discover* trending music by location and genre

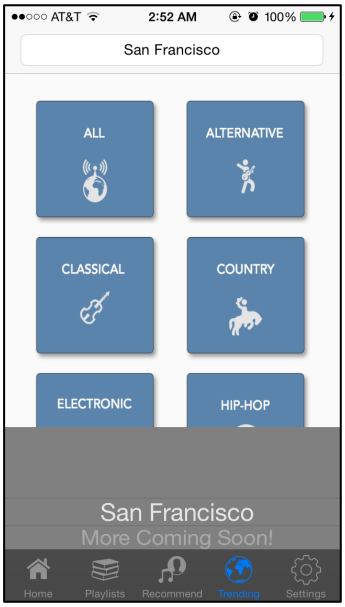

Fig. 2. Select Location

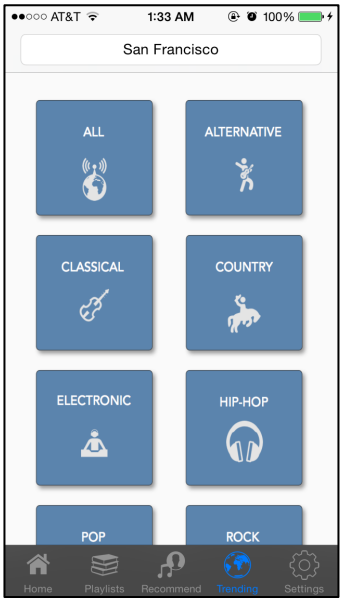

Fig. 3. Select Genre

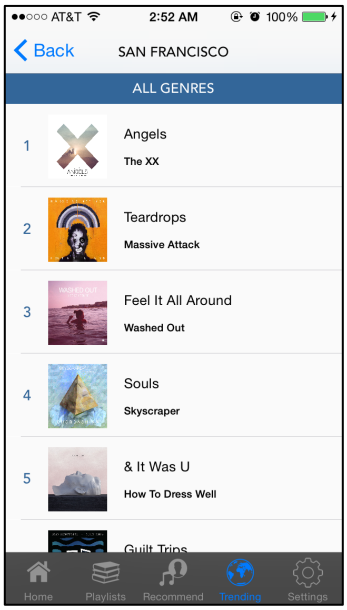

Fig. 4. Top 10 Trending Songs in SF

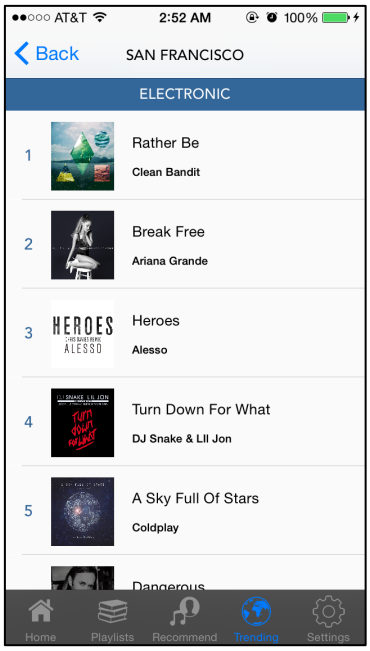

Fig. 5. Top 10 Electronic in SF

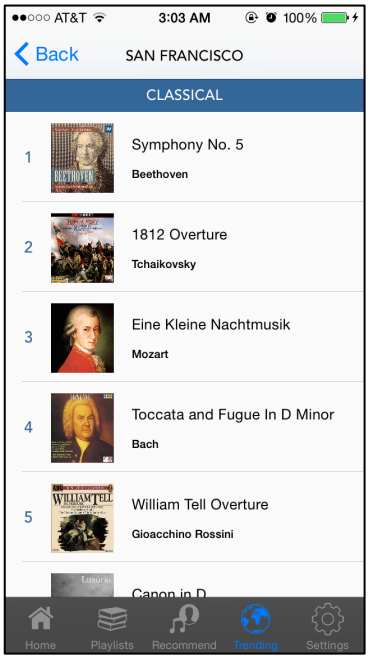

Fig. 6. Top 10 Classical in SF

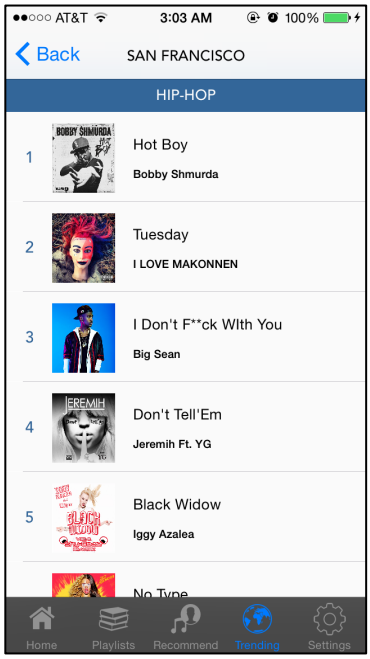

Fig. 7. Top 10 Hip-Hop in SF

To gain a sense of musical community, users will be able to look up lists of songs ranked by popularity within geographical locations (Fig. 2)—for example, the top 10 songs in San Francisco (Fig. 4). Since people who live in the same area often share the same cultural and entertainment preferences, we thought this would be a great way for people to discover new music from the network of people in their local community. To tailor the experience, users can narrow down music lists by genre (Fig. 3) to find songs in niche categories.

2. *Solicit* suggestions by creating playlists for specific occasions or genres

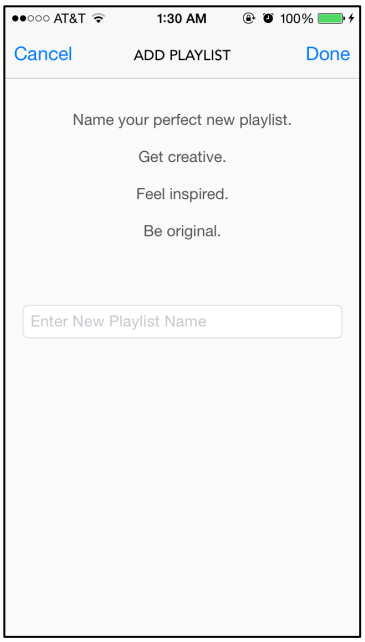

Fig. 8. Create a new playlist

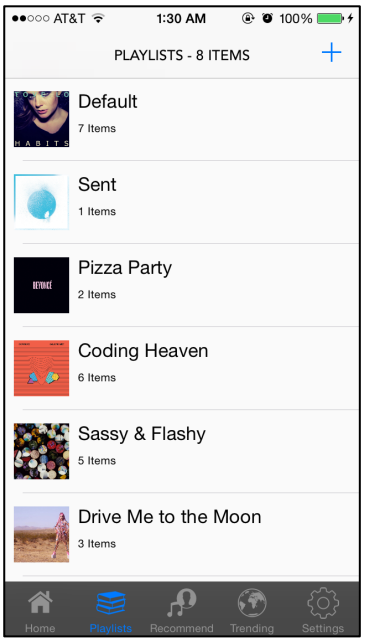

Fig. 9. View all playlists

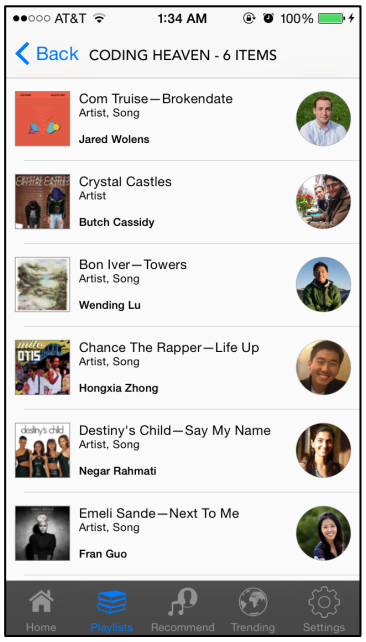

Fig. 10. Songs received in playlist

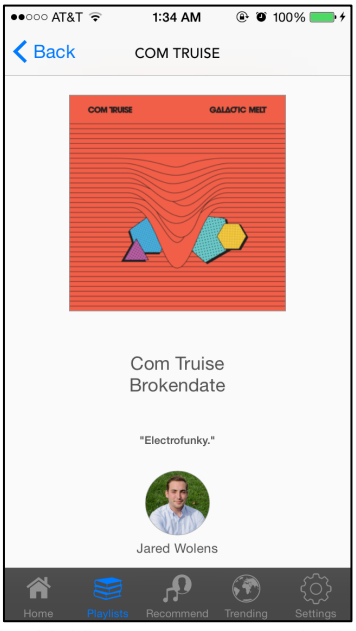

Fig. 11. View a song sent to playlist

The playlists option allows users to create separate playlists for different occasions and genres—for example, a “Coding Heaven” and an “Electro Pop” playlist. This feature was implemented because it enables users to provide direction and ideas for the music suggestions sent by their friends. Clicking on a playlist will bring the user to a list of songs that were sent to that specific playlist. Each song is accompanied by the picture and name of the friend who sent it to the user.

3. *Send* music suggestions to friends' playlists

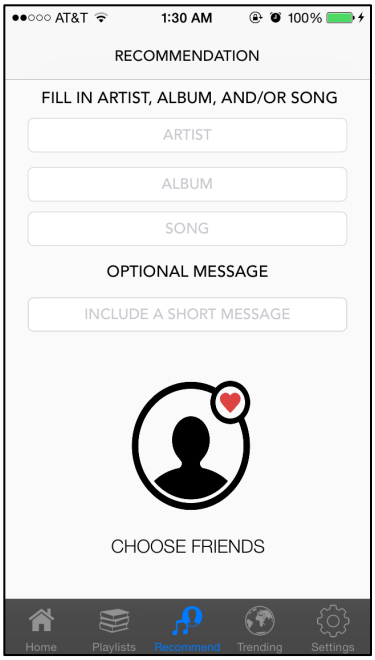

Fig. 12. Create a new playlist

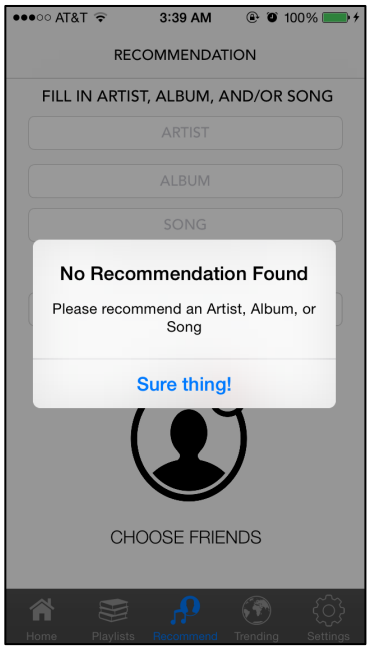

Fig. 13. Error prevention

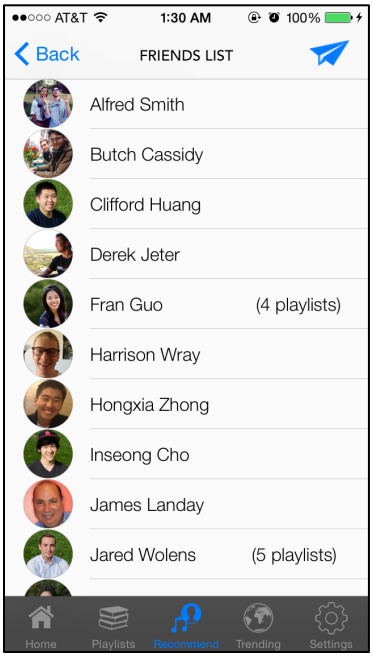

Fig. 14. Select friend recipients

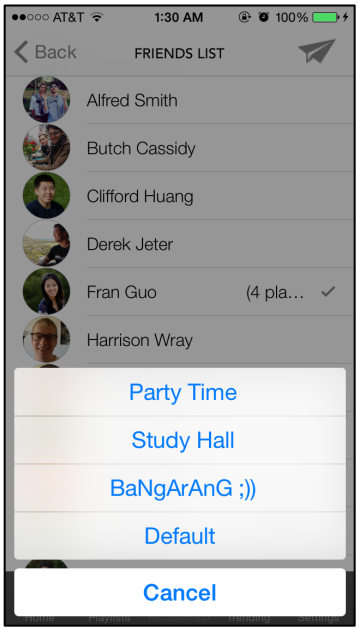

Fig. 15. A friend with multiple playlists

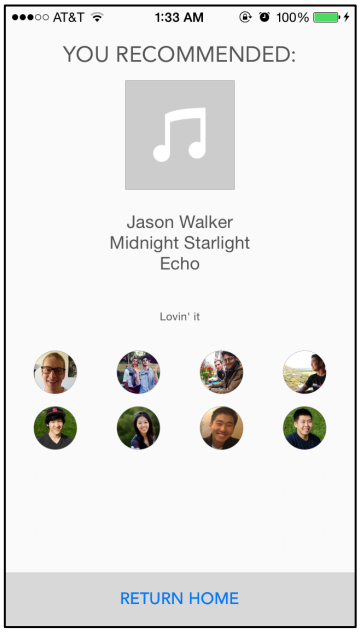

Fig. 16. Sent confirmation

Users can share their love for the music they enjoy by sending it to friends' playlists. Suggestions can be artists or albums in addition to specific songs. Good music deserves a broad audience, so we have also ensured that it is easy for users to send songs to multiple friends at once (Fig. 14). When the user is in the process of selecting friends for the music he or she sends, a list of options will pop up for users with more than one playlist (Fig. 15). The user can then select the playlist that would be the best fit for the suggestion.

MAJOR USABILITY PROBLEMS ADDRESSED

1. [H2-2 Match Between System and the Real World][Severity 3]

"I was confused about where I should click to enter my suggestions on the suggest page. This is because "Example: Apogee" in the text field looks like help documentation, rather than a user field. Should have standard text box instructions. "

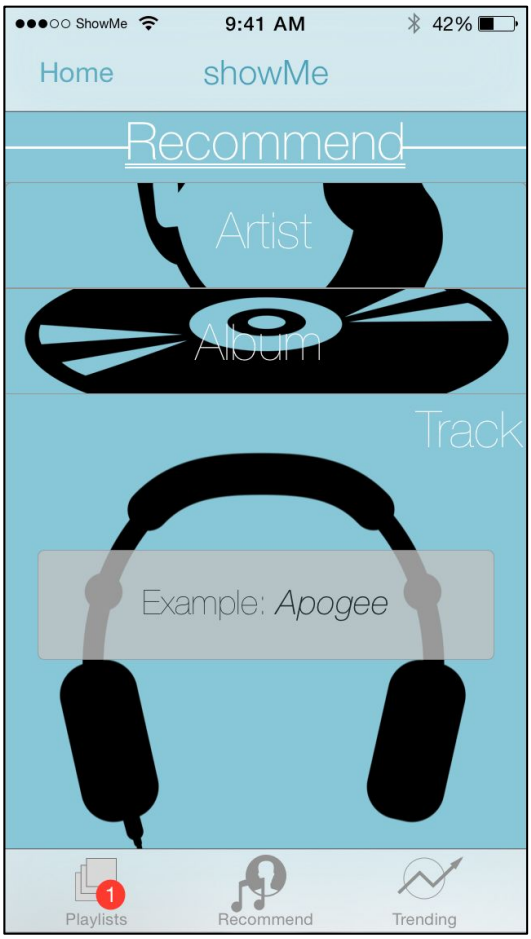

Fig. 16. Before: Confusing text box format

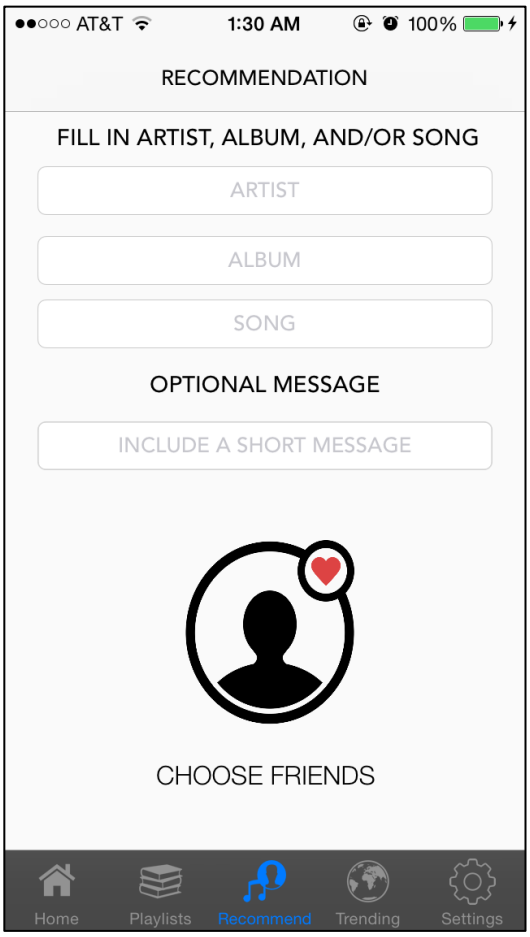

Fig. 17. After: Streamlined layout with grey text fields

FIXED (#1). The recommendations page has been simplified (Fig. 17), to mitigate confusion. Text boxes no longer carry examples, but rather simple instructions in a grey text to indicate that they are designated for user input. We removed the distracting graphics in the background (Fig. 16) to make the layout more aesthetically pleasing.

2. [H2-3 User Control and Freedom][Severity 3]

"Send to and attach options for sending recommendations are inconsistent … one of them determines the recipient by entering the friend's name into a text box while the other makes you choose from a list of friends."

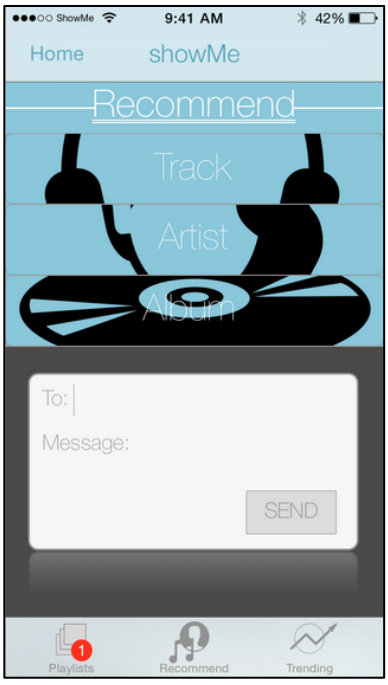

Fig. 18. Before: Type name

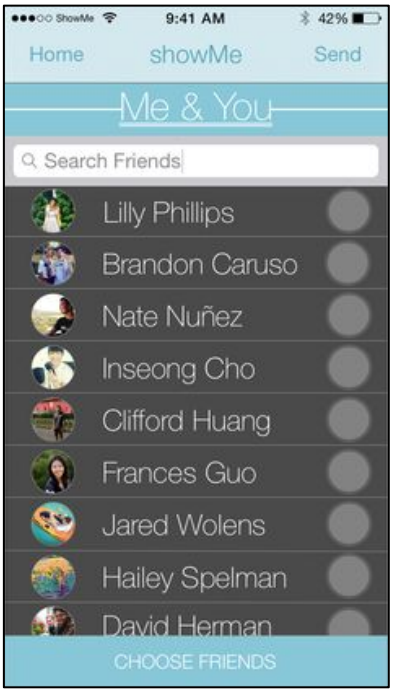

Fig. 19. Before: Select name from list

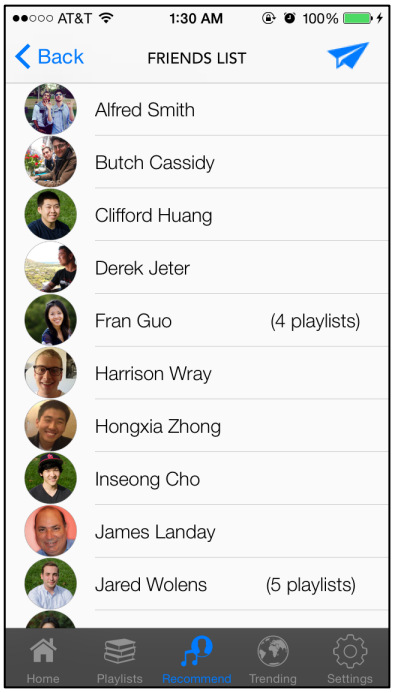

Fig. 20. After: Select name from list

FIXED (#2). We removed ambiguity from the friend selection component of the recommendation feature in order to maintain consistency. Users never have to type in the name of a friend (Fig. 16); a friends list (Fig. 18) is always provided.

3. [H2-5 Error Prevention][Severity 3]

"No error when I tried sending to a friend without giving an actual music recommendation."

(No before images because feature was missing).

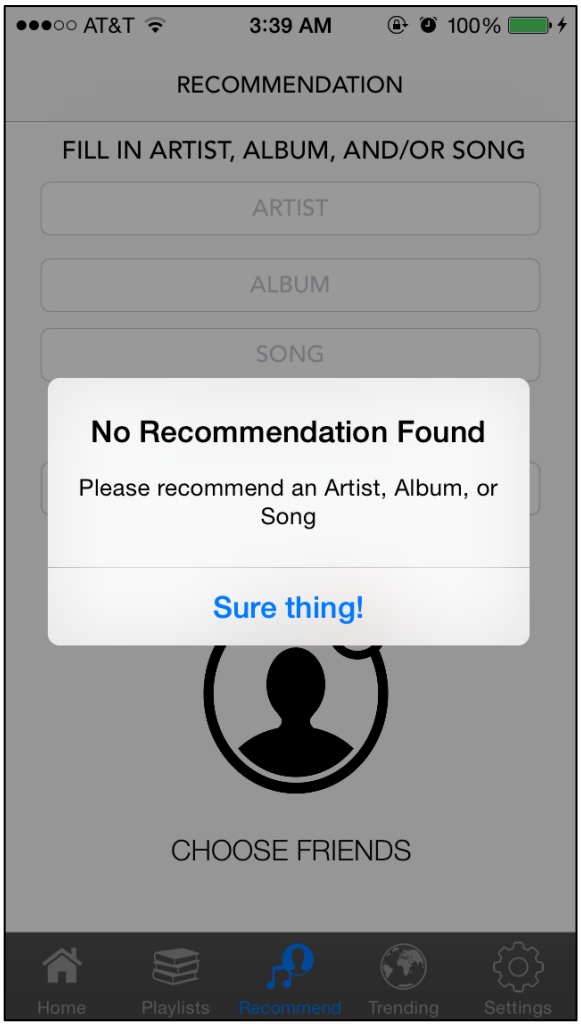

Fig. 21. After: Error prevention when song information is not indicated

FIXED (#3). We implemented error messages (Fig. 19) to improve robustness and prevent users from making the error of sending a blank suggestion to a friend.

4. [H2-3 User Control and Freedom][Severity 3]

"There does not appear to be a delete option for playlists or for any of the songs."

(No before images because feature was missing).

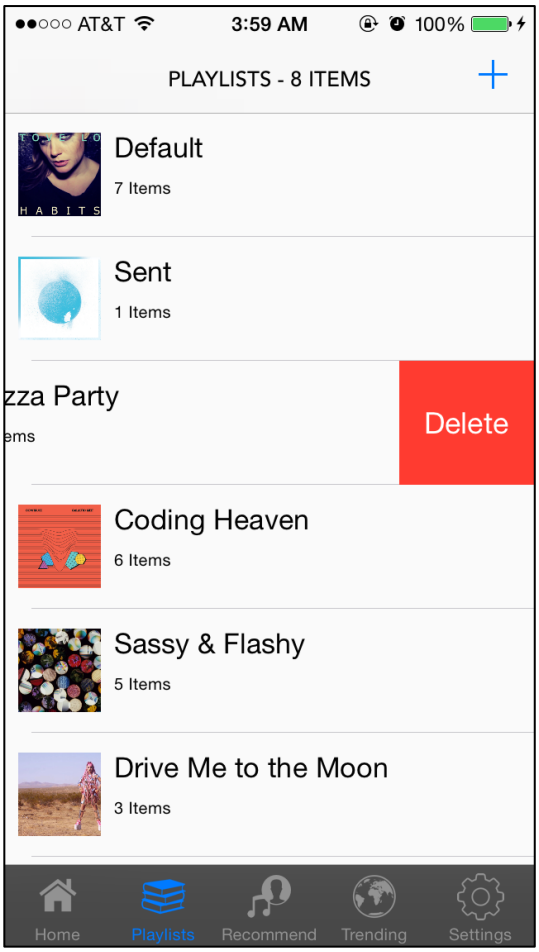

Fig. 22. After: Delete a playlist

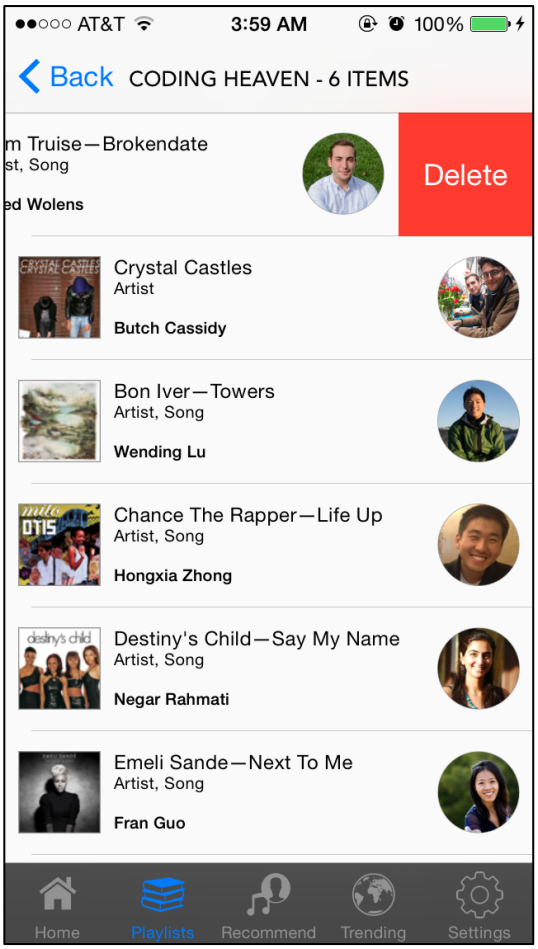

Fig. 23. After: Delete a song from a playlist

FIXED (#4). The reason the delete function was missing from the medium-fi prototype is that we were always planning to use a swipe-left-to-delete functionality, but this is impossible to demonstrate in InVision. It is definitely important to give users control over which songs remain in their playlists, so this was an obvious implementation when we started coding the high-fi prototype.

5. [H2-3 User Control and Freedom][Severity 3]

"Unable to go back to a previous field to edit information while in the process of creating a playlist."

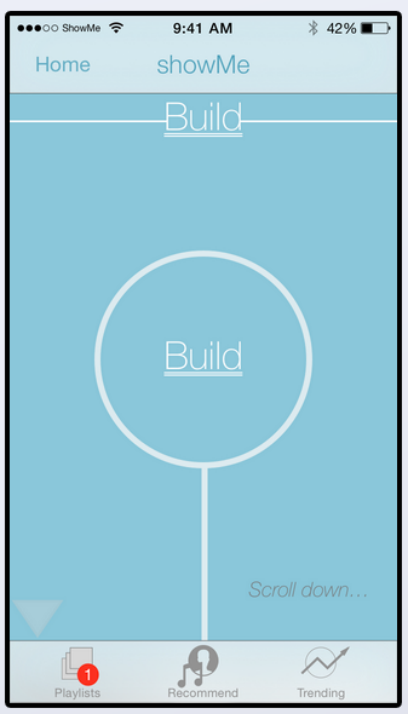

Fig. 24. Before: Build Screen 1

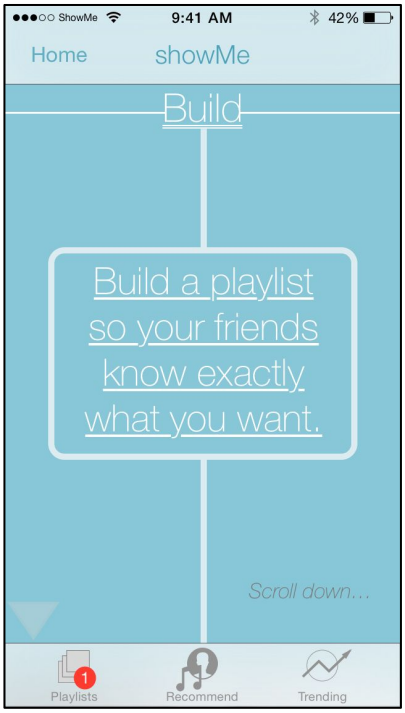

Fig. **25. Before:** Build Screen 2 Instructions

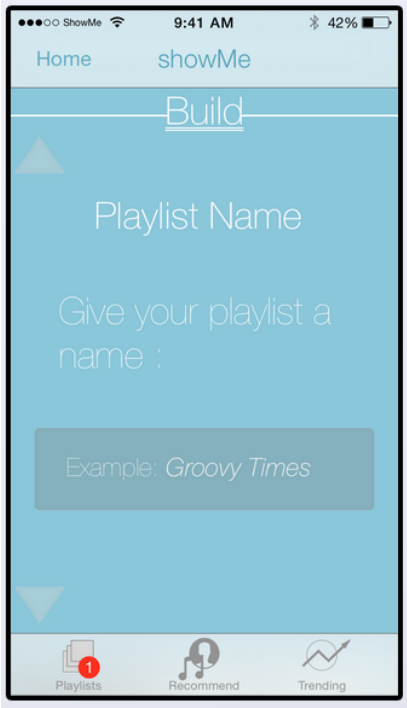

Fig. 26. Before: Build Screen 3 Name Playlist

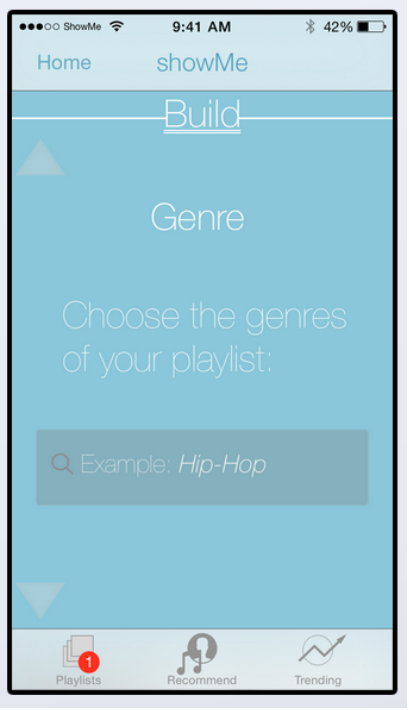

Fig. 27. Before: Build Screen 4 Choose Genre

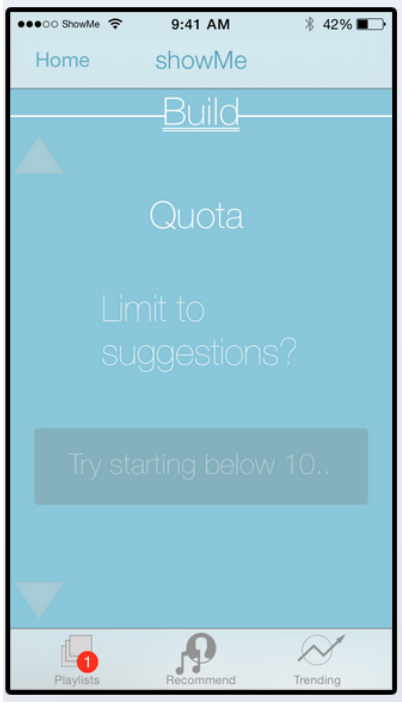

Fig. 28. Before: Build Screen 5 Set playlist quota

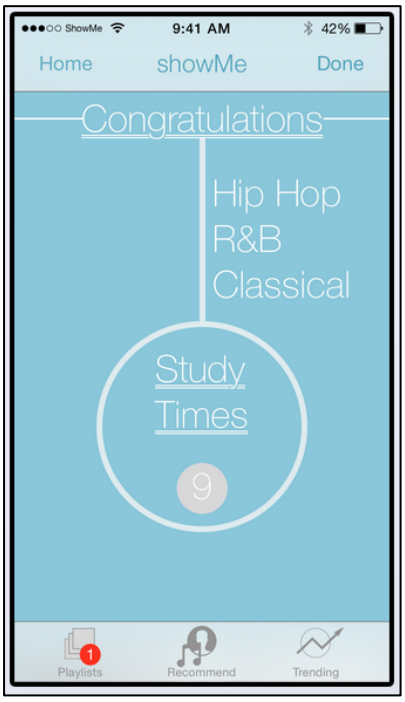

Fig. 29. Before: Build Screen 6 Confirmation

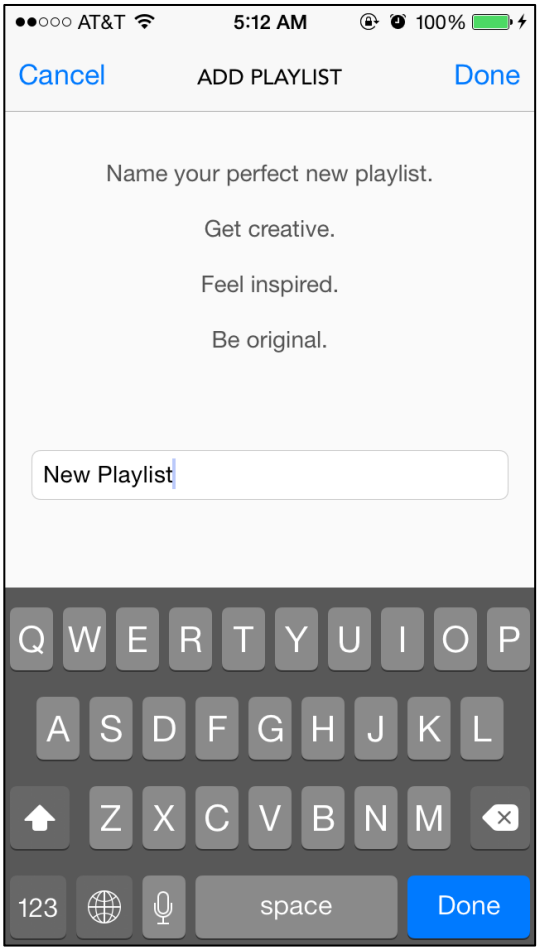

Fig. 30. After: Add new playlist

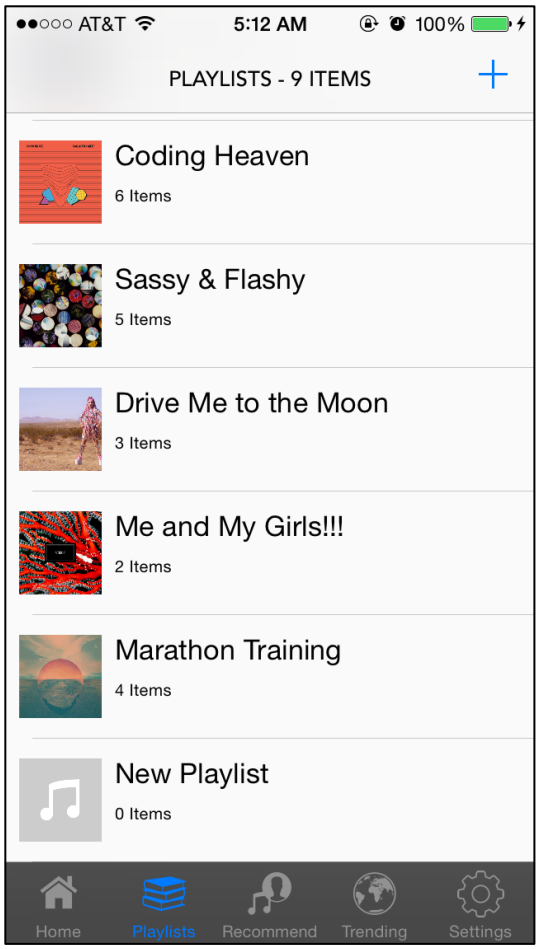

Fig. 31. After: New playlist is added to tab

FIXED (#5). By changing our create playlist page to a single page without scrolling, the user will never need to scroll back up or go back to a previous page. We decided to remove the genre and quota fields because user testing showed that users don't like to limit the number of songs that their playlists can accept, and they do not feel terribly strongly about constraining their playlists to a specific genre. The playlist name is enough to give context about the type of music that the user is hoping to receive.

6. [H2-3: User Control and Freedom][Severity 4]

"There is no exit or back button to leave the build playlist process."

FIXED (#6). It is certainly important to give the user the opportunity to exit without building a playlist. In the medium-fi prototype (See Fig. 24-29) the only way to exit was to click on another tab in the dock. Since the hi-fi prototype is only one page, we included an exit option using the cancel field in the top left corner (Fig. 30).

7. [H2-4 Consistency and standards][Severity 3]

"Build playlist interface is inconsistent and fragmented. This entire task could have been completed in one screen."

8. [H2-7: Flexibility and Efficiency of Use][Severity 4]

"Scrolling through several screens to create a playlist is cumbersome."

FIXED (#7 & #8). These two parts of the heuristic evaluation addressed the same issue, which was that the build playlist feature was too long and cumbersome for what should have been a simple task. We agreed with this sentiment, and reduced our app from a 6-page long task (Fig. 24 – 29) to a 2-page task including the confirmation (Fig. 30 – 31).

9. [H2-6: Recognition rather than Recall][Severity 4]

"In playlists, music is represented by album art without accompanying text descriptions. People who do not recognize the art are forced to click or swipe to look through the list."

Fig. 32. Before: Home playlist only shows album art

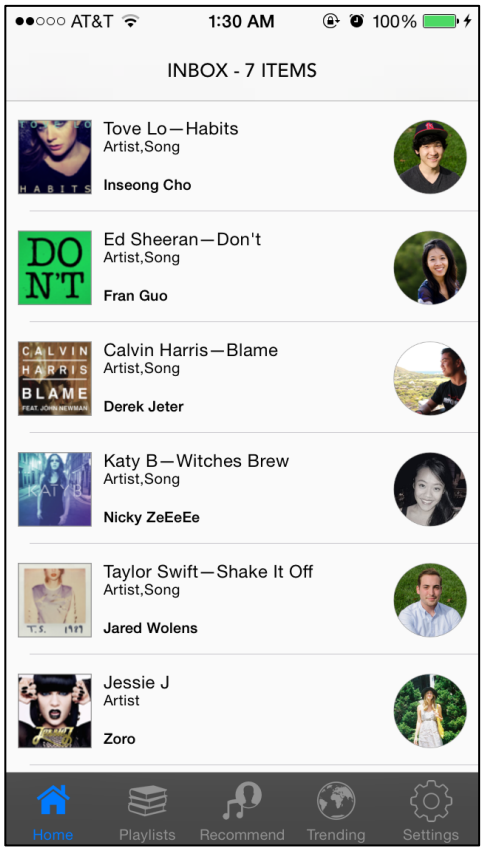

Fig. 33. After: Songs in inbox playlist with art and details

FIXED (#9). We agreed that it was very difficult to identify the song that was being shared without sliding out the panel in the card view. We ultimately decided that the stack was not a great way to represent information at a glance, despite being an interactive element. Our new design is much cleaner, with clear text. The album art is still an important element of the page, but it is more proportional with the rest of the

information. We also included the name and picture of the suggestion's sender, to add a personal touch to the recipient's playlist.

10. [H2-9 Help Users Recognize, Diagnose, and Recover from Errors][Severity 3]

"The trending list is error-prone because users must type in the descriptor. Drop-down menus or auto-completion features would help the user avoid formatting or invalid field errors."

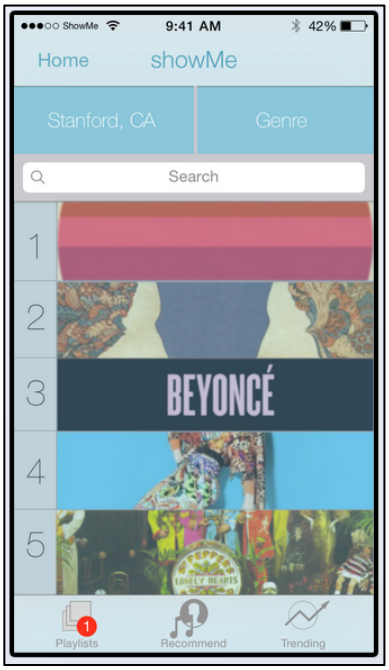

Fig. 34. Before: Trending page

Fig. 35. Before: Manually type in genre

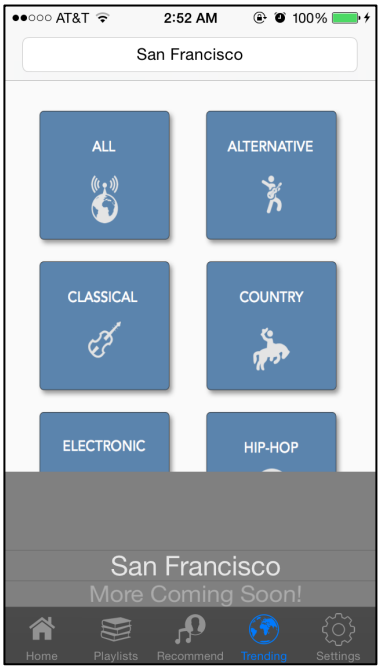

Fig. 36. After: Trending - Dropdown Locations

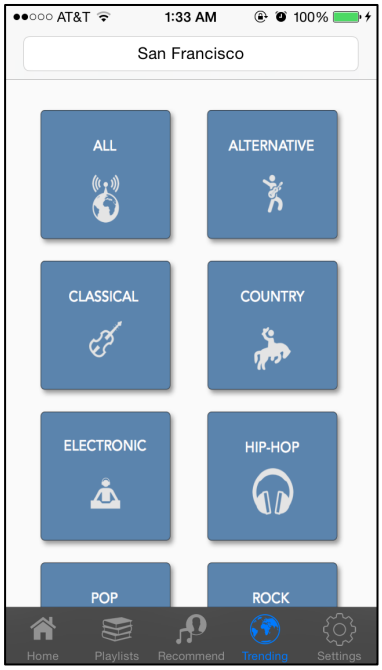

Fig. 37. After: Trending - Genre buttons

FIXED (#10). While it involves a tradeoff with user control and freedom, we decided to incorporate the feedback and present presets (Fig. 36-37) for location and genre in the Trending page, rather than ask the user to enter information manually. Ideally, we would be able to implement the best of both worlds by using autofill. For the sake of this hi-fi prototype, however, it was not possible to implement the backend database that would have allowed us to incorporate such a function, so we decided to play it safe and avoid user errors before giving the user more freedom and control over the inputs.

11. [H2-4: Consistency and Standards][Severity 3]

"On the home screen, there is a link to the home screen that is clickable."

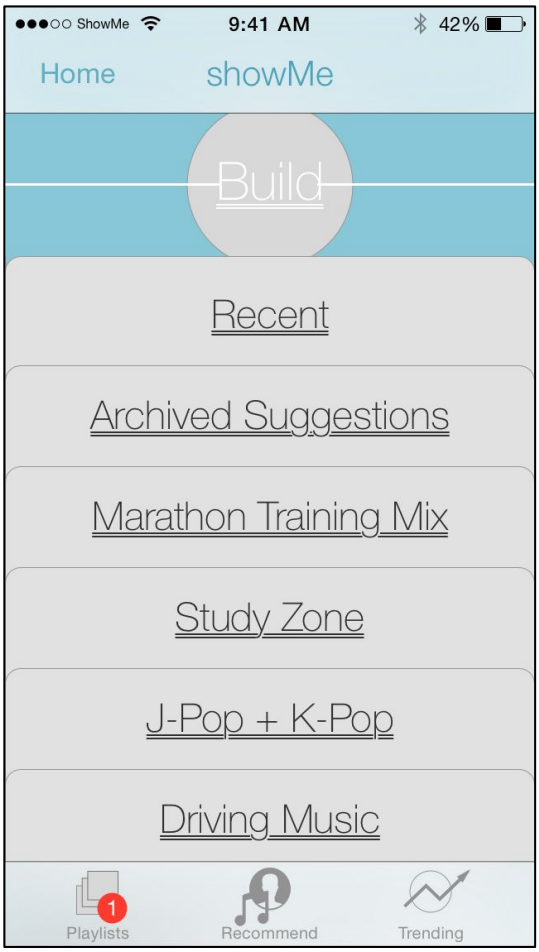

Fig. 38. Before: Home page, with home button at top left

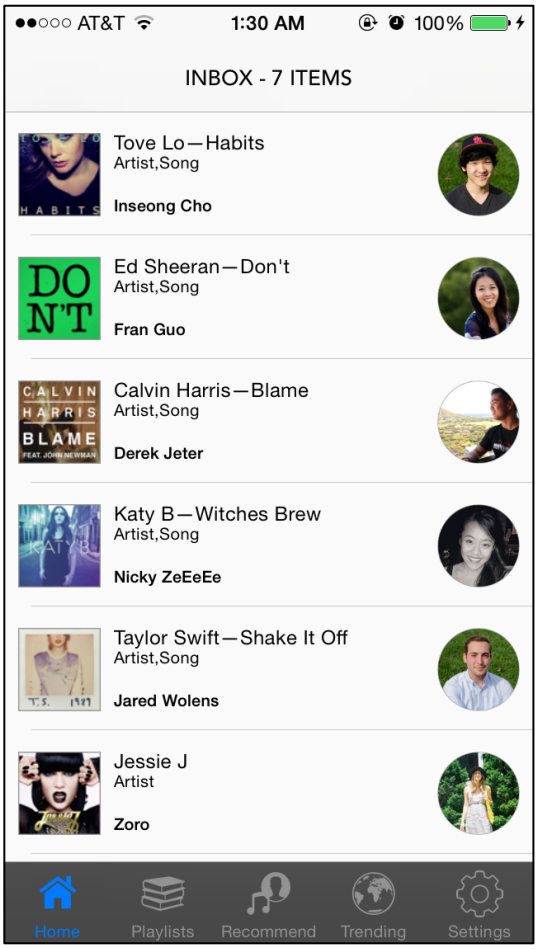

Fig. 39. After: Home page, without redundant links

FIXED (#11). This heuristic violation was an oversight on our part to remove the Home button from the home page in the medium-fi prototype (Fig. 38). We immediately fixed this when implementing our high-fi prototype, which does not contain redundant links (Fig. 39). We did, however, augment the number of icons featured in the dock. This will be explained in the section below, which describes other changes.

12. [H2-10 Help and documentation][Severity 3]

"It is unclear what criteria determined the home page for showMe (i.e. whether the suggestions were based on the user's location or genre preferences."

FIXED (#12). It is understandable that the evaluator misinterpreted the point of our home screen in the medium-fi prototype (Fig. 32). The home screen does not actually consist of suggestions that were generated by the app according to the user's settings in the Trending page (location and genre). The suggestions are songs that were sent to the user from his or her friends. To make this more evident for people, we titled our home page with an Inbox header (Fig. 39).

OTHER CHANGES

Fig. 40. Before: Dock contains Playlists, Recommend, and Trending

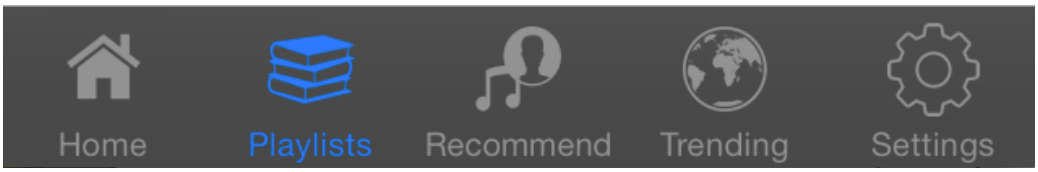

Fig. 41. After: New dock includes additional elements: Home and Settings

Dock: To avoid having a home navigation button in the top left corner of every page, which seemed like unnecessary segregation from the rest of the feature navigation buttons, we moved the Home button to the dock (Fig. 41). This makes it more easily accessible from every screen. We also added a settings icon, to accommodate for later implementations such as user account detail modifications.

.

DESIGN EVOLUTION

HOME PAGE

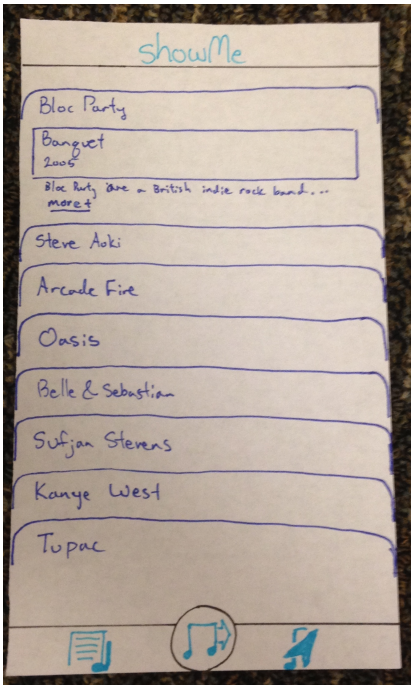

Fig. 42. Before: Low-fi home page

Fig. 43. Before: Med-fi home page

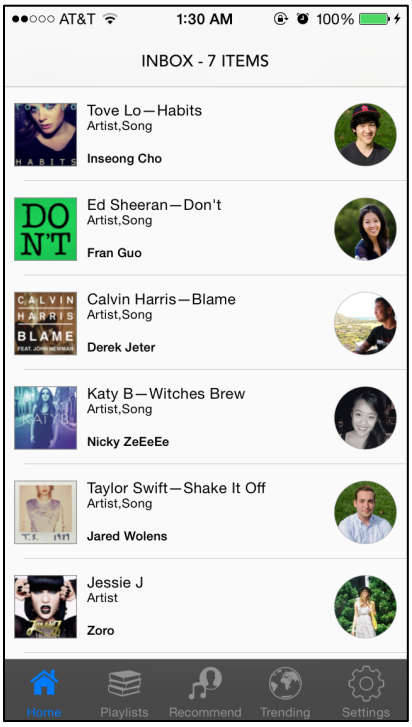

Fig. 44. After: Hi-fi home page

The overall design of our home page has always remained the same. The home page consists of an inbox of all received suggestions, with a dock along the bottom for navigation to the three features (create new playlist, send songs, search for popular local songs).

In the medium-fi prototype, Minor changes (Fig. 43) included incorporating more color for visual appeal, as well as creating better icons for easier navigation. We added numbers to the inbox so that it would be easier to quantify the number of suggestions.

For our newest prototype, we toned down the overall color scheme because we liked how a minimalistic white and black app helps the colors of the album art and profile pictures to pop against the background. We also changed the dock to include Home and Settings—it felt to have the home button in the top left during the medium-fi prototype, because it was unnecessarily segregated from the other features. We also changed the visual display of the items in the inbox playlist—each item now contains the name of the song and artist, as well as the suggester's name and image, in addition to the album art. We still wanted to quantify the number of items in the list, but we decided to do so by summarizing the number in the header rather than take up horizontal space by numbering the songs on the left.

TASK 1 EVOLUTION: TRENDING

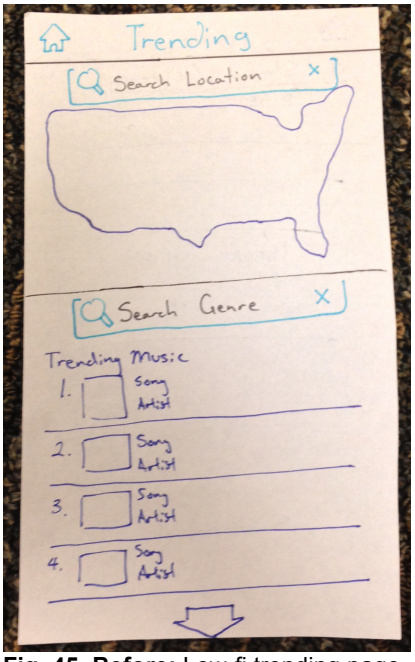

Fig. 45. Before: Low-fi trending page

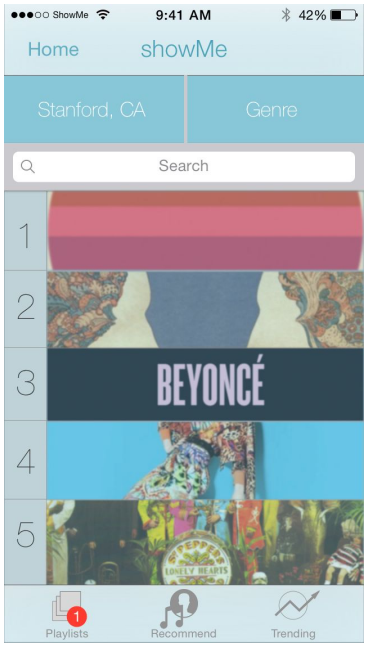

Fig. 46. Before: Med-fi trending page

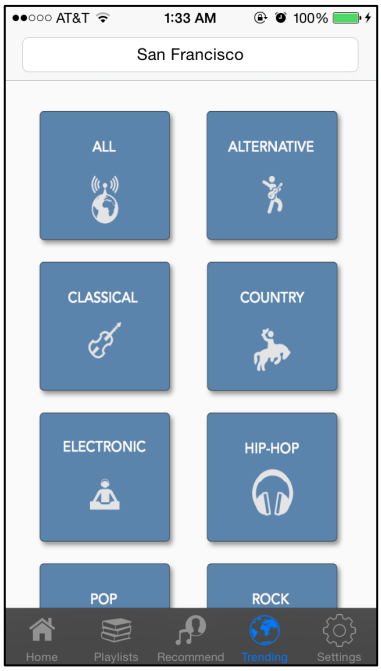

Fig. 47. After: Hi-fi trending page

We went through many design iterations to find out how to present two disjoint factors—location and music genre—effectively on the same page. During the low-fi prototype, we received lots of negative feedback about the confusing double search bars, as well as cluttered layout and irrelevant map (Fig. 45).

For the medium-fi prototype, we changed the layout to be more simple and streamlined (Fig. 46), with the list of popular songs occupying most of the space. There was a default location set to the user's current location. Clicking on either the location or genre button at the top will allow for further filtering. Unfortunately, this filtering involved manual input from the user, which risked creating the errors of misspelled cities and genres.

We further streamlined the layout for our high-fi prototype, and we also changed the logic of the section. We reasoned that it would make sense for the user to first input a location and a genre before displaying any information about the songs. As a result, the landing page is a series of buttons representing genre options, with a simple header at the top that allows the user to select a location from a dropdown menu. Asking users to indicate preferences before displaying any songs makes it mandatory for them to tailor the experience to their geographical location and preferred genre.

TASK 2 EVOLUTION: ADD/VIEW PLAYLISTS

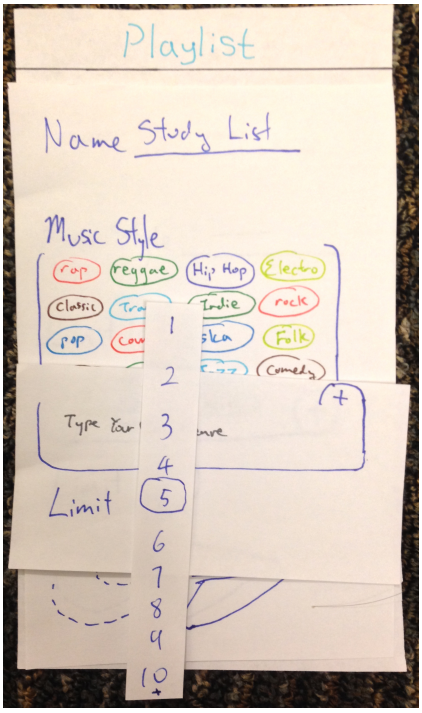

Fig. 48. Before: Low-fi add playlist

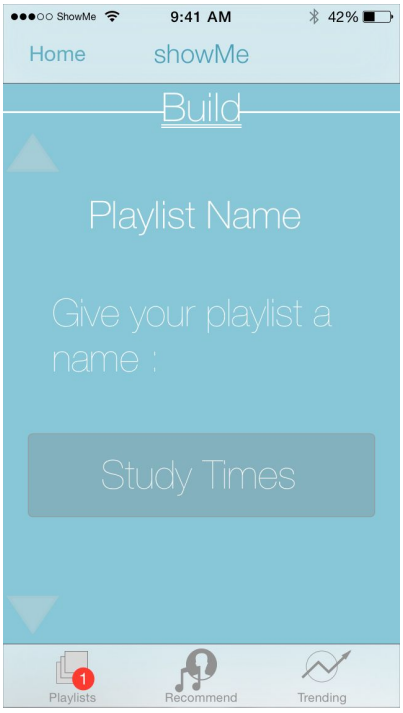

Fig. 49. Before: Med-fi add playlist

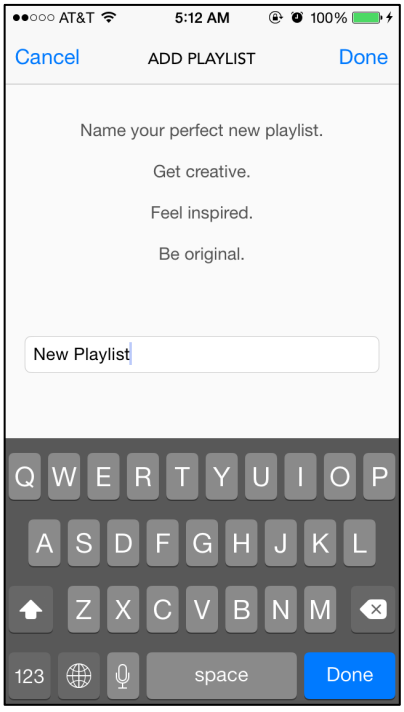

Fig. 50. After: Hi-fi add playlist

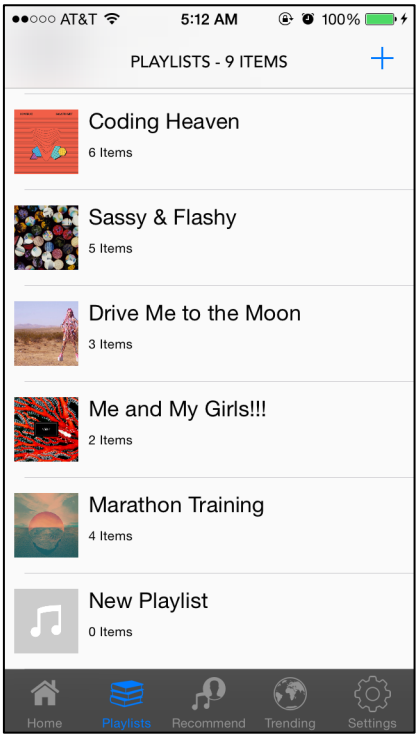

Fig. 51. After: Hi-fi view all playlists

The playlists feature was the most redesigned element of our application. Initially, our concept for playlists was that it would only be possible to have one at a time, because we

believed it would make sending to a friend's playlist more exclusive if there were limited spots for song suggestions. To this end, we also designed a song limit so that the user could further restrict their playlist (Fig. 48).

After conducting user testing, however, we learned that most users are not concerned about making their playlists exclusive; they would rather have multiple playlists at a time. It was only after this feedback that we implemented a "view all playlists" feature (Fig. 51). We tried to be very accommodating during our medium-fi prototype by creating a long page that the user would scroll through, taking each step of the process step-by-step. The heuristics evaluation, however, soon helped us realize how cumbersome it was to scroll through multiple pages in order to create a playlist. Our final prototype has much less going on, and is constrained to a single page.

TASK 3 EVOLUTION: SEND A MUSIC SUGGESTION

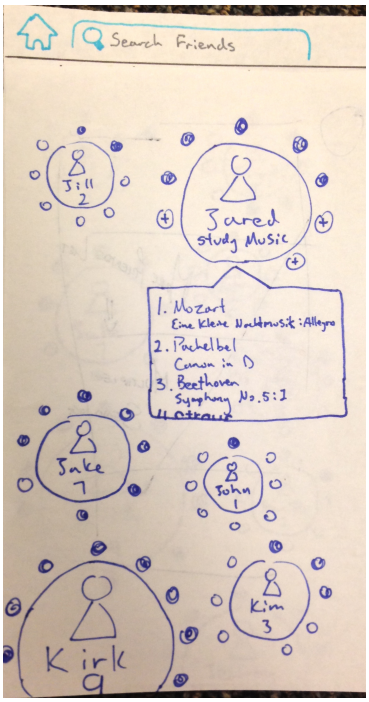

Fig. 52. Low-fi Friends' playlists

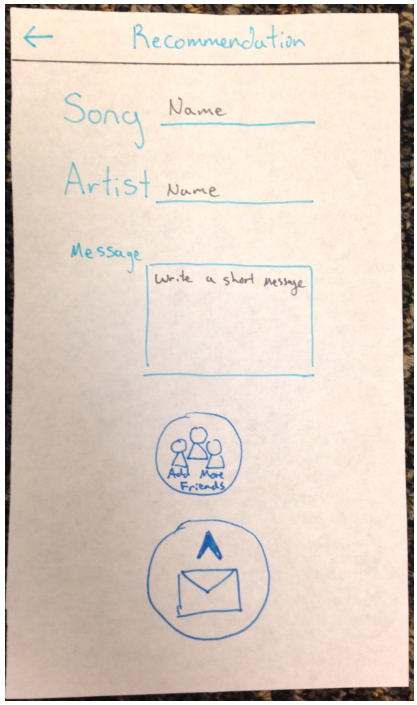

Fig. 53. Low-fi recommend music

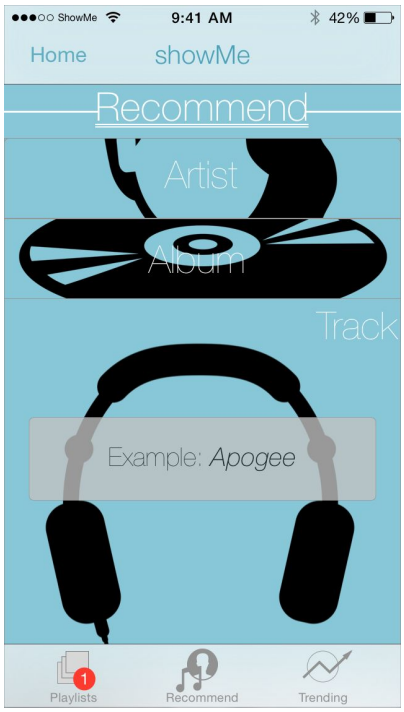

Fig. 54. After: Med-fi recommend music

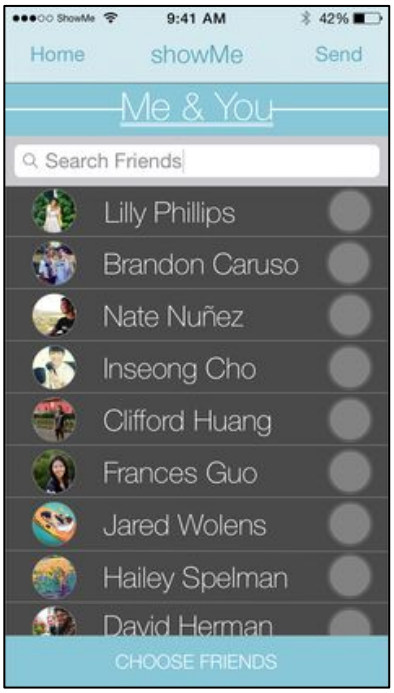

Fig. 51. Med-fi select recipients

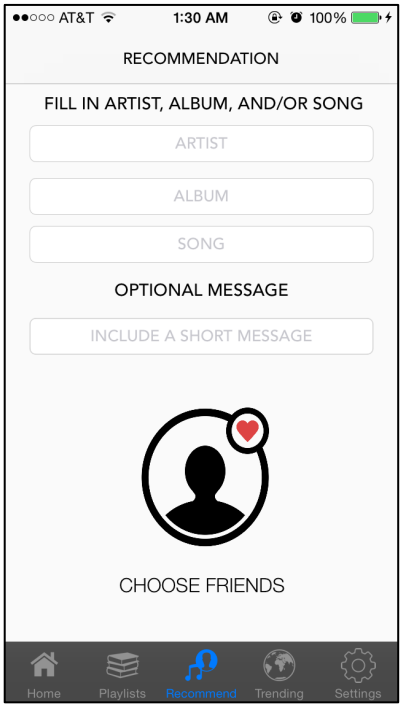

Fig. 52. High-fi recommend music

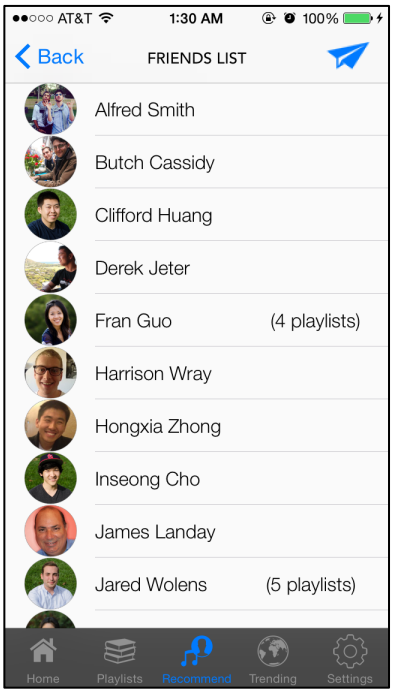

Fig. 53. High-fi select recipients

Interestingly, our suggestions page reverted to a design that is closer to our low-fi design than to the medium-fi design. The feedback for the suggestions feature in our low-fi prototype was that seeing friends' playlists represented in bubbles on the screen made the app look very cluttered. During the medium-fi prototype, we removed the feature, because there were also privacy concerns about who would be able to see music recommendations that were sent by other friends.

For the medium-fi prototype, we streamlined the recommendations page so that it no longer featured friends' playlists. From the user studies, that feature was not very useful and it cluttered the design. Instead, we redirected the suggestions button to a page with a variety of options (Fig. 53) for suggesting music: by artist, album, or track. We made it much easier to send to multiple friends at once (Fig. 51). In the low-fi prototype, selecting recipients was a sub-option and the user had to type in each person's name manually.

Our high-fi prototype discontinued the use of the stacking recommendations page, because users found it very confusing and counterintuitive. We removed the distracting background graphics and changed the format of the text boxes to make it more evident that they were intended for user-generated input.

PROTOTYPE IMPLEMENTATION

Tools

We used Sketch and Photoshop to create the design for our high-fi prototype interface. The actual application was coded in Swift, using Xcode. For reference materials, we turned to Stack Overflow. As a design tool, Sketch was simple, provided quick access to necessary tools and presets, and contained iOS templates. The drawbacks were that it often crashed and lagged and tools were occasionally too automated, which hindered image fine-tuning. Photoshop was good for image details, but unfortunately does not contain iPhone templates.

We much preferred Swift to objective C as our programming language; it was easy to learn thanks to the playgrounds. Xcode was necessary, but would often crash and lose our progress.

To keep everyone on our team in the loop, we collectively placed our files in a shared Dropbox folder so that everyone would have access to our progress.

Wizard of Oz Techniques

In our current prototype, many of the hard-coded features described below were the product of Wizard of Oz Techniques—for example, the suggested songs that are currently in the playlists. Other techniques include the imaginary back end's magical ability to fill in album art for a song that the user suggests to other friends.

Hard-coded Features

Our application is meant to operate with user data, so most of our hard-coded features were data points that we would have had to gather through actual uses. We had to hard-code the friend list user data, user playlists, suggested songs, trending songs, and album artwork.

Future Aspirations

Our first priority is to take care of unimplemented features. We will need to support password-protected user accounts, create a backend for storing data and autofilling the correct spellings of music names, and implement mobile alerts and notifications. Other technical details that need to be sorted out include implementing the delete function for the Default and Sent lists, which we were unable to figure out given our time constraints.

The overarching goal for our application is to become universal. We would like to look into music client APIs (Spotify, SoundCloud, etc.) to see if we will be able to link to their applications and augment the music sharing experience. A more ambitious goal is to integrate our application with an actual music player, so that the user does not have to leave the app in order to listen to suggestions.