team **NOTED**

Noted

where your story meets soundtrack

01

intro

the team

Alyssa

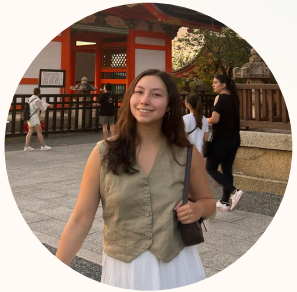
product design

Kabir

computer science

Caleb

music



Jasmine

psychology

Noted

where your story meets soundtrack

problem

Even diligent photo takers and journalers do not always look back on what they have generated, **missing out on the opportunities for reflection and introspection**

solution

Noted is a digital platform that **prompts reflection** through the **pairing of photo and song** which can be **shared** with friends and stored in a **personal timeline**

target audience

Individuals who enjoy looking at **photos** and listening to **music** that desire an organized way to **chronicle and reflect** on past experiences

table of contents

02

heuristic evaluation results

high-level summary of results

4	severity 4 violations	→	2	changes
28	severity 3 violations	→	12	changes
37	severity 2 violations	→	15	changes
13	severity 1 violations	→	5	changes
82	total violations	→	34	total changes

most violated heuristics were **H4: Consistency & Standards** (15 violations), **H2: Match Systems & World** (12 violations), and **H1: Visibility of Status** (8 violations)

severity 3 & 4 violations

Most common severity 3 & 4 violation themes...

- Changes for violations that severely impacted **user control and freedom**, such as:
  - Ensuring back buttons are functional on all screens
  - Adding the ability to delete drafts of messages to friends or photo-song pairings
  - Allowing for editing of captions for photo-song pairings
- Changes that **clarified the purpose and functions** of the app, like:
  - Adding an onboarding tutorial for new users
  - Changed the term to “Feed” in the navigation bar
- Changes that made the app **more accessible and generally easier to navigate**:
  - Adding play buttons for the songs paired the photos
  - Adding song and artist names for songs

For a full list of Severity 3 & 4 Violations and their fixes see here!

design changes from addressed violations

01 Clarifying the functionality of the photo/song pairings

Making the clickability of photo-song pairings and playability of songs more intuitive with our UI design

02 Using more descriptive language

Examples include:

- changing “collections” to “albums” on reflect page
- changing “share “ to “add more”
- changing “Remember this” vs. “check this out” on reflect page

03 Increasing user control

- Creating multiple methods of sending a photo/song pairing
- Adding functional back buttons on all nested screens
- Implementing functionality to delete shared posts and drafts
- Allowing users to edit and delete text attached to pairings

unaddressed violations - 01

Violations stemming from functionality of chosen prototype medium for the med-fi prototype

Examples:

- Navigation bar moving downwards when a permissions button is toggled
- Photo filters not working on Figma

Rationale:

These kinds of violations will be resolved by virtue of building the high-fi prototype and should not require additional revisions of our design to be fixed

unaddressed violations - 02

Low severity violations that were more a matter of personal preference than UI

Examples:

- Changing photo-song pairing album names from “latest hits” to “your entries”
- Using a different icon for regenerate

Rationale:

These kinds of violations had little impact on the usability and functionality of the app and, as a result, were deprioritized

unaddressed violations - 03

Violations for which their solution would result in a conflicting violation

Examples:

- Adding labels to universally recognizable and technologically standard icons
- Changing standard icons to text buttons

Rationale:

We determined the new conflicting violation that would result from attempting to solve the identified violation to be a greater issue, such as violating a standard in similar apps or reducing the accessibility of the app

unaddressed violations - 04

Violations that were features that the evaluator did or did not want to be added rather than a usability issue

Examples:

- Allowing users to direct message a private photo-song pairing to a friend
- Features the evaluator wanted to be added
  - Including new functionality for messaging
  - Adding search features to messages

Rationale:

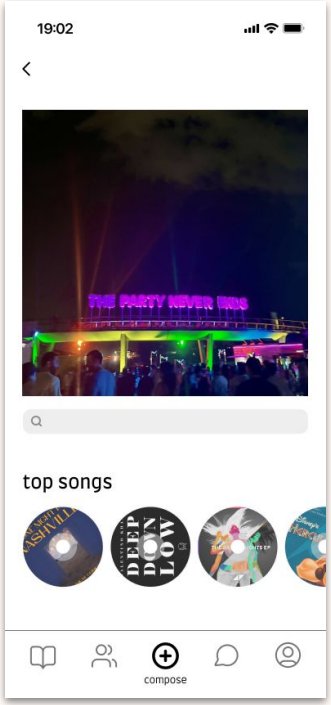
We were looking to prioritize issues that would affect usability rather than add or remove certain functionality

03

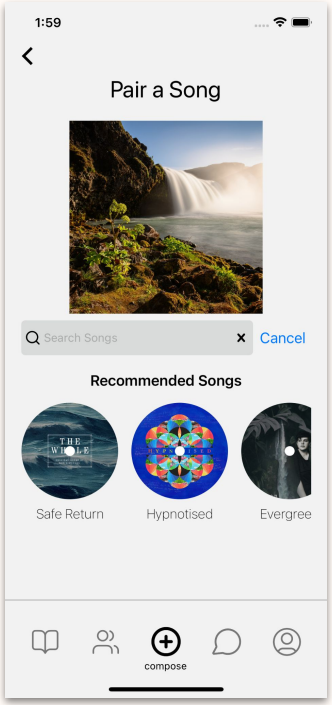
design revisions

adding in song names to records

severity 4 violation of H11



before

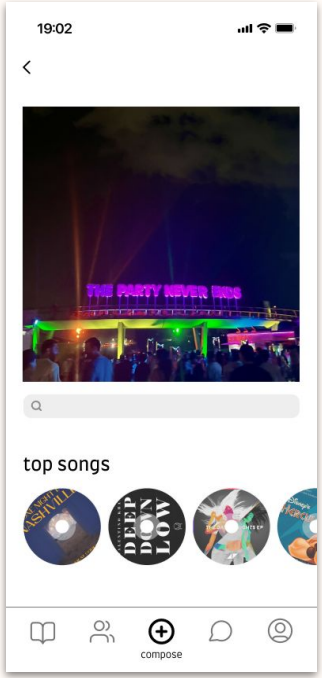


after

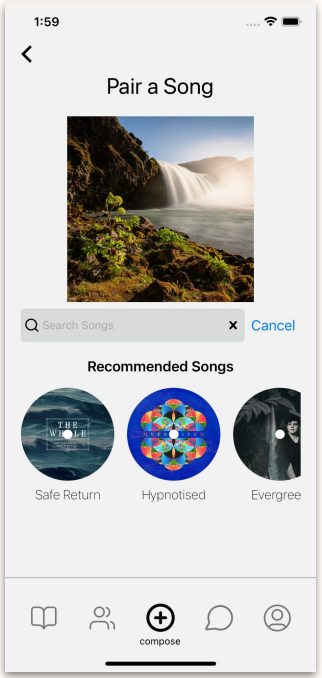
Efficiency:
Before, users had no way of identifying the specific track since only the album art was displayed. Users can now **identify specific tracks** from albums to pair to their photos

increasing size of the records

severity 2 violation of H11



before

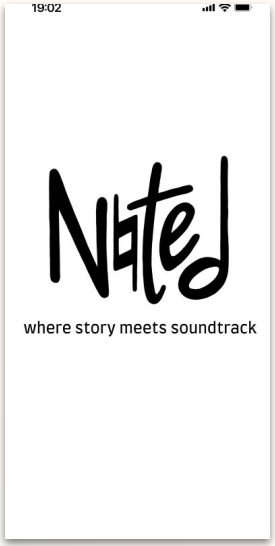


after

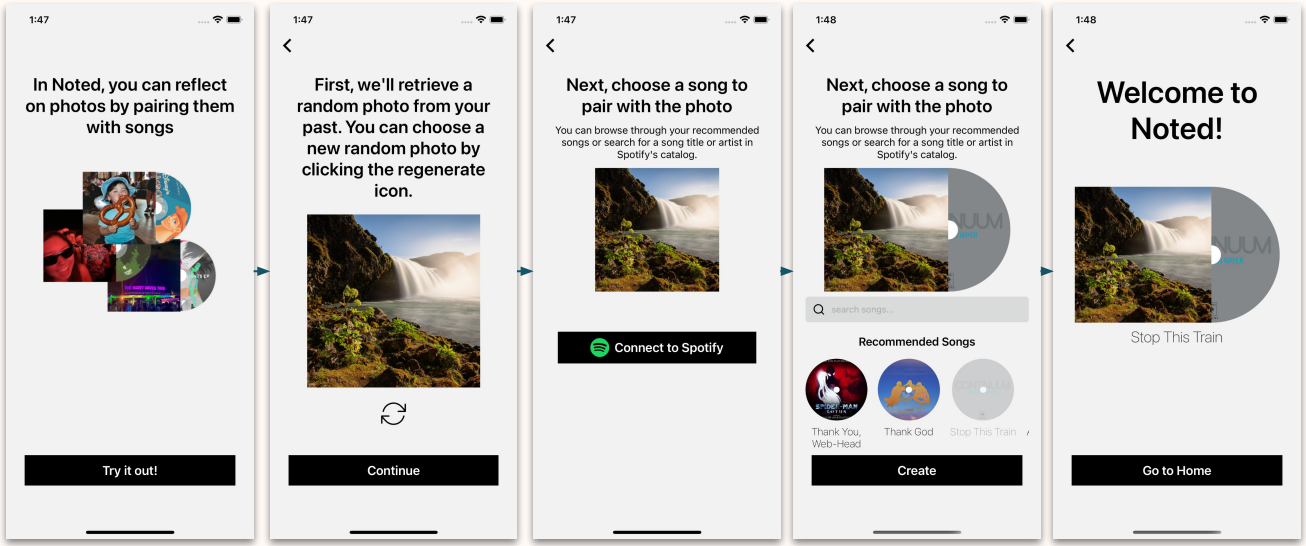
Robustness:
We increased the size of the records to **make them easier to press** and, therefore, more accessible

creating an onboarding tutorial

severity 3 violation of H2



previous opening screen
(no onboarding tutorial)

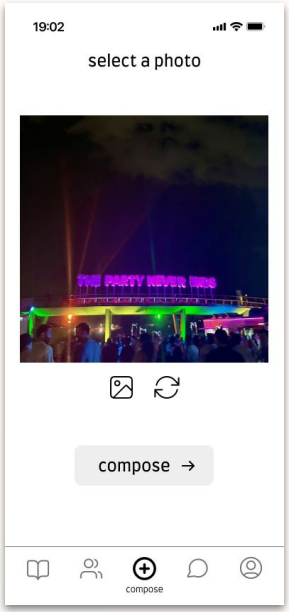


flow of onboarding process

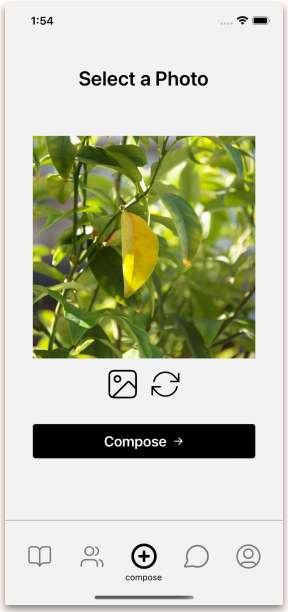
Efficiency & Robustness: The addition of an onboarding tutorial **clarifies the purpose** of the app and helps **lower the learning curve** to becoming an apt user of the app

increased color contrast of buttons

severity 2 violation of H11



before

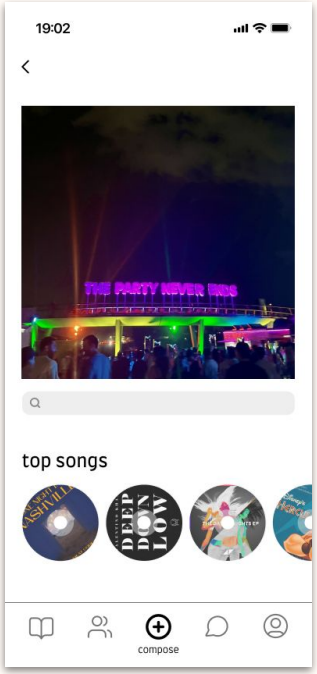


after

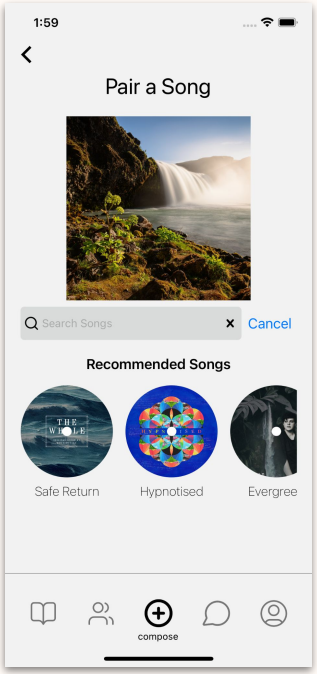
Robustness:
The light grey background color was difficult to discern from the white background. **The button is much more contrasted and easier to see**. Every button on the app is styled consistently with the one displayed in the after screenshot

added header text to screens

severity 2 violation of H8



before

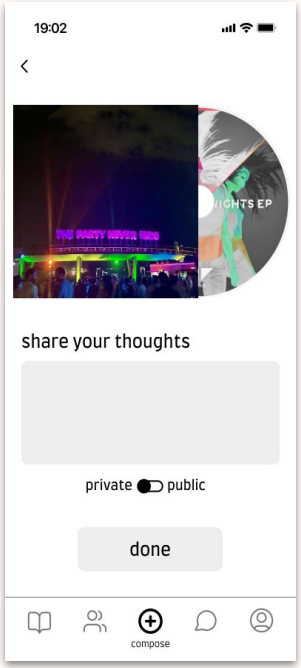


after

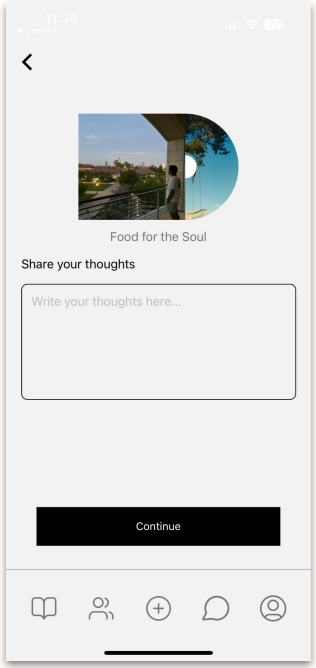
Robustness:
Test users noted that it was difficult to identify the purpose of each screen. Adding the header text makes it **clarifies the intended functionality of each screen**

added placeholder text to text inputs

severity 2 violation of H11



before

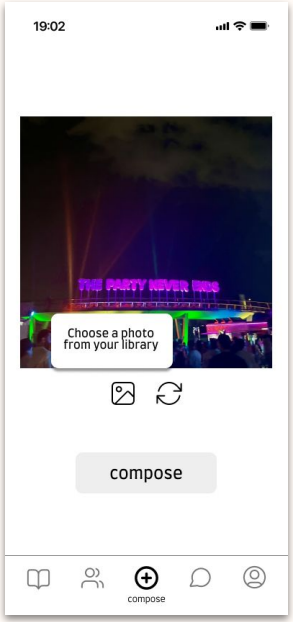


after

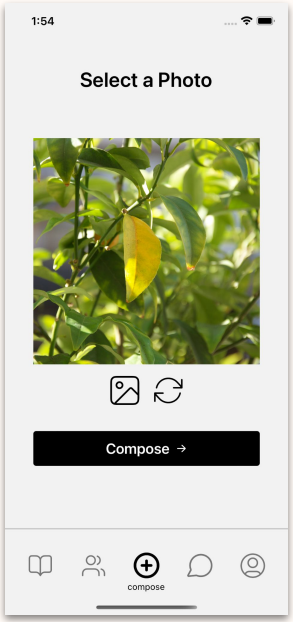
Robustness:
Users were unsure what the function of each text input was. Placeholder text helps them more easily **identify the functionality of each text input**

removed helper hover tags

severity 2 violation of H1



before



after

Robustness:
Helper tags appeared when the cursor hovered over the buttons in the med-fi prototype. Because you can't hover over components in a mobile app, we removed them. We believe the **icons are intuitive enough** for the user to understand their functionality, and instead, supplemented the icons with headers and an onboarding tutorial

04

prototype implementation status

tools being used

Github

central repository where we regularly push code to – chose because of familiarity and robust version control features

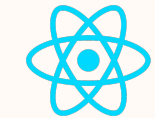
VSCode

produces code files, view the terminal, and debug errors from Expo – chose because of user-friendly interface and seamless integration with GitHub

Expo Go

allows us see our code changes and simulate our app's UI – chose because it allows for previews on real devices without requiring native builds

React Native



the framework we are programming the app on – chose because it is the most popular and widely used mobile app development framework

Spotify API

handles music recommendations and music catalog – chose because of its straightforward API and extensive music catalog

Firebase

handles database, user authentication, and image storage – chose because it is an easy to use database with efficient image storage solutions

implemented features

General:

- Stack Navigator: navigation for all main pages (Home, Feed, Compose, Profile, etc)
- Firebase database to store user data. Users can create an account and post pairings that on that account (recorded in database)
- Firebase storage to store image and song URIs
- Spotify API to fetch user's top songs and support search functionality of Spotify's catalog

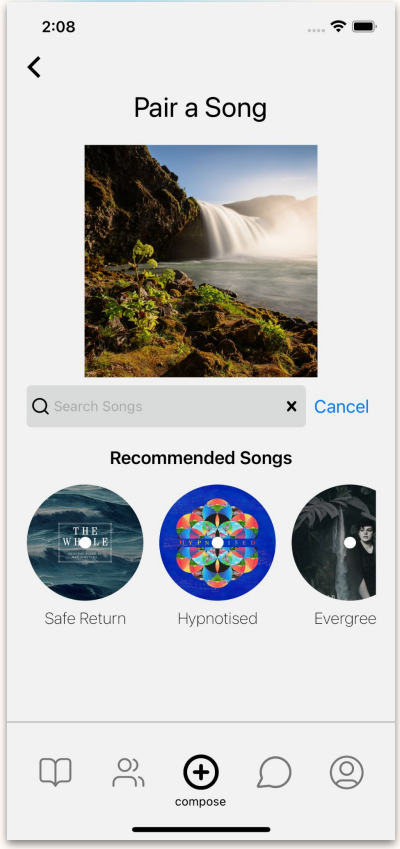
Initial Screens:

- Splash screen with logo and value proposition
- Onboarding flow with backend user account creation. Customizable profile picture and username
- Tutorial flow with overview of creating a photo/song pairing

implemented features cont.

Simple Task: Pair a song to a photo

- *Photo selection screen*
  - User is prompted with random photo from their device's photo library
- *Pair Screen*
  - User is prompted with their top songs and has the option to search for a song to pair to the photo
- *Preview Screen*
  - User may optionally add a journal entry to the pairing before sharing it with friends



Compose screen - Task 1

unimplemented features & plans to finish

- unify the UI
    - common fonts, button sizes, border radius, etc.
- create “home” feed with curated collections of photo/song pairings
- implement “messages” section
    - sending direct messages & photo/song pairings to friends
- enable private/public functionality in posting
- implement a feed with friends’ photo/song pairings
- make collections that categorize posts based on music genre
- Build out profile screen – fetch user’s posts from database
- *add animations and transitions**

aiming to finish implementation by the end of the weekend!

**unsure about implementing thing one*

wizard of oz techniques

01 2-factor SMS verification

Because the focus of this class is on user experience and design, we wanted to prioritize developing the front end portion of the app over 2FA. If we were to launch this app to the real-world, we would prioritize 2FA for security reasons

02 Logging in and out of the app

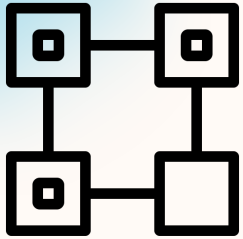
We currently are simulating a user logging into the app via a state variable. We plan to implement logging in over the weekend since active account state is not vital to implementing our simple task flow.

03 Sharing photo-song pairings

We are currently implementing our complex task through backend WOO techniques. We will be saving pairings and populating chats between users based on “sender” and “recipient” UIDs rather than building out a full messaging platform using libraries such as Twilio

We are proud to say that the majority of the app’s backend is fully functional!

hard-coded aspects



01

User Profile Page

While the profile picture and username are displayed dynamically, the app currently doesn't fetch the post data for the user. The profile page does not have to do with our simple task, so we left this implementation for after the checkpoint

02

Searching for songs using the user's Spotify account

We could integrate Apple Music API, but because Spotify and Apple APIs store songs differently, we would need to create a method of storage that is compatible for both APIs, which we think goes beyond the scope of this class

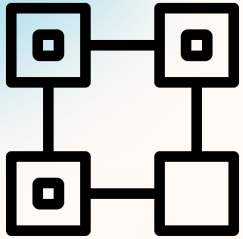
03

Fake users to simulate social network

We will be hard-coding fake users into our real database to set up a social network without having to onboard users beyond the developer accounts

hard-coded aspects cont.

these are aspects that are currently hard-coded but will be fully implemented as we continue to build the app



04 Error Messages

We currently display a pre-written error message when there is an issue with firebase or spotify fetching.
→ *Next steps: Over the weekend, we will take the actual error message that Firebase / Spotify gives back, parse it, and return it to the user*

05 Default Profile Picture

Because we haven't fully tested our profile picture uploading function, we initialize a user's profile picture to a default user icon circle
→ *Next steps: We plan on further testing the profile picture functionality to ensure users can select their own profile photo from their camera roll*

issues & questions

01 User collections

Our rough plan for making collections of posts based on genre is to utilize Spotify's API to pull the genre tags of each song and do it that way, but we would need to modify the way we are storing the pairings in our database

02 Searching for songs requires the user to have a Spotify account

We could integrate Apple Music API, but because Spotify and Apple APIs store songs differently, we would need to create a method of storage that is compatible for both APIs, which we think goes beyond the scope of this class

issues & questions cont.

03 Unable to play music in the app

Currently, our implementation does not support playing songs in the app. We know that every Spotify track has a 30 second preview url. Our plan is to read up on the Spotify API this weekend to find a way to play the 30 second preview in the app so users can better identify songs

04 No current algorithm for curating collections of pairings for users

We have created a fully detailed plan of making collections of posts based on their genre. Our rough plan is to utilize Spotify's API to pull the genre tags of each song and do it that way, but we would need to modify the way we are storing the pairings in our database

05

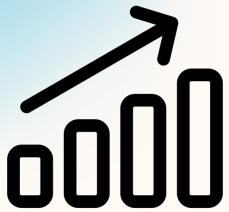
prototype demo

demo time

06

summary

progress & next steps



1. **Major progress has been made in backend functionality**
    a. database, storage, and Spotify API integrated
    b. account creation and posting is functional
2. **Need to implement moderate and complex tasks**
3. **HE feedback will continue to be incorporated as we flesh out more the of the app**
    a. Onboarding flow, delete buttons for post drafts, and other small fixes will be added

07

appendix

Figma link

→ revised med-fi prototype