Dynamic Web Design ARCH11258
ESALA: ESA The University of Edinburgh 2019-20
Submission 2 — Beta prototype

SongBox

Lucas (Zitian) Wang *s1965998*
Damyana Stoyanova *s2006146*

Links

Beta prototype on Playground: 🔗 https://playground.eca.ed.ac.uk/~s2006146/songbox

Alpha prototype on Playground: 🔗 https://playground.eca.ed.ac.uk/~s1965998/songbox

Video walkthrough: 🔗 https://www.dropbox.com/s/d7j8gsfr0b5kq6x/SongBox.mp4?dl=0

Report Link: 🔗 https://www.dropbox.com/s/77m699zewb4iyj7/SongBox.pdf?dl=0

GitHub repo: 🔗 https://github.com/jobmine/songbox

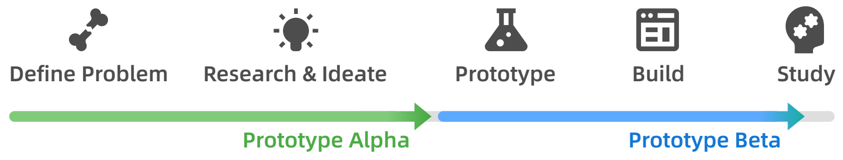


Fig. 1. Our work flow towards beta prototype

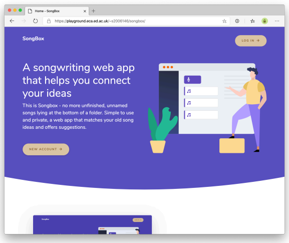
Content

After the first submission, we got a bunch of great feedback and ideas from tutors and testings. To accommodate the feedback, we redefined the scope of this project for our final submission. This report could be divided into three parts:

- **Instruction**: a detailed walkthrough and guidance of how to use SongBox,
- **Project**: the roles of the individuals in the group, motivations, methodology, and execution, and finally
- **Critique**: a brief appraisal and critique of the project.

Instruction

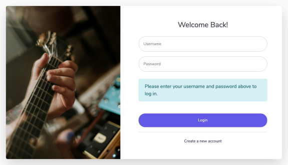
1. SongBox is a web app to support musicians and songwriters to record and write down their brainstorming ideas. To begin using SongBox, customers will land on our default product landing page. From here, potential customers could read about the products and click on New Account to sign up for an account.



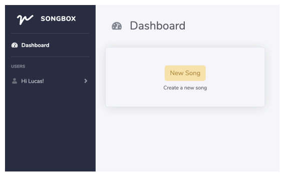
A songwriting web app that helps you connect your ideas

This is Songbox - no more unfinished, unnamed songs lying at the bottom of a folder. Simple to use and private, a web app that matches your old song ideas and offers suggestions.

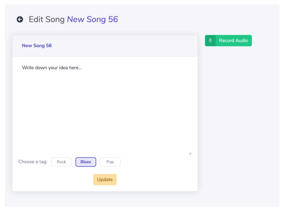
2. After signing up, the customer could sign in the account with their credential. For next time, they could also use the login button on the landing page. For demo purposes, we created a guest account, but feel free to sign up for a new account to start a new journey. Guest username: guest / password: guest1234



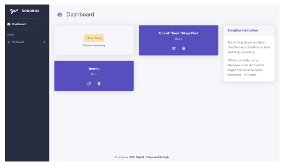
3. Then, customers will enter their dashboard, where they can take most of the actions. A new user should see a new song card, where they could start their first song idea. The dashboard also provides some general functions like sign-out and switch accounts.



4. Then, users will enter this new song page, where they take most of their creative steps, involving naming, writing down ideas, tagging, and recording. By clicking the recording button, customers could record an audio clip while writing down ideas at the same time. When everything is done, click the Update button to save it on the dashboard.



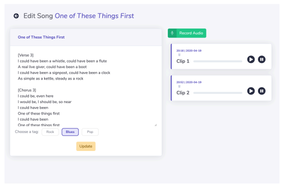
5. On the dashboard, customers could browse, modify, delete the existing songs, or create another new song. Every violet card represents a song. Click on the edit or delete icon on one of the song cards to take action.



Try writing down an idea! Use the record button to start and stop recording.

We're currently using Mediarecorder API which might not work on some browsers - IE/Safari.

6. Finally, if customers are in one of the song edit modes, they could modify or delete the content, title, tags, and audio clips on the right-hand side.



[Verse 3]
I could have been a whistle, could have been a flute
A real live giver, could have been a boot
I could have been a signpost, could have been a clock
As simple as a kettle, steady as a rock

[Chorus 3]
I could be, even here
I would be, I should be, so near
I could have been
One of these things first
I could have been
One of these things first

Project

Individual roles and collaboration

The two team members on the team are in two awkward timezones about 10 hours apart, so we mostly worked on SongBox together on the GitHub repo. With clearly discussed responsibility and collaboration needs, we separate our parts and commit to branches. We schedule weekly chats to sync on progress and merge the code to the Master. Although we collaborated on most pieces, here are the area ownership for both of us:

On Lucas(Zitian)'s plate:

- Front-end Bootstrap / CSS design and development
- Dynamic signing up / logging in functionalities
- User database setup
- Screencast recording / package delivering
- Illustrations and gif videos creating

On Damyana's plate:

- Back-end and Bootstrap front-end connection
- Songs and Clips database setup
- Dynamic creating / modifying functionalities
- JavaScript / audio interface / SVG to import custom icons
- Color palette and copywriting

Motivations, methodology, and execution

From our previous research, there's a need for a simple, straightforward songwriting app that offers suggestions on how to tag and match up old drafts. Additionally, we noticed some issues based on heuristic evaluation principles and testing with a couple of users. Some examples are:

- Registration form specifies required password length only after you've typed it in – frustrating for the user (HE)

- Record button — didn’t change states to show system status (HE & user feedback)
- Didn’t notice the song title — adjust placeholder color (user feedback)

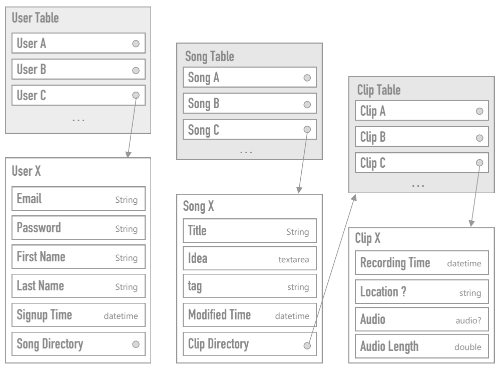
The overarching goal is still to offer a simple, functional songwriting solution made specifically for musicians. On the front-end side, problem areas were to make the interface overall more appealing visually and fit the target audience; this way, potential visitors would be drawn in, have a positive experience, and hopefully experience the app as more natural to use. Improve the affordances of buttons and overall interaction with the interface. Some example actions we took were:

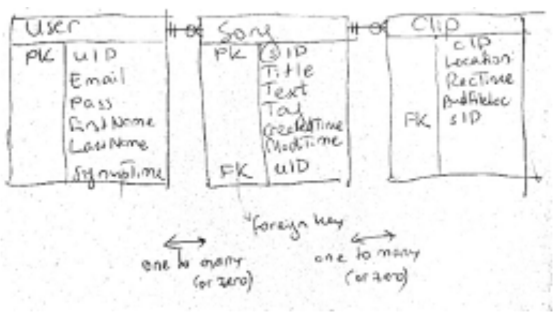
- Changing the color palette so it’s still unobtrusive, but better suited to the theme and target audience
- homepage, which required more information to entice a user — maybe gifs that show the visitor what using the app feels like - https://basedash.io/
- proper on-hover reactions of objects — like song cards
- more…

We have done many changed on the back-end as well - some examples are:

- Enable users to signup and login and see their dedicated content
- Save and load audio snippets (to an AboveRoot folder)
- Create a new clip directory for each user
- Only load clips and songs for the specific user
- more…

To execute these items, we drew the database re-constructing diagram and use software generates such schemes.





The recording features are based on official MDN documentation and Octavian Naiku's Codepen for the MediaRecorder API and a tutorial on Medium by Bryan. One of the more significant challenges was to send the audio blob, created by the Media Recorder API, to the server. Another challenge was to rename the files consequentially, according to the clipID. To solve such problems, we seek advice and feedback from the tutor, check out StackOverflow threads. Finally, we ended up using a FormData interface, appending the audio blob and all information about the recording to it, and wrote up a renaming function on top of F3's framework.

Critiques

One of the most important lessons we learned was it was easy to lose track of UX. In essence, the overall strategy views down to individual user experience using the product, when you're trying to build in functionalities. However, the silver lining is that you get a rough idea of how easy/hard it would be to implement a feature, prioritizing more effectively.

Our initial idea included being able to tag parts of the song (Chorus, Verse, Bridge, etc.) as that, combined with the style tags would make it possible to offer meaningful match suggestions of old drafts that might fit well together.

After getting tutor feedback, we realized that would involve creating a separate database table for text parts, and a button to create tagged text fields within the note-writing text element. Drag and drop functionalities for these elements, so the user isn't restricted. Additional ID records for songs and clips that define the positioning order (so interacting via drag and drop has a permanent effect in positioning). And additionally – a matching algorithm, involving filtering by certain criteria and then possibly randomly selecting up to 3 songs that match the criteria.

If we kept working on the project, here are some examples of recommendations:

- Implement the song structure elements, as they would require repeated user testing to get the interaction right (songwriting is task-oriented, so users should not be interrupted)
- Tagging clips by instruments. Widen the style tagging options.

- Drag and drop functionality: possibly implemented with Draggable JS.
- Suggestions features and additional filtering options present within the side menu.
- Encrypt the passwords (never save them in plain text). Look into other security implications.
- Restructure the app MVC architecture and use more AJAX instead of PHP templating, so the page doesn’t have to reload for each change made.
- Possibly redo the FE with flexbox from scratch, instead of the Bootstrap theme, for optimized performance.

References

John Lee's course notes http://ddm.ace.ed.ac.uk/lectures/DDM/DynamicWebDesign/

———

MDN documents

https://developer.mozilla.org/en-US/docs/Web/API/FormData

https://developer.mozilla.org/en-US/docs/Web/API/Blob

https://logicalread.com/2015/09/24/mysql-foreign-keys-mc13/

Threads from Stackoverflow:

https://stackoverflow.com/questions/15014638/recorderjs-uploading-recorded-blob-via-ajax/21055797#21055797

https://stackoverflow.com/questions/34059110/how-to-upload-file-in-fat-free-frame-work

https://stackoverflow.com/questions/40441920/saving-a-blob-to-the-server

https://groups.google.com/forum/#!searchin/f3-framework/rename/f3-framework/vbG0HycLnKQ/XDAA-vfg6TkJ

https://stackoverflow.com/questions/22640837/how-do-i-list-results-in-my-template-with-fatfree-framework-and-cortex-model

https://stackoverflow.com/questions/42705392/handling-error-in-fatfree-framework

https://stackoverflow.com/questions/30518106/fatfree-sql-error-handling/30523332

play/pause button

https://stackoverflow.com/questions/52505380/jquery-button-play-pause-audio-and-switch-images

sending audio blob to server

https://stackoverflow.com/questions/19015555/pass-blob-through-ajax-to-generate-a-file

https://stackoverflow.com/questions/15014638/recorderjs-uploading-recorded-blob-via-ajax/21055797#21055797

loading/working with multiple DB records

https://stackoverflow.com/questions/46570018/about-fatfree-framework-query

https://stackoverflow.com/questions/42614867/fat-free-framework-loading-array-of-data-from-a-query

Information from the F3 framework Google groups

https://shopify.github.io/draggable/

https://groups.google.com/forum/#!searchin/f3-framework/rename$20file/f3-framework/IVrmEgodHCw/HCK12SLL0A4J

W3 schools - Alert code html https://www.w3schools.com/howto/howto_js_alert.asp

F3 Framework website https://fatfreeframework.com/

F3 tutorial by Mark Takazs https://www.youtube.com/watch?v=XydAAp3ZF-M

Bootstrap Documentation https://getbootstrap.com/docs/4.1 and Start Bootstrap free template to start the front-end https://startbootstrap.com/themes/sb-admin-2/ MIT License Copyright (c) 2013-2019 Blackrock Digital LLC and Font-Awesome Free Icons https://fontawesome.com/license and jQuery https://jquery.org/license/

AddPipe articles & Naicu Octavian's CodePen https://blog.addpipe.com/using-recorder-js-to-capture-wav-audio-in-your-html5-web-site/ and https://codepen.io/naicuoctavian/pen/GRgzqBg

https://css-tricks.com/redirect-web-page/

https://www.freecodecamp.org/forum/t/solved-adding-font-awesome-icons-to-an-input-button/74139

https://medium.com/@thomasmarren/create-a-custom-audio-progress-bar-using-javascript-51b358811abd

https://medium.com/@bryanjenningz/how-to-record-and-play-audio-in-javascript-faa1b2b3e49b - invaluable simple example of how MediaRecorder API works

https://github.com/bcosca/fatfree/wiki/AJAX-Form-Post-example