4-Track Recorder  
A recording and mixing web application

|  |  |
| --- | --- |
|  | Glen Anderson  Dominic Cabral  Jose Flores  Ramon Meza |
|  | GUI Programming II Professor Heines |
|  | Project Proposal 1.2.2 February 4, 2016 |

# Table of Contents

Table of Contents 2

Goal statement 3

Target Audience 3

Features 3

Anonymous, registered, and authenticated users 3

Tutorial 4

Recorder/mixer 4

FX catalog 6

Mix menu 7

Future features 8

Components 8

Foreseeable Issues 9

Release Functionality Requirements 10

Project management 10

# Goal statement

Our goal is to create a web application that will allow users to record and manipulate audio in a simple and accessible way.

# Target Audience

Our target audience consists of people who are interested in sound recording. Our primary focus will be to target more experienced users with the functionality provided by the application, without alienating beginners.

# Features

## Anonymous, registered, and authenticated users

The application will be able to manage three different kinds of users; they are labeled as anonymous, registered, and authenticated users. The application user types are described in Table 1.

Table 1: User account differences

|  |  |  |  |
| --- | --- | --- | --- |
| User type | Description | Capabilities | |
| Anonymous user | A user is by default an anonymous user, an anonymous user is unknown to the application and in turn will receive a limited set of functionality. | * Able to register * Mixer access * Create new work | |
| Registered user | A registered user is known to the application, they will have the functionality of an anonymous user. To become a registered user, an anonymous user will need to register once with the application. | * Sign in * Mixer access * Storage space * Create new work | |
| Authenticated user | An authenticated user is a registered user that has logged in during their session, the application gives them the most functionality. To become an authenticated user, a registered user must sign in during their sessions. | * Sign out * Mixer access * Storage space * Create new work * Delete saved work | * Clone saved work * Open saved work * Export saved work * Inspect saved work |

## Tutorial

Due to the inexperience a user might have with technology found in our application, a user will have the option of going through a tutorial. This tutorial will be accessible to all users at any time from the menu. This menu option, though initially hidden within the menu, will be presented to an authenticated user through their first authenticated session. An authenticated user will experience a greeting during this session, that will give them an option to run the modal tooltip tutorial. If a user choses to not participate, the modal tooltip tutorial will exit and they will not be prompted again during future sessions. The tutorial scope will be the interface, application usage, and how manage work.

## Recorder/mixer

The recorder/mixer is the heart of the application; containing the four mono tracks and one stereo master track that a user will be interacting with. A proposed mixer layout is presented in Figure 1.



Figure 1: Proposed mixer layout

Each mono track, labeled as Tracks 1 - 4 in Figure 1, contain components described in Table 2.

Table 2: Mono track elements

|  |  |  |  |
| --- | --- | --- | --- |
| Component name | Label | Quantity | Description |
| Track title | Track # | 1 | Can be modified by the user. |
| Panning knob | PAN | 1 | Adjusts the stereo location of the track. |
| Equalization knobs | EQ | 3 | Adjusts the tone of treble, mid, and base range frequencies. |
| Mute button | M | 1 | Silences the track. |
| Record button | ○ | 1 | Engages the recording sequence on the track. |
| Solo button | S | 1 | Silences all other tracks. |
| Fader slider |  | 1 | Adjusts the volume of the track. |
| Volume indicator |  | 1 | Audio peak indicator. |
| Effects Slot | FX | 1 | Container that can be filled up by an effect from the effects catalog. |

The stereo master track, labeled as Master in Figure 1, contains the components described in Table 3.

Table 3: Stereo master track

|  |  |  |  |
| --- | --- | --- | --- |
| Component name | Label | Quantity | Description |
| Master title | Master | 1 | Cannot be modified by the user. |
| Location indicator |  | 1 | Indicates position in song using the format: Minutes: Seconds. milliseconds |
| Stop button | ██ | 1 | Stops the selected track. |
| Play/pause button | ►║ | 1 | Plays the selected track. |
| Rewind button | ◄◄ | 1 | Rewinds the selected track. |
| Fast forward button | ►► | 1 | Fast forwards the selected track. |
| Fader slider |  | 1 | Controls the sum of all four mono tracks. |
| Volume indicator |  | 2 | Audio peak indicator. There is one each for the left and right audio channels. | |
| Save button | SAVE | 1 | Allows a user to save their mix |

## FX catalog

The user will be able to use an effects catalog, named FX catalog, to choose effects for each of their tracks. These effects will be self-contained modules, as all controls that are needed to modify the effect will be present on the module’s graphical interface. The proposed FX catalog is presented in Figure 2.

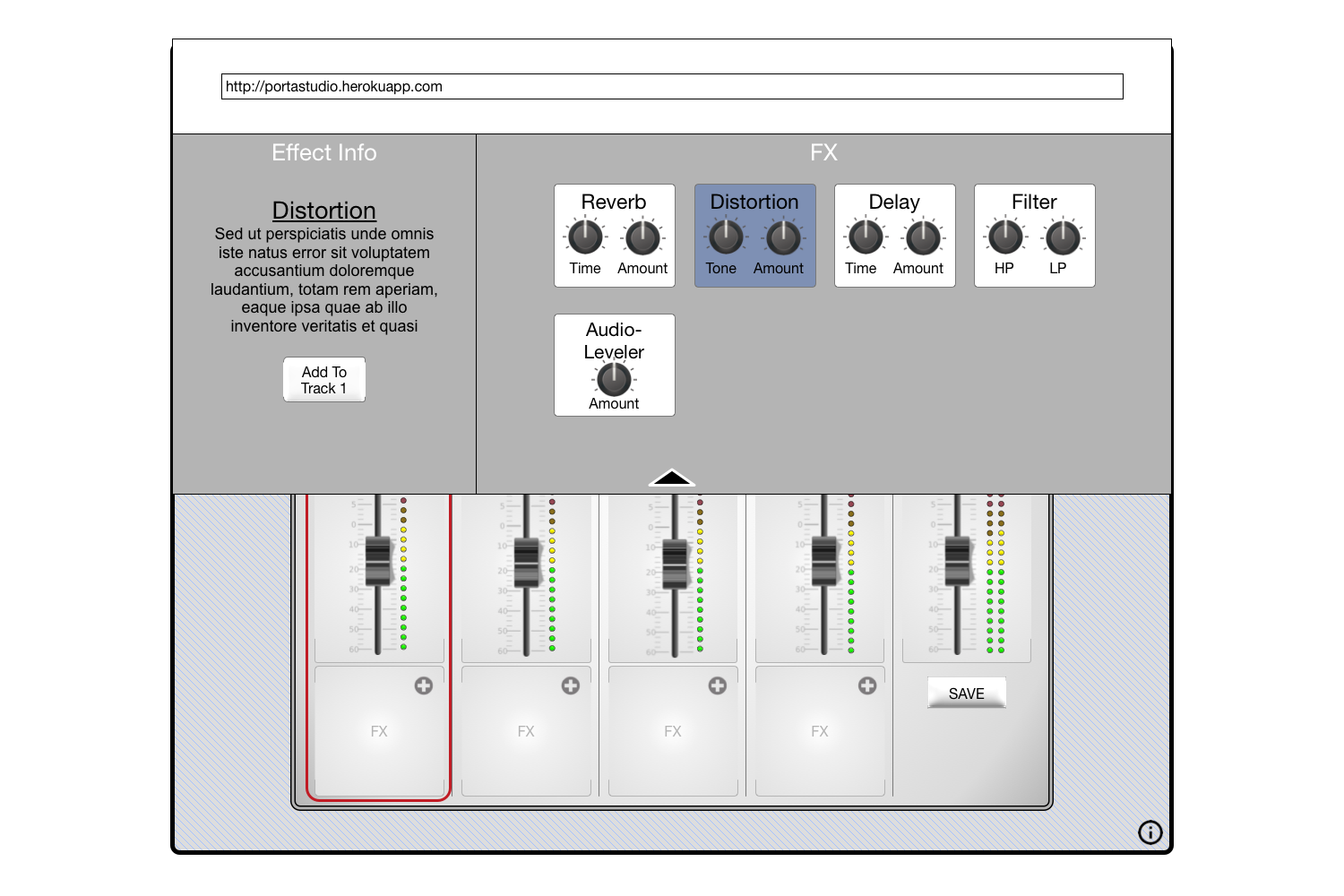


Figure 2: FX catalog

The user interface for the effects catalog is split into two panes. The left pane will hold the description of that effect as well as a selection button. The right pane will hold a list of effects, described in Table 4.

Table 4: List of effects

|  |  |
| --- | --- |
| Effect name | Description |
| Reverb | An effect where the sound produced is made to reverberate slightly. |
| Distortion | An effect that alters the sound in the harmonic (tone, timbre) domain. |
| Delay | Delay is an audio effect which records an input signal to an audio storage medium, and then plays it back after a period of time. The delayed signal may either be played back multiple times, or played back into the recording again, to create the sound of a repeating, decaying echo. |
| Filter | Remove sections of the audio spectrum. |
| Audio-Leveler | A compressor reduces the level of an audio signal if it exceeds a certain threshold. |

## Mix menu

An authenticated user that would like to manage their work, will do so in the Mix menu. This menu will be their storage space interface, and will allow an authenticated user to inspect, create, delete, clone, open, and eventually export work. The layout of the Mix menu as depicted in Figure 3 will be similar to that of the FX catalog in Figure 2. The Mix menu will be split into two panes, the left pane will display information about the selected work and hold buttons to the actions that can be performed on them. The right pane will hold a list of all past pieces of work, as well as a button that will allow the user to create a new piece of work.

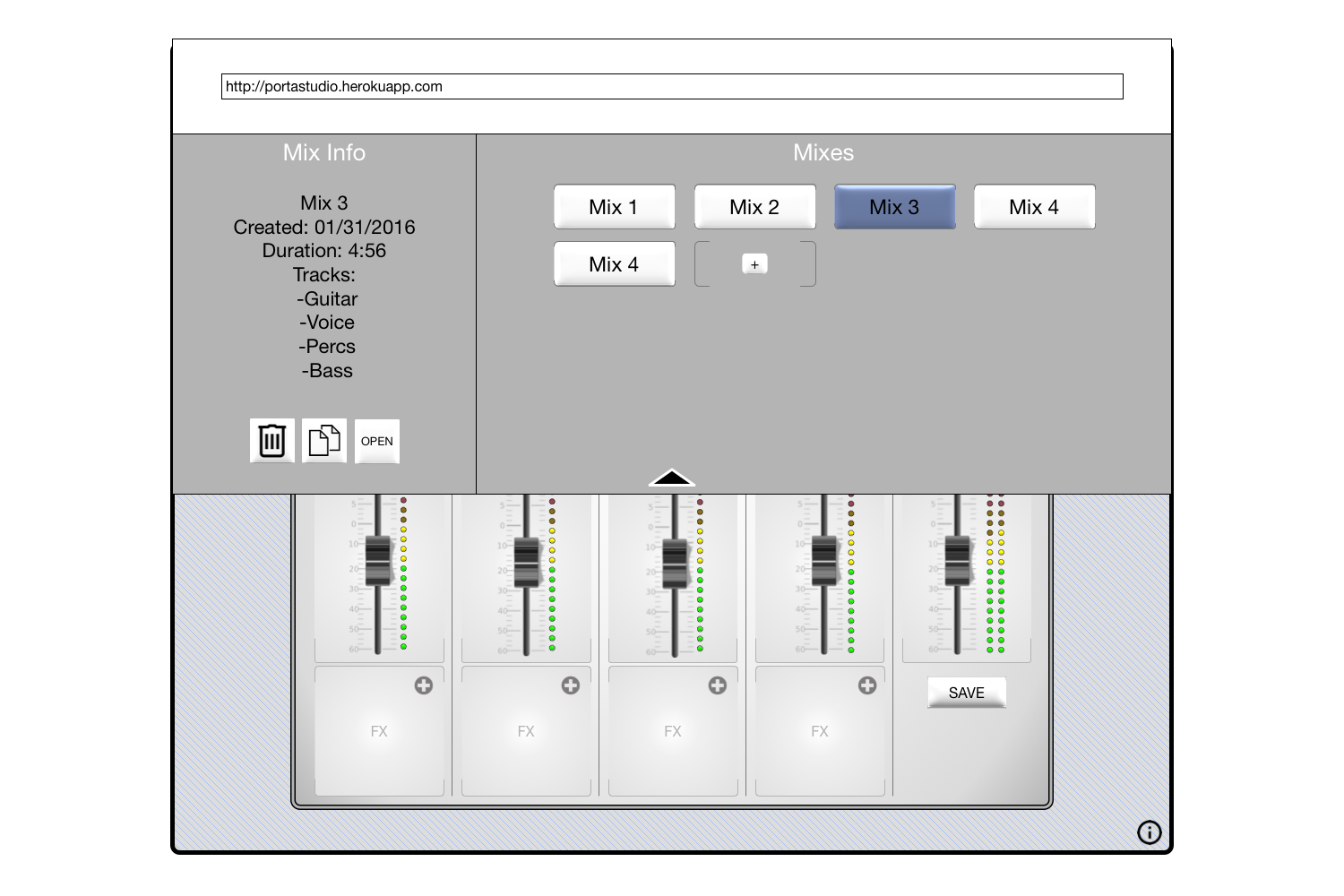


Figure 3: Mix menu

## Future features

If the application is complete and time allows, we will implement the additional features listed in Table 5.

Table 5: Future features list

|  |  |
| --- | --- |
| Feature title | Description |
| Export | Will allow authenticated users to export saved or ongoing work directly to music sharing applications, such as [SoundCloud](http://www.SoundCloud.com). |
| Sign up expansion | Will allow anonymous and registered users to register and sign in respectively with external website OAuth methods, such as those provided by twitter, Facebook, and Google. |

# Components

Most of the components that we will implement exist in some form and will be from the external component dependency list found in Table 6 that will be integrated into our project. The only item that will need to be built from scratch are the knobs that are shown in Figure 1 and described in Table 2 and Table 3.

Table 6: Dependency list

|  |  |
| --- | --- |
| Name | Description |
| AngularJS | Will be our JavaScript client side framework. |
| Bootstrap | Application layout will use the Bootstrap framework, which will keep our project responsive across devices. Bootstrap will also provide unique design features, such as form inputs. |
| Express | Is our server framework. |
| jQuery | jQuery will be used to keep our work clean and readable, it is also a dependency of jQueryUI. jQuery will reduce development time for the knobs, as they have to be made from scratch. |
| jQuery UI | jQueryUI an extension of jQuery has a lot of the base components that are shown in Figure 1 and described in Table 2 and Table 3. Using jQueryUI will eliminate a lot of development time. For example the fader sliders and visualizers described in Table 2 and Table 3 can be generated from the default slider and progress bar |
| MongoDB | We will be using MongoDB to handle our user accounts. Our database will allow us to hold all of the information that we need, such as credentials and saved work. |
| NodeJS | Our Server runs on NodeJS. |
| Web Audio API | Sound processing will be handled by the Web Audio API. Web Audio API allows us to easily capture, create, and manipulate audio using JavaScript and HTML5. |

# Foreseeable Issues

Not everyone has experience with music recording equipment and this became apparent during our first scrum meeting, because of this we are expecting our application to have a steep learning curve for beginners. Inexperienced developers will therefore need to put more effort in to learn about the application concepts before actually integrating components.

Web Audio API is new to all of the developers and getting started with the API is going to be a challenge. Thankfully there are tutorials (http://code.tutsplus.com/tutorials/the-web-audio-api-what-is-it--cms-23735) and documentation available (https://developer.mozilla.org/en-US/docs/Web/API/Web\_Audio\_API).

# Release Functionality Requirements

Our product will require a decent amount of functionality to be implemented for us to consider it a successful release. The one feature we absolutely need to implement is the recorder/mixer. The recorder/mixer needs to allow the user to record audio from their microphone, manipulate it in some way, and to export the audio as an MP3 so the user can download it to their machine. If we can accomplish this, then we have met our requirements of a functional release.

# Project management

We’ve decided to take an Agile approach to developing our application. We believe daily communication, weekly meetings, and an integrated work management system will allow us to work together efficiently. Our management tools and links are listed in Table 6 and our schedule in Table 8.

Table 7: Project management tools

|  |  |
| --- | --- |
| Name | Description |
| Heroku | Our deployment server, set to deploy when our *heroku* GitHub branch is pushed to. http://portastudio.herokuapp.com |
| GitHub | Our source control. https://github.com/gui2project/5trackportastudio |
| Slack | Used for daily team communication and application status updates. Slack has been integrated with Trello, GitHub, and Heroku notifications to keep everyone up-to-date with the project. https://gui2project.slack.com |
| Trello | Our task board, all tasks and milestones are being tracked here. https://trello.com/b/8YeXe2iu/deadlines |

Table 8: Development schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Milestone | Story | Sub Task | Responsibility |
| 01/31 | Project management software | Slack | Setup Account | Cabral |
| Integrate Heroku | Flores |
| Integrate GitHub |
| Integrate Trello |
| Heroku | Setup domain | Flores |
| Integrate GitHub |
| GitHub | Setup organization | Flores |
| Make master branch |
| Make heroku branch |
| Trello | Setup deadlines board | Flores |
| 02/02 | Project source control | Application skeleton |  | Flores |
| Development environment |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| 03/03 | Usability Test Alpha |  |  |  |
| 03/10 | Alpha Version |  |  |  |
| 03/31 | Beta Version |  |  |  |
| 03/31 | Usability Test (class) | A team run usability test |  | All |
| 04/14 | Review Beta Test Feedback |  |  |  |
| 04/25 | Class Presentation |  |  | All |
| 04/28 | Final Submission |  |  | All |

.