Bassoon Fingering Finder

Jacob Flynn, Sasha Solomon, Max Stillwell

May 10, 2012

Contents

Executive Summary

We designed a web application that helps bassoonists of all skill levels find and learn new fingerings with an easy to use interface.

Bassoon fingerings can be very difficult to learn, especially because there can be multiple fingerings for one note. Beginners and professionals alike have the need to lookup fingerings for a note, usually in a specific passage, to be sure they are using the appropriate fingering. Bassoonists need to be able to access this tool via a personal computer, a tablet (iPad, etc.), or smart-phone (Android, iPhone, etc.), and would need to be able to easily search for fingerings for a specific note. They also need to be able to add new fingerings, edit and delete fingerings, and 'like'/'dislike' fingerings, as well as view fingerings based on skill level.

This application will be beneficial to many bassoonists because it gives them an easy way to find fingerings for specific notes based on their skill level. It also gives bassoonists a social medium for sharing fingerings, as well as suggestions and preferences for certain fingerings.

Background

Motivation

The Bassoon Fingering Finder Team is very interested in evolutionary computation and artificial intelligence, as well as mobile applications. This project will not only help bassoonists, but also allowed the BFF team to explore methods for using evolutionary computation and artificial intelligence in a full software project. Also, because the application needed to be compatible across multiple platforms, it also gave the team experience with compatibility and mobile applications.

Need

Currently bassoonists have very few options when it comes to finding fingerings quickly (which many times imperative during a concert or recital). There are books listing hundreds of standard fingerings, as well as an mobile fingering application with limited capabilities. Bassoonists need a way to access fingerings quickly and be able to search for fingerings they need. Also, because bassoonists have no social medium for discussing preferences or suggestions for fingerings, or receiving help for certain fingerings, it would also be helpful to have an application that not only helped bassoonists find fingerings, but also allowed them to communicate with fellow bassoonists.

Benefits

This project will benefit bassoonists, from beginners to professionals, and will provide a way for bassoonists to communicate through a bassoon specific application. Bassoonists will be able to add, edit, delete, view, and search for bassoon fingerings with an easy to use graphical interface, as well as manage fingerings they have added, 'like'/'dislike' fingerings, and access the application via personal computer or mobile devices.

Problem Definition

Goals and Deliverables

Create an interactive web application that would include

- fingerings add/edit/delete/view/search
- sources for fingerings
- examples in literature
- comments
- likes/dislikes
- a graphical user interface that makes it easy to enter in fingerings
- accessibility via mobile devices as well as personal computers
- users that must login to be able to add/edit/delete fingerings

Specifications and Constraints

- Ruby v. 1.9.3
- Rails web framework v. 3.2.0
- Postgres database v. 9.1.3
- Hosted through Heroku
- Viewable via PC/tablet/smart-phone
- Supports Chrome/Firefox/Safari/Opera

Project Plan

Tasks and Schedule

1. Infrastructure
    - (a) Database
        - i. For users
        - ii. For note/fingering combinations
    - (b) Users
        - i. Authentication
        - ii. Basic user
            - A. Administrative user
        - iii. Email verification
        - iv. Time zone specification
        - v. Skill level
2. Web page
    - (a) Information organization
    - (b) Professional appearance
3. Application
    - (a) Fingering chart
    - (b) Scale for notes
    - (c) Input of a note
    - (d) Input of a fingering
    - (e) Store a note/fingering combination
    - (f) Approve a note/fingering combination
    - (g) Search a note for a list of associated fingerings
    - (h) Sort list of associated fingerings by rating, based on skill level
    - (i) Add a series of notes
    - (j) Search a series of notes
    - (k) Like/Dislike a note/fingering combination

Team Responsibilities

Jacob

Jacob's primary task was the infrastructure. He worked to get the user authentication, time zone specification, and skill level functioning. He setup the ability for users to be able to view and edit their 'profile' information, containing their skill level, email address, etc. Once those tasks were done, he moved to work on bug reports and assist with the application section when needed.

Sasha

Sasha's primary task was implementing the mailer and search capabilities. The mailer was used to automatically email users when they registered for the site, as well as when they forgot their password (the mailer would email a new temporary password). Implementing the search involved querying the database for appropriate fingerings, rating fingerings, and sorting/prioritizing fingerings. She created an algorithm that would appropriately rate fingerings based on their like/dislike scores that would appropriately weight fingerings. This algorithm then took into account the skill level of users and prioritized fingerings that had skill levels close to the user's skill level. Thus, when users searched for fingerings of a specific note, the resulting fingerings were ordered to show users fingerings that were rated highly and that are appropriate for the user's skill level. Sasha also designed and completed the BFF poster for the Engineering EXPO.

Max

Max's primary task was the JavaScript, HTML5, & CSS related parts. He mainly worked on the HTML5+JavaScript canvas for the note/fingering entry, editing, and display. He also worked on the browser side validation JavaScripts for the various forms on the website. He worked on adapting a free CSS template for use on the site and added various CSS rules for things like tabs and tables. Lastly he handled filtering most bug reports to their respective developers, i.e. filtering user bugs to Jacob and search bugs to Sasha.

Concepts Considered

Implementation Model: Web application vs. Fully Mobile

The initial project proposal requested a mobile application be created for iPhone/iPad. The benefits of a mobile application are a more complete control of the mobile device for better optimization, and the potential to charge for the application. The benefits of making a Web application are a significantly extended user base, as well as being able open up the choice of programming languages.

Programming Language: Python/Django vs. Ruby/Rails

Originally, we considered using python with the django web framework to develop the application. Both languages are scripting languages that lend themselves to web development. All members of the team had experience working with Python and Django previously on the software engineering project for CS383/384. However, Ruby/Rails is ubiquitous in web development currently and would be an opportunity to learn a new language and web framework. While keeping with a language similar to those the team is used to working with.

Applet Language: Java vs. Flash vs. JavaScript+HTML5

Java and flash are both often used in web development to build applets and therefore are well supported on most modern PC browsers with up to date plugins. However, mobile browsers this is not the case with Java applets unable to run on Android and iPhone, and Flash only running on newer versions of the Android OS. JavaScript+HTML5 is fairly new but runs on any browser including mobile devices albeit with varying levels of support for various features, and requires no extra plugins to install and keep up to date.

Validation Methods: In-house vs. Devise

Because the application was fairly small and specific, the team decided in-house validation may be easier and better suited to the application. However, as the application continued to develop, it became a much larger application than originally modeled. The validation plug-in, Devise, provides built-in validation tools that are easy to integrate with new applications. Some of the primary features we looked at, and needed are:

- Basic users with a user name, password, and email
- Administrative flag
- Basic functions for New, Show, Edit, and Delete

Search Results Rating Algorithm: Ratio Rating vs. Confidence Interval Rating

The simple rating algorithm involved finding the ratio of 'likes' and 'dislikes' of a fingering (i.e. likes divided by likes plus dislikes). However, this rating algorithm could possibly rate fingerings inappropriately. For example, if fingering one had 100 likes and 50 dislikes, while fingering two had 1 like and no dislikes, fingering 2 would be rated higher than fingering one, even though fingering 1 should be rated higher.

Concept Selection

Implementation Model: Web application vs. Fully Mobile

The team had little to no practical knowledge of the programming language used for iPhone development, objective C. They also believed the extended user base, allowing for all mobile devices as well as computers, would be a great benefit to the application. Therefore Web application was selected.

Programming Language: Python/Django vs. Ruby/Rails

After each team member wrote a small application with Ruby/Rails it was decided to go with Ruby/Rails. It was clear that Ruby has syntax very similar to that of Python, and that Rails presented many features we wanted during our pervious development with Django.

Applet Language: Java vs. Flash vs. JavaScript+HTML5

Since the application was to be mobile device compatible JavaScript+HTML5 was chosen for the language of the applet that was to handle the entering, display, and editing of bassoon key fingerings.

Validation Methods: In-house vs. Devise

We made a few attempts to use Devise in our application. Each of these attempts was met with more problems that solutions. It appeared we would spend more time working on fixing the Devise conflicts that we would simply building the features from scratch.

Search Results Rating Algorithm: Ratio Rating vs. Confidence Interval Rating

The team decided to use a more accurate rating algorithm that take into account the volume of likes/dislikes and how well received the fingering is (i.e. if it is ‘controversial’). The algorithm used is as follows:

$$(\hat{p} + z^2_{\frac{\alpha}{2}} \pm z_{\frac{\alpha}{2}}\sqrt{(\hat{p}(1-\hat{p}) + z^2_{\frac{\alpha}{2}}/4n)/n})/(1 + z^2_{\frac{\alpha}{2}}/n)$$

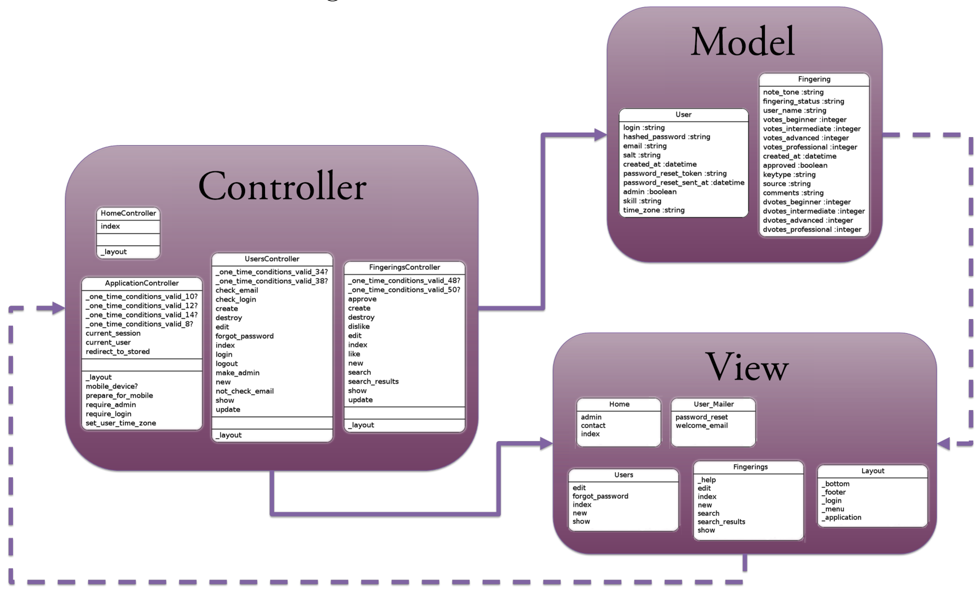
Where $\hat{p} = \frac{1}{n}$ and $z = 1.96$ (for a 95% confidence interval).

System Architecture

Conceptual Design

Conceptually, the web application would be based on a Model-View-Controller architecture:

Figure 1: MVC Architecture



- The user interacts with the user interface (e.g. adding fingerings).
- The controller handles the input event from the user interface, and converts the event into an appropriate user action understandable for the model.
- The controller notifies the model of the user action (e.g. adding the fingering to the database).

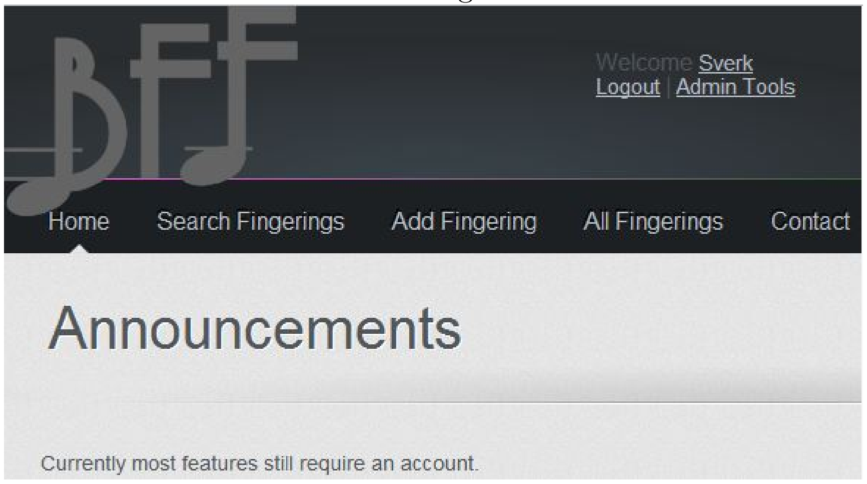
A view queries the model to generate an appropriate user interface (for example the view lists the shopping cart's contents). The view gets its own data from the model. In some implementations, the controller may issue a general instruction to the view to render itself. In others, the view is automatically notified by the model of changes in state (observer) that require a screen update. The user interface waits for further user interactions, which restarts the control flow cycle.

Components and Integration

Components

The main site is shown below in both the mobile and the full site views. The site has locations for basic users to input new fingering/note combinations, view and edit their profile, and search for fingering/note combinations.

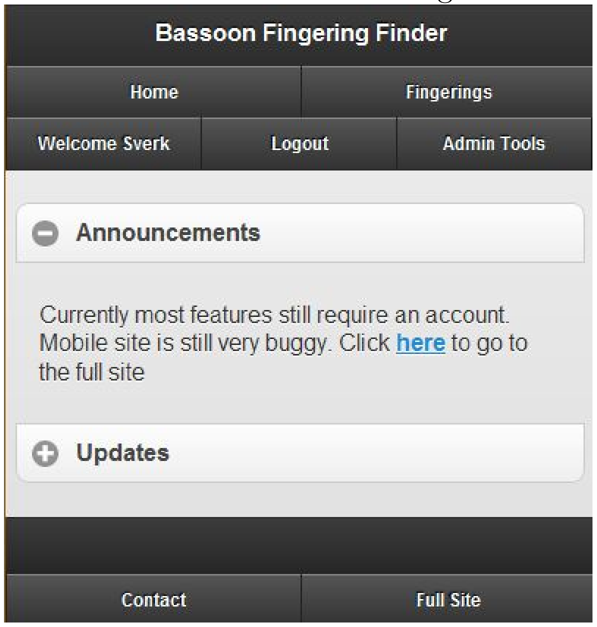
Figure 2: Full Site View



Announcements

Currently most features still require an account.

Figure 3: Mobile Site View

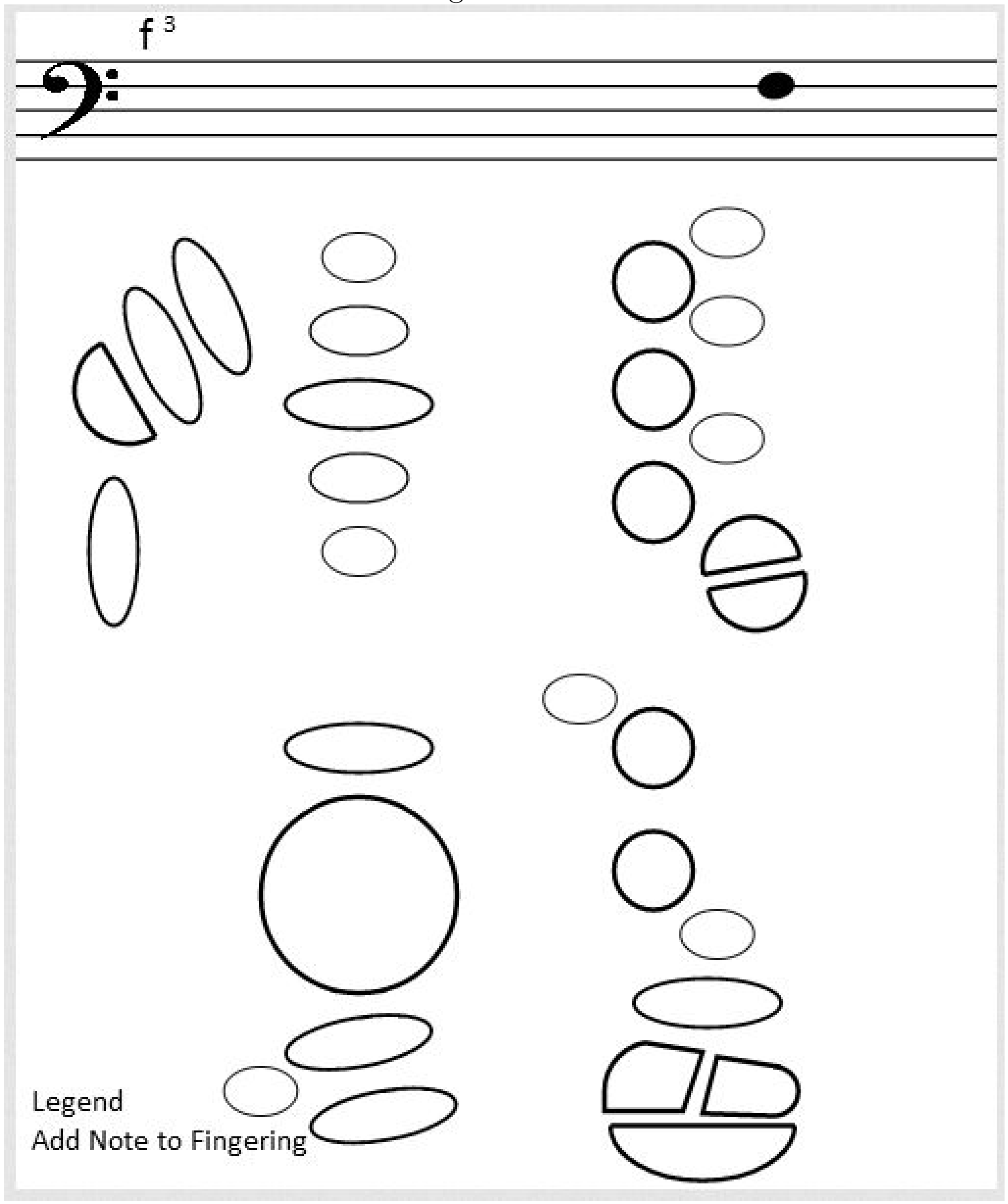


Currently most features still require an account. Mobile site is still very buggy. Click here to go to the full site

The administrative view of the site is only slightly different. A link is added to allow administrators to view all users. This allows them to view/edit/delete and set users as administrators. There is also a page to view all fingerings. This allows administrators to approve new fingerings.

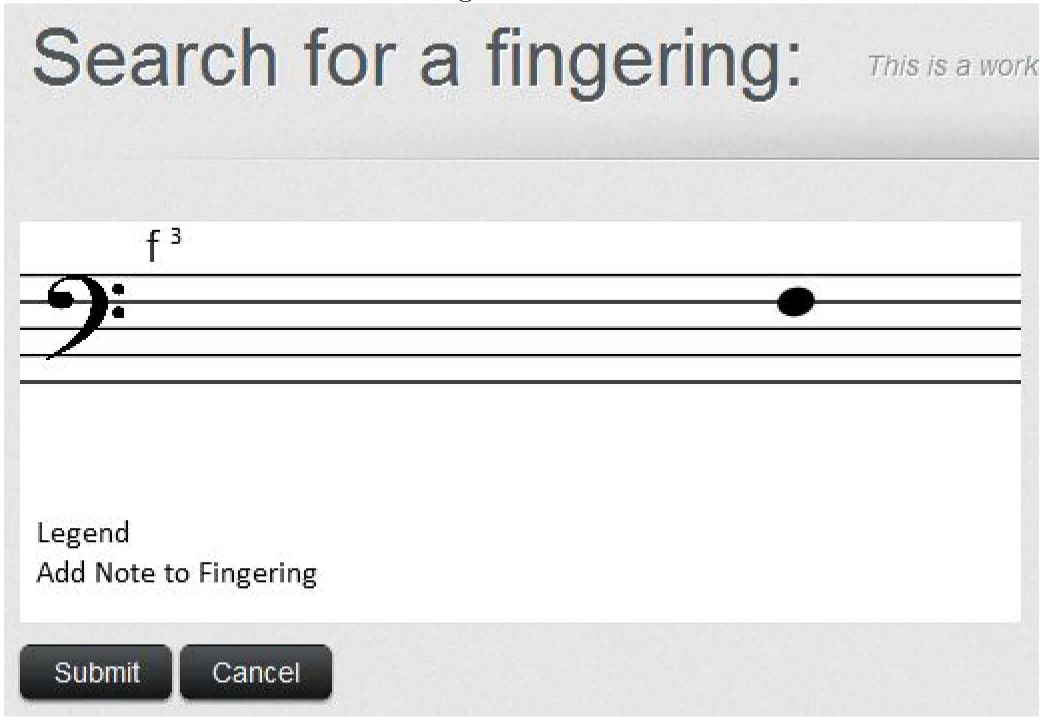
The initial step in the application is to add a new fingering. You can see the staff at the top to set which note this will be, and the fingering chart to set which keys should be pressed, and how.

Figure 4: Full Site View



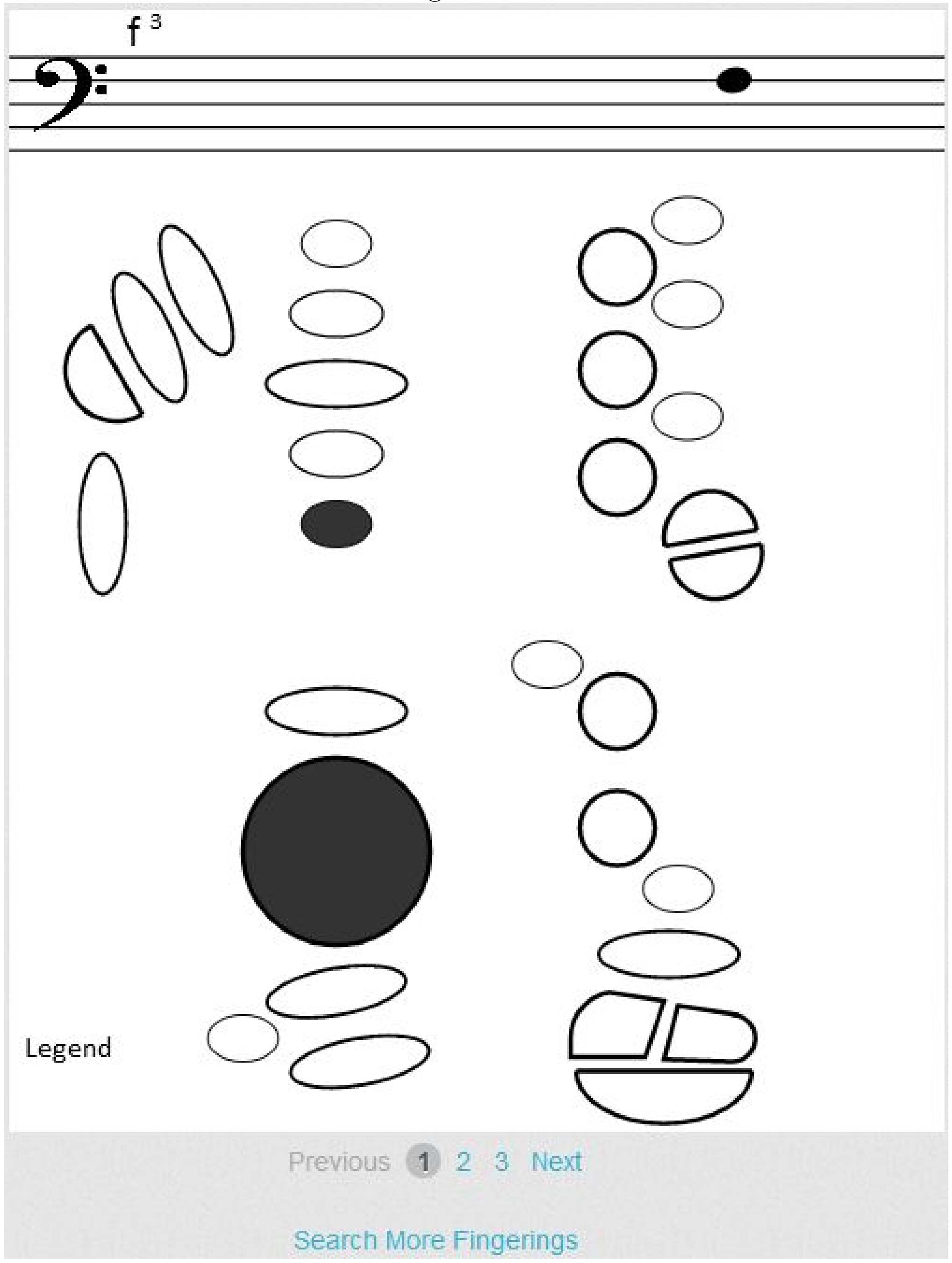
The next step is to search for that fingering. First you see just the staff for a note. This will search the database for any note/fingering combination containing that note.

Figure 5: Full Site View



It will then display a sorted list of the note/fingering combinations, and provide a next button to access the rest of the notes.

Figure 6: Full Site View



Integration

The features shown and explained above are all integrated using a PostGres database. The database has two main sections, a list of users, and a list of note/fingering combinations. The application will pull this information off the database depending on which page it is on. The page also decides how to display the information, using a CSS to enforce a similar style across the pages.

Results from Tests/Analysis

The initial beta group, consisting primarily of our sponsor Dr. Susan Hess, response was reasonably positive. They submitted several bug reports, and a few feature requests but overall provided fairly positive feedback about the features that were functioning and fully implemented.

The response remained positive after we moved to a larger test group including students and associates of our Dr. Hess. While Dr. Hess continued to be the primary tester, submitting the majority of bug reports and feature requests, the other users did not mention any major concerns about the application.

We were also able to get some feedback during the Engineering EXPO. This was also fairly positive. One of the biggest questions was if we could, or if we planned, on expanding the application to other instruments. We also had a number of non-bassoon players take a look at the application and while they were initially confused about the layout of the fingering chart, they appreciated the application for the problem it solves.

Future Work

Work for this project is intended to continue next semester in order to implement the current application on a mobile platform. Though the current web application is usable via mobile browsers, a mobile application would provide the flexibility and flow specific to mobile devices.

In addition to a mobile version, the web application version would be continued to include features available in the premium version of the web application. These premium features would include, but is not limited to the following:

1. Being able to hear the sound of each note as it is selected.
2. Searching more than just standard fingerings, (i.e. the fingerings searchable on the site will only be basic, and in order to view many alternate fingerings one would have to pay for the app)
3. A mobile optimized version of the site.
4. Input of multiple fingering/note combinations.
5. Searching of a fingering to find the note associated, in addition to searching the note for the fingerings.

The implementation of a mobile application, as well as a premium web application, would require a full semester to complete and could be implemented by a future Computer Science Senior Design Team.

Appendices

Software Requirements

Web Browsers	
Description	Be viewable on most web browsers including mobile web browsers.
Notes	• None.

Application Requirements

Show Fingering/Note Combinations	
Description	Show a fingering combination for a note or a series of notes.
Notes	• None.

Enter Fingering/Note Combinations	
Description	Ability to add new fingering/note combinations.
Notes	• None.

Edit Fingering/Note Combinations	
Description	Ability to edit existing fingering/note combinations.
Notes	• None.

Remove Fingering/Note Combinations	
Description	Ability to remove existing fingering/note combinations.
Notes	• None.

User Requirements

Fingering Request for Notes	
Description	Ability to select a series of up to three fingerings and request a fingering progression for those notes.
Notes	• Can request alternate fingerings. • Can view example music for the fingerings. • Can sort fingerings by hole coverage or expertise level. • Can "Like" or "Dislike" the given fingerings. (Requires Login) • Can submit a new fingering for the chosen note progression. (Requires Login) – Can attach example music to the new fingering. (Requires Login) – Can set the expertise level of the new fingering. (Requires Login)

Administrative User Requirements

Add New Fingering	
Description	Ability to add a new fingering for a given set of notes. (Requires Login)
Notes	• None

Remove Existing Fingering	
Description	Ability to remove a new fingering from the database. (Requires Login)
Notes	• None

Update Fingering	
Description	Ability to update and existing fingering in the database. (Requires Login)
Notes	• None.

Approve User Added Fingering	
Description	Approve a fingering added by a user for use in the database. (Requires Login)
Notes	• None.

Deny User Added Fingering	
Description	Deny a fingering added by a user. (Requires Login)
Notes	• None.

Freeze User Like/Dislike Options	
Description	Ability to prevent users from Like/Disliking specific fingerings or all fingerings. (Requires Login)
Notes	• None.

Unfreeze User Like/Dislike Options	
Description	Ability to allow users to Like/Dislike specific fingerings or all fingerings. (Requires Login)
Notes	• None.