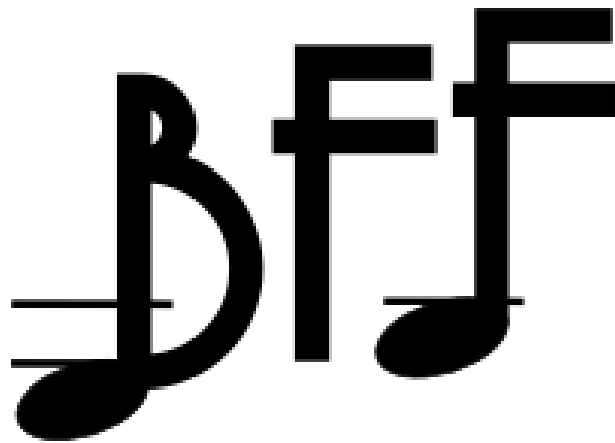


Bassoon Fingering Finder

Jacob Flynn, Sasha Solomon, Max Stillwell

April 20, 2012



Contents

Executive Summary	4
Background	4
Motivation	4
Need	4
Benefits	5
Problem Definition	5
Goals and Deliverables	5
Specifications and Constraints	5
Project Plan	6
Tasks and Schedule	6
Team Responsibilities	6
Concepts Considered	6
Implementation Model: Web application vs. Fully Mobile	6
Programming Language: Python/Django vs. Ruby/Rails	6
Validation Methods: In-house vs. Devise	6
Search Results Rating Algorithm: Ratio Rating vs. Confidence Interval Rating	7
Concept Selection	7
Implementation Model: Web application vs. Fully Mobile	7
Programming Language: Python/Django vs. Ruby/Rails	7
Validation Methods: In-house vs. Devise	7
Search Results Rating Algorithm: Ratio Rating vs. Confidence Interval Rating	7
System Architecture	8
Conceptual Design	8
Components and How Integrated	8
Results from Tests/Analysis	8
Future Work	8

Appendices	9
Software Requirements	9
Application Requirements	10
User Requirements	11
Administrative User Requirements	11

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 smartphone (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 basoonists 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/smartphone
- Supports Chrome/Firefox/Safari/Opera

Project Plan

Tasks and Schedule

Team Responsibilities

Concepts Considered

Implementation Model: Web application vs. Fully Mobile

Though the project proposal suggested the model be implemented via a web application, our sponsor voiced the need to have the application be functional on mobile devices as well. Though it would be an interesting opportunity to build a fully mobile application, the team had no experience with building mobile applications.

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.

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 modelled. The

validation plug-in, Devise, provides built-in validation tools that are easy to integrate with applications.

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

Because the sponsor still wanted a mobile application in addition to a web application, the team decided to implement a mobile compatible web application to cater to users of both PC’s and mobile devices.

Programming Language: Python/Django vs. Ruby/Rails

However, the team decided that gaining experience working with Ruby on Rails, which is ubiquitous in web application development, would be prudent.

Validation Methods: In-house vs. Devise

However, after trying to implement Devise on partially implemented application, it became apparent that the application was too far along in development to gain use from Devise.

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:

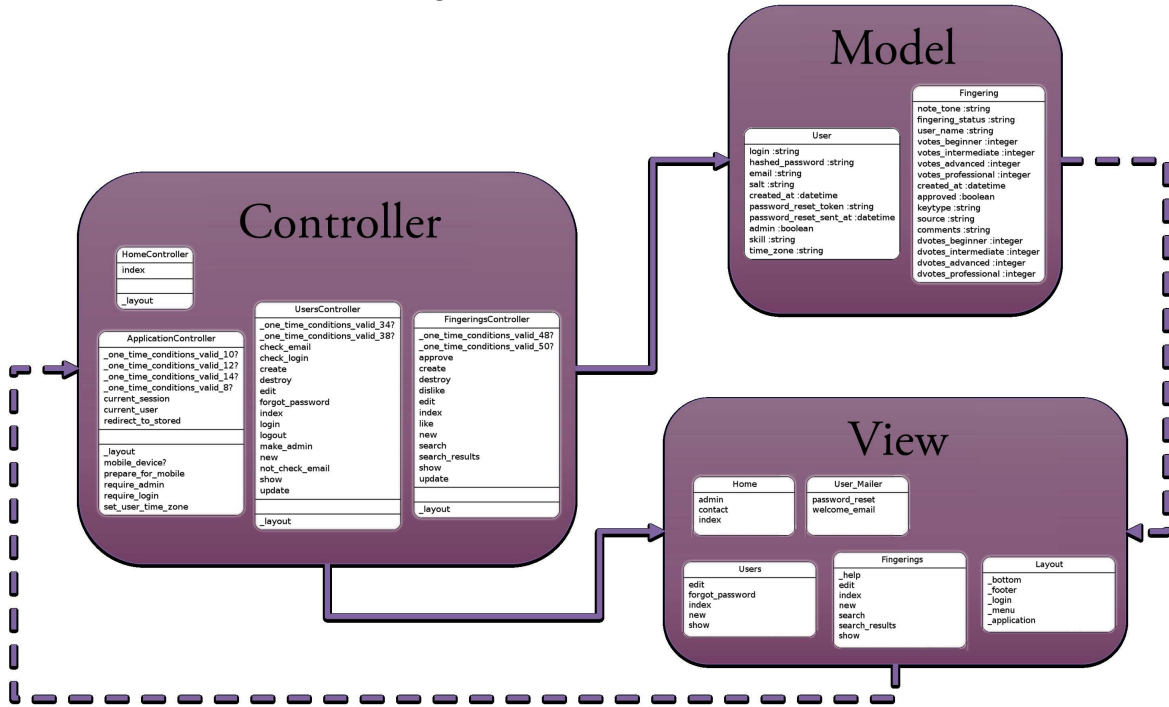
$$(\hat{p} + z_{\frac{\alpha}{2}} \pm z_{\frac{\alpha}{2}} \sqrt{(\hat{p}(1 - \hat{p}) + z_{\frac{\alpha}{2}}^2/4n)/n}) / (1 + z_{\frac{\alpha}{2}}^2/n)$$

System Architecture

Conceptual Design

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

Figure 1: MVC Architecture



Components and How Integrated

Results from Tests/Analysis

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	<ul style="list-style-type: none">• None.

Application Requirements

Show Fingering/Note Combinations	
Description	Show a fingering combination for a note or a series of notes.
Notes	<ul style="list-style-type: none">• None.

Enter Fingering/Note Combinations	
Description	Ability to add new fingering/note combinations.
Notes	<ul style="list-style-type: none">• None.

Edit Fingering/Note Combinations	
Description	Ability to edit existing fingering/note combinations.
Notes	<ul style="list-style-type: none">• None.

Remove Fingering/Note Combinations	
Description	Ability to remove existing fingering/note combinations.
Notes	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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)<ul style="list-style-type: none">– 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	<ul style="list-style-type: none">• None

Remove Existing Fingering	
Description	Ability to remove a new fingering from the database. (Requires Login)
Notes	<ul style="list-style-type: none">• None

Update Fingering	
Description	Ability to update and existing fingering in the database. (Requires Login)
Notes	<ul style="list-style-type: none"> • None.

Approve User Added Fingering	
Description	Approve a fingering added by a user for use in the database. (Requires Login)
Notes	<ul style="list-style-type: none"> • None.

Deny User Added Fingering	
Description	Deny a fingering added by a user. (Requires Login)
Notes	<ul style="list-style-type: none"> • None.

Freeze User Like/Dislike Options	
Description	Ability to prevent users from Like/Disliking specific fingerings or all fingerings. (Requires Login)
Notes	<ul style="list-style-type: none"> • None.

Unfreeze User Like/Dislike Options	
Description	Ability to allow users to Like/Dislike specific fingerings or all fingerings. (Requires Login)
Notes	<ul style="list-style-type: none"> • None.