BSharp Final Technical Design Document

Guy Cockrum - GUI
Nick Morris - Database
Robert Stewart - GUI
Ryan Tanner - GUI/Database
Cameron Keith - Database
Chris Linstromberg - Database

Table of Contents

Introduction

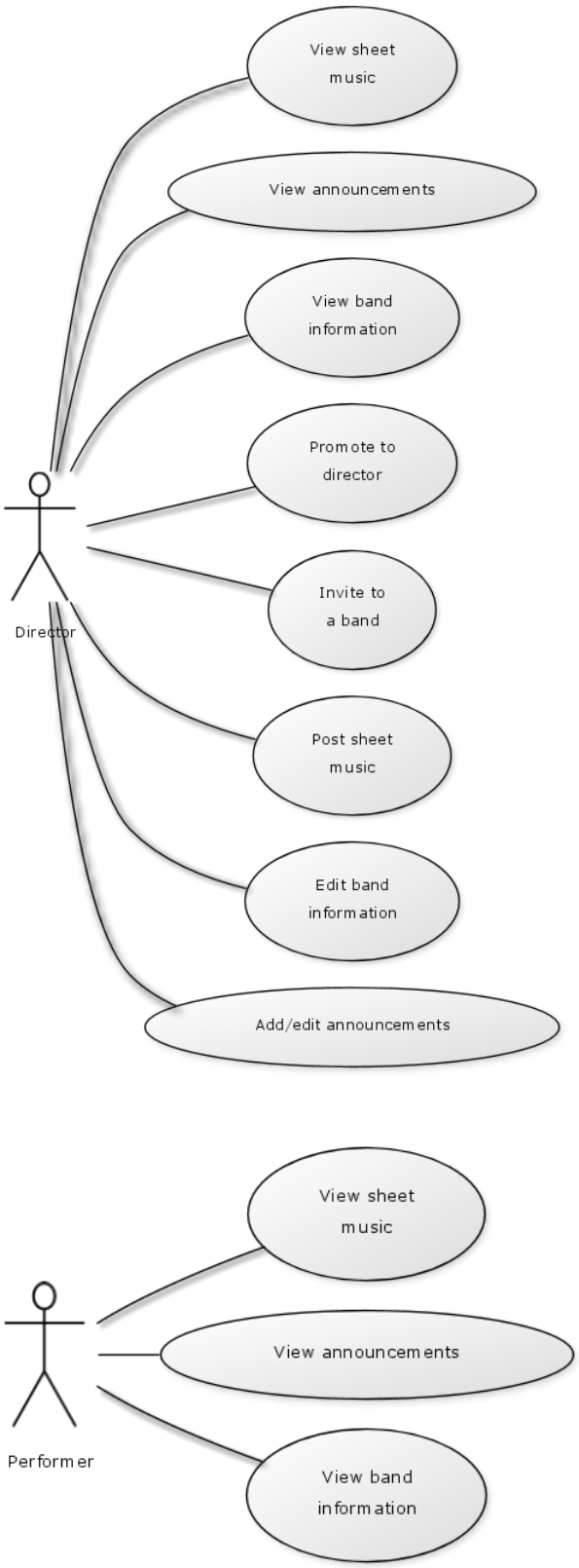
The storage of sheet music is a problem almost every major performing ensemble faces. Bands, orchestras, and choirs have libraries full of different sheet music they can use, which can include hundreds, sometimes thousands of pieces. Every time a band distributes a new piece, a band director or band librarian has to make copies for every part of a piece. Depending on the size of the ensemble, this can include hundreds of pieces of music. Naturally, this creates a great amount of extra work for the director or librarian, and in addition, it costs a significant amount of money per year.

Marching bands have an even greater cost. These tend to be the largest performing ensembles of all, and because they play football games, parades, and shows in all weather, there is a greater likelihood of damage or destruction of sheet music. Because of this, the band has to make copies more frequently, creating unnecessary spending on paper and ink, in addition to wasted time.

Here at BSharp, we have developed a website that can expedite the distribution of music. Our databases programmers include Ryan Tanner, Chris Linstromberg, Cameron Keith and Nick Morris, and our GUI programmers include Ryan Tanner, Guy Cockrum and Robert Stewart. In this report, we will describe BSharp in detail, including its software features, database model and software architecture, GUI screenshots and descriptions, our testing procedures, a reflection on the development of BSharp, and a preview of future developments.

Ryan Tanner - Databases and GUI - PHP API, Some CSS, Database Design, Javascript, Development server webmaster, MySQL script and database management, and helped get Android working with the server.
Nick Morris - Databases - Help with Concept Development, Database Design, Facebook Login research, Development Server Facebook App manager, some PHP, MySQL, uploaded sample PDFs for testing.
Chris Linstromberg - Databases - Facebook Login Integration, PHP API, Database Design and Implementation, MySQL Diagnostics, GitHub issue tracking and resolving.
Robert Stewart - GUI - CSS, javascript (selecting tabs along with hiding and making content visible), facebook login research
Guy Cockrum - GUI - CSS, Android App.
Cameron Keith - Database - Original Concept creator, Production Server Facebook App Manager, Production Server webmaster, Database Design, Creation of Database Creation SQL Script, Database Population SQL script, uploaded sample PDF's for testing.

Use Case Diagram

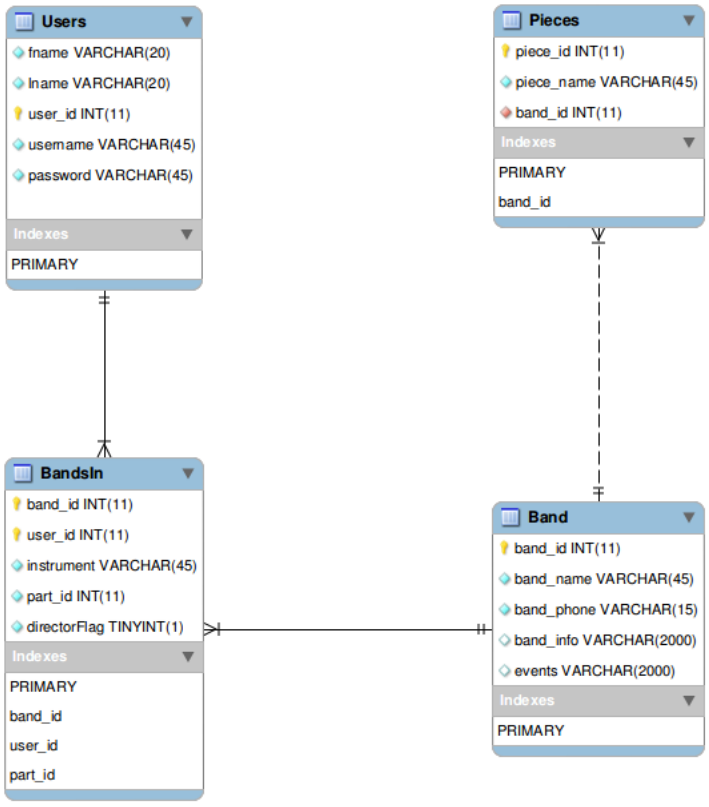


Final Software Features

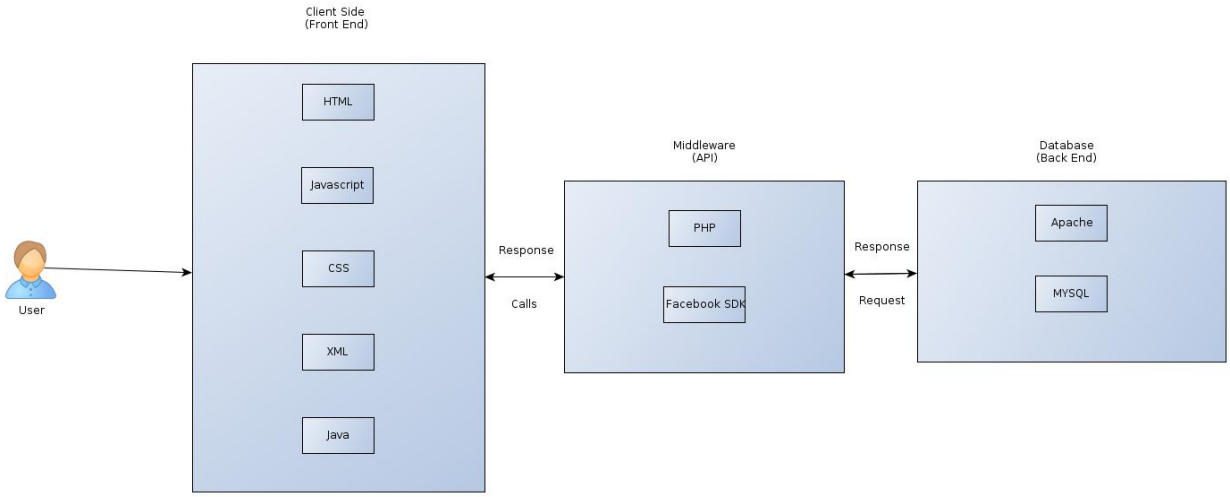
Users will be able to go to the site and either create an account with us, or use their Facebook account to log in quickly. Once they have logged in to the site, users will be able to create a band, adding in information for the band when prompted and becoming a director for that band automatically. Band directors will be able to access the site and manage their band by editing the band's information and announcements, uploading sheet music, and adding members to the band. Band directors can also mark new members as band directors. After they are added to a band, performers can view all band information and view PDFs of their part for each piece. Directors can edit band information and view all music parts for a band.

Database Model

The database's Users table contains all the information that is globally used for a user in the system. The BandsIn table references the primary key of Users, user_id. This table links a user to each band they are in, using the aggregate primary key of band_id and user_id (both individually are foreign keys). The Band table stores information for each band. The Pieces table, which contains the information about each piece of sheet music in the database, references the Band table, as the two are related by the band that owns a particular piece.

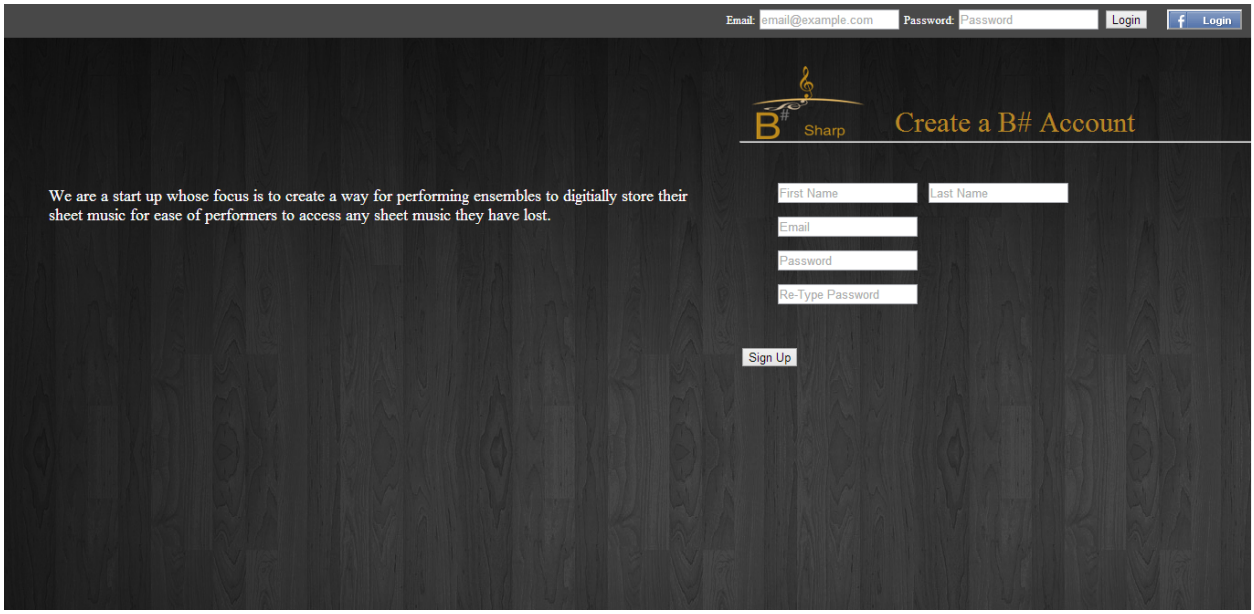


Software Architecture

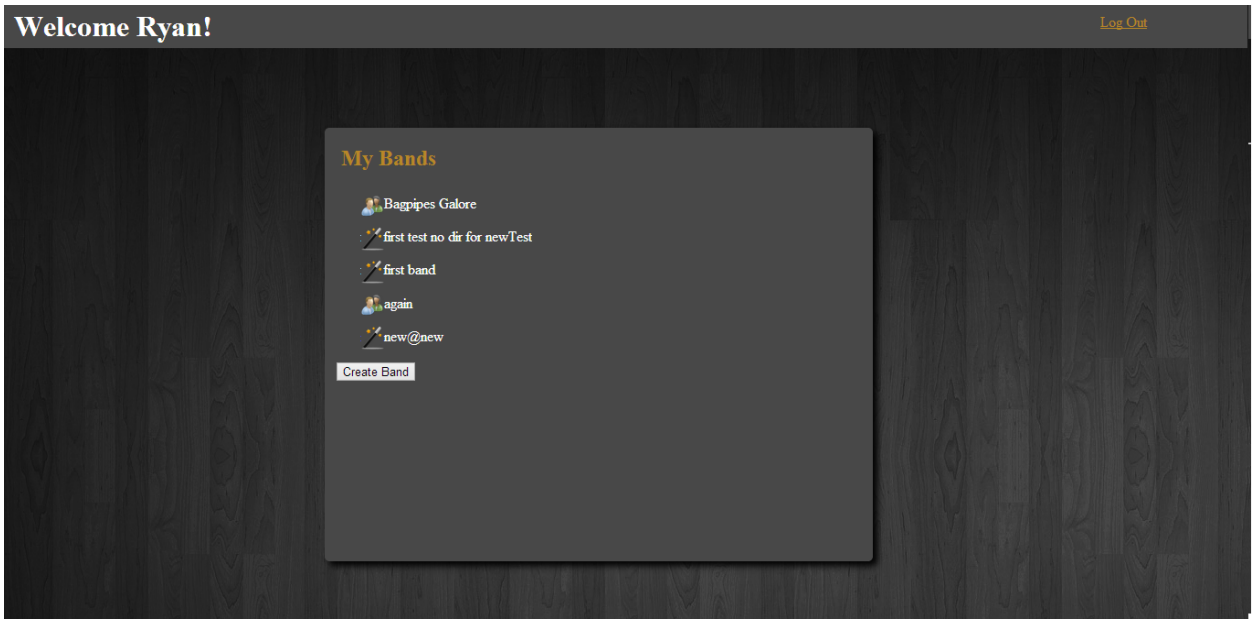


- **HTML**
  - Used as the structure of the webpages we used.
- **Javascript**
  - Used for actions that run on the page without reloading the page
- **CSS**
  - Used for formatting and stylising the webpages.
- **XML**
  - Used for the formatting, styling, and structure of the Android app
- **Java**
  - Used for the logic of the Android app, all actions the app made from HTTP requests to click listeners.
- **PHP**
  - Used to serve as an intermediary transmission step between the front end of the website and the MySQL database back end.
- **Facebook SDK**
  - Used to integrate logging into the BSharp website with a users Facebook account.
- **Apache**
  - Used to allow HTTD to run and allow our website to be accessed publicly.
- **MySQL**
  - Used as a means of storing user information, as well as information pertaining to bands songs, information, and members.

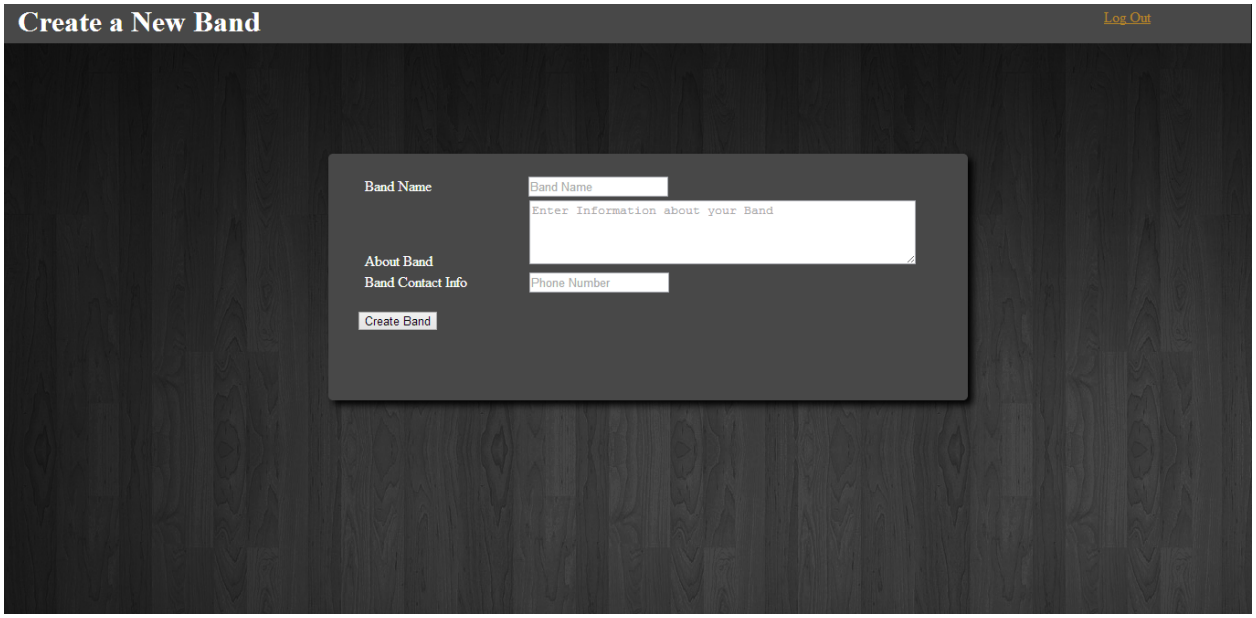
GUI Screenshots and Descriptions



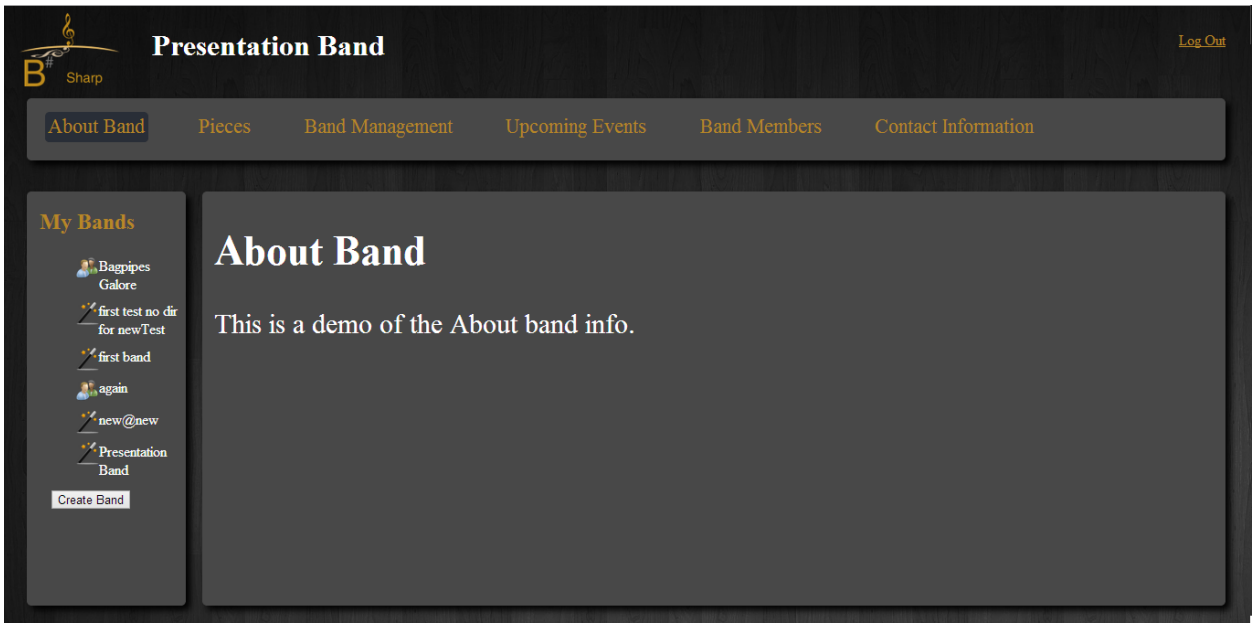
This is the home page of our site. We went for a simple design with a small “about us” section next to where visitors can create an account. Log In with Facebook or our own system is in the top right.



This is the User Page. It is a very simple page that greets the user on log in or account creation, gives the user a list of bands they are currently in, and gives the user the option to create a new band. If the user is a director of a particular band, a wand icon will show next to the band name. If the user is a member of a band, a person icon will show next to the band name.

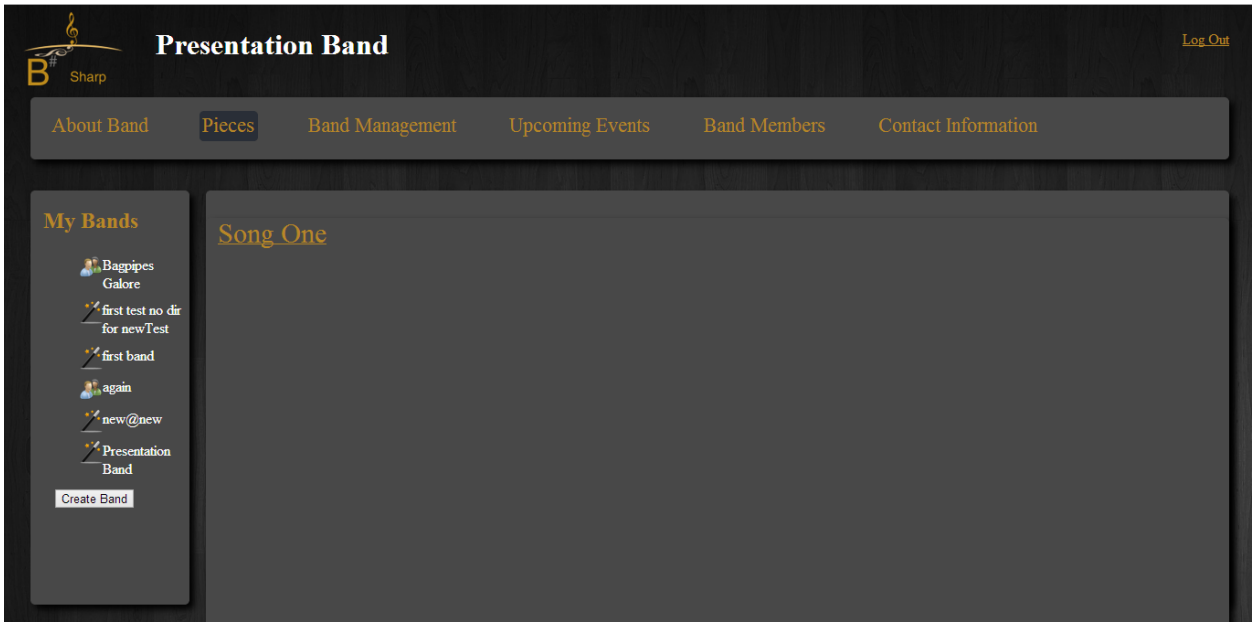


This is the Band Creation page, where a user is taken when he/she is creating a new band. The page prompts the user for a band name, a few lines of text for the “About Band” section, and a phone number for the “Band Contact Info” section.

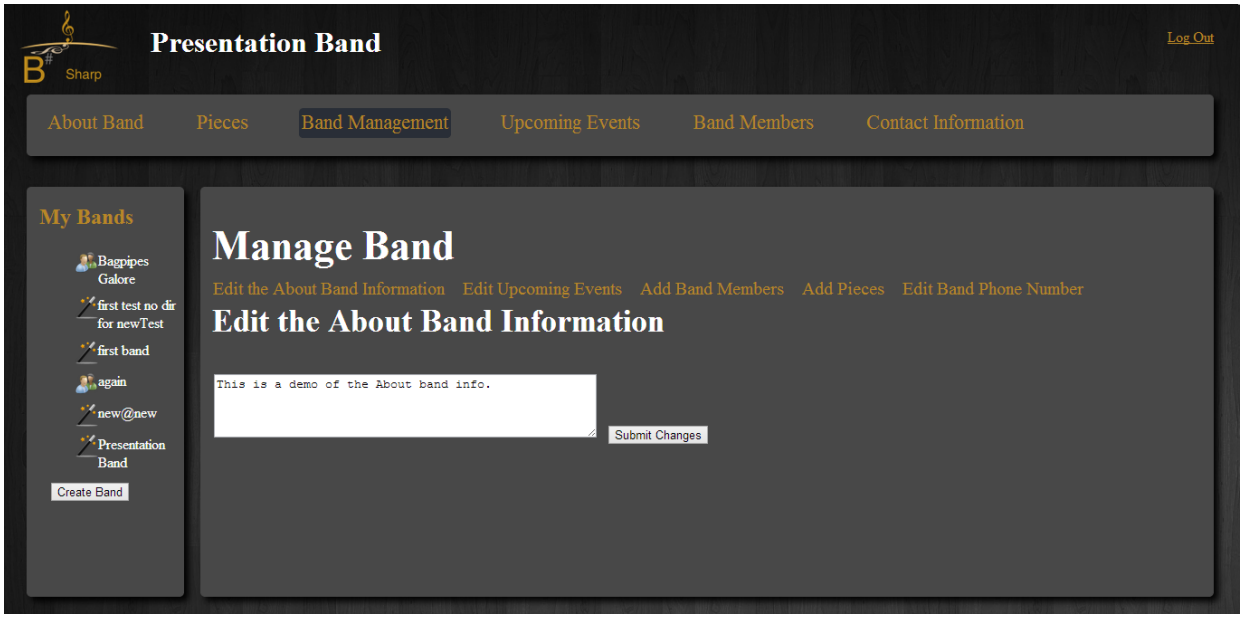


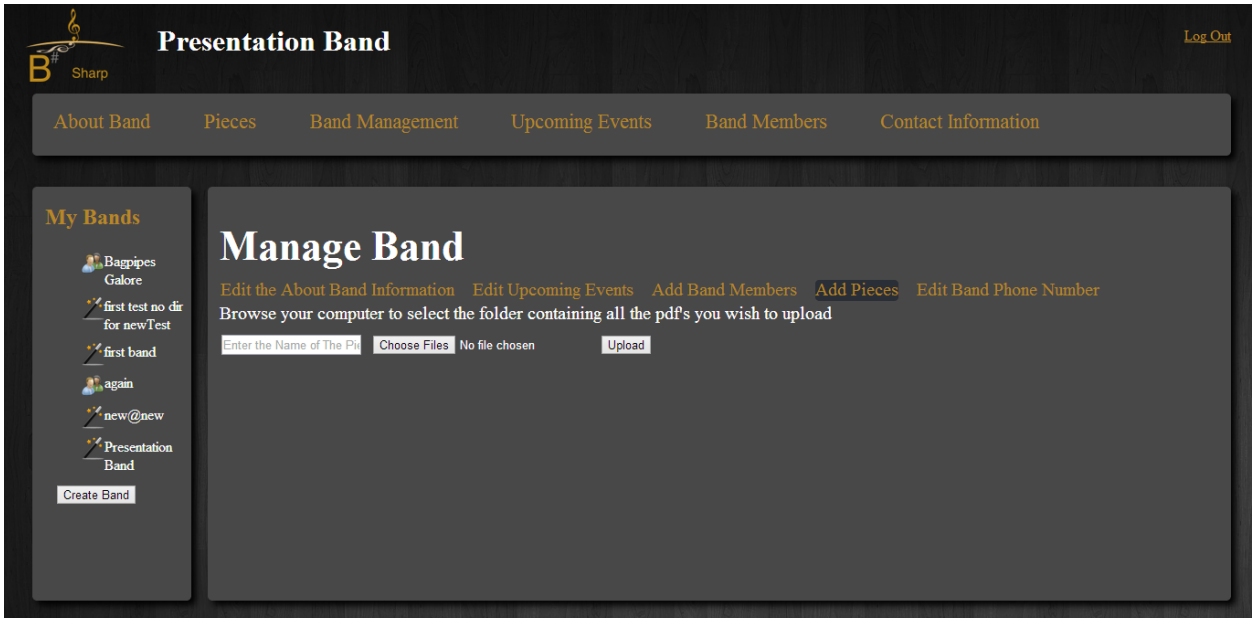
This page is the Band Page. It will always show the band name in the top left of the screen, next to the BSharp logo. It will also always show a user’s list of bands on the side for navigation between bands. Depending on whether the user is a director or a member, the page shows About Band, Pieces, Band Management (only viewable by directors), Upcoming Events, Band Members, and Contact Info. These are all clickable and display different content.

- About Band - the default information shown on the Band Page, it is the About info the directors can enter for their band to say what they want about the band.

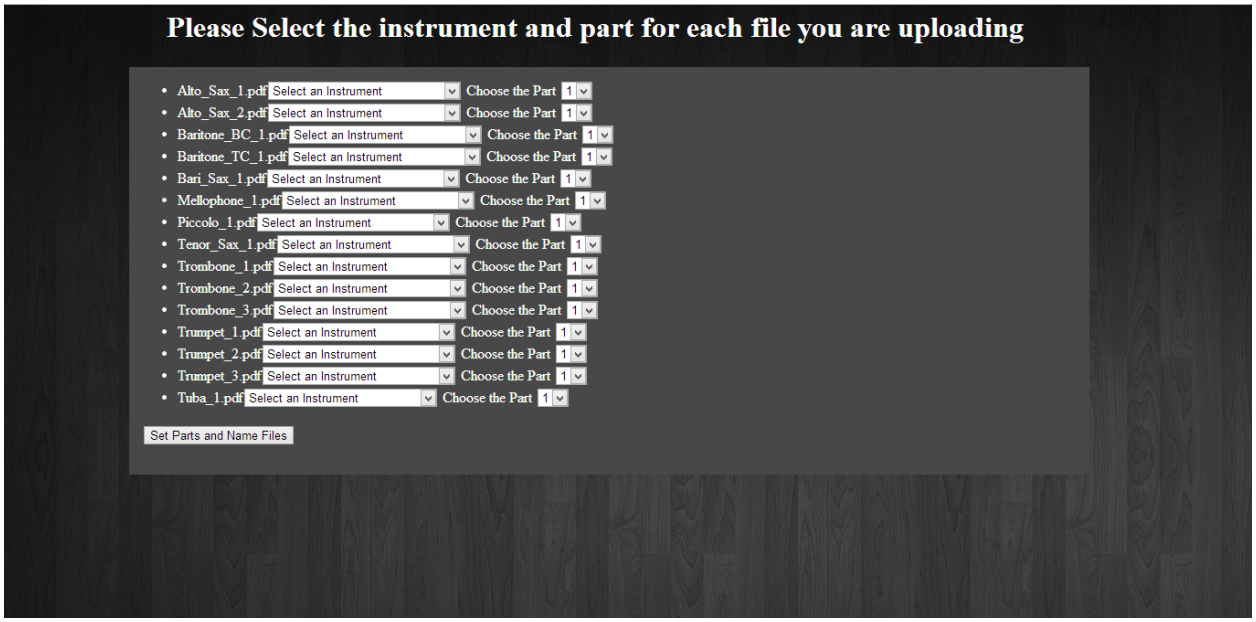


- Pieces - this page will list all the pieces that the user can view and when the user clicks a link it opens a new tab that displays the PDF of the sheet music for their part of that Piece.

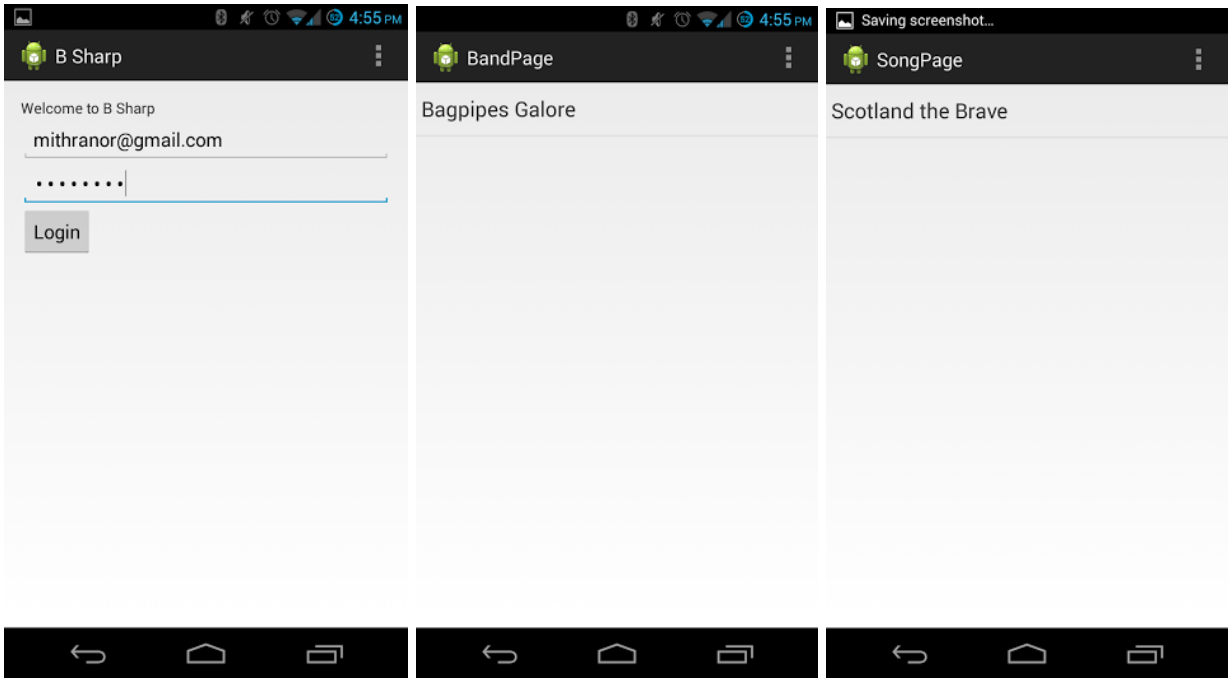




- Band Management - This is only accessible by directors. It is used for editing all the information about the band, including adding music for the band and adding members.
- Upcoming events - This section is created by the director in Band Management. It can be formatted how he/she wants, but will keep new lines properly so that the user can create a list of upcoming events.
- Band Members - This contains a list of all the members of the current band.
- Contact Info - This lists the phone number provided for the band and all email addresses associated with director accounts for this band.



This page is what is used to prompt the user for how to name the files they are uploading. It lets the user select the instrument for that part (a file) as well as the part number for that music. Once the user has set the appropriate names and parts for all the files, they can submit it and it will automatically rename and index the files they uploaded to be accessed by users.



These are the Android screens. It is a simple UI since the users can only view PDFs for the bands they are in. First is just the log in screen, second is The list of Bands the user is in, and third is a list of Songs the user can view.

Testing Practices

To test the product, members of our team tried to create different problems with the site, such as entering something not allowed in a field, trying to upload the wrong contents, and so on. We did not get a chance to have people outside of these classes test the product because of time constraints (for us and anyone who would have been able to test it) and because we have had our hands full fixing bug. However, we did have a little feedback on the general design of our website, which helped focus our development.

Our test team was very helpful. They found lots of bugs with our website that we did not notice because we knew what was supposed to happen and they were a fresh set of eyes. We found their feedback to be very useful. We took what they said about our project and tried to communicate with them to help us fix or improve our site and app. We fixed all the bugs they submitted (at least the ones that were actual issues that could be solve).

We tried to find bugs in the team's site and app we were testing but it seemed that from iteration 1 to iteration 2 there weren't many changes in their product so we didn't find many more bugs in iteration 2. Also from iteration 1 to iteration 2 there were still many bugs that went unfixed which made finding new bugs more difficult because old ones were still in the way. Also, we recently discovered that the group we were testing added features after the iteration 2 testing was complete so there are some features that we couldn't test because they had not implemented them yet.

Team Reflection

One of the main technical challenges we dealt with was Android development. Android took us quite a bit of time. It felt like we knew what we were doing but there were constant issues with implementing the Android app. There always seemed to be some small bit of code that we used just barely incorrect and then we would figure it out and move on.

Another technical challenge was implementing Facebook login. First of all, Facebook login does not apply very well to our site since users are linked to each other through bands, to solve this we just had to make it so that when a user logged in through Facebook the site would check whether or not they already had an account and if not create one for them. Also, simply getting the Facebook login to work took a lot of work, and we eventually switched methods from Javascript authentication to PHP authentication and then were able to finish it rather quickly.

One of the main things that we had to learn mostly on our own was PHP. We had a small intro to PHP in Databases but starting on the project was like going from the kiddy pool to the high dive. There was a lot about PHP that was still unclear to us going into the project so we had to figure it out as we went.

Another thing we learned along the way was how to connect to a remote database with Android. This was key in our app and probably most groups apps, and took us a surprising amount of time to figure out. Eventually we discovered we were just overcomplicating it and got it to work just fine.

We also had to learn how to display PDF both in a webpage and on Android. This turned out to be much easier than anticipated on the website since there was lots of documentation to help out. On Android we learned how to call other applications to view the PDF since integrating our own PDF viewer in the app was beyond our skill level.

If we started this project now with the knowledge we have gained we would create more versatile PHP functions to re-use instead of the very specific functions we have now.

BSharp 2.0

First, we would make the director view for a specific piece not be the normal browser view for a server file system. Right now it just takes the director to the file system view for the server for the piece of music they clicked on because we did not have time to implement security measures to keep them contained to their own files or create our own listing of the files on a different web page.

Additionally, a feature we would add to the next feature set would be to have more customizable home pages. Currently, the site only lets the director customize the “About Band” section and the “Upcoming Events” section of the band. Implementing a feature that would allow for the band director to customize their page to be themed more like their actual band would be a good feature to add. Adding a wysiwyg in place of the text fields for the “About Band” section and “Upcoming Events” would also add more versatility for the director to customize their page.

Appendix A: Data Dictionary

- Users: Stores the login and personal information for each user in the system.
    - Attributes
        - fname: Stores the first name of the user. Varchar length 20
        - lname: Stores the last name of the user. Varchar length 20
        - user_id: Stores the internal ID number of the user. Integer, range= 1-n
        - username: Stores the email address of the user. Varchar length 45
        - password: Stores the password of the user. Varchar length 45
    - Indexes
        - user_id: Primary key, used to uniquely identify users in the system
- Band: Stores the about information regarding each band in the system.
    - Attributes
        - band_id: Stores the internal ID number of the band. Integer, range=1-n
        - band_name: Stores the name of the band. Varchar length 45
        - band_phone: Stores the phone number of the band. Varchar length 15
        - band_info: Stores information about the band. Varchar length 2000
        - events: Stores information about upcoming events for the band. Varchar length 2000
    - Indexes
        - band_id: Primary key, indicates the unique band to be referenced
- Pieces: Stores the identifying information for each piece in the system
    - Attributes
        - piece_id: Stores the internal ID number of the piece. Integer, range=1-n
        - piece_name: Stores the name of the piece. Varchar length 45
        - band_id: Stores the internal ID number of the band owning this piece. Integer, range=1-n
    - Indexes
        - piece_id: Primary key, indicates the unique piece of music in the system
        - band_id: Foreign key, references Band.band_id, indicates the unique band to be referenced
- BandsIn: Links the Band table, the Users table, and the Pieces table, along with providing for lookup on parts.
    - Attributes
        - band_id: Stores the internal ID number of the band. Integer, range=1-n
        - user_id: Stores the user ID in the previously mentioned band. Integer, range=1-n
        - instrument: Stores the name of the instrument that the previously mentioned user plays in this band. Varchar length 45
        - part_id: Stores the ID of the part number this user has been assigned. Integer, any valid integer accepted.
        - directorFlag: Indicates whether or not the user is a director of this band. Tinyint of size 1, valid integers are 0-1 exclusive.

- Indexes
  - band_id: Foreign key, references Band.band_id, indicates the unique band to be referenced
  - user_id: Foreign key, references Users.user_id, indicates the unique user to be referenced