CS 4500 Group Project
Professor Dr. Keith Miller

Amanda Rawls
Stefan Rothermich
Jeffery Calhoun
James Steimel

Project Description

Music App for Gary Brandes, chair of the Music Department, as a student group project in Dr. Keith Miller's CS4500 class.

Mr. Brandes had a set of three primary requirements:

1.) Communication
    a. One way communication from the Music Department to the student
    b. Protect information about students' whereabouts by preventing unauthorized persons from viewing announcements
2.) Calendar for Events
    a. shows the events that fulfill students' attendance requirements and other important departmental events; updated by the UMSL Music Department through Google Calendars.
3.) Faculty Directory
    a. Centralize information for the student.

User Guide

Menu

The same menu appears on all pages and is visible to all users. In the full-screen web view, the menu will appear as a row of text button with two drop down menus. In the mobile or reduced-screen web view, the menu will appear as a single text link on the left and a menu button on the right.

The left most text button in all views is "UMSL Music Department". This will redirect the browser to the home page. The remaining text links that appear in the full-screen web view, or those that appear in the drop-down menu, appear in the following order: "Announcements", "Calendar", "Contact" as a drop-down or expandable menu, "Manage Users" as a drop-down or expandable menu, and "Account" as a drop-down or expandable menu. "Announcements" will redirect the browser to the announcements page, "Calendar" will redirect the browser to the event calendar. The "Contact" submenu contains "Department" and "Faculty". "Department" will redirect the browser to view information about the department. "Faculty" will redirect the browser to the Faculty Directory page. The "Manage Users" submenu contains "Add User" and "Find User". "Add User" will redirect the browser to the account creation page, and "Find User" will redirect the browser to the search user page. The "Account" submenu contains "Login" and "Logout". "Login" will redirect the browser to the log in page, and "Logout" will log the user out, displaying a notification that the user is logged out, then redirect the browser to the log in page.

Student

Log in

This page accepts a user email and a password. It displays a Login button that when pressed, will log the user in if the credentials are correct. If the fields are empty, a field will be highlighted prompting the user to complete it. If the fields are filled with incorrect values, the page will display a notification that sign in failed, and clear the fields for the user to try again. There is also a Forgot Password? link which redirects the user to the reset password page. When the user successfully logs in, they will be redirected immediately to the announcements page.

Reset Password

A back button will take the user back to the log in page. This page allows a user to enter an email. Once the reset button is clicked, If the email is found in the database, a reset password email will be sent to the email given, and the user notified of that fact.

Announcements

If a person attempts to view this page without logging in the browser will be redirected to the log in page.
After a "New Announcement" button, this page displays all announcements for the user, only displaying the ones that are intended for the user to see. This is determined based on the groups their account is associated with.

High priority messages are displayed with a red title, medium priority messages have a green title, and low priority messages have a black title. Each message displays the title, the author, and the date and time it was written, and the contents of the message. If the contents are lengthy, the full message is displayed in the overlay full message display that appears when the message is clicked. A message that the user has not yet clicked on is marked as New.

The full message display includes the title of the message, followed by the message, and then it gives the Posted By:, On:, and To:, information fields at the bottom right. To dismiss the full message display, click outside the message or click on the x in the upper right hand corner. Once a message has been seen in full message display mode the New tag is removed.

At the bottom of the announcements page a link "Back to Top" returns the user to the top of the page.

Event Calendar

The calendar is imported from Google Calendar and is updated by the UMSL Music Department. There are two views displayed: the monthly view, and the list view.

Contact Department

This page provides a short description of the department and gives the address and phone numbers to contact the department.

Faculty Directory

This is a static list that will need to be manually updated. It is derived from the list of faculties on the existing umsl.edu Music Department website. Individuals are presented in alphabetical order by last name, although first name precedes the last name in the listing. The faculty member's areas of interest or speciality are listed, followed by their email and phone number if available. Each entry will show a deeper shadow when the mouse is hovering, to assist the user in isolating the information they require.

Inaccessible Areas

A student, or viewer that is not logged in, cannot view Find User, add User, or add Announcement pages. Attempts to view these pages will result in the browser being redirected to the log in page.

Faculty

The faculty has all the privileges of the student.

Find User

If a person attempts to view this page without the correct privileges they will be informed they do not have the privileges, and the browser will be redirected to the announcements page.

A drop-down menu is displayed. Names in the list are displayed last name first followed by first name and are displayed in alphabetical order. The first user in the list is selected by default, as indicated by the name displayed in all capitalization. Once a user is selected and the submit button is selected, the browser is redirected to the edit user page.

Edit User

If a person attempts to view this page without the correct privileges they will be informed they do not have the privileges, and the browser will be redirected to the announcements page.

This page displays the name and email and groups the user's account is currently associated with. These fields are editable. If no group is selected the app defaults to including the user in all groups. An invalidated field will make the edit and delete buttons unresponsive. Once changes are made, the user can click on edit to commit them, or delete the account altogether.

When an account is deleted, the browser is redirected to the find user page.

Create an Announcement

If a person attempts to view this page without the correct privileges they will be informed they do not have the privileges, and the browser will be redirected to the announcements page.

Add User

This page displays fields required for creating a new user account.
An account requires a first name, last name, and a valid email.
A faculty member can only create an account and add it to the Groups to which they belong themselves.

Admin

The admin has all the privileges of the student and the faculty.

Add User

See Faculty→Add User for more information.
If an account was deleted recently the database rejects the creation of an account with the same email. This email will only be cleared out of the database if the user attempts to log in to the app again. Please see Firebase documentation for more detail.

An admin viewing this page will be able to choose from any existing groups and add faculty privileges. If they select faculty privileges, they will be able to enter a phone number and an office number, and select admin privileges. This page warns the admin that creating another admin gives them full admin privileges.

Technical Documentation

Admin Access

The credentials to log into a default admin account on the app and to access the Firebase dashboard are stored privately by Amanda Rawls. Should this project be developed further, details can be requested by emailing her at jarfk6@mail.umsl.edu.

File Organization

- root
  - *project html files*
  - *README*
  - css
    (.css files required for Bootstrap, see documentation)
  - font-awesome
    (web icons from w3schools)
    - css
    - fonts
    - less
    - scss
  - fonts
    (glyph icons)
  - functions
    - *project .js files*
  - images
    - *project image files*

- js
  (.js files required for Bootstrap, see documentation)
- styles
  - *project .css files*

Firebase

Quick Facts

- Firebase is a real-time, NOSQL cloud database
  - syncs data across all clients, regardless of apps platform
  - Data stored as JSON objects
    - JSON supports arrays as properties which makes aggregating data very easy
      - e.g. Groups can have an Users[] as a property to simplify data associations
  - Advantages for our application: scalable, easy to install/configure (schemaless), minimal scripting overhead for queries
  - Potential issues: data layer hosted, workarounds available but must commit resources
  - Firebase creates unique keys(id) for each entry, which can be used to reinforce DB redundancy with little overhead
  - If you come from a relational database background, there is a learning curve
  - However, compared to even JDBC framework, firebase is very concise

About

Firebase is a Google-owned startup that provides backend infrastructure services for applications. These services include commonly required features, such as user authentication, databases, web hosting, push notifications, and analytics. Taking advantage of the features Firebase provides requires very minimal time for configuration, and is easy to learn for someone with experience working with alternative implementations of these services.

Furthermore, the entire API used to access the Firebase infrastructure is cross-platform. Firebase provides a unified set of API calls in various languages, such as JavaScript, Swift, Objective C, and Java – making it highly suitable for developing applications across platforms, including web, mobile, and desktop.

Because of the aforementioned ease of adoption and cross-platform functionality, we decided to integrate Firebase into our application. It allowed us to quickly build much of the core functionality our customer requested with only one codebase, and fully supports both iOS and Android mobile platforms, in addition to the web interface. In terms of specific features required for our project, it was essential to provide a registration/authentication system; a database for storing and accessing users, groups, and announcements; and some form of notification for new announcements. It is

important to note that the storage database Firebase provides to applications is separate from the authentication database Firebase uses to store user credentials; the latter is not directly accessible by the API methods. This results in improved security, as the credential storage database is not vulnerable to SQL injections through the application's user interface.

Authentication

Instead of allowing users to register themselves, as is standard in many applications, this project required a specific authentication system tailored to the needs of the customer's department. In order to avoid allowing non-students to view information that they shouldn't, the customer requested the ability for faculty and administrators to control who could add users to the system. Users would be added by email, given a random password unknown to both faculty and student, and be immediately sent an email to set their own custom password. Additionally, users must be allowed to reset their password if they were to forget.

Firebase provides the ability to register accounts and send password-reset emails out-of-the-box. After a faculty member submits a provided form containing profile data for a user, the system generates a random alphanumeric password, sends the password reset email to the user, and saves their profile in the database. To avoid a state where the user registration is only partially complete for any reason (such as a loss of internet connection), if one step fails, the proceeding steps are not executed. For instance, if the API call to register a user with a provided email and password fails, the faculty user would be notified, and the password reset email would never be sent.

If it was also required that faculty can delete user accounts in cases where the student was no longer a student, and the admin to remove a faculty account for a member who leaves the department. However, the Firebase API does not allow the ability to delete a user account without that user's unique OAuth token. Thus, while a user could potentially can delete their own account, there is currently no API method for one user to delete another's account. Instead, we accomplished this feature through a workaround involving the Firebase database. Because a user entry is created in the storage database when faculty registers a new user, it can be assumed that all currently registered users have a profile entry in the database. The system loads this profile data each time a user logs in. By allowing faculty to delete the user profile data from the database, we were able to simulate account deletion. While the user's login credentials would still be valid immediately after their profile data was deleted, upon logging in, the system would determine that there is no profile data to be loaded for this user, and then invalidate and delete that user's authentication credentials, and log them out. At this point, the user would have no profile information saved, and no valid credentials to log in.

Real-time Database

While there were alternative database solutions we could have used to store users, announcements and ensembles, Firebase provides a rather unique form of database that's useful in mobile-oriented applications. The database works as a sort of tree, where each "table" (in RDBMS terms) could be represented as a child node of the root.

Children of these "table" nodes could be entries of that table, or "tables" themselves. For example, one upper-level table node would be "announcements", which would have child nodes for each group. These group nodes would be "table" nodes, and each announcement sent to a group would be a child entry of this "group" node. Reading and writing to the database would involve traversing the tree to the node of interest, which is simplified by the Firebase API.

Because this application is to be primarily used on mobile, it wasn't enough to just provide a method to store and retrieve data. On a mobile phone, it could be considered a poor user experiences to have to press a button to reload data, or navigate back and forward again, such as to pull new announcements for the server. Implementing some sort of automatic server-polling feature would overcome this problem, but results in more development time. Firebase removes the need to implement this polling manually. By subscribing to listen for events on a node, client device can automatically be notified of changes to a node and it's sub-tree. A particular user's device listens on all nodes corresponding to the groups/ensembles that user is a part of, and when a faculty member posts a new announcement, a child node is added to that group sub-tree. Every listening client device is automatically sent that new announcement data, which is then rendered and displayed in real-time. All together, users can see new announcements as they are posted, without the need to refresh their device manually.

Data Structure

Our basic structure:

- 3 types of data objects
    - Announcements
        - functions/announcementmodel.js
    - Groups
        - functions/ groupmodel.js
    - Users
        - functions/usermodel.js

The combination of Announcements/Groups makes it very easy to filter announcements based off user accessing application. **Groups** has Users[] as a property and vice versa; the redundancy increases response time when retreiving information from the database.

Bootstrap

About

"Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools. Before Bootstrap, various libraries were used for interface development, which led to inconsistencies and a high maintenance burden." - From the Bootstrap website.

Bootstrap has an MIT Open Source license allowing you to do what you want with it. The license can be found at https://github.com/twbs/bootstrap/blob/master/LICENSE. Bootstrap has already seen a great deal of success: 20% of the most popular websites that use JavaScript today also use Bootstrap, so the likelihood that a team picking up this project to develop it further will have some familiarity with it.

Use in This Project

Since this project is intended to be used across all platforms it was essential that the appearance remained consistent. Bootstrap enabled us to accomplish this task quickly and make building this app in a few short weeks a feasible goal. Adding responsiveness was made easy with Bootstrap.

Thanks to Bootstrap, it was easy for the multiple team members to work on the graphic user interface and keep the style consistent across the growing number of .html pages. Bootstrap already comes with the capability to respond to the platform on which the app is viewed, meaning despite the possibility of being viewed anywhere, the app will be consistent and the development team only had to work with one copy of the design code. This helped us to develop an app that would work for mobile, without having to make separate Android and iOS apps.

The standard page design is classic Bootstrap page with taskbar along the top, followed by a carousel. The taskbar is recreated on each page, so it gives each page a similar look right away. Attempts at externalizing the taskbar proved challenging, but doing so would make updating the taskbar much easier, as it is currently duplicated across every page that requires it.

Sources:
https://w3techs.com/technologies/details/js-bootstrap/all/all
https://trends.builtwith.com/docinfo/Twitter-Bootstrap

Additional Notes

The Music Department Facebook page is added with Facebook plugin. Same is done with Google Calendar on the calendar page. Colors for buttons use the UMSL color scheme.

Future Functionality

Privileges

Hide button elements from users if they do not have the privilege to view the content of the pages they lead to.
Ensure users cannot edit their own details, and can only edit the details of users of lower privilege.

Announcements

Delete announcements, only by the author of the announcement.

Groups

Group creation, edit, and deletion only by a user with admin privileges.
Generate a random token for the group so that new users can use the token to add themselves to a group and remove the burden of adding hundreds of students at the beginning of the semester from faculty and admin.

Add User

Only allow umsl.edu or Missouri University System suffixes.
Given one or more tokens (generated at random and distributed by faculty to students) and entered by the student), notify the faculty responsible for the group that the student requests access to announcements. For safety reasons the faculty must approve students before they can see announcements for any given group (except All).

Edit User

Group selection. The default functionality that stipulates no group selection registers the user in all groups will be removed, and a "no group" option displayed under the Groups heading instead.
Add an option to edit privileges, to be displayed only to a user with admin privileges.

Calendar

Event creation, edit, and deletion from within the app to centralize all tasks related to student communication, only by faculty and admin.

Faculty Directory

Populate this list dynamically from the Faculty registered in the app itself.
Include office locations for faculty members if available
Search directory to find a faculty member by group or name.

Known Issues

Edit User

The buttons on this page may not operate correctly in Safari. This is a known problem and is to be corrected.

Team Documentation

Customer-led Decisions

To Build or Not To Build On the Previous Group's Work

As we discussed the user requirements over Spring Break, we decided to approach the customer with a web-based solution and explained its advantages considering our limitations.
Due to the costs and regulations incurred with the Apple Store, it would not be easy to facilitate a working, separate, Android and iOS app. However, since nearly all mobile devices are capable of displaying a responsive, adaptable website, we thought this would be the most flexible solution. The customer agreed to this approach. As a result, very little of the previous group's work was carried over into our project. We at first thought at least the directory could remain the same, but it was poorly organized and needed to be overhauled.

User Story Priority

In order to stay in keeping with the book, when we confirmed the user stories with the customer, we asked him to set the priority of user stories in the first iteration. Beyond the first iteration, Mr. Brandes requested that we decide user stories based on logical progression of functionality.

Iterations

The standard iteration length was decided to be 24 hours of work over the course of 1 week. This was determined based on the rule of thumb that a student study 2 hours for each credit hour of the course. With 3 credit hours and 4 students, this led us to conclude 24 hours was a reasonable amount of work in a given week.

Spring Break

The project was assigned just before Spring Break. The team met with the customer before Spring Break to gather requirements. The team worked over the break to develop user stories descriptions and estimates and confirmed these user stories with the customer after returning from break.

1st iteration

After Spring Break, we asked Mr. Brandes to decide what functionality he would most like to see first. With communications being the most important, he decided on the log in, log out, send announcement and read announcement user stories. We commenced with a velocity of 0.7 and successfully delivered all user stories at the end of the iteration. We also provided a demo with which the customer expressed satisfaction.

2nd iteration

At this point Brandes had requested that we decide our own user stories. We decided to expand upon the read and send announcements functionality with privileges enforced on both sides. We also added the calendar and a rudimentary contact page for the department, and a reset password page.

3rd iteration

During this iteration, we continued to develop the privileges of different account types, this time with the user stories involving account creation, edit, and removal. The Faculty Directory was also added during this iteration. This iteration was kept small due to high workloads in other courses. Another customer demo occured and with most of the work being "under the hood" the customer was still satisfied with the project but did not seem as enthusiastic.

Clean up

With Group Presentations and course finals approaching, we entered clean up mode. We worked on completing the functionality that had been started and ensuring form validation, adjusting the visual appearance of the website, creating and rehearsing the group presentation, and finalizing the documentation.

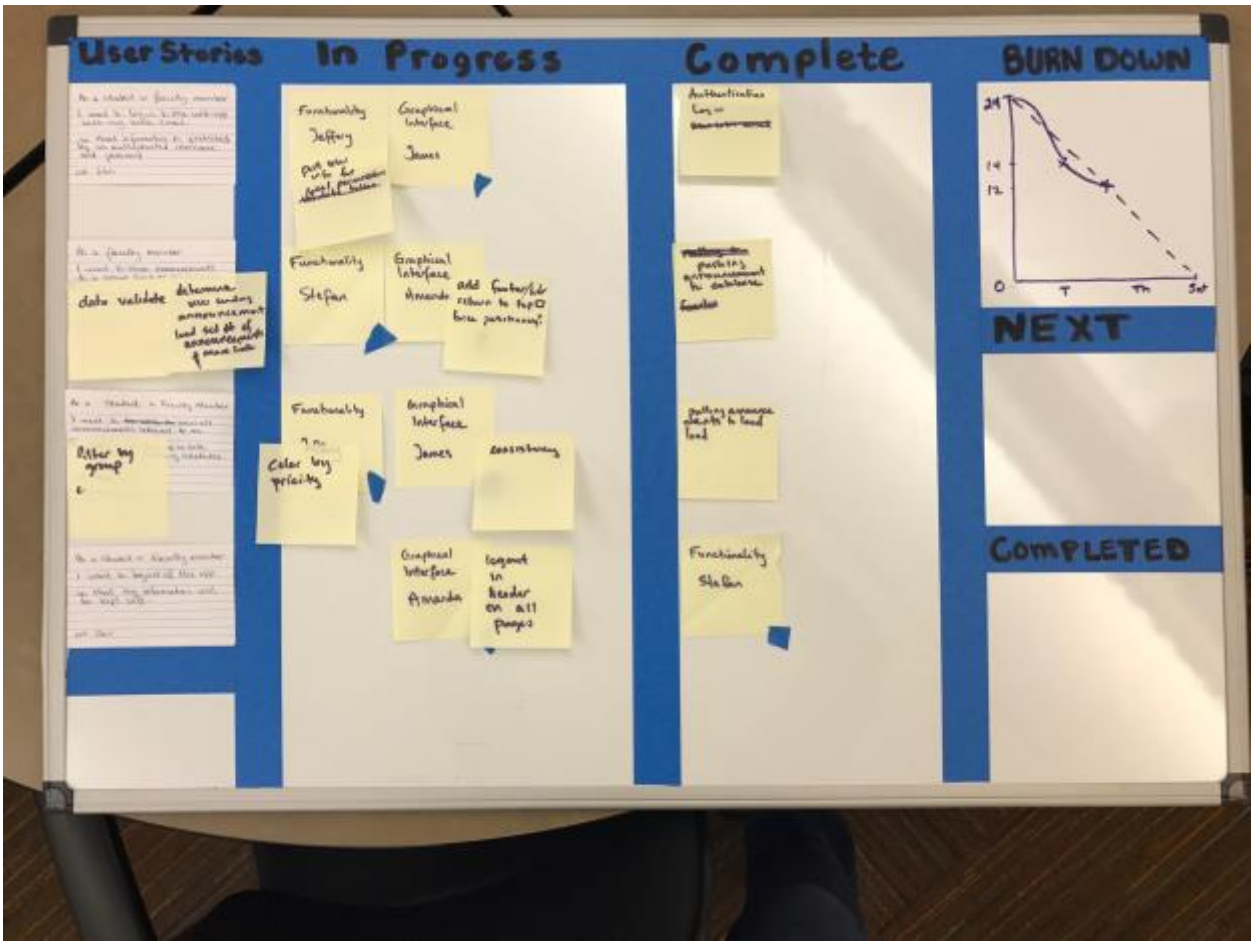
The Big Board

User stories

The user stories were written in the format
As a <user type> I want to <functionality> because <purpose of functionality>.
User stories were kept minimal. For instance, there was a separate user story for "add account", "edit account", "delete account" for all account types. This ended up being redundant. This also reduced the necessity for separate tasks. It is the opinion of Amanda Rawls, the project leader, that the user stories should have been rewritten/revisited more thoroughly at the end of each iteration.

Unsatisfactory Performance of the Big Board

Due to the disparate and conflicting schedules of the team members, most of the work done on this project was done in relative isolation. As such, it was not always possible to update the big board when work had been done. While the board was stored on campus for equal access, most of the work as also done off-campus on the weekends. The board was essentially inaccessible. While the board was updated twice a week during class meetings, only once did someone remember to take a photograph of it. This is how it appeared at that time:



Communications in lieu of the Big Board

Slack became our "Big Board". At the beginning of each iteration, a post as "pinned" to the channel, stating what was scheduled to be done. Discussions about bugs and individual user were maintained inside in threads.

Advantages

Universal access and ease of communication reduced the isolation of the work of individuals. Questions were asked and responded to quickly, and resources were shared with ease. Slack allows a channel to pin important messages and threads, and also to maintain threads within the channel to isolate discussion on a particular topic from the general chatter.

Disadvantages

The overview and the daily tracking of the burn down rate provided by the big board was lost.

Testing and Bug Resolution

Testing was done by all team members as they used the application. As one team member completed and pushed their functionality to the Git repository, other team

members would access the website hosted on GitHub and test the functionality. Reports were made and tracked in Slack.

Unfortunately, Slack does not allow free accounts to export private channels as originally thought, so it is not possible to include complete testing and bug resolution documentation. However, here is an example of one such thread:

Fixing bug where new announcements aren't being colored based on priority

Jeffery Calhoun [8:14 PM]
Can you figure out why the color isn't being set for new announcements? In the function that generates announcement-card html, there's a function that should check the priority level and add a class to the card based on that. My guess is the priority it's getting is undefined, or doesn't fit one of my switch cases. I was expecting 0, 1, or 2

Stefan Rothermich [8:15 PM]
Yup is the code pushed?

Jeffery Calhoun [8:16 PM]
Yeah

Stefan Rothermich [8:16 PM]
Np

Stefan Rothermich [8:38 PM]
Fixed the usual JS crap, casted it as int and fixed it, was string

[8:38]
Want me to push it or u can just fix it really easy just the usual parseInt(announcement.proority)

[8:38]
When I refreshed all we're colored

Jeffery Calhoun [8:38 PM]
Go ahead and fix it if you don't mind

Stefan Rothermich [8:39 PM]
Sounds good ,

Jeffery Calhoun [8:39 PM]
Or push it since you already fixed it

Stefan Rothermich [8:39 PM]
Sure np

[8:44]
Pushed

Stefan Rothermich [10:25 PM]
Fixed/Pushed coloring bug as described above, and issue related to displaying all groups associated with an announcement in modal (announcements.html). The latter is functional but not sure I like the formatting for line breaks (implemented to prevent long strings from causing display issues). Probably look into formatting a tad but shouldn't cause any issues for a demo Monday.

[10:27]
Maybe just have it L justified or something