CS496

FINAL PROJECT DELIVERY

Dylan Feuerman &
Joseph Cornelius

Introduction:

Prior to the semester, our group reached out to Brian Haunert, the Director of Recreation and Wellness at Loyola University of Maryland. We contacted Mr. Haunert in anticipation of our senior project, after noticing that the mobile application for this department had not been updated since 2019. Upon pointing this out, our team offered to modernize the application by completing a full redesign as well as adding any features that the Department desired to the app. Once he obliged, we set up a meeting to go over our vision of the new application and see what features that he desired for us to add.

Mr. Haunert's specific requirements will be detailed in the next section, but it was a constructive meeting. By the end our team had a clear idea of the desired app for Loyola University of Maryland's Department of Recreation and Wellness, including their fitness and aquatic center (FAC). Our original vision aligned with our client's goals, and it was time to get to work. As a team, we made the decision to begin coding this application in JavaScript and see how far that would take us.

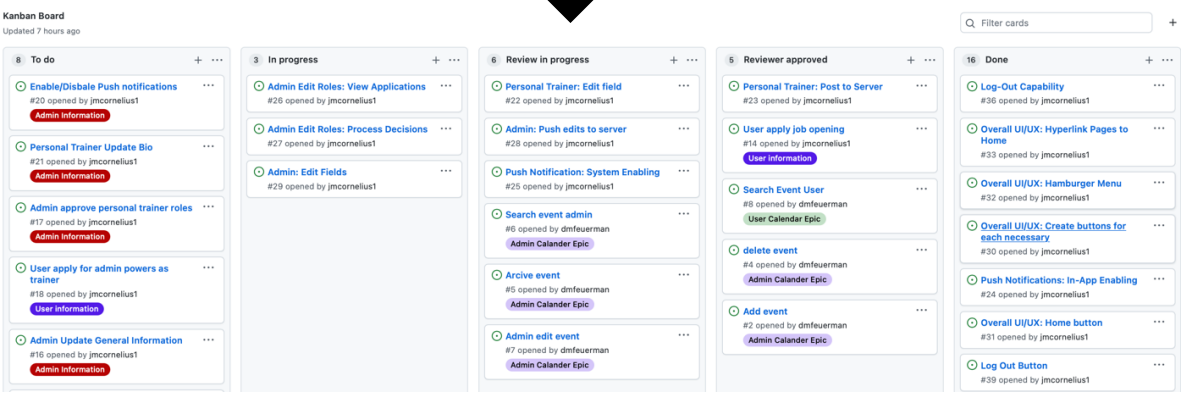
Requirements & Iterations:

Once our meeting concluded with Mr. Haunert, the two of us laid out the baseline requirements that were needed to create a functional, updated app for the FAC. As we learned from working together last semester, once the overarching baseline requirements were in place, we needed to break them into precise user stories. We organized these in a spreadsheet before transferring them to the Kanban board we are currently using on our git repository.

Below are brief progression screenshots from our initial logging, to part of our planning of each iteration, and finally a glimpse at our Kanban board from a previous point in time.

ID	Story Title	Priority	As a/an [user type]	I want [functionality]	So that [value]
17	Admin Delete job openings	Low	admin	to be able to delete outdated job openings	I will not continue to receive applications
15	Admin approve personal trainer roles	Medium	admin	to be able to accept or deny user applications for admin roles	I can promote trainers and deny unqualified requests
14	Admin Update General Information	high	admin	to be able to update the general information on any page - open swim, class times, rock climbing wall, etc.	it is easy for all users to access and see
13	Admin Update Policies	High	Admin	ability to update the sites various policies	rules are clearly stated, and can be edited as they change
11	Admin Update contact info	Medium	Admin	I want to be able to add new staff members	Users can get in contact with users
9	Admin Add student job opening	Medium	Admin	to post job openings for student employment	We can accept qualified applicants and may fill necessary positions
5	Admin Edit Event	Medium	Admin	to be able to edit an existing item	I can update
4	Search Event Admin	Medium	Admin	to search for an event	I can run admin features on it
3	Archive Event	High	Admin	remove event from calendar and store	users no longer see the item but can be re added
2	Delete Event	High	Admin	remove event from calendar	users no longer see the item
1	Add Event	High	Admin	add event to the calander	users know about upcoming functions
19	Personal Trainer Update Bio	Low	admin (trainer specific)	create/update an about me page	users feel more comfortable selecting me as their personal trainer
18	Enable/Disbale Push notifications	Low	admin or user	enable or disable my push notifications for the app as desired	I can be on alert or not bothered at my choosing
16	User apply for admin powers as trainer	Medium	user	to apply for admin role if I become a trainer	I may receive admin role and begin to schedule my classes/appointments with users
12	User apply job opening	Medium	User	to be able to apply for available student employment positions	I may be considered for the job
10	User access contact info	Medium	User	I want to be able to locate staff members	I can get in contact with them
8	User Sign up for Events	Medium	User	I want to be able to sign up for an upcoming event	I can participate in the activity
7	User request to Add Event	Medium	User	I want to be able to request the admin for a new event	I can have my event added to the calendar
6	Search Event User	Medium	User	I want to be able to search for an event	I can locate specific events

Competed:	ID	Story Title	Priority	As a/an [user type]	I want [functionality]	So that [value]
		Iteration #1, Delivery on March 02, 2022				
Done	1	Front end login form	High	Admin/User	Want to be able to log into my account	I can use the fetures of the application
	Story points Id	Name/purpose				
	1.1	Text box for username/email				
	1.2	Text box for password				
	1.3	Submit button				
	1.4	Switch to sign up				
Done	2	Front end sign up form	High	Admin/User	Want to be able to sign up for an account	I can use the fetures of the application
	Story points Id	Name/purpose				
	2.1	Text box for username/email				
	2.2	Confirm text box for username/email				
	2.3	Text box for password				
	2.4	Submit button				
	2.5	Switch to sign up				
	3	Front end Admin Delete job openings	Low	admin	to be able to delete outdated job openings	I will not continue to receive applications
	Story points Id	Name/purpose				
	3.1	select job box				
	3.2	Select delete button				
	3.3	Select cancel button * 2				
	3.4	Select confrim delete button				
	4	Front end Admin approve personal trainer roles	Medium	admin	to be able to accept or deny user applications for admin roles	I can promote trainers and deny unqualified requests
	Story points Id	Name/purpose				
	4.1	Select training request				
	4.2	Delete request button				
	4.3	Confirm request button				
	5	Front end Admin Update General Information	high	admin	to be able to update the general information on any page - open swim, class times, rock climl	it is easy for all users to access and see
	Story points Id	Name/purpose				
	5.1	change information in text box				
	5.2	Cancel button				
	5.3	Save/publish button				
	6	Front end Admin Update Policies	High	Admin	ability to update the sites various policies	rules are clearly stated, and can be edited as they change
	Story points Id	Name/purpose				
	6.1	change information in text box				
	6.2	Cancel button				
	6.3	Save/publish button				



Key Requirements from Client:

- Mobile Compatible
- General Info
- Student, Trainer, Admin Account Types
    - Accounts stored on backend Database
    - Login/Log Out
- Events
    - Add, Edit, Delete
- Job Applications
    - Apply/Review
- Search
- Hamburger Menu
- Navigation Bar
- Main App Styled Home Page

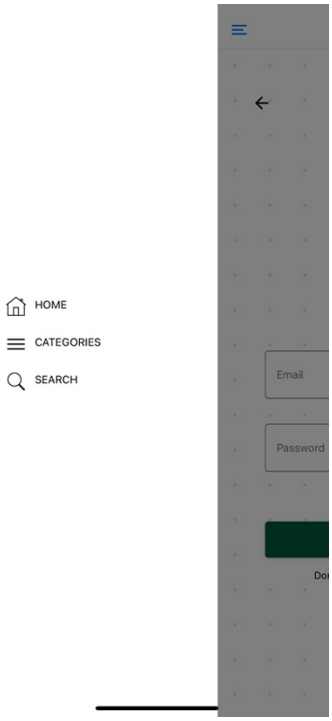
The steps completed at each iteration have been detailed previously and tracked on GitHub through our Kanban board. Further, most of our work was completed through the Expo Go web interface, often with zip folders too large for GitHub. To verify our frequent coding updates and continual improvement + teamwork, we exported a version control of our code updates to our Git Repo under SeniorCapstone/Version_control_Dylan_Fuerman.pdf. On top of all of these items to track our progress, the use of QR codes through Expo Go shows our project as it stands at different points throughout the semester and at each iteration.

Iteration 1:

Iteration one saw our team create a very barebones application, with the overall feel of a current mobile app, and none of the true features, data, or backend. This satisfied our goal to this point, as we were able to make some crucial design decisions, and successfully completed the bulk of the skeleton code. The main user stories completed at this iteration were:

- Overall UI/UX: Home screen
- Overall UI/UX: Create buttons for each necessary page (collection)
- Overall UI/UX: Hyperlink pages to buttons
- Overall UI/UX: Hamburger Menu

There were not many screenshots to be taken at this point that are not included later in the presentation but attached below is a quick screenshot of the desired hamburger menu, specific to this iteration.

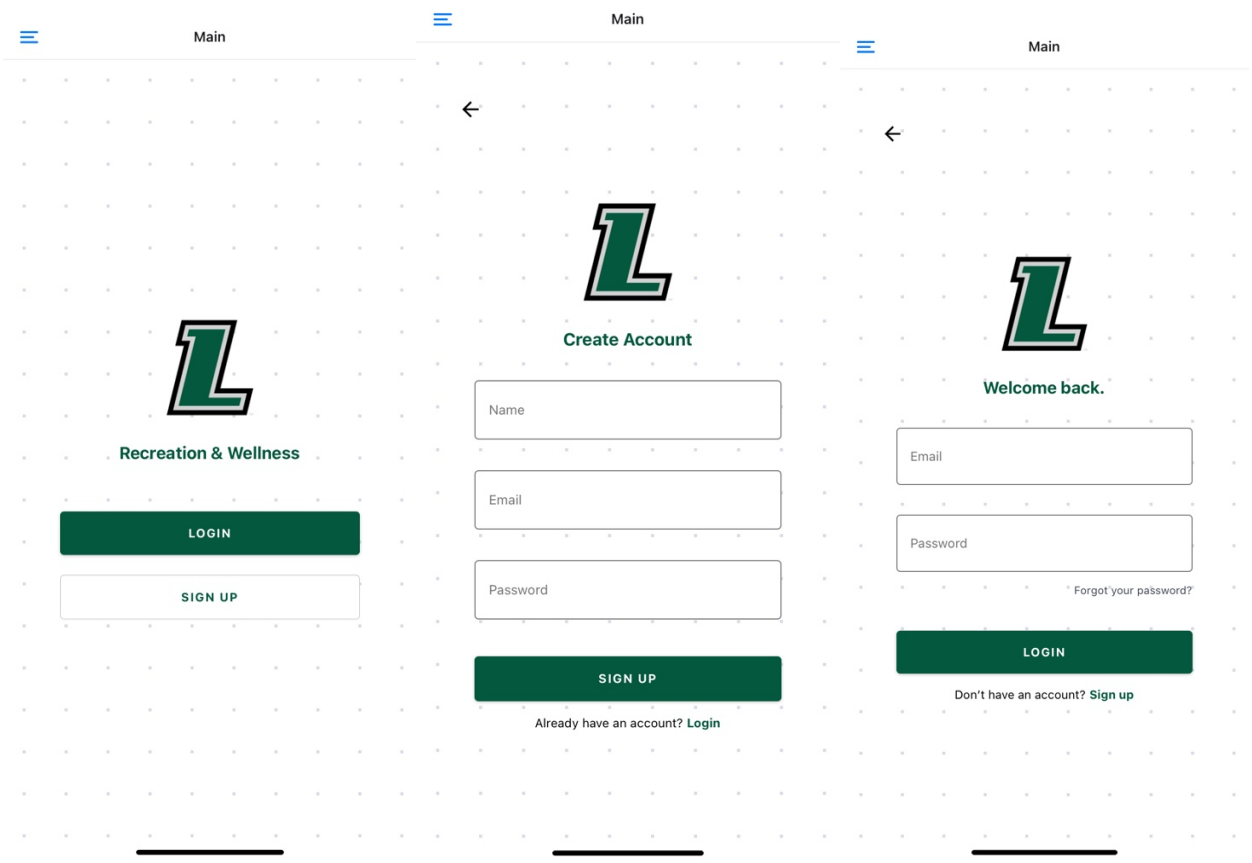


Iteration 2:

The team focus for iteration two was incorporating a user signup, sign-in, sign-out, and forgot password portions. For this we completed different user interfaces entirely dependent on whether the user that was logging in had the role of user, trainer, or admin. While the front end was prototyped then coded with little error, this iteration saw our team begin to be become conflicted with various databases, and the backend work of handling all logins. In this iteration we completed our original user stories of:

- Log-In screen
- Log-In capability
- Register screen
- Register an account
- Log-Out button
- Log-Out capability
- Forgot password page

Below are pictures attached of our finalized initial, sign-in, and register pages. It can be noticed that each of the pages is hyperlinked into one another for ease of navigation and a simplistic, up-to-date UI/UX.



Iteration 3:

Iteration three proved to be our most challenging stop along the development process. The two of us went back and forth on trying various databases and developing the backend that would 'hook-up' to the database. After countless hours of trial and error, we switched our focus to building the front end for the pages we knew would be necessary.

From there we began to brainstorm all the forms that would be necessary, how the forms interconnected to the various pages, and thus began our entity-relationship diagram construction. This was a particularly frustrating iteration because even though we developed

more working code (see unit testing), there was less functionality within the app, and mostly limited between the front end and the back end. We finally decided to use Google's *Firestore* NoSQL database and got the backend server to work properly on the day of the delivery. This meant that we had all the work we completed for the design ready to go, but only a limited amount of it functional at the time of delivery.

Iteration 4:

The fourth and final deliverable iteration, and subsequent time since is really when our team put all the pieces together. We linked our database to our application, so they now update one another in real time. The team linked the forms into their individual pages, thus integrating our collections to their appropriate in app buttons and subsequent pages. Given that so much of our project came together towards the end, many of our user stories, any containing database data were completed in the final iteration, and to a high level of success, see unit testing HTML screenshot. Screenshots from inside the application will be provided later. The user stories finalized for collaborative review in this stage were:

- Admin update general information
- Admin update policies
- Admin
- Admin add event
- Admin edit fields
- Admin update contact information
- Admins push edits to server
- Admin edit roles: View Applications

- Personal Trainer edit field
- Apply for trainer roles as user
- User request to add event
- User sign up for events
- Archive event
- User view job openings
- User apply job opening

Future Iterations:

Our hang-up on the connection from front end UI to the backend database storage caught us up for longer than expected, and consequently shortened our expected timeline. In terms of functionality our app runs at a high success rate, with 0 downtime and real-time updates. It is a very usable application that our team is proud of. However, this sacrifice of time towards the end cost us in terms of user experience (UX) due to the overall appearance. Our goal was to have a modernized application that fits into current aesthetic standards. While it is an efficient app that 'gets the job done', it is admittedly a bit 'clunky' and old-fashioned in appearance. Cleaning up the appearance would be a focus for us in the immediate future were we to have more time or continue with this project. In addition to appearance, some of the coding became tedious and we found ourselves in over our head given only a 15-week semester. With more time, some goals that our client did want would likely have been achievable. Though both of us have built search functions before, our in-app search function never had the entirely correct functionality during testing, so while in progress, that needs more work. Secondly, we figured out how to assign roles when registering, but never finalized approving different role changes,

and reassigning them the necessary permissions. Thus, the original user stories that did not get completed but were in progress:

- Admin approve personal trainer roles
- Admin Process Decisions
- Search Functionality (All roles)

Meanwhile, some of the features our client had not requested but that we had originally set out to incorporate presented themselves to be beyond our scope of capability. Since we were not launching this application into the actual app stores on the different devices, our team did not have access to the permissions for system notification settings. So, here are the original user stories that did not get completed and our group had yet to start:

- Push Notification: System enabling
- Push Notification: In-App enabling
- Enable/Disable Push Notifications

Architecture & Design:

After analyzing the task at hand, our team opted to take a layered approach to designing our application for the FAC. We knew that we would have a database at the core, and that while not every portion of the app actively used the database, the different user interfaces would be entirely dependent on which type of account was accessed from the database. Also, that each NoSQL collection would be linked to a button on the home screen, each with its own page.

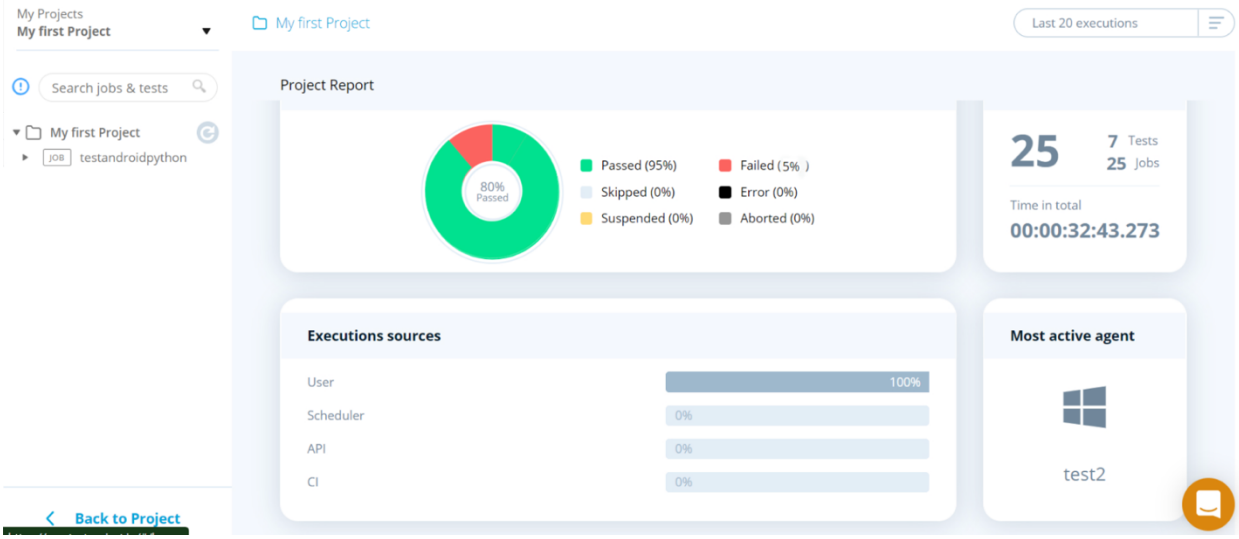
As for design, we decided to build our application using 'Expo Go'. It is an open-source web-based platform with partnered apps on both the Google Play and App Stores. This pairing of

web to mobile interfaces allows for easily accessing and testing projects, as well as keeping all coding and files both separate and organized. Upon every code update, our team could generate a QR code to be scanned on any mobile device, with the free Expo app installed, and our own application could be run. This was decided as the best option since it is free to use and the QR codes allow for unique version control as our team presents the various iterations to both client and professor. For the design language, our application was built using JavaScript, as it was a language we were both familiar with, easy to use, and supported by Expo Go. The only other language we used for this project was CSS/HTML to build the presentation webpages for our unit testing.

Finally, for the database our team chose to use Google's *Cloud Firestore* in conjunction with their app development interface, Google's *Firebase.* This is a NoSQL database that can be remotely accessed and supports all three interfaces we would need to use build a database for: Apple applications, Android applications, and web applications. Some key features that led to us choosing *Cloud Firestore* was its ability to store hierarchical data and more importantly was its offline support. This meant that our database caches data that the app is actively using, so that even if the device is offline, it will automatically sync in real-time as the device reconnected. Though our app was never fully completed and deployed, this would have been a necessary component were it to be used by Loyola. This database handles all login information, information entered and stored from each of the previous forms, and any general information as set by the admin, including files. The individual pages and subsections of the app each represent a different NoSQL collection within our database, detailed later.

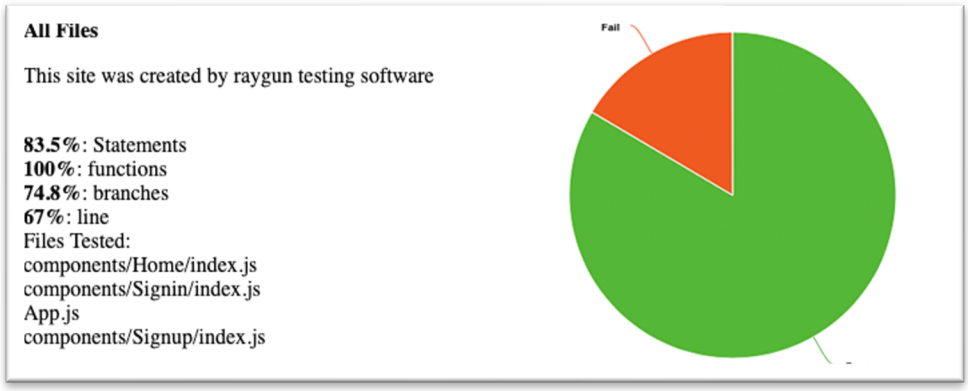
Testing & Internal Quality:

For our first iteration, we used software we found online to test our code. We only had a barebones layout of our application up to this point, so there was not much to be tested. Below a screenshot from the unit testing interface can be seen. As one may notice, the pie chart does not align with the data values shown. So, while the unit testing may be accurate, the interface was not, and we did not use this software for our future iterations.

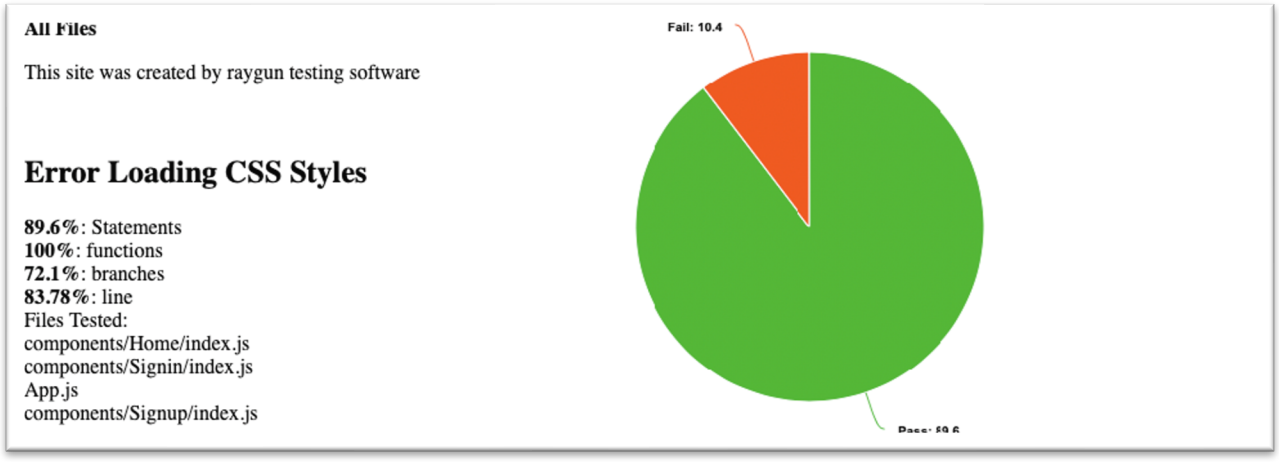


After seeing the shortcomings from our last unit testing software, Dylan recommended we used a software that he had helped develop. One that the company he completed his internship for ended up approving and began using themselves going forward. As it was trusted, we made the decision as a team to experiment with it as well, to which we saw great results. Rather than take screenshots from the program, we decided to transfer over our results and with them, designed custom HTML webpages to display our data in a concise fashion. As a team we felt like this was easiest for our client and others to view and take value from.

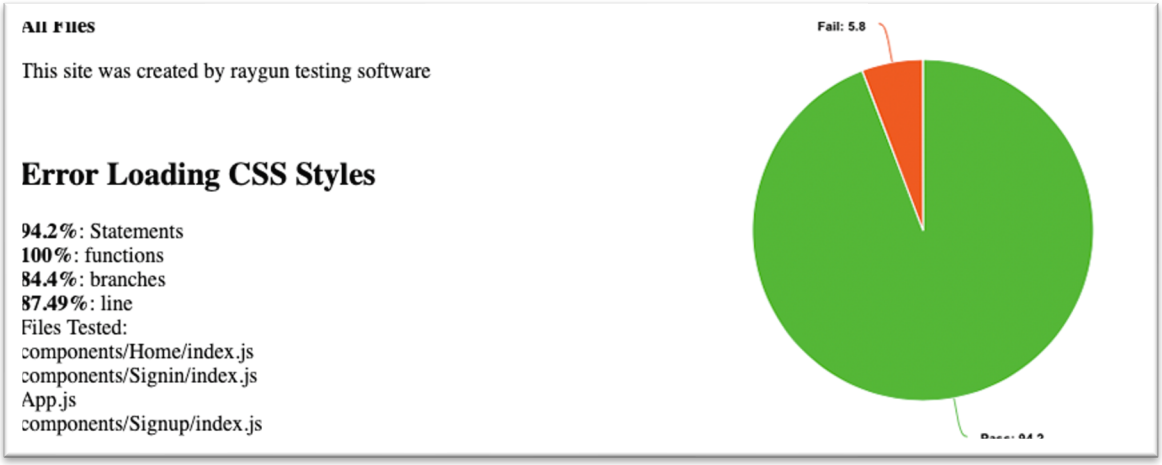
As is shown below, are our three different webpages showcasing our unit testing, one for each of development iterations 2-4, ordered. Across the board, there are gradual increases in the pass rates of each section. The only spot in which this trend does not hold true is the drop in branch coverage between iterations two and three. This is attributed to our difficulty at the time fully integrating and connecting our database. However, as is seen between iterations three and four, once this problem was resolved the 'branches' coverage increased drastically again.



Iteration #2



Iteration #3

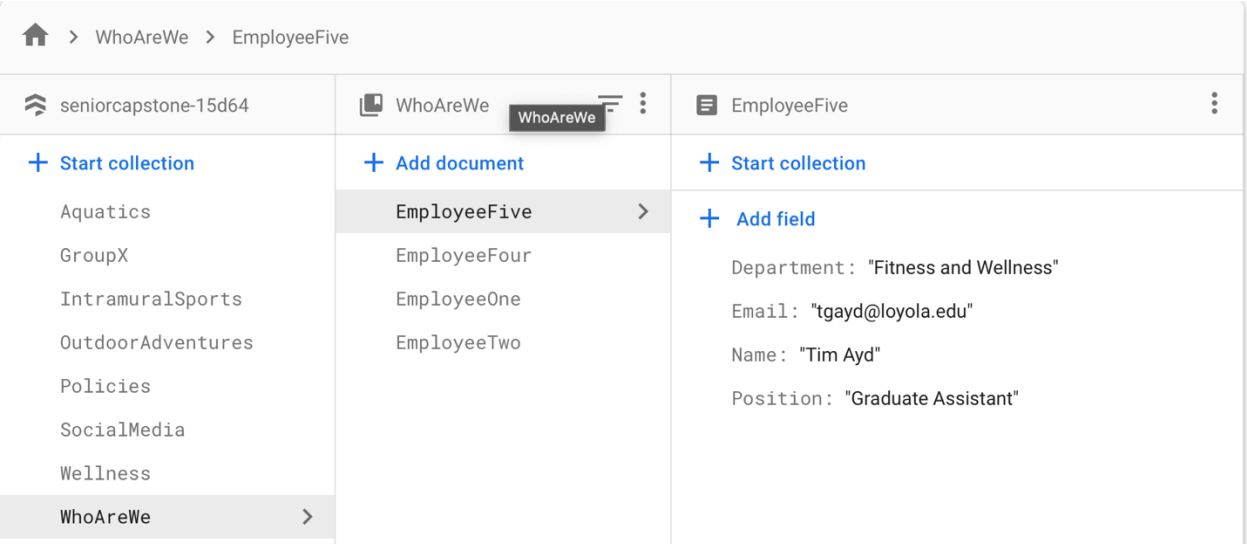


Iteration #4 (Final)

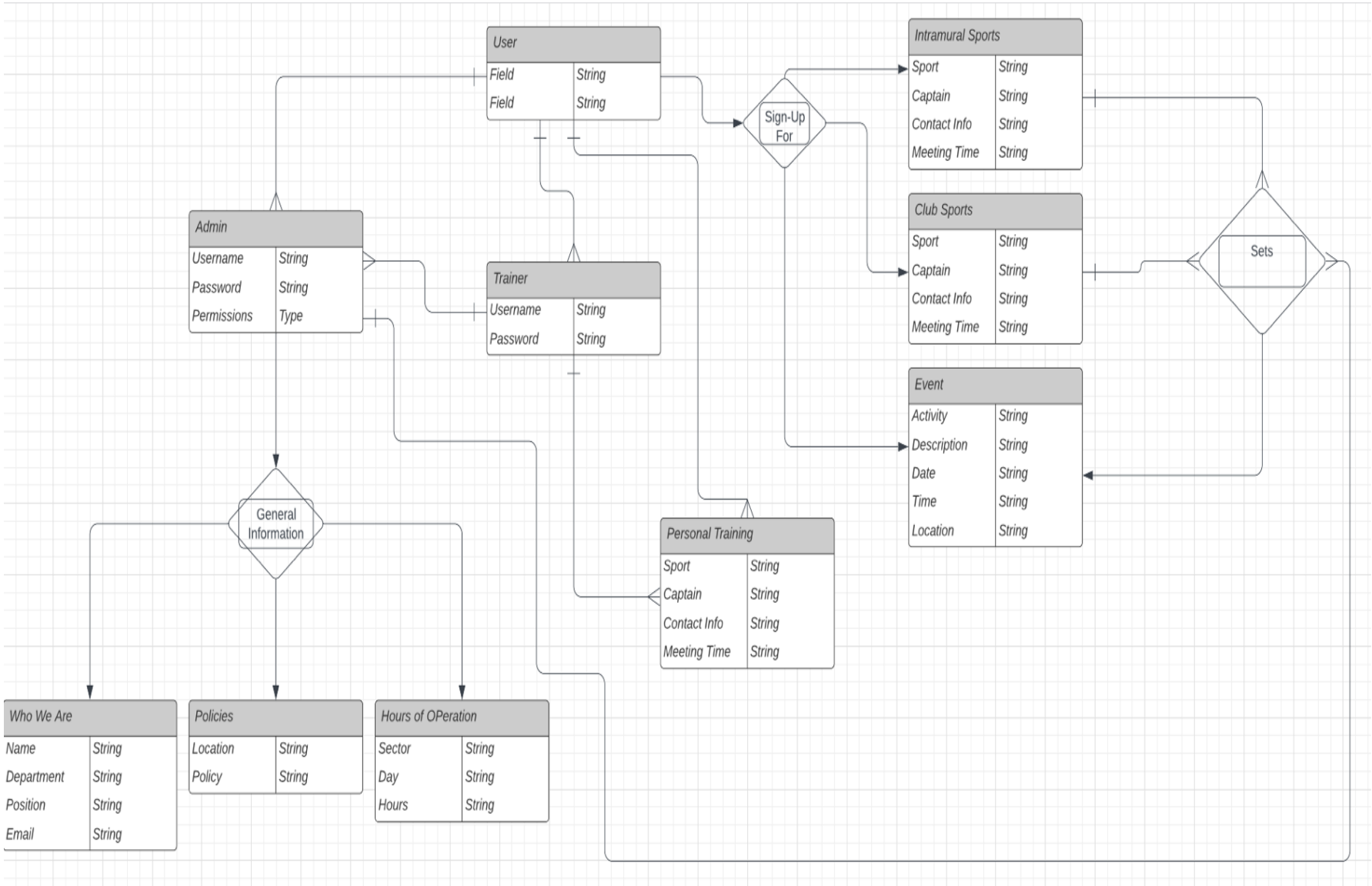
Finally, we had an initial goal of 95% or greater of our total code passing unit testing. Unfortunately, as is seen in the last webpage, we missed this mark by 0.8%. Though 95% was an optimistic goal, we are still proud of our achieving 94.2% coverage. Especially for an application that is not 100% finished.

Data:

To recap, our application stored all the necessary data on Google's *Cloud Firestore.* This is again, a NoSQL database. NoSQL databases utilize 'collections' rather than tables, and as such in our design we assigned each subsection of the app interface to its own collection. Below is a screenshot from Google's web interface, showing an example admin set field under the 'Who Are We' tab, accessed from the application's home screen.

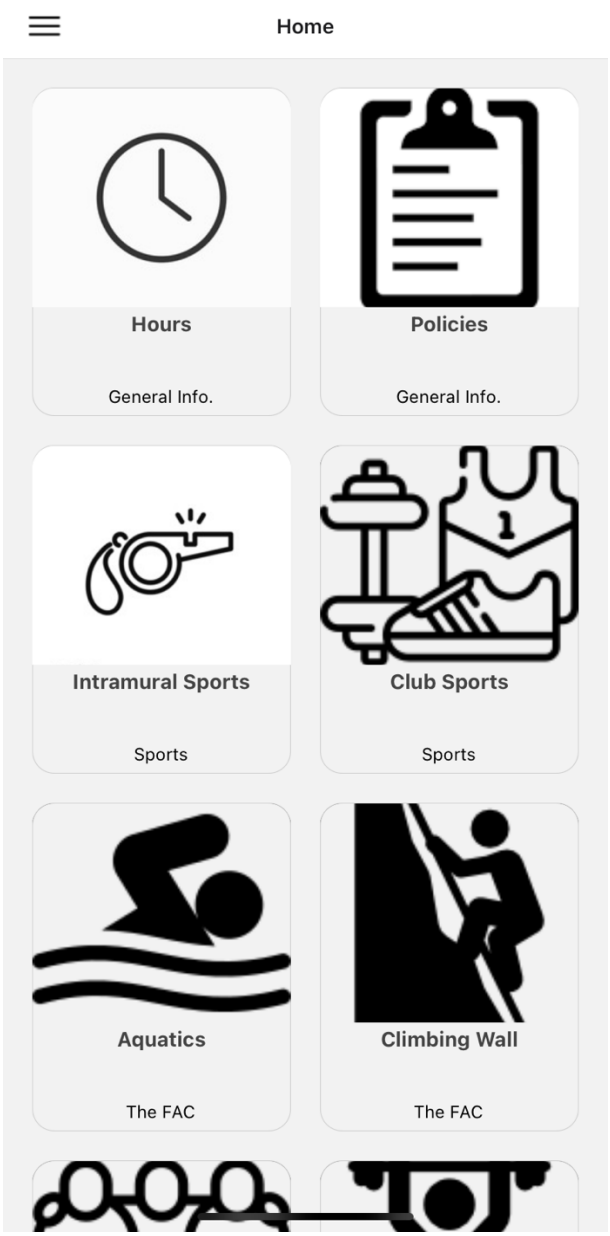


Further, to best highlight our overall interface of the main collections, below is a team generated entity-relationship diagram (ERD). This shows the general overall flow of the various entities currently stored within our database, their properties, and how the UI/UX interconnects them all in a way that is advantageous to both users, trainers, and admins.

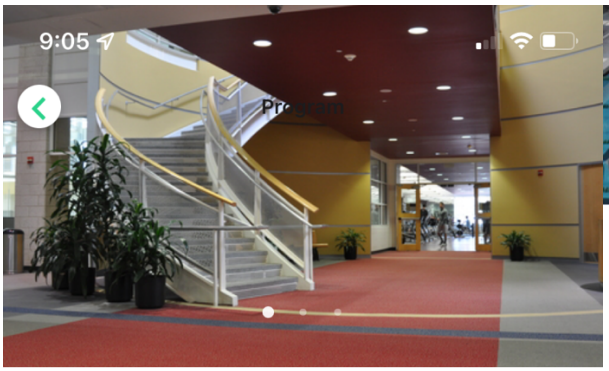

User
Field String
Field String
Sign-Up For
Intramural Sports
Sport String
Captain String
Contact Info String
Meeting Time String
Club Sports
Sport String
Captain String
Contact Info String
Meeting Time String
Sets
Admin
Username String
Password String
Permissions Type
Trainer
Username String
Password String
Event
Activity String
Description String
Date String
Time String
Location String
General Information
Personal Training
Sport String
Captain String
Contact Info String
Meeting Time String
Who We Are
Name String
Department String
Position String
Email String
Policies
Location String
Policy String
Hours of OPeration
Sector String
Day String
Hours String

UI:

Initial Load Up Screen
When Entering App



Application Home Screen



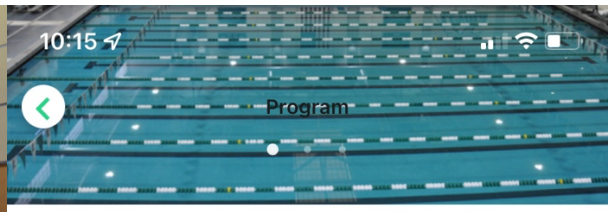
Policies

GENERAL INFO.

Location:
Basketball Court
Policy:
All equipment must be cleaned and put away

Location:
Weight Room
Policy:
All weights must be re-racked once finished

General Policies Screen – As Seen By User



Aquatics

THE FAC

Monday

Pool Hours:
6:00am-11:30pm

Tuesday

Pool Hours:
6:00am-11:30pm

Wednesday

Pool Hours:
6:00am-11:30pm

Overview of Aquatics Hours Screen – Sorted By Day, Scroll For More

Club Sports

SPORTS

Name

Name

Grade

Enter your Grade

Sport

What Sport?

Email

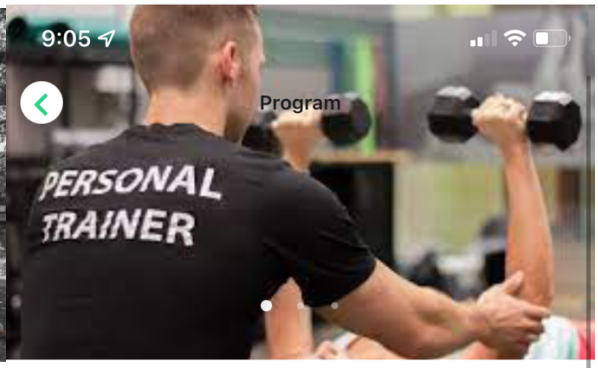
Email

Years of Experience

Years

Submit

Example of Form – Application to Join



Personal Trainers

THE FAC

Name

Name

Grade

Enter your Grade

Trainer

Specific Trainer? (If not leave empty)

Email

Email

Time

Time

Date

Date

Apply For Personal Training Session – User View

Student Employment

ABOUT US

Name

First Name

Grade

Enter your Grade

Job Type

What job do you want?

Days Available Per Week

Days

Times Available

Time

Submit

Intramural Sports

SPORTS

Baseball

Captain:
John Smith
Meeting Time:
2:00-4:00 M,W,F
Contact Info.
Jsmith@loyola.edu

Basketball

Captain:
Hank Rower
Meeting Time:
W,F 4:00-6:00
Contact Info.

Student Employment Application Form

Intramural Sports – View Current

Who Are We?

ABOUT US

Tim Ayd

Position:
Graduate Assistant
Department:
Fitness and Wellness
Position:
Graduate Assistant

Emily Fornatora

Position:
Assistant Director
Department:
Youth and Family Programs
Position:

About Us – Staff Page User View

Contact Us

ABOUT US

Name

First Name

Grade

Enter your Grade

Email

Email

Message

Message

Submit

Contact Form – Admin Receives Notification With Info