# HBDA Score Tracker Sprint 3

Zach Bender • Tyler Blair • Trey Thorne • Tessa Tryon

Source Code: https://github.com/tessa-hudson/Capstone_Fall2021

Project Site: https://tessa-hudson.github.io/Capstone_Fall2021

# Project Focus

## Who it's For

Hemophilia and Bleeding Disorders of Alabama (HBDA) for use with event attendance and summer camp scoreboard.

## What it Does

- Use a point tracking system for a friendly competition between groups of campers throughout the week
- Provides an efficient way for camp administrators to keep track of all the attendees and the events they participate in

# Sprint Backlog

- Each event should have a corresponding scoreboard on the homepage (~3 hours)
- Superadmins should be able to add attendees to groups in each event (~6 hours)
- Attendees should not exist in more than one group within an event (~3 hours)
- Superadmins should be able to remove attendees (~4 hours)
- Superadmins should be able to remove attendees from a group (~6 hours)
- Superadmins should be able to remove groups (and subsequently remove link between group and its attendees) (~6 hours)
- Admins should be able to remove events (and subsequently delete all groups in the event)
- (~6 hours)
- Superadmins should be able to update attendees (~4 hours)
- Superadmins should be able to update groups (~4 hours)
- Superadmins should be able to update events (~4 hours)
- Superadmins should be able to confirm or deny requests to change points (~8 hours)
- Campers/attendees should have read-only access to the app (ie they cannot create, update, or delete any data) (~4 hours)
- There should be an admin role (between campers and Superadmins) that can request point changes but can make no other changes to the data (~6 hours)
- API endpoints should require user authentication so unauthorized users cannot access data
- ( ~5 hours)
- User interface should be easy to view/use on a mobile device (~10 hours)
- App should have simple navigation (~10 hours)
- Add theme/styling to app (~12 hours)
- There should be separate environments (databases and environment variables) for testing, development, and production (~5 hours)
- There should be no hardcoded secrets in the app (~2 hours)
- App should have unit tests for at least 80% of API endpoints (~10 hours)

## Sprint Goals

- Restrict features based on user roles
- Style the app and make it accessible for mobile users
- Update and remove functionality for attendees, events, groups
- Set up separate environments for development, testing, and production
- Create Unit tests

# Sprint Metrics

- Stories
  - Sprint 1 - 7 Stories Completed
  - Sprint 2 - 11 Stories Completed
  - Sprint 3 - 17 Stories Completed
  - 3 Incomplete Stories
- Testing
  - Sprint 1 - 19 manual tests (Postman)
  - Sprint 2 - 29 manual tests (Postman)
  - Sprint 3 - 62 automated unit tests

## Contributions

## Contributions - Zach

- Front end functionality
  - Created the ability for users to update and delete attendees, groups, and events
  - Created the ability for users to add and remove attendees from groups
  - Created a page that can be used to view the attendees in each group
- Greatly improved the styling of the web application
- Created a Google Developer project to allow users to sign in with Google

## Contributions - Tyler

- Environment Configuration
  - Set up environment variables to allow code to work in both a production or dev/test environment without significant code changes
  - Create scripts for launching a
- Set up test database
  - Set up universal user to increase security
  - Uphold production data integrity; able to separate testing data and production data

# Contributions - Trey

- Optimized connection class
    - Separated bloated connection class into optimized and easier to understand subclasses
    - Subclasses contain the functions necessary for their respective tables while the connection class handles only the connection and holds the cascading deletion functionality
- Finished up Pointlog functionality
    - Point requests can be accepted or declined
    - When accepted, the point request is set as accepted and the corresponding group/attendee has their points modified
    - Added cascading deletion functionality, so if a group/event/attendee is deleted, so are any point requests in relation to them.

## Contributions - Tessa

- Auth0 integration
  - Set up Auth0 scopes for different roles
  - Restricted API Endpoints based on scopes
  - Added Auth data to frontend requests
- Frontend migration from Class Components to Functional Components
- Unit Tests for all endpoints

## Challenges

- Refactoring Code to meet requirements for Auth0
- Incorrect configuration of Test Database exhausted most of our Azure Student Credit
- New environments = New bugs
- Troubleshooting for the Production environment requires new deployments for each change
  - Backend Deployment: ~8 minutes
  - Frontend Deployment: ~5 minutes

## Lessons Learned

- Blocking tasks should have higher priority especially if it blocks another team member
  - When blocked by another team member’s task, a clear list of requirements should be provided to that team member so they know what to prioritize
- Testing environment should be added much earlier on in the project

# Demo

New Website Functionality

- Updating
- Deleting
- Adding Attendees to Groups

Restricted Access to Site

- Depending on time