CS482/495/496 Software Project Proposal: Youth Football League Portal

Oscar Roat, Brenden Bokino, Loren Kim, Reece Watkins

2025-11-25

1 Client Information

By sharing this client information and the rest of this document, you are stating that this client has provided this project as something they want (not something you created and asked if they wanted), and that they are interested in having you complete this project for your capstone.

- Client name: Dr. Sibren Isaacman
- Client title: League
- Client email address: snisaacman@loyola.edu
- Client employer: Loyola University Maryland
- How you know the client: CS Professor

2 Project Description

2.1 Overview

Our project is a web application for a youth football league called "Loyal Order of Youth League Athletes". Dr. Isaacman wants an all-in-one site to advertise manage, and broadcast the league. Here, games are scheduled and live streamed, teams are managed, and stats are tracked.

2.2 Key Features

[At this point you should have a basic understanding of your client's needs. List out the key features of the software system the client wants you to build.]

- **User Roles:**
  - Admin: manage calendar, games, playoff brackets, carousel, stats, live broadcasts, verify parental approval.
  - Coach: invite adults to create youth profiles, manage their own team.
  - Adult: oversee and update their child's youth profile, communicate with youth.
  - Youth: create a personal profile including photo (with privacy restrictions).
  - Guest: comment during live games, rate players/coaches, limited access to youth pages.
- **Calendar:** schedule and display matches with date, time, type (regular, playoff, finals), and participating teams.
- **Team Management:** create, update, and delete teams; include team name, colors, and roster.

- **Game Management:** add, edit, delete games; update and track game statistics; broadcast live games with live stats.
- **Playoff Bracket:** auto-updating bracket to reflect wins/losses during the season.
- **Profiles:**
  - Youth: profile picture and details, linked to adult accounts.
  - Adult: contact info required, can manage youth profiles.
  - Coaches/Users: manage own accounts, invite/manage youth.
- **Communication:** adults can send messages to youth; users can comment on live games and review teams/coaches.
- **Frontpage Features:** carousel of seasonal images to enhance visual appeal.
- **Privacy and Access Control:**
  - Youth cannot chat with other youth.
  - Youth profile pictures visible only to logged-in users.
  - Coaches/adults have additional permissions for managing teams and profiles.

2.3 Areas of CS required

Software Engineering/Testing, Web Development, Data Management, (maybe some) Sports Analytics

2.4 Potential Concerns and Questions

[Is there any aspect of this project that makes you unsure if it will work, either due to your own interests/background, or that you aren't sure if it fits the requirements? Are there questions you have about this project that you want instructor feedback about?]

3 Requirements

3.1 Non-Functional Requirements

[Non-functional requirements are just as important as functional requirements. Dont forget to specify them.]

ID	NFR Title	Category	Description
NFR1	NFR Example 1	Usability	Description of the NFR (it does not follow a user story template)
NFR2	NFR Example 2	Security	Description of the NFR (it does not follow a user story template)

Table 1: Non-Functional requirements

3.2 Functional Requirements (User Stories)

[In CS482, all functional requirements are written as User Stories. In CS496, some projects may use a different template to write the requirements. The table below is an example of writing the Stories. Adapt accordingly to different templates or if you want to display more info.]

ID	Story Title	Points	Description
S1	Story Example 1	5	As a user, I want to write a user story example, so that people will understand them.
S2	Story Example 2	2	As a user, I want to write a user story example, so that people will understand them.

Table 2: Functional requirements as User Stories.

4 System Design

4.1 Architecture

We will be using an MVC architecture for this project. Since there are four of us working together, the separated design will be the easiest to integrate. For the *Model*, we will need a way to access data for individual users (admin, coaches, adults, youths), teams, and games (to add to the calendar). The *View* will need landing pages for the home screen, team, calendar/games, profile, and livestreams, including comments and game stats.

The main modules involved in this project have to do with common functionalities on the website. Adding games to the calendar is a primary admin function, and there is a lot of info that goes into each game, such as the date, time, location, and teams playing. All of these should be added to the calendar together as a new "game."

Messaging is another key feature, but there are different types of messages and restrictions based on user types. There is messaging between two adults, which should be private for those individuals. Adults can also message youth to inform them of games, but youth cannot message back/each other. Finally, there are live comments during games. Those should be public to everyone watching the live stream.

The score board during live streams is another important piece. Having an efficient way to update the UI during games is important so the score bug is up-to-date. Things that need to be updated on it are what quarter it is, the clock, the down and distance, and each teams score.

Other key features that could be built into modules are getting data from the database(s), user authentication, the front page picture carousel, and profile creation/updating

4.2 Diagrams

[CS482, on sprints/iterations 2-3, you need to create and update a diagram (check the assignment for which type of diagram). On CS496, since before sprint/iteration 1 you should have a class diagram and keep it up-to-date.]

4.3 Technology

We will be using **JavaScript** for building the application. For the front-end, we will use **Bootstrap** and **Jquery** for UI. For the backend, we will use **Express** for server connection and **MongoDB** for the database. For testing, we will use **Jest**.

4.4 Coding Standards

Tests will be required for every new commit. Obviously, we want as high coverage as possible, but the baseline for every commit should be **70%**. In person meetings will happen at least once per sprint, preferably earlier so everyone will be on the same page going into the sprint.

4.5 Data

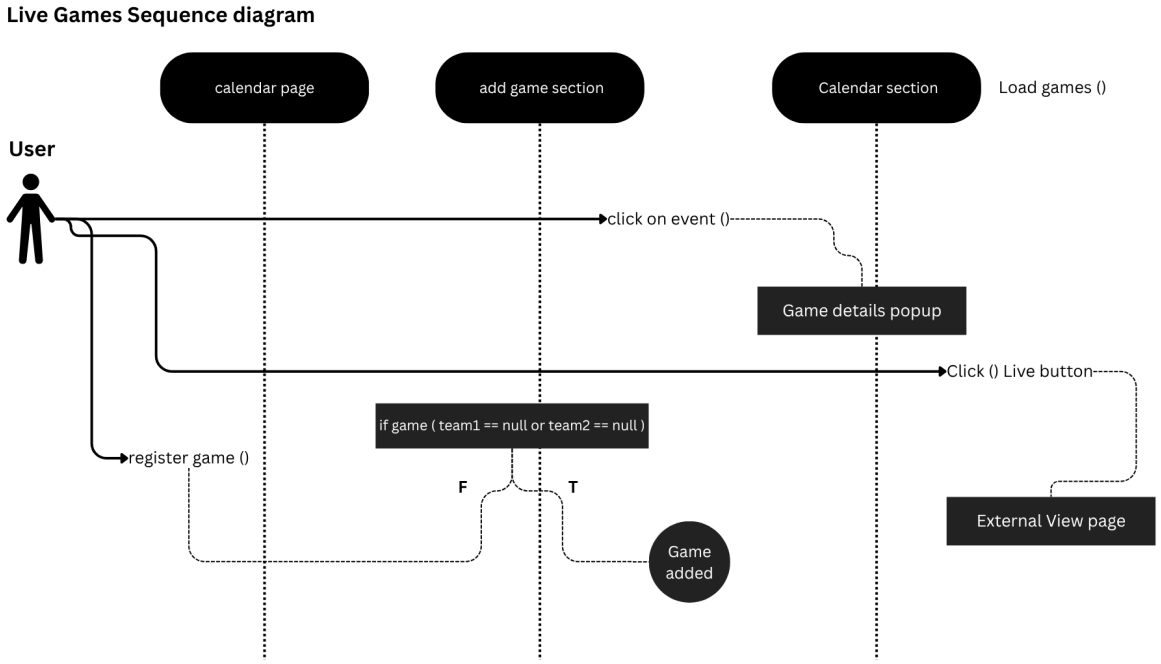
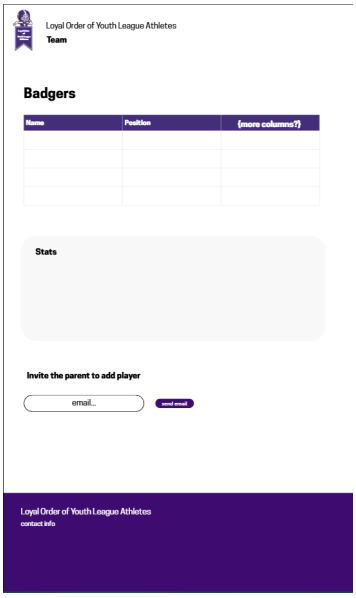


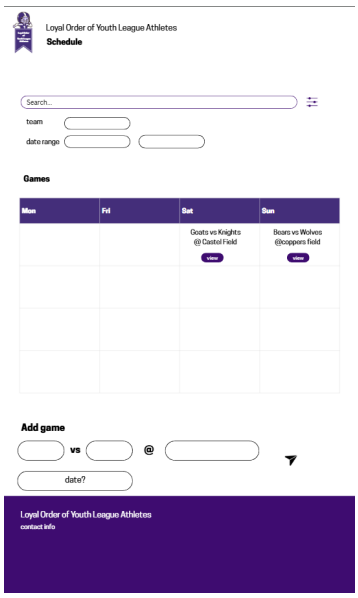
Figure 1: Live games sequence design.

4.6 UI Mocks

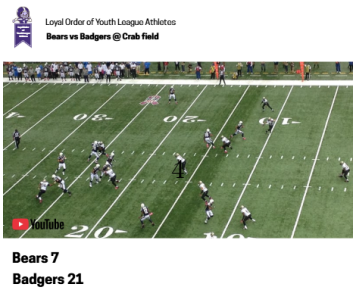
(a) The home page. Information about upcoming games and some ratings.

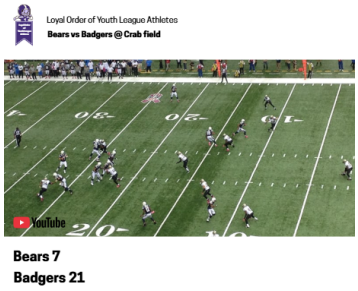


(b) Team page. View team stats and roster, coaches invite from here.



(c) Games page. View games, admins add games. Calendar and bracket views.





Live Games simplified diagram | description

live links are saved in the game attribute. These links can then be displayed throughout our site using the full calendar package.

Events for the calendar are created using the games in our data base. Once rendered the event click attribute triggers a popup, displaying the game with the live link

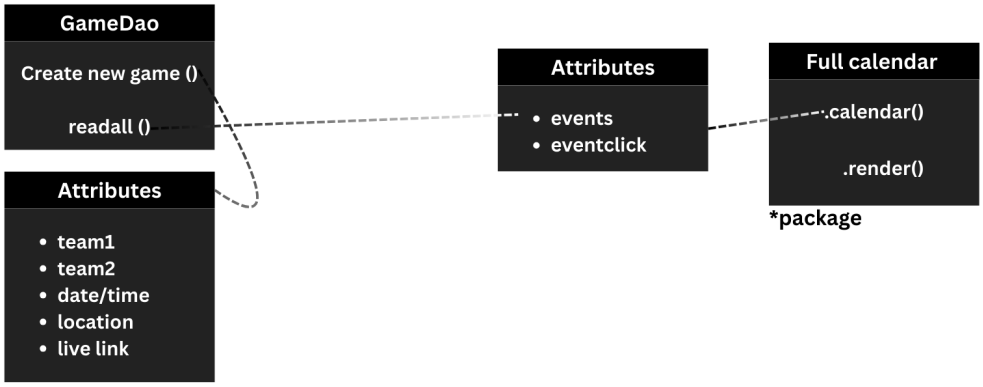


Figure 2: Class diagram for live games.

5 Iterations

5.1 Iteration Planning

Iteration	Dates	Stories	Points
1	10/09 - 10/26	U5 User Account, U15 Login/Logout, Spike Database Image Upload, C8 Youth Team - Create Team, A0 Calendar - Create Game	15
2	10/26 - 11/09	U2 Communications, U5 User Account - UI, C7 Invite Youth, A10 Youth Profile - Adult View, Tech Task UI Design, U1 Welcome Images, U15 Login/Logout - Sessions & Encryption, A6 Playoff Bracket, U4 Live Broadcast, C8 Youth Team - UI & Testing, A0 Calendar	23
3	11/06 - 11/23	U12 Game Chat, U2 Communications - Replies & Media, A10 Youth Profile, A9 Parental Consent, C7 Invite Youth, A6 Playoff Bracket, U11 Reviews, U1 Image Carousel, A3 Track Stats, U4 Live Broadcast	18
		Total:	56

Table 3: Iteration Planning for Incremental Deliveries

5.2 Iteration/Sprint 1

5.2.1 Planning

- Loren and Brenden:

Stats simplified diagram | Description

Stats are structured by type (tackle, touch down etc), the player who made the play & the game it was done in. On the Game details page, a admin can add stats corresponding to players on either type and those stats will be added to that users list of stats. The game id will be stored in the url on the page --- making it easy to associate a users stat with a specific game.

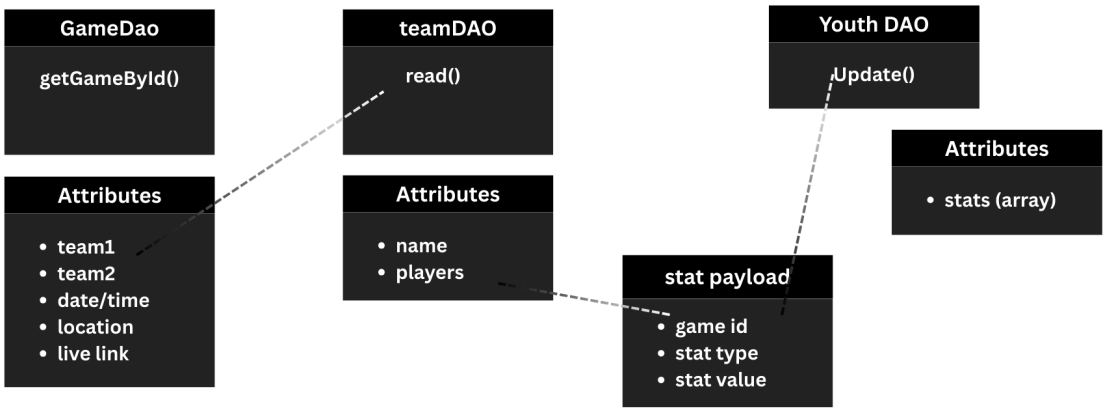


Figure 3: Class diagram for live stats.

1. (U5) User Account: 8 points

- Oscar:
  1. (Spike) Images to Database: 2 points;
  2. (U15) Login/Logout: 1 point
- Reece:
  1. (C8) Youth Team - Create Team: 1 point
  2. (A0) Calendar - Create Game: 1 point
  3. (Spike) Calendar and event handling: 2 points

Total: 15 points

5.2.2 Work Done

Loren and Brenden:

Oscar: The login/logout functionality was not totally finished. The actions of logging in and logging out worked, but saving the info in a session to remain logged in while browsing the site was not completed. The spike was completed, though.

Reece:

5.2.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

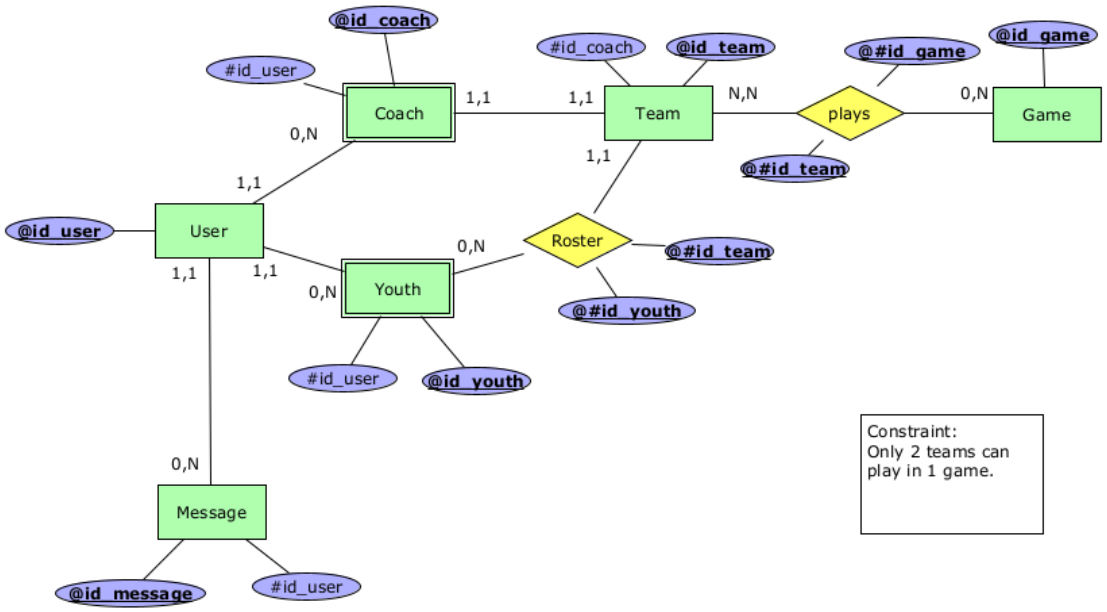
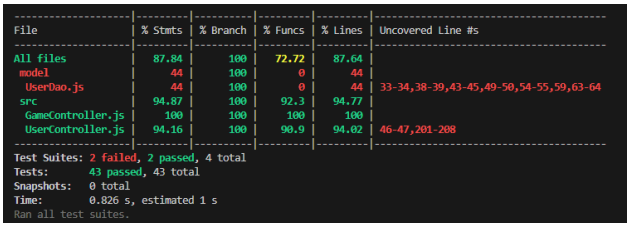


Figure 4: ERD for the data structure.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	87.84	100	72.72	87.64	
model	44	100	0	44	
UserDao.js	44	100	0	44	33-34,38-39,43-45,49-50,54-55,59,63-64
src	94.87	100	92.3	94.77	
GameController.js	100	100	100	100	
UserController.js	94.16	100	90.9	94.02	46-47,201-208

Test Suites: 2 failed, 2 passed, 4 total
Tests: 43 passed, 43 total
Snapshots: 0 total
Time: 0.826 s, estimated 1 s
Ran all test suites.

Figure 6: Sprint 1 Testing Coverage

5.2.4 Retroespective & Reflection

Brenden
The setup of the project was much more difficult than I anticipated. The process itself is not that long, but it was a headache trying to understand how it work and implement it in our own project. That is the major thing I learned: How to set up a web server and what the interactions between the server and client look like. Another pitfall was the UI. I was not aware of how crucial UI would be, and did not focus on it as much as I needed to this iteration.
Oscar
I definitely underestimated the amount of work/difficulty of work for this project. For this sprint, each task that I had to work on weren't too hard on their own, but having to learn and adjust while writing the code was still difficult. In the coming weeks, when the work will get harder, I need to manage my time better.

5.3 Iteration/Sprint 2

5.3.1 Planning

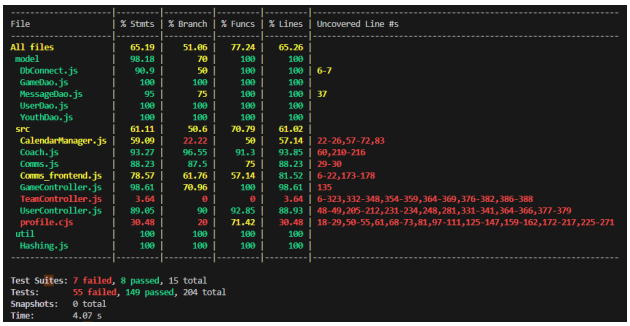
- Loren:

1. (U2) Communications: 8 points
2. (U5) User Account - Finish UI: 1 point

- Brenden:
  1. (C7) Invite Youth: 3 points
  2. (A10) Youth Profile - Adults can view youth profiles: 1 point
  3. Technical Task Webpage UI Design: 1 point
- Oscar:
  1. (U1) Welcome Images: 3 points
  2. (U15) Login/Logout - Finish password encryption and user sessions: 1 point
  3. (A6) Playoff Bracket -¿ UI and interactivity: 1 point
  4. Refactoring: 1 point
- Reece:
  1. (U4) Live Broadcast: 3 points
  2. (C8) Youth Team - Finish UI and tests 1 point
  3. (A0) Calendar - Create game: 1 point,
  4. (A0) Calendar - Create calendar: 2 points
  5. (A0) Calendar - Add game to calendar: 1 point

5.3.2 Work Done

Loren
I partially completed communications up to 8 points. I did the full crud, including UI and testing, and need to finish the reply option for other users to reply to messages. I completed the UI for user account. I also completed the UserDao tests (which I counted as less than a point, but wanted to keep track that it was completed)
Brenden
I completed my technical task and part of A10 Youth Profile. I enabled adults to view youth profiles and ended up spending a lot of time creating youth accounts and profiles themselves. Profiles have dynamic links a DAO was created for a youth account. There isn't much left to do for youth profiles besides individualized information such as a youth players game statistics, so I'd bump the points up to 2 for this iteration. I've also partially completed C7 Invite Youth. The system I ended up going with is one where the coach promotes an adult, the adult creates the account for the youth, and the coach can see all youth not in a team and add them to theirs. It all works except that the button to add doesn't add them currently, the models for teams/coaches need a bit more work before it is ready for that. What is left to do isn't difficult, so I would give myself 2 points for this task, totaling to 4 points.
Oscar
I completed the Login/Logout functionality that I had leftover from iteration 1, that being password encryption and saving user info in sessions. On the profile page, there is a script that shows that /loggeduser continues to return the user, even after reloading. It doesn't actually show user information yet, but that is more of a semantic issue. I have partially completed the Welcome Images section. The website does link to the photos stored in the db, and the images show up in the image carousel. However, I still need to add admin control over what images are seen in the carousel. Finally for the playoff bracket, I set the UI and added some interactvity. I still need to link the teams in the playoff bracket to the teams in the league, I need to allow only admins to update the bracket, and I need the info to be stored through reloads. **Reece**
I finished up my work on the Game, team implementations. Then I did two spikes one for figuring out how to do a calendar and another for how to do live games in our web app. I used a package called full calendar in js and for the live feeds just leveraged embeddings.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	65.19	51.06	77.24	65.26	
model	98.18	70	100	100	
DbConnect.js	90.9	50	100	100	6-7
GameDao.js	100	100	100	100	
MessageDao.js	95	75	100	100	37
UserDao.js	100	100	100	100	
YouthDao.js	100	100	100	100	
src	61.11	50.6	70.79	61.02	
CalendarManager.js	59.09	22.22	50	57.14	22-26,57-72,83
Coach.js	93.27	96.55	91.3	93.85	60,210-216
Comms.js	88.23	87.5	75	88.23	29-30
Comms_frontend.js	78.57	61.76	57.14	81.52	6-22,173-178
GameController.js	98.61	70.96	100	98.61	135
TeamController.js	3.64	0	0	3.64	6-323,332-348,354-359,364-369,376-382,386-388
UserController.js	89.05	90	92.85	88.93	48-49,205-212,231-234,248,281,331-341,364-366,377-379
profile.cjs	30.48	20	71.42	30.48	18-29,50-55,61,68-73,81,97-111,125-147,159-162,172-217,225-271
util	100	100	100	100	
Hashing.js	100	100	100	100	

Test Suites: 7 failed, 8 passed, 15 total
Tests: 55 failed, 149 passed, 204 total
Snapshots: 0 total
Time: 4.07 s

Figure 7: Sprint 2 Testing Coverage

5.3.3 Testing Coverage

Loren
I think this coverage for model and util is good enough. I do think that half of the files in src are good enough, while the other half will need to increase in coverage. I worked on the coverage for MessageDao, UserDao, Coach, Comms. I can increase the coverage in Coach and Comms by writing tests for the remaining 6-7 lines uncovered. Comms-frontend.js is only at 81% because it was more difficult writing the tests that had the html component, document, etc. So I only wrote for some of the functions that were repetitive (ie. Successful delete, delete error code, etc)
Brenden
Coverage on my end was okay. I worked a lot on front-end Javascript and had trouble figuring out the testing for it. I ended up stepping back from those tests and improving coverage of some of the weaker scripts. Coverage for other things like UserController and YouthDao turned out well. I'll hopefully be able to tackle the testing issues this next iteration.
Oscar
The password encryption tests are successful, but there is an issue with the login/logout tests. Even though logging in does work on the website, there is an issue with the test setup that I could not fix after hours of debugging. Furthermore, for the tests that require password hashing, the tests have an issue with calling the password hashing function, even though it is correctly imported in the test file. **Reece**
Test coverage went well on my end. Had some trouble thinking about testing my calendar implementation but beyond that nothing too pressing. I must however make a point of testing early and often.

5.3.4 Retrospective & Reflection

Loren
The challenges were adding delete and update especially as buttons in the UI. The hardest part was trying to convert the script from the html code into js. I didn't expect it to be difficult, but it was very tricky getting it to work with the posts and document. I did write the test cases so I can try to go back and make the js frontend work with the html. I can avoid this mistake by planning better in the future. I learned how to add buttons in the messages based on ID as well as updating a message both in the database and in the frontend. I started writing everything in the Comms.js and quickly realized that a lot of the logic would also be in the html with the delete/update buttons as well as the view/hide toggling. I learned how to write the tests for posts in Jest as well as start working with error messages. I learned a lot in this iteration and will learn from my mistakes and successes to code better in the next iteration.
Brenden
This iteration was pretty fun. I enjoyed putting together the UI for the profile page and creating the navbar using Bootstrap. What really frustrated me was the testing. I'll be sure to test along the way next time because handling the tests afterward is a huge headache. Another issue I was having was thinking about how I'd handle inviting youth players. I think going to the client for a clearer idea would have been good.
Oscar
This iteration went much better than the previous. That is mostly because of the time I spent working on

the project. I spent a long time trying to add admin control to the image carousel, but I wasn't able to figure it out. That will be a primary focus for the next iteration. I am still having trouble with my test writing, both in terms of coverage and effectiveness. I need to write more tests immediately after adding new functionality, so I don't fall behind in the future. **Reece**
This iteration went nice. More clarity on what to do and to what level of detail. One thing is that I have to make a habit of testing after developing so that work doesn't pile up.

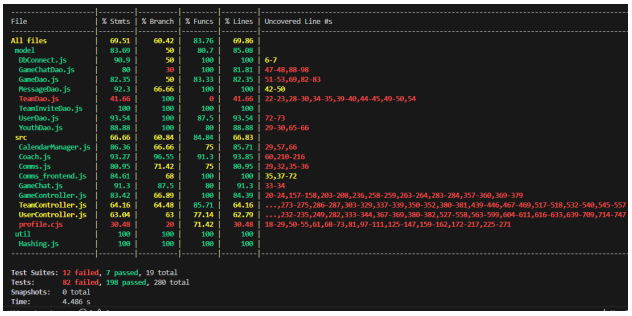
5.4 Iteration/Sprint 3

5.4.1 Planning

- Loren:
    1. (U12) Game Chat: 5 points
    2. (U2) Comms Reply: 1 point
    3. (U2) Add images/videos in chat: 1 point
- Brenden:
    1. (A10) Finish Youth Profile: 2 points
    2. (A9) Parent Consent: 1 point
    3. (C7) Finish Invite Youth: 2 points
- Oscar:
    1. (A6) Playoff Bracket: 3 points
    2. (U11) Reviews: 2 points
    3. (U1) Finish Image Carousel: 1-2 points
- Reece:
    1. (A3) Track Stats: 2 points
    2. (U4) Finish other work from iteration 2: 2 points
    3. (U16-New story) Integrating teams with coach profile: 2 points

5.4.2 Work Done

Reece
I worked on finishing up things from the last iteration and implementing a track stats method/ extending what is possible with the live games.
Brenden
I worked on parental consent, invite youth, and team/game/coach reworks this iteration. In order to have all the different entities connected and have invites work how I wanted them to, some big changes had to be made to the way our data was set up, and that ended up taking a lot of time to rework. I was not able to focus on youth profile improvements.
Oscar
I completed the Reviews and the Image Carousel. The reviews are very similar to regular messaging, expect that it is public and reviews can be sorted by the team they are reviewing. Only logged users can review a team, but anyone can view and filter them. For the Image Carousel, the images are fed in from the database with a script, and there is a separate page where you can upload and delete images to have them appear in the carousel. I did not completely finish the Playoff Bracket. I have it linked to the teams in the database, and the teams are sorted by total wins and placed accordingly in the bracket, and there is some interactivity with clicking on teams. However, I did not add a way for an admin to permanently update the bracket and allow other users to view the changes.
Loren



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	69.51	60.42	83.76	69.86	
model	83.69	50	80.7	85.08	
DbConnect.js	90.9	50	100	100	6-7
GameChatDao.js	80	30	100	81.81	47-48,88-98
GameDao.js	82.25	50	83.33	82.35	51-52,69,82-83
MessageDao.js	92.3	66.66	100	100	42-50
TeamDao.js	41.66	100	0	41.66	22-23,28-30,34-35,39-40,44-45,49-50,54
TeamInviteDao.js	100	100	100	100	
UserDao.js	93.54	100	87.5	93.54	72-73
YouthDao.js	88.88	100	80	88.88	29-30,65-66
src	66.66	60.84	84.84	66.83	
CalendarManager.js	86.36	66.66	75	85.71	29,57,66
Coach.js	93.27	96.55	91.3	93.85	60,210-216
Comms.js	80.95	71.42	75	80.95	29,32,35-36
Comms_Frontend.js	84.61	60	100	100	35,37-72
GameChat.js	91.3	87.5	80	91.3	33-34
GameController.js	83.42	66.89	100	84.39	20-24,157-158,203-208,236,258-259,263-264,283-284,357-360,369-379
TeamController.js	64.56	64.48	85.71	64.16	...,273-275,280-287,303-329,337-339,350-352,380-381,439-446,467-469,517-518,532-540,545-557
UserController.js	63.08	63	77.14	62.79	...,232-235,240,282,333-344,367-369,388-382,527-550,563-599,604-611,616-633,639-709,714-747
profile.cjs	30.48	20	71.42	30.48	18-29,50-55,61,68-73,81,97-111,125-147,159-162,172-217,225-271
util	100	100	100	100	
Hashing.js	100	100	100	100	

Test Suites: 12 failed, 7 passed, 19 total
Tests: 82 failed, 198 passed, 280 total
Snapshots: 0 total
Time: 4.486 s

Figure 8: Sprint 3 Testing Coverage

I completed the game chat. Users can only chat during the start/end time of the game. After the game is over, they can still view messages, but not post. I had to refactor games (id was generated rather than letting mongo create the ids, there were no start/end times, and the date was not being processed correctly). I also worked on posting photos/videos and being able to view them from the data base in view messages. Finally, I worked on adding a reply option. Users can reply to anyone (including themselves). The reply is saved below the message using the authors name and time with shading to differentiate the reply.

5.4.3 Testing Coverage

This coverage is okay. There is a slight improvement from last iteration, but that is not including many of the newer front end JavaScript files that are in the view folder. Front end testing has been harder to replicate than back end. Most areas that lack testing on the front end are due to not giving myself (Brenden) enough time to test.

5.4.4 Retroespective & Reflection

Oscar
I thought that this was my best iteration. Throughout this sprint, I became much better at understanding all of the code and how it works together. This made adding new features easier and less stressful. Also, since most of the foundation was done in the previous cycles, adding new features was mostly making modifications to things already present. I did slightly better on testing, but not by much. Unfortunately, I am still behind on testing, and I wasn't able to make the time to catch up while staying on schedule with the new stuff I needed to add. However, if I wrote new code, I made sure to add testing.
Loren
This iteration seemed much harder than the last because I was working on someone else's code. It was much harder to integrate into previously written code than to write the class myself. There were bugs in the code which ended up taking me a really long time to find all the bugs and fix them. Its also more difficult than when we initially started because I had to comb through much more code to fix the bugs. I think I learned the most through this iteration. I started with saving photo/video one way and realized half way through that it was the wrong approach. I also learned the importance of debugging early on in the code.
Brenden
This iteration was difficult. Similarly to Loren, I had to work with someone else's code. I decided to change the way our games, teams, and youth interact in the back end, and that affects alot of different scripts. I should have given myself more time to make this transition, I got too hung up on whether I wanted to do it or not. The code base is getting larger and larger, and it is becoming more and more apparent that good organization is crucial for a project. Something cool I learned that I will mention is custom HTML attributes. I used them to hide certain groups elements from different types of users.

6 Final Remarks

6.1 Overall Progress

[Have you completed everything? If so, present evidence on how you brought value to your client, and the overall client satisfaction. Otherwise, estimate how much progress you done and how long it would take to finish this project.]

6.2 Project Reflection

[Your personal reflection on the project. What lessons did you learned. What would you have done differently. How can you do better work in future projects? You may write this as a team or per person (or both)]

Appendix

[Appendix section if needed]