CS482 Software Project Proposal: YBT Sports League

Joel Robinson, Ekvtime Itchirauli, Yohann Gouin, Nishant Gurung

2025-12-07

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. Nguyen Ho
- Client title: Assistant Professor
- Client email address: tnho@loyola.edu
- Client employer:
- How you know the client:

2 Project Description

2.1 Overview

[Add a few paragraphs describing your project succinctly. What problem are you trying to solve, what is the purpose of your project? Why does your client want this project?]

Our client wants to build an interactive website that'll help her and the organization in planning and schedule tournaments while providing an online experience that'll allow fans to engage with the community and watch their favorite teams.

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.]

- Support 6-month seasons.
- Organize basketball tournaments targeting kids underage.
- Up to 50 teams (5–10 players each).
- One team manager maximum per team.
- Randomized, continuously updated brackets for seasons.
- Nationwide events with location, time, and date scheduling.
- Ability to buy tickets for games online.
- Create/manage teams.

- Sign up self and children.
- View/post photos (only visible to registered users).
- Livestreaming of games (live and archived).
- Comments, likes, dislikes on matches and posts.
- Display of game schedules.
- Bracket generation and continuous updates.
- Promote kids/teen sports through the platform.

2.3 Why this Project is Interesting

Building a modern, fully integrated tournament management system allows us to solve a real community problem while applying a wide range of computer science skills—from full-stack development and database design to role-based security, distributed systems, and UI/UX. People should care about this project because it supports youth involvement in sports, improves organizational fairness and transparency, and provides a professional, reliable platform for small leagues that typically do not have access to such advanced tools

2.4 Areas of CS required

Combines a React front end with an Express/Node.js backend to deliver a dynamic, role-driven tournament platform. Database systems play a central role through complex MongoDB schema design, indexing, aggregation pipelines, and secure credential storage. The architecture also incorporates distributed systems concepts, with a cloud-hosted MongoDB Atlas instance, cross-origin communication, and asynchronous client–server interactions. Strong software engineering practices appear throughout the project, including modular controllers, middleware pipelines, unit testing with Jest, and support for maintainability and scalability. Additional subfields such as computer security (session management, bcrypt hashing, RBAC), algorithms and data structures (bracket organization, event/match filtering, efficient lookups), and human–computer interaction.

2.5 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	Salting	Performance	Passwords shall be stored using strong one-way hashing (bcrypt) with a minimum of 10 salt rounds.
NFR2	Authentication	Security	All authenticated sessions shall be securely stored via express-session using encrypted cookies and/or a MongoDB-backed session store.
NFR3	Connection	Security	All authenticated sessions shall be securely stored via express-session using encrypted cookies and/or a MongoDB-backed session store.
NFR4	RBAC Model	Security	The system shall enforce role-based access control (RBAC), ensuring Admin-only pages and endpoints (e.g. /api/admin/...) are protected by middleware.
NFR5	System Availability	Reliability	System shall target 99.5 percent availability during tournament seasons.
NFR6	MongoDB	Reliability	MongoDB write operations shall maintain data consistency using transactional updates where applicable (e.g., multi-document writes for children + links).
NFR7	Query Fields	Performance	Frequently queried MongoDB fields (e.g., eventId, teamA, teamB, status) shall be indexed to ensure efficient query execution.
NFR8	Consistent Design	Usability	The UI shall maintain a consistent design using the established YBT color palette (black, red, yellow) and Bootstrap components
NFR9	Modular Architecture	Maintainability	The system shall follow a modular architecture separating controllers, models, routes, and middleware to support long-term maintainability.
NFR10	Statement Coverage	Maintainability	The project shall maintain greater than 70 percent statement coverage using Jest for backend logic (controllers, middleware, DB helpers).
NFR11	Feedback	Usability	Users shall receive clear visual feedback on successful or failed actions (e.g., form submissions, errors, login attempts).
NFR12	System Logs	Observability	System logs shall capture all authentication events, match creation, role updates, and errors with timestamps.
NFR13	Data Backup	Observability	MongoDB shall be backed up daily during active tournament seasons.

Table 1: Non-Functional requirements

3.2 Functional Requirements (User Stories)

ID	Story Title	Points	Description
S0	Scheduling	8	As a/an Admin, I want to maintain a seasonal event schedule, so that I can schedule competitions for teams.
S1	Signing Up Children	2	As a/an Adult, I want to register my children on the website, so that they can be assigned a team for basketball
S2	Team Manager Volunteer	2	As a/an Adult, I want to volunteer as a team manager, so that I can assist coaching teams.
S3	Match Bracket	2	As a/an Admin, I want a randomized match bracket , so that teams can compete in an organized manner
S4	History of Matches	3	As a/an User, I want match history and videos, so that I can watch old matches at a later date.
S5	Match Updates	5	As a/an Admin, I want the ability to create match updates, so that I can relay cancellations, delays, and other issues about a match.
S6	Sponsor	1	As a/an Sponsor, I want the ability to reach out to the organization, so that I can potentially sponsor the events/organization.
S7	User Posts	2	As a/an User, I want the ability to post pictures and comments, so that I can add posts and talk about the games taking/having taken place.
S9	Team Photos and Logos	2	As a/an User, I want be able to see team logos and match photos, so that I can distinguish different teams and observe their match photos
S10	Live Stream	8	As a/an Admin, I want stream live matches, so that the viewers can follow the games from wherever they may be.
S11	Live Chat	5	As a/an User, I want a live chat box , so that the viewers can talk to each other.
S12	Archive Folder	2	As a/an Admin, I want an archive folder, so that I can save the live streams and images
S13	Buying Tickets	2	As a/an User, I want to buy tickets online securely, so that I don't waste time when I reach the venue
S14	Contact Page	2	As a/an Admin, I want a contact page or section for users , so that they can reach out to us or the sponsors
S16	Guest Viewing Livestreams	1	As a/an Guest, I want to have the option to view a livestream, so that making an account is will not be necessary
S17	Display Match Info	1	As a/an User, I want to see where and when the match is being held, so that I can know what time to watch or potentially attend the game in person
S19	Sponsor Info	2	As a/an Sponsor, I want to display my info as a sponsor, so that I can properly advertise and support the league/team
S20	Admin Support Messages	3	As a/an Admin, I want to read and change statuses on messages, so that I can properly prioritize and mark off support tickets.
S21	Registration	2	As a/an User, I want to register as different user types, so that I can be identified as a child/adult/sponsor
S22	Admin CRUD	8	As a/an Admin, I want to add users, edit their info, and delete users., so that I can have access to editing from the website.

Table 2: Functional requirements as User Stories.

4 System Design

4.1 Architecture

Our team is following a **Layered (n-tier) Architecture** combined with elements of the **Model-View-Controller (MVC)** pattern to maintain a clean separation of concerns, improve scalability, and support future feature expansion. The system is composed of the following main modules:

1. **Presentation Layer (Frontend / Client-Side)**

   Provides the main user interface for players, coaches, parents, and administrators.

   Developed as a responsive web application using *JavaScript*.

   Handles user interaction, navigation, and data visualization (e.g., brackets, live scores, and media galleries).

   Communicates with backend services via *RESTful APIs* or *GraphQL* endpoints.

2. **Application Layer (Backend / Server-Side Logic)**

   Built using *Node.js with Express*.

   Implements all core business logic such as user authentication, bracket generation and updates, live event scheduling, and content management.

   Coordinates data flow between the frontend, database, and external services.

   Manages roles and permissions (e.g., admin, coach, player, viewer).

3. **Media and Live Streaming Module**

   Handles *live video streaming* for games and events using integrated streaming services such as YouTube (idk if we can use this, feel free to change).

   Manages *video and photo archives*, allowing users to view past games, highlights, and team media galleries.

   Integrates with *cloud storage* (or other) for efficient and scalable content hosting.

4. **Bracket Management Module**

   Provides tools for creating, updating, and displaying tournament brackets in real-time.

   Supports automatic bracket progression based on game results entered by administrators or referees.

   Integrates with the database to ensure live updates and accurate standings.

5. **Data Access Layer**

   Responsible for database interactions through an ORM such as ...

   Handles all CRUD operations for users, teams, games, brackets, and media assets.

   Ensures data integrity and efficient querying.

6. **Database Layer**

   Uses a *relational database* (e.g., PostgreSQL or MySQL) to store structured data such as user profiles, team information, game results, and tournament brackets.

   A *cloud storage service* (such as...) is used to store and retrieve large media files.

7. **External Integrations**

   Integrates with *authentication providers* (e.g., Firebase Auth or OAuth 2.0) for secure login and role-based access.

   May include *email notification*(for event registrations or new logins).

   Optionally connects to *analytics tools* for monitoring website performance and user engagement.

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.]

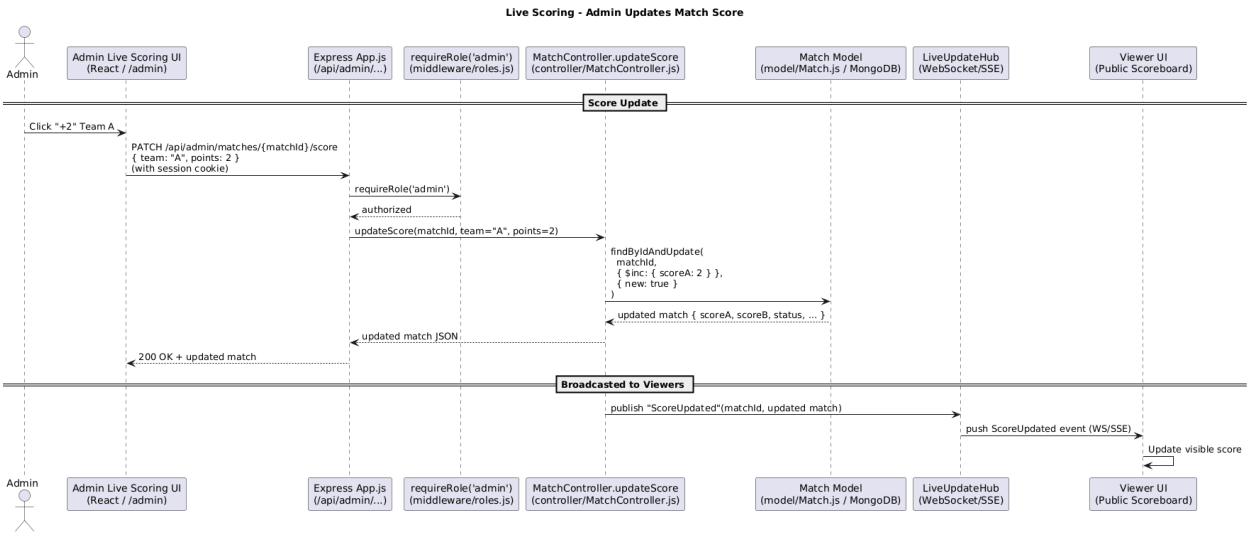


Figure 1: Score Diagram

4.3 Technology

We plan on building the website using JavaScript as our programming language. For the frontend, we'll use Bootstrap, a CSS framework for responsive UI design. For the backend, we'll use Node.js as the runtime environment which will allow us to communicate with our MongoDB database and send responses back to the browser. Finally, Jest will be used as our framework for node.js testing.

4.4 Coding Standards

The coding standards we will be following are using singular names for database entities, only allowing code with unit tests and 70% coverage and above tests. We will also be using snakecase for all method names going forward.

4.5 Data

The Data Structure is given below.

4.6 Database Schema

- **Users**
  - int id (PK)
  - string email
  - string password_hash
  - string display_name
  - string phone
  - boolean is_verified
  - datetime created_at
- **Roles**
  - int id (PK)
  - string name
- **UserRoles**
  - int user_id (FK)
  - int role_id (FK)
- **Adults**
  - int id (PK)
  - int user_id (FK)
  - string legal_name
  - string address
  - string gov_id_type
  - string gov_id_last4
  - string photo_url
- **Children**
  - int id (PK)
  - string full_name
  - date birthdate
  - string photo_url
- **AdultChildLinks**
  - int adult_id (FK)
  - int child_id (FK)
  - string relation
  - boolean is_primary
  - int consent_id (FK)
- **Consents**
  - int id (PK)
  - int child_id (FK)
  - int consenting_adult_id (FK)

  - string type
  - datetime signed_at
  - string document_url

- **Seasons**
  - int id (PK)
  - string name
  - date start_date
  - date end_date
  - int max_teams
  - int min_players
  - int max_players

- **Teams**
  - int id (PK)
  - int season_id (FK)
  - string name
  - string logo_url
  - string color_primary

- **TeamManagers**
  - int team_id (PK, FK)
  - int adult_id (FK)

- **RosterMembers**
  - int team_id (FK)
  - int child_id (FK)
  - int jersey_number

- **Venues**
  - int id (PK)
  - string name
  - string address
  - string city
  - string state
  - int capacity

- **Tournaments**
  - int id (PK)
  - int season_id (FK)
  - string name
  - string bracket_style
  - boolean seeded

- **Brackets**

  - int id (PK)
  - int tournament_id (FK)
  - datetime generated_at
  - string status
- **Matches**
  - int id (PK)
  - int tournament_id (FK)
  - int round_number
  - int bracket_slot
  - int team_a_id (FK)
  - int team_b_id (FK)
  - int winner_team_id (FK)
  - string status
- **Games**
  - int id (PK)
  - int match_id (FK)
  - int venue_id (FK)
  - datetime starts_at
  - datetime ends_at
  - int home_team_id (FK)
  - int away_team_id (FK)
  - int score_home
  - int score_away
  - string status
- **GameUpdates**
  - int id (PK)
  - int game_id (FK)
  - string type
  - string message
  - int created_by (FK)
  - datetime created_at
- **Videos**
  - int id (PK)
  - int game_id (FK)
  - int uploaded_by (FK)
  - string url
  - int duration
  - boolean is_archived
- **Livestreams**

  - int id (PK)
  - int game_id (FK)
  - string provider
  - string stream_key
  - string playback_url
  - string status
- **Photos**
  - int id (PK)
  - int uploaded_by (FK)
  - int team_id (FK)
  - int game_id (FK)
  - string url
  - string visible_to
- **Posts**
  - int id (PK)
  - int author_user_id (FK)
  - string title
  - text body
  - datetime created_at
- **Comments**
  - int id (PK)
  - string parent_type
  - int parent_id
  - int user_id (FK)
  - text text
  - datetime created_at
- **Reactions**
  - int id (PK)
  - string parent_type
  - int parent_id
  - int user_id (FK)
  - string kind
  - datetime created_at
- **Sponsors**
  - int id (PK)
  - string org_name
  - string contact_email
  - string website_url
  - string logo_url

- **Sponsorships**
  - int id (PK)
  - int sponsor_id (FK)
  - int season_id (FK)
  - string tier
  - date start_date
  - date end_date
- **SponsorInquiries**
  - int id (PK)
  - string name
  - string email
  - text message
  - string status
- **Announcements**
  - int id (PK)
  - string scope
  - int scope_id
  - string title
  - text message
  - int created_by (FK)
  - datetime created_at
- **Subscriptions**
  - int id (PK)
  - int user_id (FK)
  - string type

4.7 UI Mocks

The UI mocks are in our github repository. They feature central pages, and a login page, there is also an admin dashboard planned not shown within the UI mocks.

5 Iterations

5.1 Iteration Planning

[In CS496, you plan all iterations beforehand. In CS482, you update the planning here at each iteration.]

5.2 Iteration/Sprint 1

5.2.1 Planning

Joel: S1 Signing Up Children (2), S21 Registration (2) Nishant: S9 Team Photos and Logos (2), S14 Contact Page (2) Ekto: S11 Live Chat (5) Yohann:S13 Buying Tickets (2) and Paired Programming S11 Total Points: 15 The reasoning we started with these is because it was crucial to start with things like signups, children integrations, team formation and pages, contact pages. Live Chat seemed to be something that would take additional effort so it would make sense to get out the way first and pair program for it.

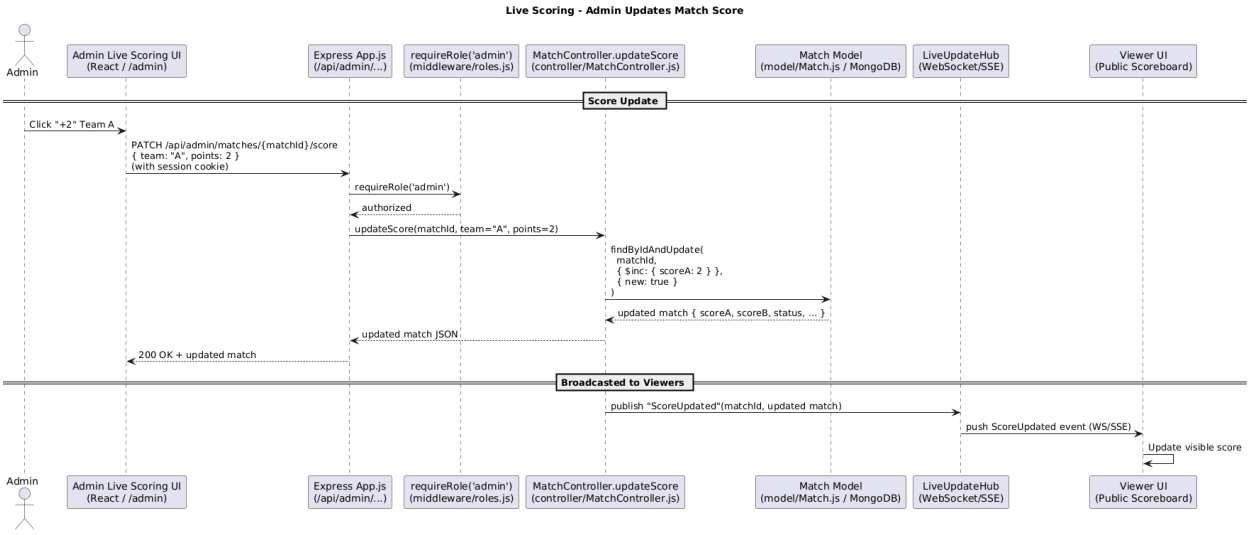


Figure 2: Score Diagram

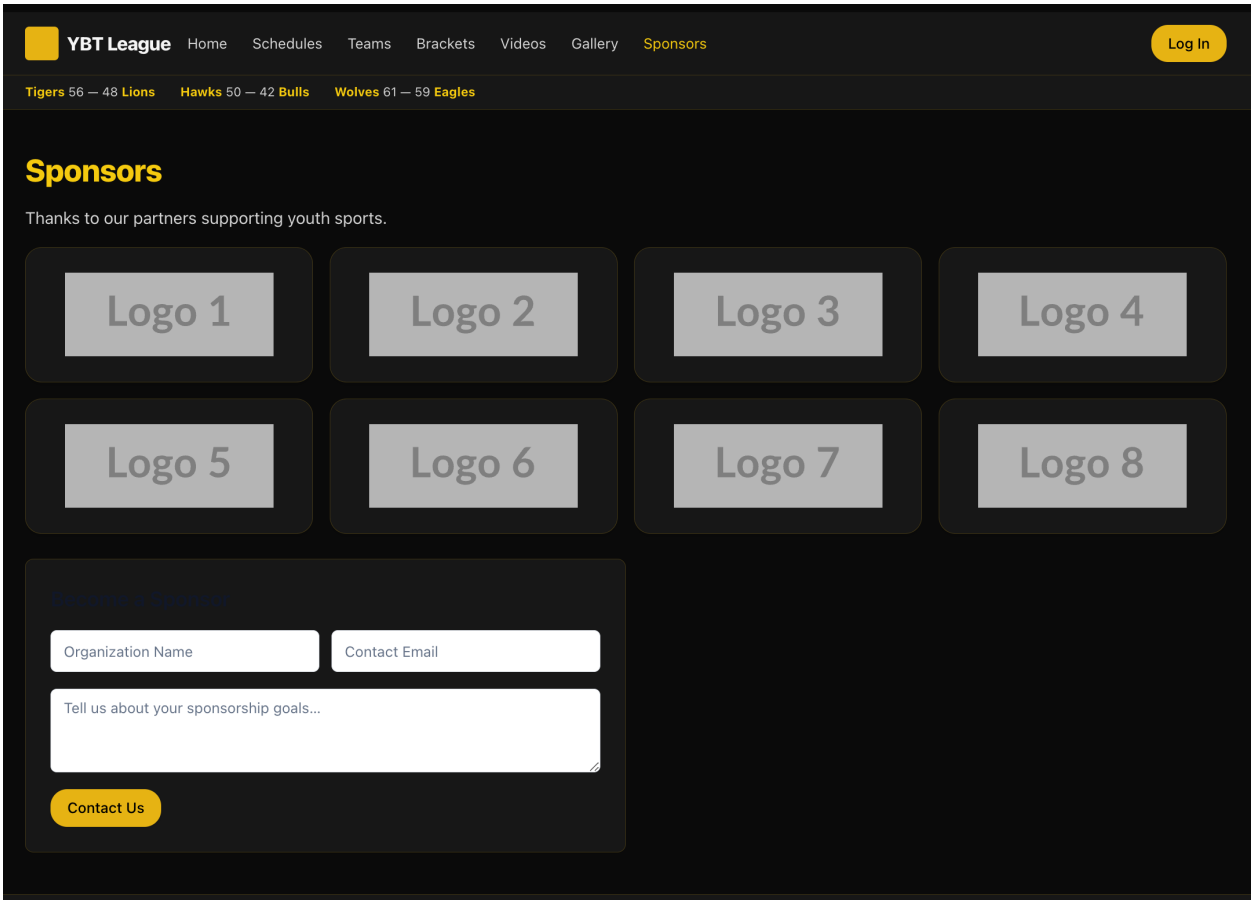


Figure 3: Sponsrs

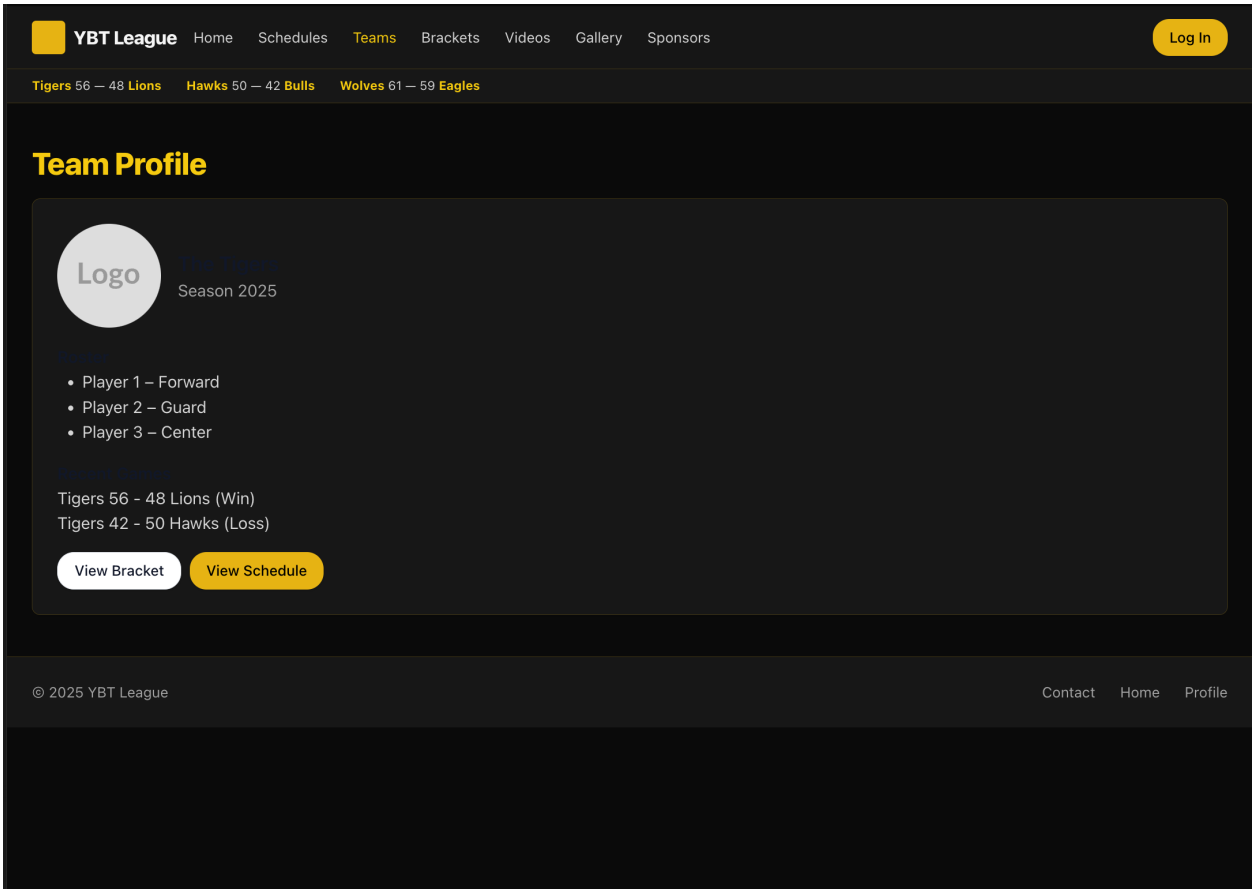


Figure 4: Team Profiles

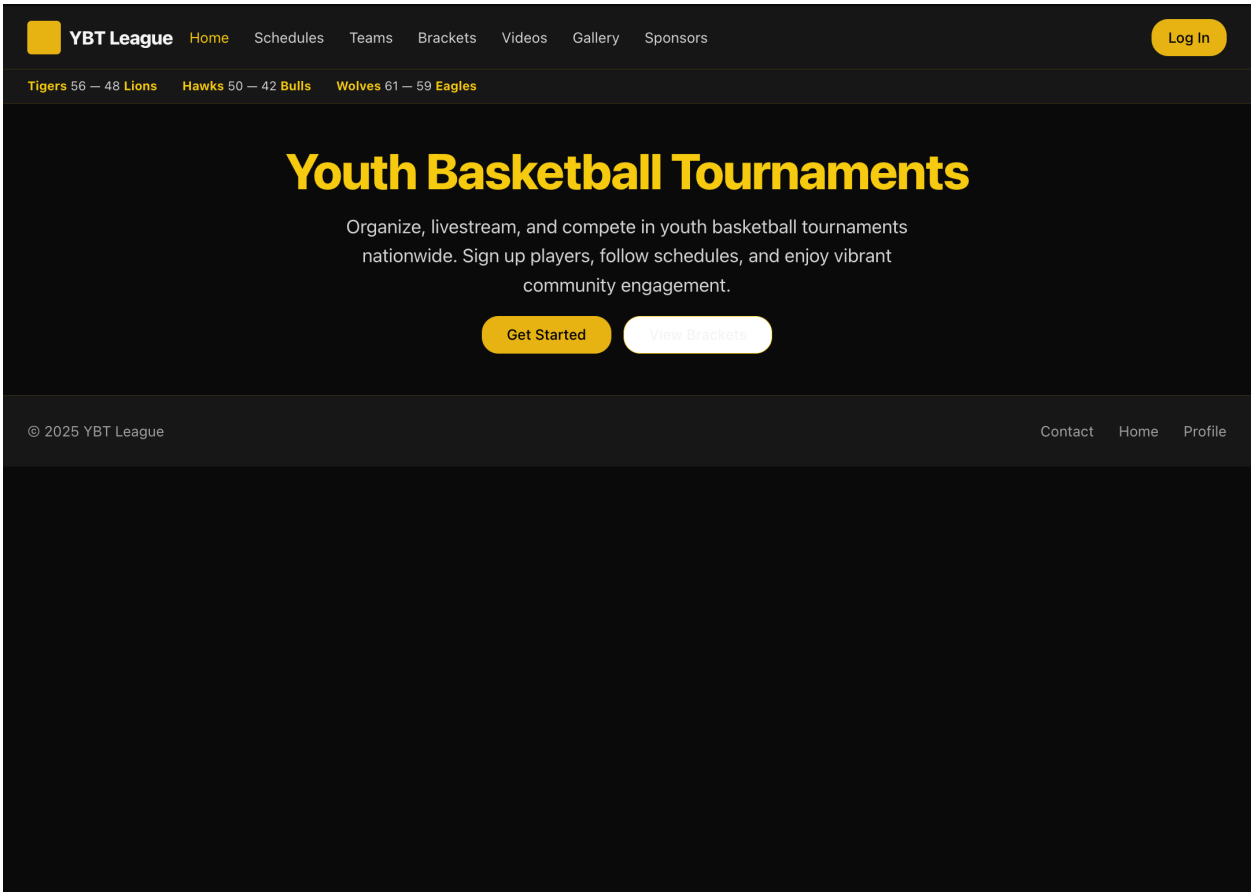


Figure 5: Main Page

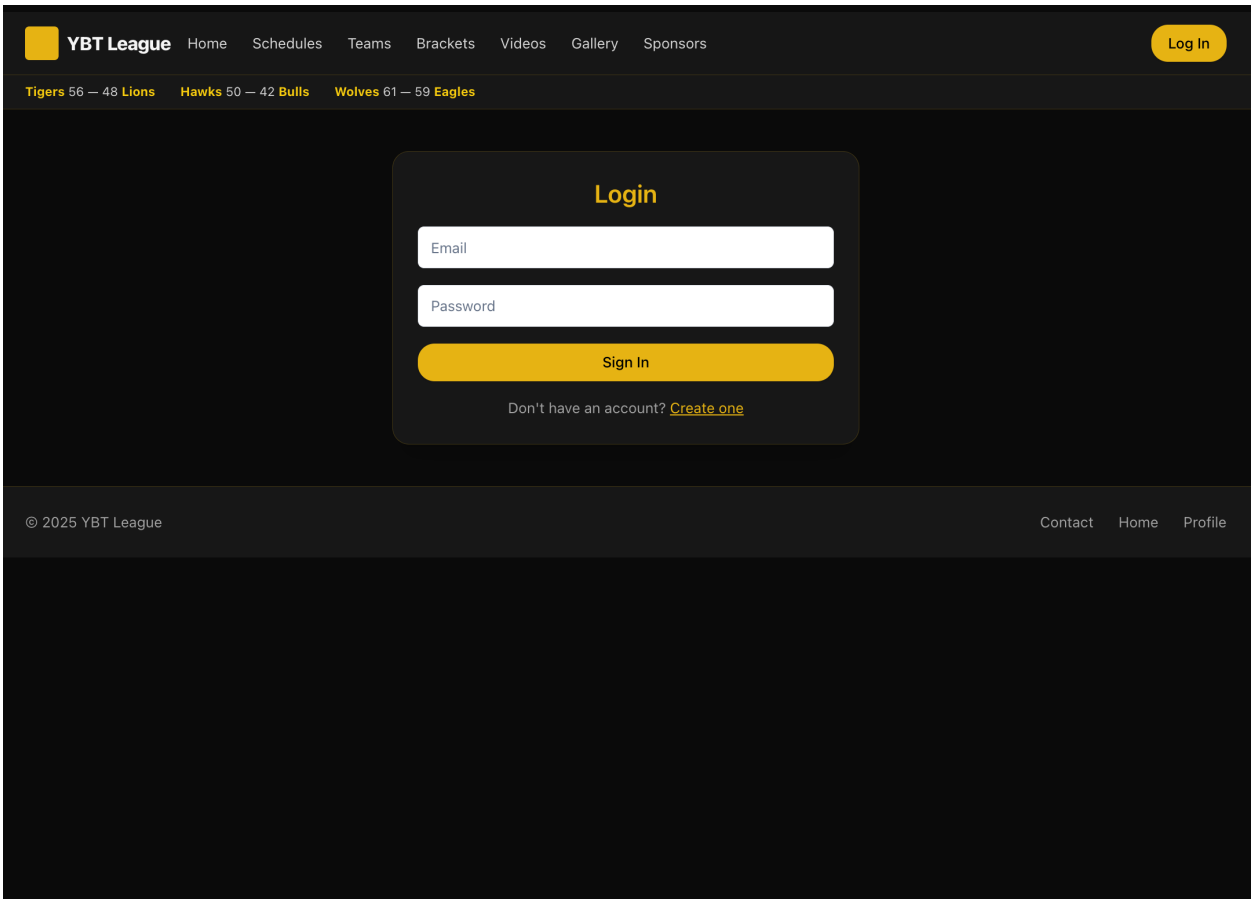


Figure 6: Login Page

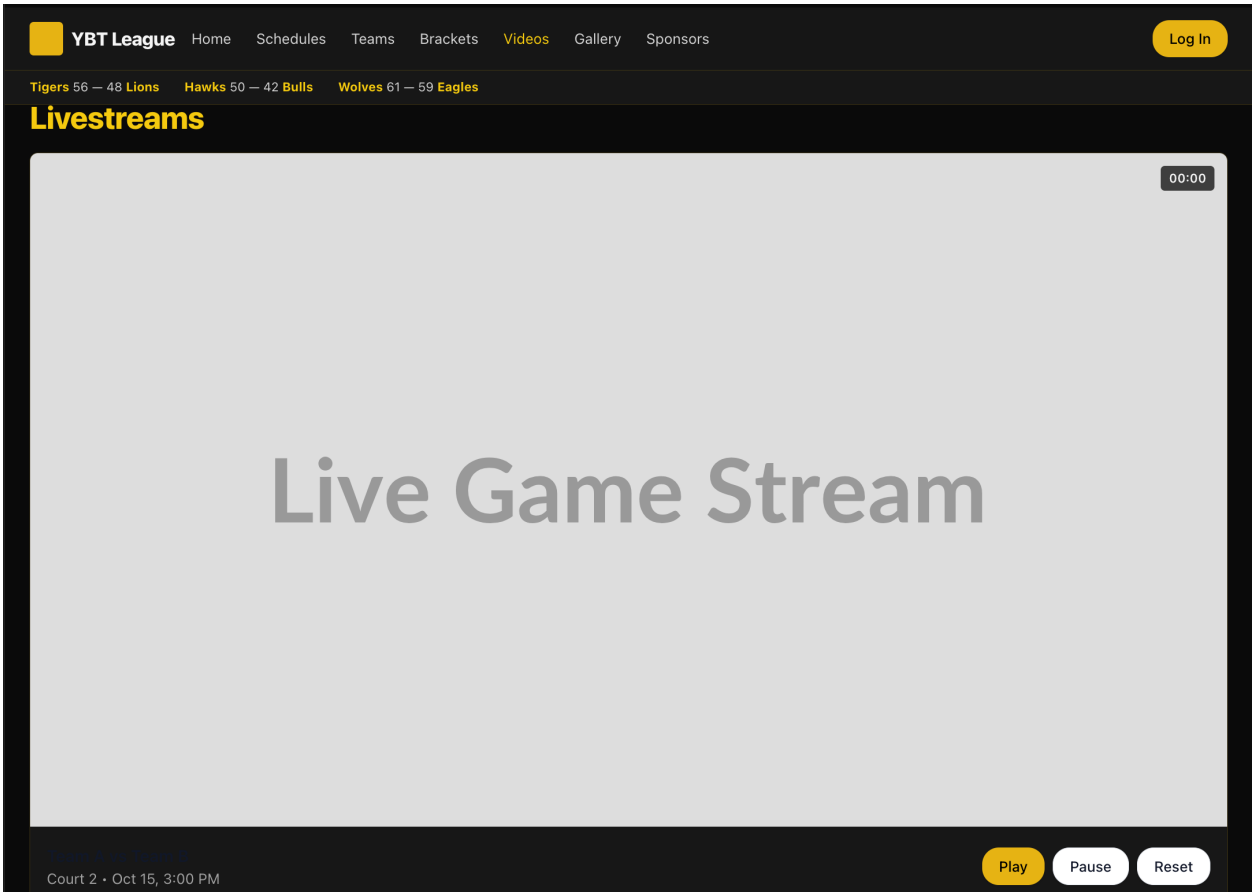


Figure 7: Livestreams

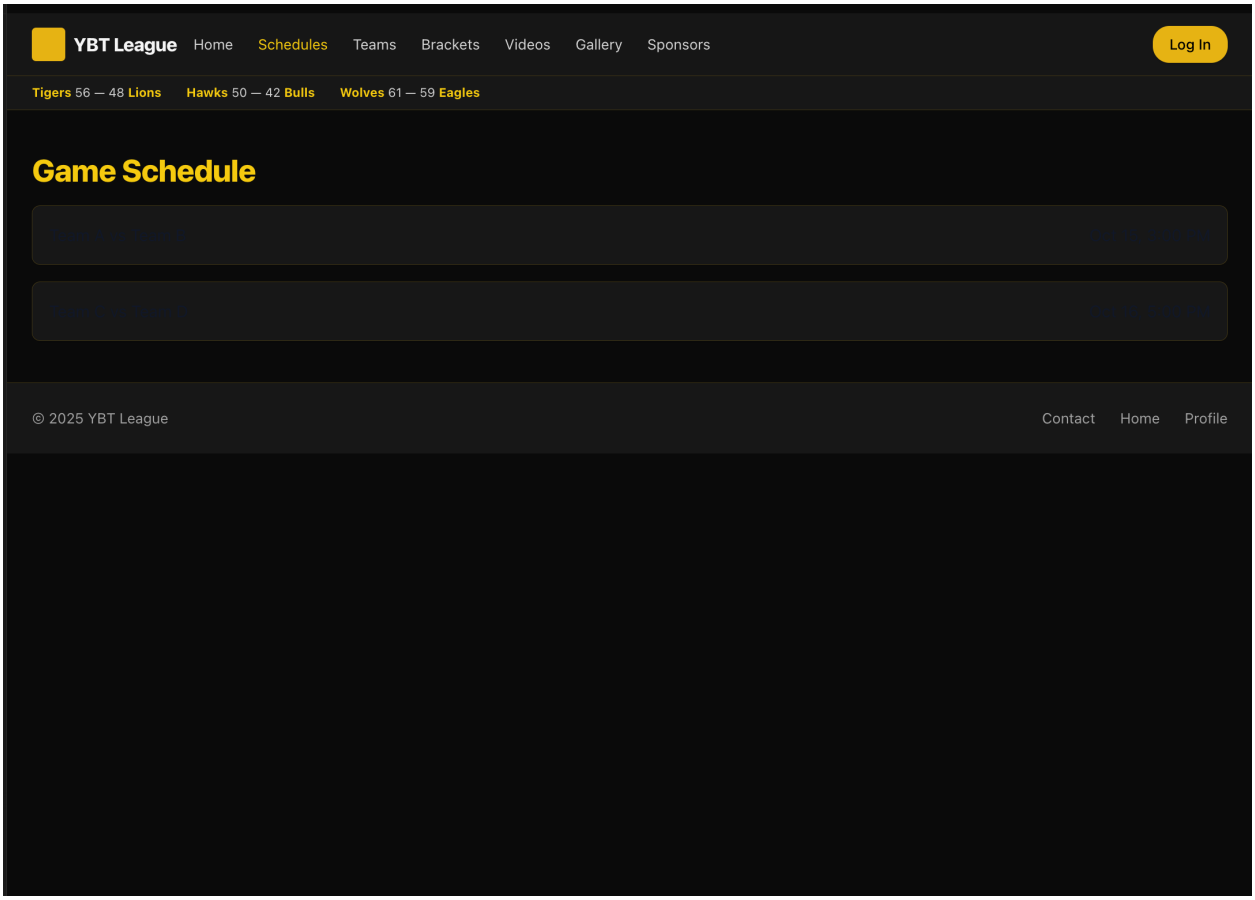


Figure 8: Game Schedules

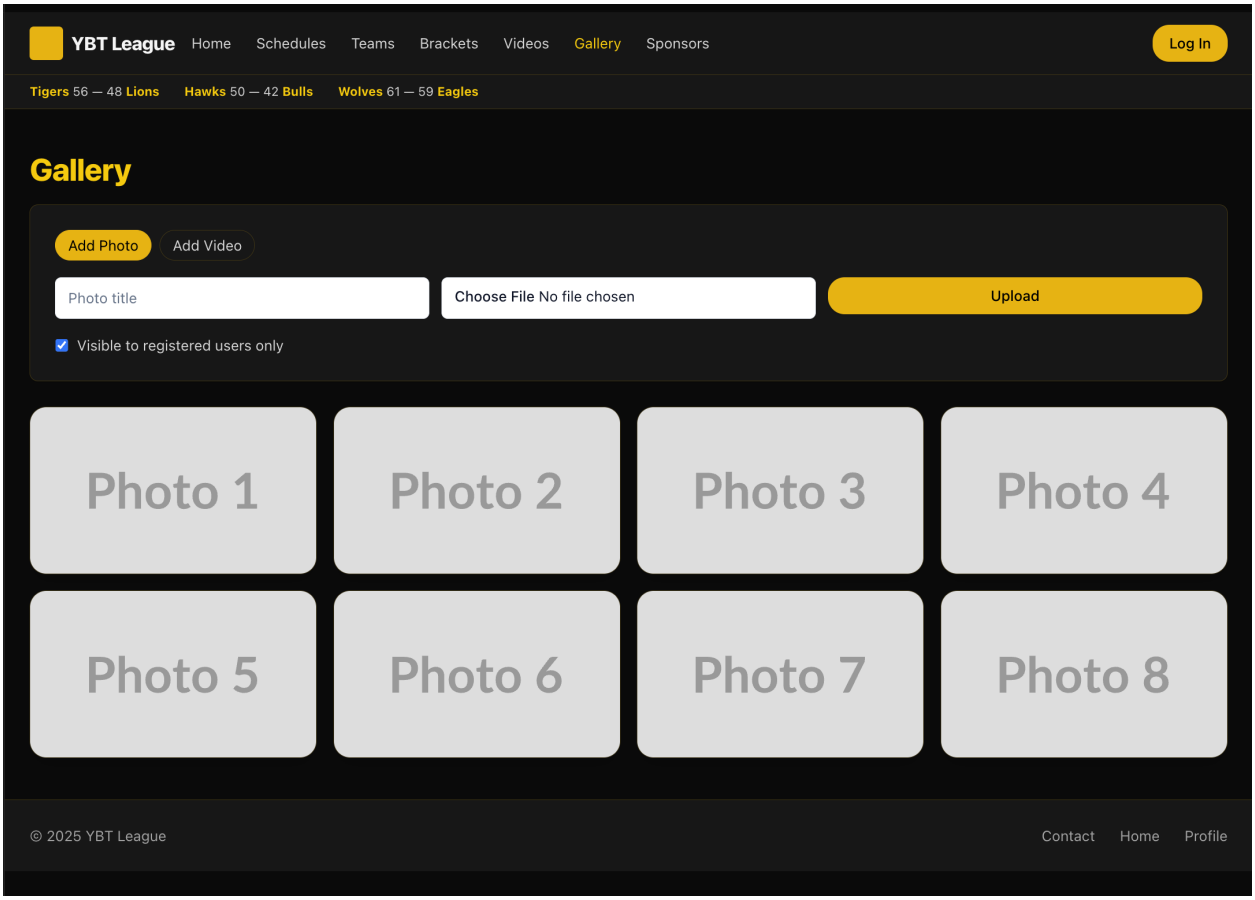


Figure 9: Gallery

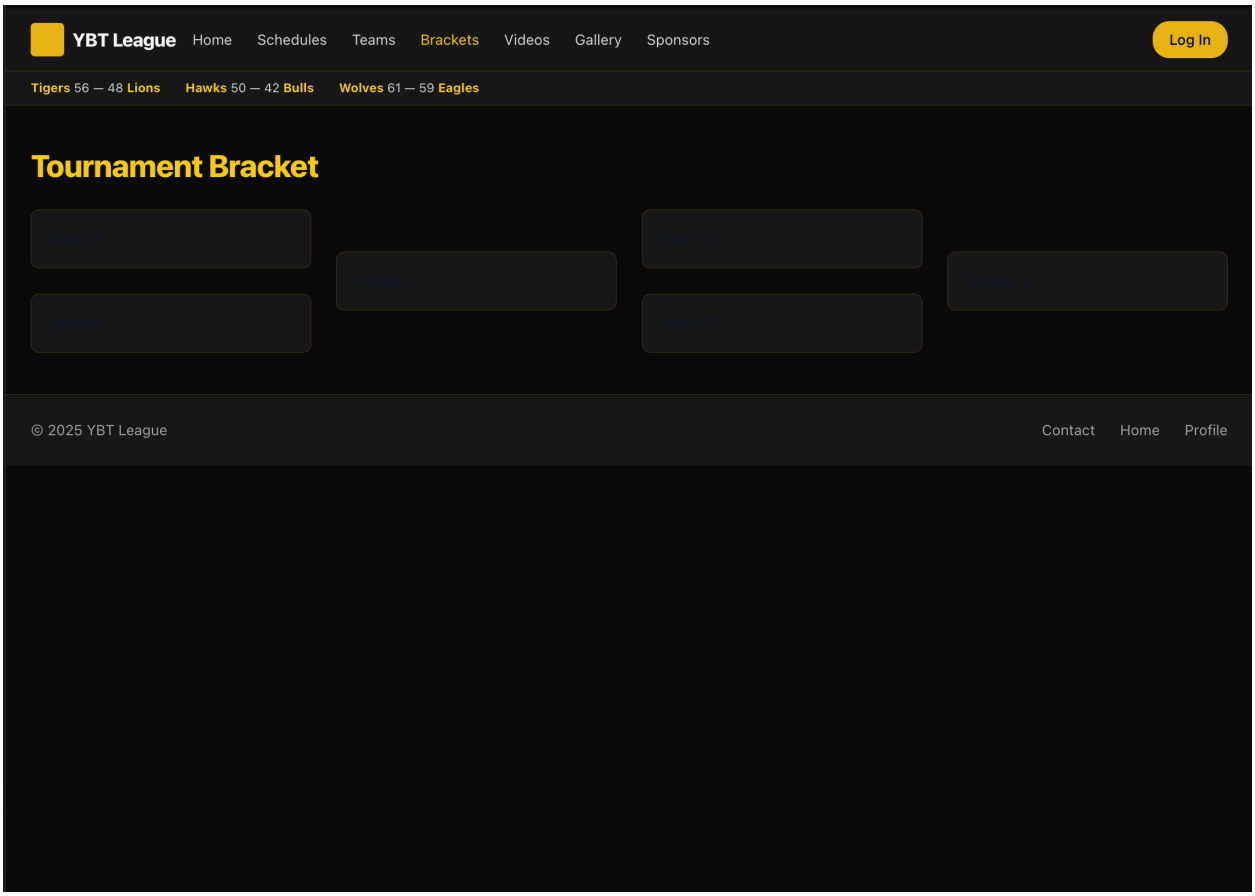
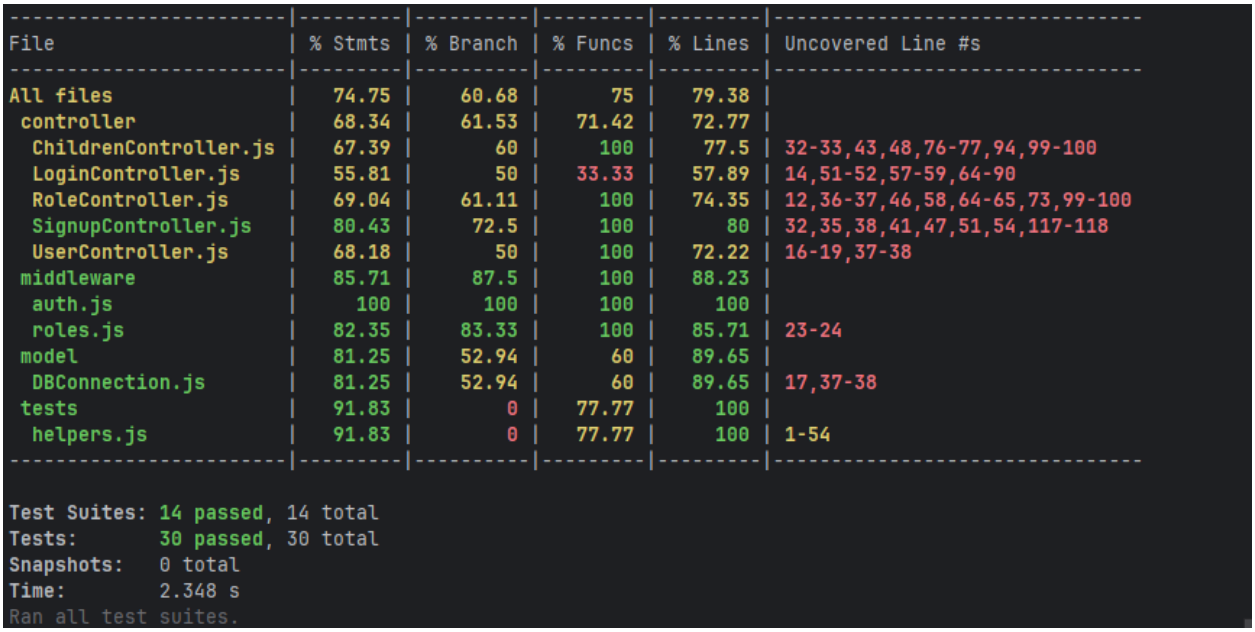


Figure 10: Bracket

Iteration	Dates	Stories	Points
1	10/09 - 10/23	S1 Signing Up Children, S21 Registration, S9 Team Photos and Logos, S14 Contact Page, S11 Live Chat, S13 Buying Tickets	15
2	10/23 - 11/06	S22 Admin CRUD, S0 Scheduling, S11 Live Chat, S13 Buying Tickets, S19 Sponsor Info	25
3	11/06 - 11/20	S3 Match Bracket, S4 History of Matches, S5 Match Updates, S11 Live Chat, S12 Archives	17
		Total:	57

Table 3: Iteration Planning for Incremental Deliveries



```
-----------------------|---------|----------|---------|---------|-------------------------------
File                   | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------------------|---------|----------|---------|---------|-------------------------------
All files              |   74.75 |    60.68 |      75 |   79.38 |
 controller            |   68.34 |    61.53 |   71.42 |   72.77 |
  ChildrenController.js |   67.39 |       60 |     100 |    77.5 | 32-33,43,48,76-77,94,99-100
  LoginController.js    |   55.81 |       50 |   33.33 |   57.89 | 14,51-52,57-59,64-90
  RoleController.js     |   69.04 |    61.11 |     100 |   74.35 | 12,36-37,46,58,64-65,73,99-100
  SignupController.js   |   80.43 |     72.5 |     100 |      80 | 32,35,38,41,47,51,54,117-118
  UserController.js     |   68.18 |       50 |     100 |   72.22 | 16-19,37-38
 middleware             |   85.71 |     87.5 |     100 |   88.23 |
  auth.js               |     100 |      100 |     100 |     100 |
  roles.js              |   82.35 |    83.33 |     100 |   85.71 | 23-24
 model                  |   81.25 |    52.94 |      60 |   89.65 |
  DBConnection.js       |   81.25 |    52.94 |      60 |   89.65 | 17,37-38
 tests                  |   91.83 |        0 |   77.77 |     100 |
  helpers.js            |   91.83 |        0 |   77.77 |     100 | 1-54
-----------------------|---------|----------|---------|---------|-------------------------------

Test Suites: 14 passed, 14 total
Tests:       30 passed, 30 total
Snapshots:   0 total
Time:        2.348 s
Ran all test suites.
```

Figure 11: Iteration 1

5.2.2 Work Done

Finished Work: Joel: S1 Signing Up Children (2), S21 Registration (2) Fully finished with page creation and connection from frontend to backend. Nishant: S9 Team Photos and Logos (2), S14 Contact Page (2) Fully finished with page creation and connection from frontend to backend.

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.] It is good enough because it is 75 percent of the code coverage overall.

5.2.4 Retrospective & Reflection

The challenges and issues had with this iteration is people not realizing the depth of the stories they chose, and how much work would be going into it, aswell as teamwork not really being at the forefront, and now we will try to work towards that. S11 AND S13 were not finished or any files even committed, with no real reason explained as to why.

5.3 Iteration/Sprint 2

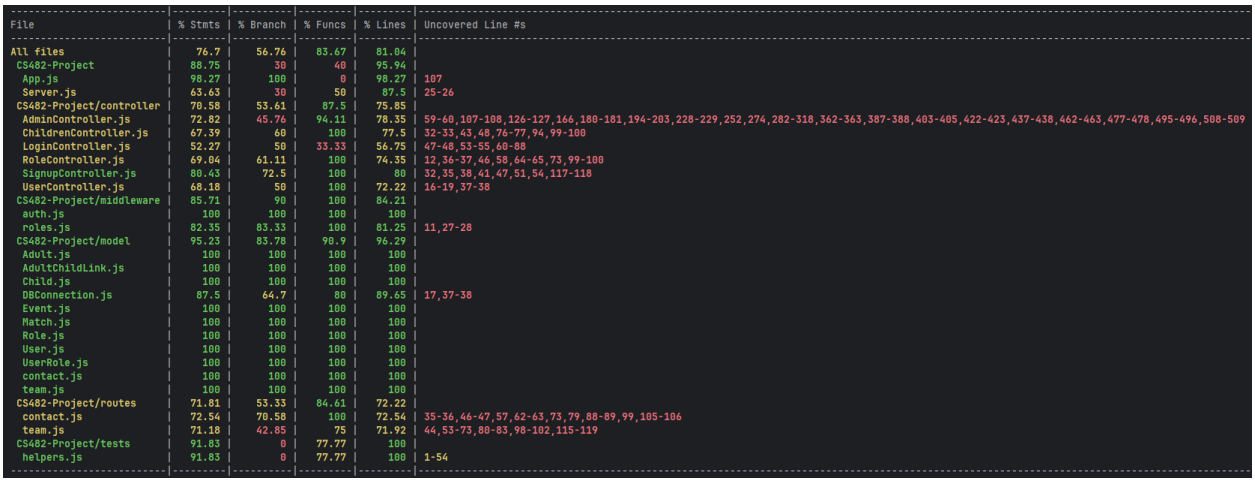
5.3.1 Planning

Joel will be doing S22 Admin CRUD worth 8 points. Nishant will be doing S0 Scheduling worth 8 pts Ekto will continue working on S11 Live Chat worth 5 pts. Yohann will be continue working S13 Buying Tickets and additionally work on S19 Sponsor Info worth 2 pts each.

5.3.2 Work Done

S22 Admin CRUD finished (but more could be edited later), S13 finished with examples included, although there is no true payment functionality because Professor Rocha advised against it due to the complexity in the long run. S0 had no work done towards it and S11 still doesnt seem to have any progress.

5.3.3 Testing Coverage



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	76.7	56.76	83.67	81.04	
CS482-Project	88.75	30	40	95.94	
App.js	98.27	100	0	98.27	107
Server.js	63.63	30	50	87.5	25-26
CS482-Project/controller	70.58	53.61	87.5	75.85	
AdminController.js	72.82	45.76	94.11	78.35	59-60,107-108,126-127,166,180-181,194-203,228-229,252,274,282-318,362-363,387-388,403-405,422-423,437-438,462-463,477-478,495-496,508-509
ChildrenController.js	67.39	60	100	77.5	32-33,43,48,76-77,94,99-100
LoginController.js	52.27	50	33.33	56.75	47-48,53-55,60-88
RoleController.js	69.04	61.11	100	74.35	12,36-37,46,58,64-65,73,99-100
SignupController.js	80.43	72.5	100	80	32,35,38,41,47,51,54,117-118
UserController.js	68.18	50	100	72.22	16-19,37-38
CS482-Project/middleware	85.71	90	100	84.21	
auth.js	100	100	100	100	
roles.js	82.35	83.33	100	81.25	11,27-28
CS482-Project/model	95.23	83.78	90.9	96.29	
Adult.js	100	100	100	100	
AdultChildLink.js	100	100	100	100	
Child.js	100	100	100	100	
DBConnection.js	87.5	64.7	80	89.65	17,37-38
Event.js	100	100	100	100	
Match.js	100	100	100	100	
Role.js	100	100	100	100	
User.js	100	100	100	100	
UserRole.js	100	100	100	100	
contact.js	100	100	100	100	
team.js	100	100	100	100	
CS482-Project/routes	71.81	53.33	84.61	72.22	
contact.js	72.54	70.58	100	72.54	35-36,46-47,57,62-63,73,79,88-89,99,105-106
team.js	71.18	42.85	75	71.92	44,53-73,80-83,98-102,115-119
CS482-Project/tests	91.83	0	77.77	100	
helpers.js	91.83	0	77.77	100	1-54

Figure 12: Iteration 2

I believe this is good enough coverage because it tests every important file above 60 percent and more, there may be an additional increase in coverage as these get more fleshed out. The testing didnt change, but was added to with these new controllers and models.

5.3.4 Retroespective & Reflection

The pitfalls this iteration had to do with mostly stagnancy on the project from some members, and we will be continuing forward trying to make sure that everyone contributes the same amount of work towards their user stories chosen. Not everything went to plan, with there being little development done with scheduling and the live chat. There was learning in additional website and database connections for team and child links, and fields/attributes being added as we get more complex with these fields.

5.4 Iteration/Sprint 3

5.4.1 Planning

Joel Robinson: S3 Match Bracket (2) S4 History of Matches (3) Nishant Gurung: S5 Match Updates (5) Yohann: S12 Archive Folder (2) Ekto: S11 Live Chat (5) We aimed to implement a majority of the important match features this iteration, and the deliverable should be able to show them all.

5.4.2 Work Done

S3 Match Bracket and S4 History of Matches were completed by Joel. S5 Match Updates were completed by Nishant, S11 is partially (and almost fully) finished by Ekto. Yohann completed Archive Folders.

5.4.3 Testing Coverage

This coverage is okay, there wasnt much time to put out very good test suites for our new controllers and javascript files, but something needed to be done.

5.4.4 Retroespective & Reflection

The pitfalls of this iteration were a lack of communication across the team, and something needs to be done going forward for everyone to know what they are doing before the iteration planning needs to be submitted. Everything completed within the iteration went well and alot was learned about updating items and showing them within the page. No one was able to finish the diagram in time either, so it must be added at a later date.

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.] Overall, We completed roughly 70 percent of the work we set out to finish. The time it would take to finish this project would most likely span over another 2 iterations, as there were strides to be made within the coach viewing teams and assigning positions, aswell as bracket updating and sponsor pages.

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)] Joel: The lessons I learned from this project are very linked with the planning that would come with each iteration and the beginning of the project. We oversimplified how much backend creation would need to happen for iteration 1 to come to fruition, which made it a mess when we were approaching the deadline, I would have personally taken more time and piecemealed it, aswell as maybe integrating similar tasks together for more coherency. Overall I think we attempted this project very well and have a semi-favorable end product for the time we had worked on this and the amount of programming it consists of total.

Appendix

[Appendix section if needed]

File	% Stmts	% Branch	% Funcs	% Lines
All files	72.59	56.83	74.69	75.62
CS482-Project	69.01	15.62	16.66	72.93
App.js	92.77	50	0	92.68
Server.js	35.59	10.71	20	41.17
CS482-Project/controller	72.35	57.66	87.93	76.95
AdminController.js	72.82	45.76	94.11	78.35
BracketController.js	89.47	75	100	88.88
ChildrenController.js	67.39	60	100	77.5
LoginController.js	53.06	44.44	28.57	57.14
MatchController.js	82.45	87.87	100	83.01
RoleController.js	69.04	61.11	100	74.35
SignupController.js	80.43	72.5	100	80
UserController.js	68.18	50	100	72.22
CS482-Project/middleware	85.71	90	100	84.21
auth.js	100	100	100	100
roles.js	82.35	83.33	100	81.25
CS482-Project/model	88.16	83.72	69.23	88.55
Adult.js	100	100	100	100
AdultChildLink.js	100	100	100	100
Child.js	100	100	100	100
DBConnection.js	87.5	64.7	80	89.65
Event.js	100	100	100	100
Match.js	100	100	100	100
Role.js	100	100	100	100
User.js	100	100	100	100
UserRole.js	100	100	100	100
contact.js	100	100	100	100
schedule.js	96.87	100	90	96.87
supportMessage.js	90.9	75	80	90.9
team.js	52	100	0	52
CS482-Project/routes	64.22	55.24	75	66.02
admin.js	100	100	100	100
contact.js	72.54	70.58	100	72.54
publicEvents.js	25	0	0	28
publicMatches.js	21.05	0	0	22.58
schedule.js	73.52	85	100	73.52
signup.js	90.47	100	100	90
supportMessage.js	65.95	77.77	77.77	65.55
team.js	71.18	42.85	75	71.92
CS482-Project/tests	91.83	0	77.77	100
helpers.js	91.83	0	77.77	100

Figure 13: Iteration 3