CS482/495/496 Software Project Proposal: add your tentative project title here

your name(s) here

2025-11-30

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: Eric Cui
- Client title: Teen league manager
- Client email address: lcui@loyola.edu
- Client employer: Loyola University Maryland
- How you know the client: Through our Professor

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?]

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.]

2.3 Why this Project is Interesting

[Why did you decide this project was interesting enough to you to be a capstone project? What about this project is enticing? Why should anyone care?]

2.4 Areas of CS required

[What subfields of computer science seem most likely to be relevant to your project? A capstone must involve multiple.]

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?]

2.6 Summary of Efforts to Find a Project

(Not necessary for 482) [Briefly list out when/how you've discussed with this client, and if you've discussed with other clients who either didn't work out or didn't respond. If you considered a different project and it didn't work out, why didn't it work out?]

[Most CS495 projects end here. The sections below are for CS482 and CS496 software projects].

2.7 Comparison to Draft

[For CS496 only, focus on highlighting the major differences between the draft proposal in CS495 and this one here. If there are no major differences, you can remove this subsection.]

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	Color Scheme	Visual/Design	The color scheme for the app should be a blue
NFR2	Permission	Access	Invite/login in and out system
NFR3	Security	Protection	Verification/Accounts
NFR4	Managing	Updating	changing and adding things
NFR5	Viewing	Security	Being able to use the website

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
A4	Managing Teens account	3	As an Adult, I want to be able to watch and manage my kids/teens accounts, so that I can look and accept any invites while also seeing what they're doing on their accounts
A11	Live Scores	5	As an Admin, I want to be able to update live scores, so that I can keep viewers up to date on live matches
A16	Full Match Control	3	As an Admin, I want to be able to check each match in the brackets, so that I can make sure everything is good to start the match, and to allow teams to enter
AD1	Calendar control	8	As an Admin, I want to be able to add matches and change/edit the bracket date so that I can update recent changes on the calendar/matches
TM5	Invite System	3	As a Team Manager, I want to invite members to a team, so that I can have a soccer team of 11 or more
TM6	Posting Team Info	5	As a Team Manager, I want to manage the team I have, so that I can update/change what's going on with the team
U3	Login in/sign up screen	3	For all users (except Guest), I want to be able to sign up or log in to the site with an account, so that I can save my data and be able to join/work on a team
U7	Home screen	2	As a user, for all users, I want to see posts and live matches, so that I can look at the current score/results while also interacting with the post
U8	Using Posts	2	For all users (with exceptions), I want to be able to interact with posts, so that I can like/dislike and comment on posts while also knowing that teens can't see flagged posts
U12	Watching History	2	For all users (except Guest), I want to be able to see the history of games I have played in or watched, so that I can have memories of what I have done or seen
U13	Profile management	2	For all users (except Guest), I want to be able to go into my own profile, so that I can change my info and description about me
U15	Full Match display	2	For all users, I want to be able to see time, date, place, teams competing, type (normal, playoff, final), so that I can keep with the recent results of a match
U2	Inbox screen	2	For all users, As a/an For all users (except Guest), I want to be able to see notifications from the website, so that I can see if I get any news and invites from any games, teams, or posts that have come in
U17	Settings screen	3	As a/an For all users (except Guest), I want to be able to change anything in my profile, so that I can either delete, switch, or change anything for my account
U18	Forget password	1	As a/an For all users, I want to be able to reset my password, so that I can update my password in case I forget it
T19	Parent permission	1	As a/an Teen, I want to be able to have a parent or guardian know that you're signing up, so that the website knows if you're allowed to make an account
U9	Profile viewing	3	As a/an For all users, I want to be able to look at others' profiles, so that I can see who is who and the stats they have

Table 2: Functional requirements as User Stories.

4 System Design

4.1 Architecture

We will be using a Model-View-Controller (MVC) architecture since the Rails API is designed to support MVC. Frontend will utilize react, backend is using Redis and MongoDB, while the Rails API handles all the endpoints and logic.
Potential Main Modules:

- User Management Module
- Team Management Module
- League Management Module
- Analytics Module
- Admin Panel Module
- Notification Module

4.2 Diagrams

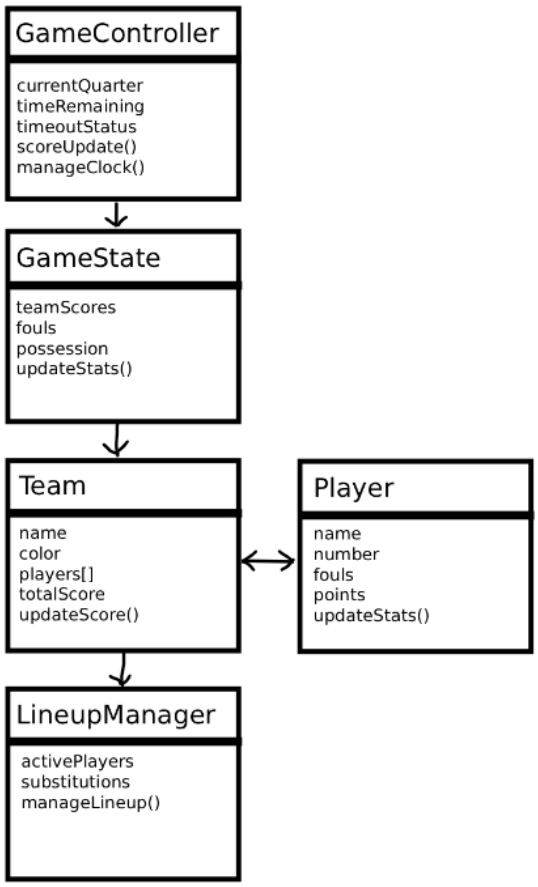

Figure 1: Live Score example created by Justin

4.3 Technology

Tech	Type	Usage
Ruby	Language	Backend
Rails	Framework	Web application framework
Docker	Containerizer	Containerize the app for consistency
Redis	RAM-Based Database	Quick data retrieval and updates; ideal for real-time data
React	UI Framework	Used for UI design and implementation
MongoDB	Schema-Based Database	Long-term data storage; performance not critical
RSpec	Testing tool	Used to test ruby code

Table 3: Technologies used in the project

4.4 Coding Standards

4.4.1 Naming

- **Global Variables:** All caps and underscores for spaces. Examples: `VARIABLE`, `TEST`, `MIN_NUM`
- **Classes:** Capitalized and CamelCase. Examples: `Class`, `ClassGuide`, `ExTwo`
- **Functions:** Lowercase with underscores for spaces. Avoid using numbers. Examples: `function`, `function_two`, `print_all_one`
- **Variables:** Same as functions, but numbers do not require underscores. Examples: `n`, `x`, `x2`, `print_status`
- **Files:** File names should be descriptive and follow variable naming rules unless another convention is necessary. Examples: `test_file.txt`, `grid_layout.py`, `App.jsx`
- **Folders:** Same as files. Keep names general and simple. Examples: `folder`, `folder_two`, `src`

4.4.2 Code Formatting

- Include a blank line between each function.
- Use consistent and readable loop structures.
- Place comments above functions and significant code blocks.
- Inline comments should be aligned to the right of the code line they describe.
- Maintain consistency throughout the codebase.

4.4.3 Commenting Rules

- Comments must be placed above each function explaining its purpose.
- Comments must be placed above each class.
- Comments should be detailed and easy to understand.
- Inline comments (for specific lines) must appear to the right of that line.
- Comments for code blocks should be placed above the block and explain its function.
- Create and follow standard templates for class and function comments.

4.4.4 Coding Principles

- Follow the **DRY (Don't Repeat Yourself)** principle when designing code.

4.4.5 Testing

- Only allow code with unit tests and with at least 65% coverage to be committed.

~~4.5 Data~~

4.6 Data

Our data will be in MongoDB. Below is the Entity-Relationship list that we will implement to the database:

- **Entity: Admin**
  - Attributes:
    - Name
    - Username
    - Password
  - Relationships:
    - MODIFY Teams, Profiles, Matches, Posts, and Calendar
    - Can RATE Matches and Posts
    - Can COMMENT on Posts
- **Entity: Team Manager**
  - Attributes:
    - Name
    - Username
    - Password
    - Team
  - Relationships:
    - INVITE Profiles
    - ADD a Team
    - VIEW Teams, Profiles, and Calendar
    - Can RATE Matches and Posts
    - Can COMMENT on Posts
- **Entity: Adult**
  - Attributes:
    - Name
    - Username
    - Password
    - Team
    - Children
  - Relationships:
    - ACCEPT INVITES from Team Manager
    - ACCEPT INVITES to their Teen from Team Manager
    - VIEW Teams, Profiles, and Calendar
    - Can RATE Matches and Posts

        * Can COMMENT on Posts

- **Entity: Teen**
  - Attributes:
    * Name
    * Username
    * Password
    * Team
    * Adult Supervisor
  - Relationships:
    * CHILD OF a Adult
    * VIEW Teams, Profiles, and Calendar
    * Can RATE Matches and Posts
    * Can COMMENT on Posts
- **Entity: Guest**
  - Relationships:
    * VIEW Matches, Teams, Profiles, Posts and Calendar
- **Entity: Team**
  - Attributes:
    * Players
    * Team Name
    * Team Logo
    * Team Manager
- **Entity: Match**
  - Attributes:
    * Score
    * Teams
    * Rating
    * Comments
    * History
  - Relationships:
    * Can VIEW a Team
- **Entity: Calendar**
  - Attributes:
    * Matches
  - Relationships:
    * VIEWS Matches
- **Entity: Profile**
  - Attributes:
    * Name
    * Team Name
- **Entity: Post**
  - Attributes:
    * Comment
    * Reactions

4.7 UI Mocks

Figure 2: login screen with the ability to sign up or reset your password (plus being able to be a guest)

Figure 3: Once you log in or join as a guest, you'll start from the home screen with the option bar on the left

5 Iterations

5.1 Iteration Planning

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

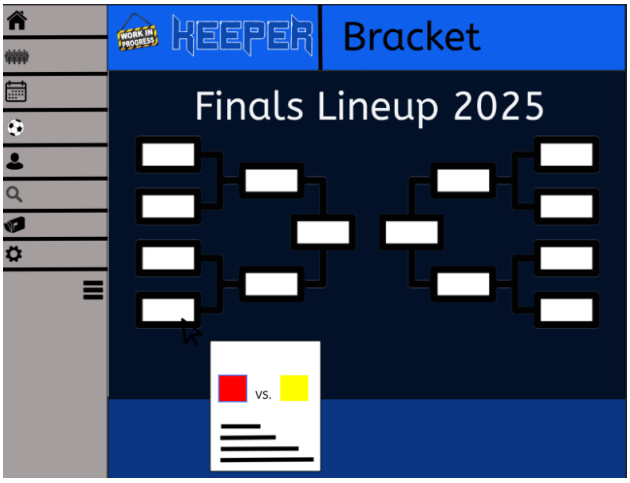

Figure 4: This is where you can see the current tournament matches that are happening while being able to click to see the info of the two teams.

Iteration	Dates	Stories	Points
1	10/14 - 10/23	U7 Homepage screen, U3 Login/sign-up screen, U13 My Account screen, U2 Inbox screen	10
2	10/28 - 11/06	U17 Settings screen, U18 Forgot Password, T19 Parent permission	5
3	03/01 - 04/01	S7 Story Title, S8 Story Title, S9 Story Title, S10 Story Title, S11 Story Title	21
~~4~~	~~04/01 - 05/01~~	~~S12 Story Title, S13 Story Title, S14 Story Title, S15 Story Title~~	~~19~~
~~5~~	~~05/01 - 06/01~~	~~S16 Story Title, S17 Story Title~~	~~06~~
		Total:	70

Table 4: Iteration Planning for Incremental Deliveries

5.2 Iteration/Sprint 1

5.2.1 Planning

- ⋆ Justin/Jordan: U7 Home page screen 2pts
- ⋆ Alberto/Ose: U3 Login screen 1pt, sign up screen 3pts
- ⋆ All: U13 My Account screen 3pts
- ⋆ Alberto: U2 Inbox screen 2pts

5.2.2 Work Done

We did not complete any stories because we did not fully understand how Ruby works and had conflicts with assignments from other classes that week. So far, we have partially completed U3/U7 (Justin/Jordan does U7, and U3 Alberto/Ose does U7).

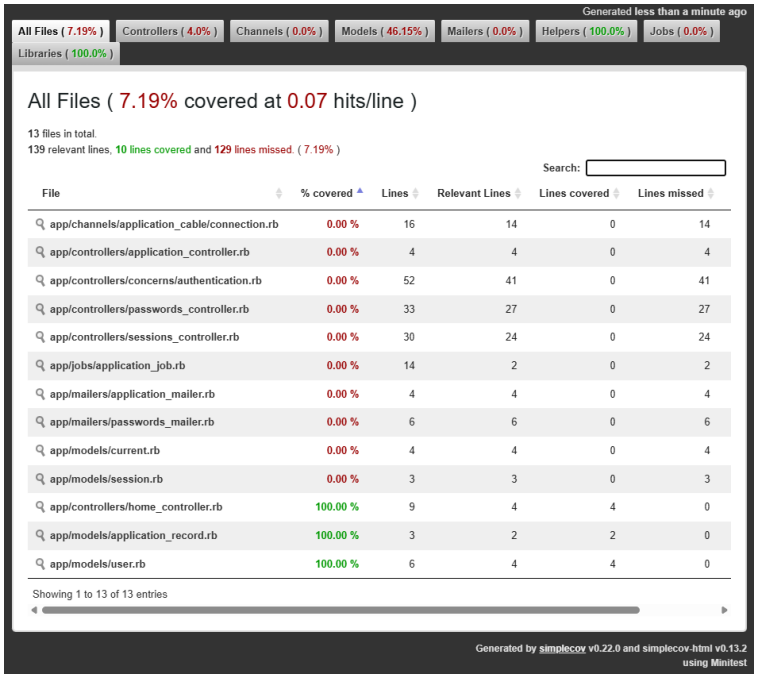

Figure 5: Test coverage from Alberto's end

5.2.3 Testing Coverage

5.2.4 Retroespective & Reflection

What we have learned from the first iteration as a group is that we need to rely on each other more, rather than having one person be the lead on programming. A lot of people were busy with different things to the point where they couldn't work on this project, in which we should have picked up where they left off. Another reason is that this is a new language that most of the group has not done at all, which makes it harder/longer for others pick up how to do it. Hopefully, in this next iteration, we can have 1 fully finished and work on the smaller stories in this one.

5.3 Iteration/Sprint 2

5.3.1 Planning

Goal:

- ⋆ Justin/Alberto: U17 Settings screen 3pts
- ⋆ Ose: U18 Forgot Password 1pt
- ⋆ Jordan: T19 Parent permission 1pt
- ⋆ Total points: 5pts

Stretch Goal:

- ⋆ T10 Teen verifier 2pts
- ⋆ U20 Creating posts 5pts
- ⋆ Total points: 7pts

Previous Goal:

- ⋆ U7 Home page screen 2pts
- ⋆ U3 Login screen / sign up screen 3pts
- ⋆ U13 My Account screen 3pts
- ⋆ U2 Inbox screen 2pts

5.3.2 Work Done

Alberto got U2 Inbox Screen completed and fully integrated. This feature has real time notifications and a mark as read feature. Logged in users will receive notifications if that notification is tied to them. He also worked on the backend to implement ActionCable and channel subscriptions for any future features. He also refactored the authentication system in the frontend to make authentication easier in the future. Protected routes were also created to add a layer of security. Ose has partially worked on U18 Forgot Password feature but it is not fully functional. The user model was modified to implement role, date of birth, and names. Other goals have not been met as of writing this.

5.3.3 Testing Coverage

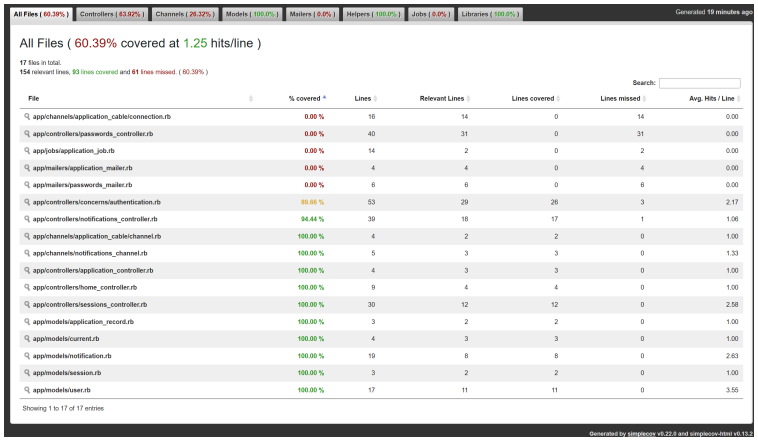

Figure 6: Test coverage from Alberto's end

5.3.4 Retroespective & Reflection

What we have learned from this is that we need more teamwork and communication. We also need to have everyone complete and work on their tasks in a timely manner. We cant have people work on their tasks last minute in case there is a major issue with the project as a whole. Implementing GitHub actions is something that will be necessary from now on. People need to commit enough, at least once daily and with good documentation. We need to utilize GitHub issues to track and document any bugs or features being implemented correctly. Using PRs is also included in this. Having other team mebers review each other's code is also something we should do.

5.4 Iteration/Sprint 3

5.4.1 Planning

Goal:

- ⋆ Justin/Alberto: U15 Full Match Display 2pts
- ⋆ Ose/Jordan: U9 Profile Viewing 3pts

Stretch Goal:

- ⋆ U8 Using Posts 2pts
- ⋆ U20 Creating posts 5pts

Previous Goal:

- ⋆ U7 Home page screen
- ⋆ U3 Login in/Sign up screen
- ⋆ T19 Parent Permission
- ⋆ U13 Profile Management
- ⋆ U17 Settings Screen

5.4.2 Work Done

For this Iteration, we got a lot of previous iterations done while having done the current iterations in mine. First, Alberto worked on the functions of the live match while Justin worked on the template (design) on said page. Second, Ose and Jordan was working on displaying the profiles of users once you click on it. For extra things, Justin worked on the homepage, login screen, and settings. Jordan worked on the homepage and sign-up page, Ose worked on the forgot password page, and Alberto worked on more of the backend of things.

5.4.3 Testing Coverage

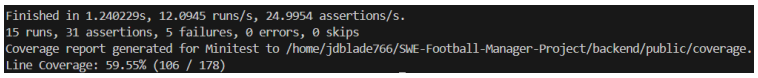

Figure 7: Test coverage from Justin's end

5.4.4 Retroespective & Reflection

We achieved a lot of things from our backlog this iteration, while also getting most of our current iteration completed. The challenges we had this Iteration were communication and having problems with merging certain branches. For example, Justin had made a lot of changes in his own branch to which was a huge improvement from previous iterations, but the branch he was in was not on GitHub, so it made it harder to bring it over to Alberto's branch to have it reviewed. Another big thing was a bunch of redesigns from different pages, while being able to click to go to them.

~~5.5 Iteration/Sprint 4~~

~~[CS496 has 5 sprints. CS482 only has only 3 sprints (remove Iterations 4 and 5 from this doc if you are writing a doc for 482]~~

~~5.5.1 Planning~~

~~[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]~~

5.5.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

5.5.3 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.]

5.5.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

5.6 Iteration/Sprint 5

5.6.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

5.6.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

5.6.3 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.]

5.6.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

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]