Sports Wagering Technical Reference

An informal and frequently changing container for policy, requirements, design notes, sketches, etc. in order to help out with building this thing.

Team members

- Madeline Rys
- Christian Wettre
- Mahima Chandan
- Dale Pippert
- Amrutha Ravi

GitHub URLs

Sports wagering project site

https://github.com/madelinerys/CS546-Final-Project

Sports wagering project proposal

https://github.com/madelinerys/CS546-Final-Project/blob/main/doc/ProjectProposalSportsWagering.pdf

Sports wagering database proposal

Original is available here: https://github.com/madelinerys/CS546-Final-Project/blob/main/doc/DatabaseProposalSportsWagering.pdf

However this is no longer being maintained. Instead the database schema has been incorporated into this document.

Sports wagering technical reference (this document)

https://github.com/madelinerys/CS546-Final-Project/blob/main/doc/TechnicalReferenceSportsWagering.md
https://github.com/madelinerys/CS546-Final-Project/blob/main/doc/TechnicalReferenceSportsWagering.pdf

Terminology

Admin. A user with additional privileges over those of a normal user. The data model as submitted does not have a flag for this, but needs one.

Bet. A monetary value in dollars you are investing, in the hopes of winning some percentage return on your investment.

Bettor. Used interchangeably with user, player.

BYE. A week where a given team does not play.

House. This is the sponsor or owner of the game. A player bets against the house. If a player wins a bet they get the house's money for that bet. If a player loses a bet they are losing it to the house. For example, if our system was adopted by the State of New York, U.S., then New York state government would be the house.

Juice. See vigorish.

Line. A catch-all term to describe types of bets. For an example of lines, see https://www.espn.com/nfl/lines.

NFL. National Football League, http://nfl.com.

OFF. Some games do not have posted betting lines for various reasons. These are considered "off" games and cannot be bet.

Player. See bettor.

PPP. Project Pitch Proposal.

Push. A tie with the house. The player is refunded their original bet amount in full.

Resolve. A bet is considered resolved when either the player has won and been credited with the winning amount, *or* has lost the bet, *or* has pushed in which case there money is refunded.

Surcharge. See vigorish.

User. See bettor.

Vig. See vigorish.

Vigorish. This is a surcharge tacked onto bets. In a perfect world this is how the house makes money. Vigorish is only returned to the player on bets won, not bets lost. On bets lost, the house keeps the vigorish which is their primary means for realizing revenue from the activity.

Wager. See bet.

Technologies

I believe the professor introduces us to Bootstrap in the weeks ahead. Never used personally. However it may offer needed assistance for the "responsive" requirement. So I was going to propose we stick to:

- Bootstrap (which brings with it jQuery and Popper whether you want them or not) (Note: I'm waivering on this, if we don't need responsive I would suggest leaving Bootstrap out.)
- Node.js and Express (required)
- Handlebars
- Axios (needed for API to sports information)
- Bcrypt (encrypt password)
- Mongo (required)
- HTML (required)
- Javascript (required)
- CSS (required)

NOTE: Do we need "responsive"? I can't seem to find that requirement anymore. If we don't need it then we don't need Bootstrap.

I'm proposing we stay away from front-end frameworks such as React/Angular. Not enough time for me/us to learn these technologies. If we go down in flames we should do it with technologies that are covered in the class, if for no other reason hopefully the professor shows some empathy.

System rules

A list of rules that don't necessarily pertain to any one page or database collection but rather are more global in nature.

1. Whole dollars only. Do not display or deal with cents anywhere. Any rounding/truncation that needs to occur should be done in a direction that favors the house.
2. U.S. dollars only, we will not deal with international currencies or cryptocurrencies.
3. Over/under and straight bets are made with a 10% surcharge (vigorish). This charge is not returned on bets lost, but is returned on bets won.

The next several sections cover database collections. Refer to this document for information on database from now on, the database proposal document has been subsumed here. This keeps everything in one place without having to spend time on cross-referencing or trying to guess what is covered in which document.

Bets collection

Stores bets. Each document is a bet from a bettor aka user. Bettors use the system's user interface to enter bets. As the bets are entered, the system stores them to this collection.

Bets are made against lines. This collection as originally submitted in the database proposal was designed to store a *reference* to a line for each bet made. This has been changed to now store the actual value of the line.

Bets schema

Field	Type	Description
`_id`	ObjectId	Mongo-generated key for the document
`bettorid`	ObjectId	`_id` of the bettor from Bettors collection
`gameid`	String	`_id` of the game as returned from Lines API
`bettype`	String	type of bet, one of: AML, ASP, HML, HSP, OV, UN [1]
`num`	Number	the player's *number*, the meaning of which depends on the *bettype*.
`amount`	Number	dollar amount of the bet
`pays`	Number	dollars this bet pays on a bettor win
`collects`	Number	total dollars this bet collects on a bettor win; this is equal to amount + pays
`paid`	Number	dollar amount this bet collected, or null if bet is still live; may be zero indicating this bet has resolved and was a loss for the bettor [2]
`entered`	Date	time and date of the bet
`resolved`	Date	time and date the bet resolved, or null if bet is still live [2]

Notes

1.
   - AML Away Money Line.
   - ASP Away Spread.
   - HML Home Money Line.
   - HSP Home Spread
   - OV Over.
   - UN Under.
2. These fields are necessary in order for the system to know whether it has resolved the bet or not.

Bets example document

```
{
  _id: "5eeb5b6186fbfca1f18ed313",
  bettorid: "5ee77f8c75ee6029745ca8ac",
  gameid: "ari-sea-2020-11-19",
```

```
    bettype: 'ASP',
    num: 3,
    amount: 77,
    pays: 70,
    collects: 147,
    paid: null,
    entered: "2020-11-17T15:30:22",
    resolved: null
  }
```

Scores collection

Fka Games collection.

Each document captures a single NFL game final score. An NFL season is 17 consecutive weeks, with 14 games per week, so at the conclusion of a full regular season (not counting postseason), this collection would have 17 * 14 = 238 documents in it. The system inserts final scores into the collection in an automated fashion, as they become avaialable.

Score documents may be inserted with a null `awayScore` and null `homeScore`. After the game is played, a system background job is responsible for updating `awayScore` and `homeScore` with the game's final score.

Scores schema

Field	Type	Description
`_id`	String	format of awayTeam-homeTeam-date-of-game (see example document)
`gameDate`	String	YYYY-MM-DD
`awayTeam`	String	designator for away (visiting) team
`homeTeam`	String	designator for home team
`week`	Integer	Week of the season, 1-17. NFL weeks start on Tuesday and end on Monday. As a frame of reference, 10/21/2020 is in Week 7. If you want to know, for example, what is the *current week right now* as a frame of reference, you can go here and it should default to bring up the current week pre-selected.
`awayScore`	Integer	Away team final score or null if game not complete or not yet updated.
`homeScore`	Integer	Home team final score or null if game not complete or not yet updated.

Scores example document

```
{
  _id: "htx-kan-2020-09-10",
  gameDate: "2020-09-10",
  week: 1,
  awayTeam: "hou",
  awayScore: 20,
  homeTeam: "kc",
  homeScore: 34
}
```

Scores notes

1. NFL is the National Football League.
2. GMT is four hours ahead of EDT, and five hours ahead of EST. For example, 1:00 PM EST is 6:00 PM GMT.
3. Designators for teams defined in Teams collection as `Teams.abbrv`.

Bettors collection

These are users aka bettors that have signed up. Possibly (time permitting) seeded with 1000 bettors for demo purposes.

Bettors schema

Field	Type	Description
`_id`	ObjectId	Mongo-generated key for the document
`username`	String	
`pwd`	String	bcrypt hash of password
`balance`	Number	dollar balance in account

Bettors example document

```
{
  _id: "3e85908c9dad05d2589ae104",
  username: "foghorn5",
  pwd: "$2a$16$7JKSiEmoP3GNDSalogqgPu0sUbwder7CAN/5wnvCWe6xCKAKwlTDq",
  balance: 250.00
}
```

1. Surprisingly, this collection will be seeded with exactly the same 1000 people from an earlier people.json lab. Apparently they all like to gamble!

Teams collection

A seeded reference collection to store static identities for all 32 NFL teams. This collection has exactly 32 documents in it, one per team.

Teams schema

Field	Type	Description
`_id`	ObjectId	Mongo-generated key for the document
`abbrv`	String	three-letter unique business key to enhance readability
`fran`	String	franchise name, typically locale/city/state of team
`nn`	String	team nickname

Teams example document

```
{
  _id: "5f85808c9dad05d358aae011",
  abbrv: "ten",
  fran: "Tennessee",
  nn: "Titans"
}
```

Lines API

Lines API is available from application as: http://localhost:3000/api/lines/nfl

This API is what the rendering logic for display of betting page must use, via handlebars.

Example return from Lines API

Only shows two games coming back, there will usually be up to 14 except on Mondays when there may only be one (for Monday night game).


```
[
{
  "gameTime": "8:20 PM",
  "gameDate": "Thursday, November 19",
  "gameDateJulian": 1605835200000,
  "lineDate": "20201117221505",
  "awayShort": "ari",
  "awayML": 150,
  "awayPts": 3,
  "awayOE": 57.5,
  "awayLong": "Arizona Cardinals",
  "homeShort": "sea",
  "homeML": -170,
  "homePts": -3,
  "homeOE": 57.5,
  "homeLong": "Seattle Seahawks"
},
{
  "gameTime": "1:00 PM",
  "gameDate": "Sunday, November 22",
  "gameDateJulian": 1606068000000,
  "lineDate": "20201117221505",
  "awayShort": "phi",
  "awayML": 160,
  "awayPts": 3.5,
  "awayOE": 45.5,
  "awayLong": "Philadelphia Eagles",
  "homeShort": "cle",
  "homeML": -180,
  "homePts": -3.5,
  "homeOE": 45.5,
  "homeLong": "Cleveland Browns"
},
]
```

Betting page

Normal users would be directed straight here after login. This is where they make their bets.

The main and most important page of the application. It is composed of an account balance text box at the top and a series of betting panels below that.

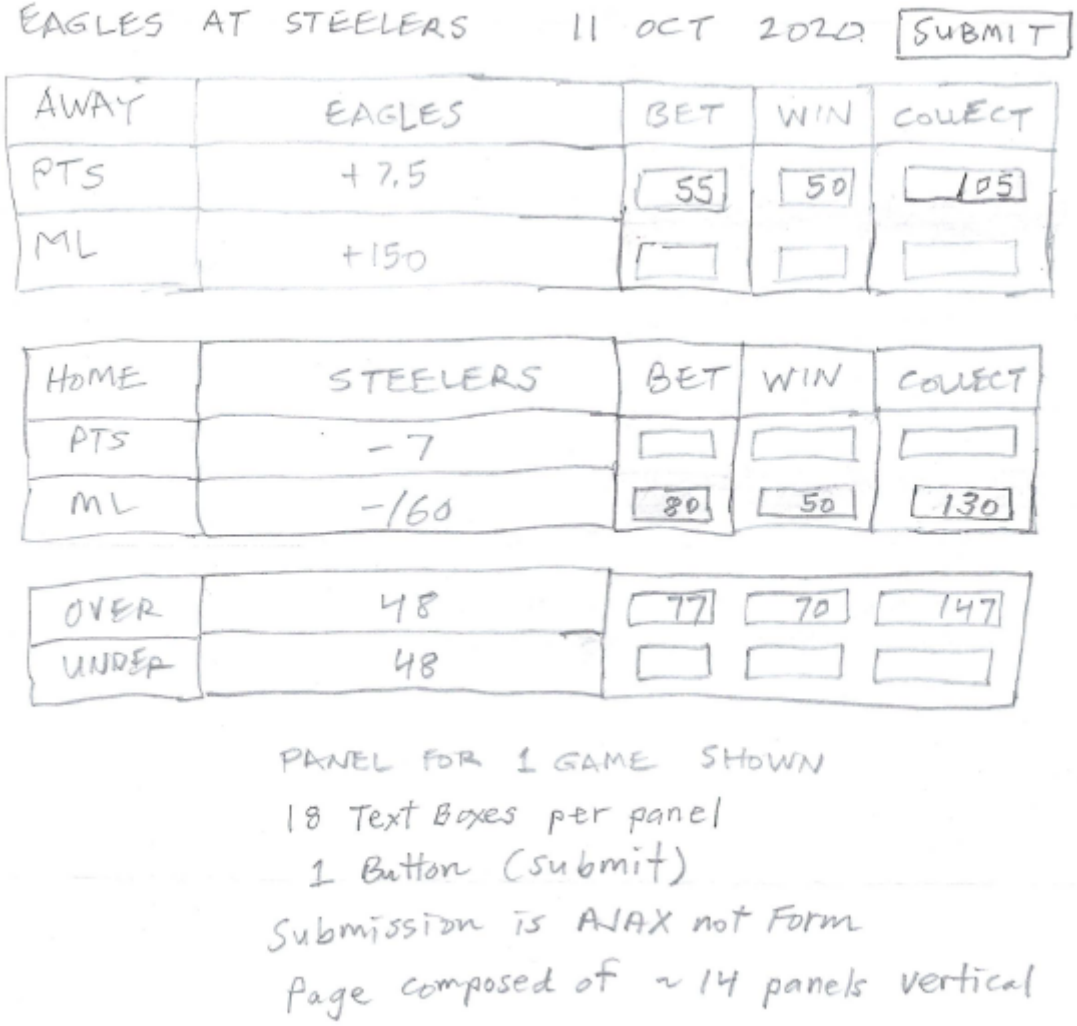
The rest of the page has 12-16 betting panels with clear separation between each one. They are arranged veritically one after another. All must be visible at the same time i.e. the page could be reduced greatly by having a drop down to select which single betting panel to view at a time, but that will not satisfy. The user must be able to see all panels and the state of each panel on the page by simply scrolling their device.

Account balance operation

1. Displays account balance for the user from `Bettors.balance`.
2. Input control is some type of read only large text box with large font.
3. Label clearly states this is the user's balance.
4. I see it having larger font that what is available on the betting panel.
5. User cannot type in this account balance text box, it is disabled from any changes. Only the application can make changes to it.
6. Balance is whole dollars only.

Betting panel sketch

Only one betting panel shown. The page itself has up to 16 of these, one per game that week, arranged vertically one after another with clear delineation between them (e.g., space, or color change or something visual to break them up).



Betting panel operation

1. Input for this panel is sourced from http://localhost:3000/api/lines/nfl which then must be rendered through a handlebars view and out to the user.

2. Submit button enables only when one or more of the 18 text boxes contains a non-zero amount, and is disabled otherwise.

3. BET and WIN text boxes work together. They start out empty. User chooses, by typing in *one* of these two boxes, how much they either want to BET, or WIN.

   a. If the user enters a BET amount, the WIN box will update with the BET amount, minus 10%. The BET box will remain enabled and the WIN box will be disabled in this case.

   b. If the user enters a WIN amount, the BET box will update with the WIN amount, plus 10%. The WIN box will remain enabled and the BET box will be disabled in this case.

4. The COLLECT box always shows BET amount plus WIN amount.

5. All text to left of Bet/Win columns is either static to the template page or static as rendered from the database. User cannot change this static text.

6. User will be prompted after pressing Submit, e.g., "You are about to wager $212 on this game. Are you sure?" Y/N buttons.

7. Bets against panels that have a `gateDateJulian` value less than the current Julian date must be disallowed. This check should be done on the front end and back end both.

8. User will receive confirmation once Submit has sucessfully processed on the backend by insert bet into Bets collection. User's balance will deduct by amount of wager. Confirmation can be as simple as "Your wager totalling $212 is in!" Perhaps it is some type of popup with buttons *Continue* and *Logout* on it.

   a. Pressing *Continue* dismisses confirmation box and returns user to the betting screen.

   b. Pressing *Logout* logs user out and returns application to the login screen ready for the next customer.

   c. Perhaps we change Submit button to *Start Again* so that user does not get confused and try to Submit bets a second time.

9. Submit must occur by AJAX callback so as to a) meet requirement of the project per professor and 2) so that screen does not have to flash as it reloads all other live panels (one panel per game). The AJAX call should only result in the current panel refreshing.

10. Panel is shown in its already smallest dimension with only one panel stationed horizontally per row. On larger screens the panels should to some extent "unstack" and start to appear horizontally, perhaps two or three of them will fit horizontally. This is to demonstate that our application is *responsive* which is another requirement. (NOTE: Check this, I can't find this requirement anywhere.)

11. This page generally has 1-16, one for each game for that week but depending on when the page is pulled up some of the games may have already played and are therefore not available for betting. Panels are arranged responsively but on a small screen they would stack vertically one after the other with comfortable space between each. A drop down where you would first select one of the up to 16 games then only that panel appears does not feel right to me which is why I want all live panels to display at once. Bettors need to be able to see in one fell swoop what the lines are, what looks appealing to bet on, and what they have already bet on.

12. Lines displayed for the betting panel are read from routes in the system that are in turn connected to a real-time API, NOT from the Lines collection. Repeating, do not use Lines collection to populate fields in betting panel, use API which is available from: http://localhost:3000/api/lines/nfl. Lines collection will eventually be populated but this collection is strictly for use by the API and not for use by the application proper.

13. The exact number of betting panels on the page can vary due to several factors. You may for example wager on a Friday, in which case if there was a Thursday night game that week, it will not be available for betting. Other reasons for the varying panels can include BYE week considerations, or OFF games due to player injuries or what have you. Generally it should run from 12-16 panels for any given week of the season. The logic should not rely on having a fixed number of panels per week as it can and will vary.

Sign up or sign in

This is the landing page for the application. Unlike some applications, you are not allowed to do anything until you sign in. If you don't have an account you can sign up here.

1. Information to be entered for new users (sign up flow) must be enough to populate a new document entry in the `Bettors` collection.
2. New users will need to enter a password (twice to make sure no typos). The PPP spoke of having state-of-the-art security, so this password should have some restrictions on it e.g. 1 special, 1 number, 1 upper case, 1 lower case and so forth.
3. CAPTCHA (i.e., "I'm not a robot") would be a nice touch but we did not commit to this. Look into it if you would like to do so or time permitting.
4. Successful sign in or sign up takes user to the fund account page.

Fund account operation aka payment portal

These rules not in chronological order.

1. Page should display user's current balance in text box same as or similar to what is done on betting page.
2. Only available to users who are already authenticated with the system.
3. Users chooses between several offered credit card types e.g., Mastercard, VISA, AmEx, Discover, ??
4. User enters amount to add to their current balance. Whole dollars only.
5. User enters credit card number, expiry month/year, CVV, name, address.
6. Some validation provided for card number e.g. VISA and Mastercard are each 16 digits long.
7. User submits transaction to the system.
8. System delays for a bit then returns a simple confirmation and simultaneously update the current balance text box with the amount just funded.
9. This page backend is updating `Bettors.balance` for the current user.
10. In real life this page is making an actual credit card transaction. We are simulating that experience only.
11. Except for displaying user's balance at the top of the page, you can get ideas for this page from anywhere on the web that you buy things with your credit card. This is just a generic payment screen nothing special towards sports betting except for displaying the user's balance.
12. In real life we would be taking mostly cash not credit card payments. Perhaps one could show that somehow on the page e.g., "Put in your $5, $10, or $20 below." with an image of a cash collector or something. First cut not worried about this but I think it would be a nice touch since sports wagering "on premises" would frequently be a cash business just like buying lottery tickets is a cash business.
13. Once account is funded with some minimum amount there should be a live link they an click to head to the betting page.

Player history

This page shows history of bets for a player, including win/loss and financials.

Work Breakdown

Thoughts I have is that it's best to have each team member take on a particular feature in a "full stack" manner meaning they are responsible for user interface, the Express routes, and getting whatever data is needed in and out of Mongo. I think working at a distance with a short calendar this has the best chance of success. Team members can work mostly independently on their piece initially. Comments?

We would then identify the main pieces of functionality and team members would volunteer what they want to take on.

Item Developer

Project ProposalDale Project Database ProposalDale Project Pitch ProposalMadeline Betting Panel, full stackChristian Fund account via credit card, full stackMahima Sign in or Sign up, full stackAmrutha Player historyMadeline API to betting lines including database storageDale API to game results including database storageDale

Business

Target market

Our end target market is state governments who are looking to expand into sports betting as a means to increase revenue into the state. We are looking to possibly partner with established organizations (including potential competitors) to license our software into their betting terminals and kiosks.

Deployment for betting terminals and kiosks would be gas stations, convenience stores, newstands, and the like.

Potential competitors

1. William Hill
2. Scientific Games
3. Fan Duel
4. Draft Kings
5. International Game Technology
    - https://www.igt.com/products-and-services/sportsbetting
6. And others (see PPP).