System design:

Repo: https://github.com/mofas/CSCI-P565-30620

We will use javascript as our primary language to develop the fantasy game in both frontend and backend. It provides several advantages compared to other languages.

1. The browsers must run javascript, so you must use javascript as the front-end language.
2. If we use the same language in both frontend and backend, we can increase our productivity and decrease learning curve.
3. Javascript is an async language, which is good for building high traffic server.
4. Javascript is the most popular language in the world, there is lots of resource and third parties library available.
5. Since we use the same language, we can use the same tools/ecosystem (IDE, compiler, bundler) as well.

Development Prerequisites:

Node.js (https://nodejs.org/en/):
Node.js is environment for execute your javascript outside browser, which is built based on V8 engine (developed and maintained by Google). It also has built in package management system npm (https://www.npmjs.com/)

Yarn (https://yarnpkg.com/en/):
Yarn is package management client developed by Facebook, which is ultra faster compare to npm.
I highly recommend to use this one to manage your package.

Babel (https://babeljs.io/):
Babel is a compiler for writing next generation javascript, so we don't need to handle the awkward old style javascript. In addition, it can help you to prevent certain potential bugs.

Front-end framework and library:

React (https://facebook.github.io/react/):
React is view library built by Facebook, which is the most popular front-end library in the world to build single page application (SPA). It has shadow-dom to improve render performance and synthesis event system to handle browser's inconsistency.

Redux (http://redux.js.org/):
Redux is a state management library. If React is responsable for View in MVC structure, then redux manages MC.
Actually, we call this structure "flux".

React-Router (https://reacttraining.com/react-router/):
React Router provides universally routing components that compose declaratively with your application, it help us to construct complex navigation behavior in SPA with little effort.

Immutable.js (https://facebook.github.io/immutable-js/):
Immutable.js (developed by facebook as well) allow you to create persistent/immutable data structures in javascript, which prevent certain common bugs related mutable data structures. In addition, it can work with react to achieve “pure rendering” to improve the application performance.

Create-react-app (https://github.com/facebookincubator/create-react-app#getting-started):
Create-react-app is a scaffolding tool for building a new react project, also developed by Facebook. It can save us lots of time on initial configuration all development tools, like babel, webpack, css preprocessor, bundler, and assets compress.

CSS-module (https://github.com/css-modules/css-modules):
CSS module allow developers to write the same class name for styling without worrying about notorious global class name pollutions problem in SPA. Every class name you write in css file will be translated to unique id in CSS declaration.

Back-end framework and library:

Express (https://expressjs.com/):
Express is a light but powerful web application framework, which is pretty easy to set up for a restful API server. It also provide many common modules or “middleware” for building fully functional web application. For example, cookies, session, CSRF, etc.

Passport (http://passportjs.org/):
Passport is a flexible library of node.js if we want to allow users has many different way to login into our system. It currently support authentication using a username and password, Facebook, Twitter, and more.

GraphQL (http://graphql.org/):
GraphQL is very powerful query language for API service aiming to replace restful API. It also open sourced by Facebook as well. It can save us lots of time on common task which restful API usually ask to do, like validation parameter, formatting return data. The best of all, it will automatically generate API document for you through graphiQL (http://graphql.org/swapi-graphql/).

Database:

We plan to use mongoDB (https://www.mongodb.com/) is this project for several reasons.

1. mongoDB is schema free, which means we can change our data schema as we want at any time, that is good for rapid requirement change project.
2. mongoDB save data in JSON format, and JSON is the format most restful API will use.
3. We use javascript in both frontend and backend, and therefore handling JSON data is like breeze for us.
4. mongoDB is high performance and easy to config, backup and restore.
5. There are several free mongodb host services available, so we can have developing, staging databases for free.

Dataset:

We use nflgame (http://pdoc.burntsushi.net/nflgame) as our data API. Nflgame is a free API that real-time NFL data. The API is a Python 2 module. We wrote Python scripts to obtain data from the API calls and wrote the data to a csv file. We have data engineering scripts to create new features such as player rank. We will write scripts to import the csv file into the MongoDB database. The Python scripts will run every week to obtain current NFL game statistics.

Draft:

The draft will proceed in a serpentine fashion. We will choose who gets each pick randomly. Then we will reverse the order of picks for the second round, saving the round and who is picking in our database. The draft will be 20 rounds long with the suggested number of each position to be picked displayed to each user. The league is locked until the draft phase is complete, meaning that no games can be played. Once complete, the schedule will be generated and the league will begin.

Content Delivery / Server:

We tend to use nginx (https://nginx.org/en/) as our proxy to deliver our static files or API if we need to host it on school machines.Otherwise, we can use other free hosting service like github, heroku (https://www.heroku.com/).

Data Schema:

ACCOUNTS: (accounts)

```
{
  _id: ""
  email: "admin",
  password: "password",
  role: "player", // admin, player
  status: "0", // -1 for not validated, 0 for normal user.
  valCode: "w34tesgbw34trg4retq4rtwg",
}
```

PLAYERS: (players)

```
{
  _id: xxxxx,
  picture: "./img/player"
  name: "Tom Brady",
  position: "QB",

  team_id: xxx,
  fancy_score_rank: 1, //Integer

  passing_yds: 12, //Integer
  rushing_yds: 12, //Integer
  receiving_yds: 12, //Integer
  passing_tds: 12, //Integer
  rushing_tds: 12, //Integer
  receiving_tds: 12, //Integer
  kicking_fgm: 12, //Integer
  kicking_fgmissed: 12, //Integer computed by kicking_fga - kicking_fgm
  kicking_xpmade: 12, //Integer
  passing_int: 12, //Integer
  fumbles_lost: 12, //Integer
  defense_int: 12, //Integer
  defense_ffum: 12, //Integer
  defense_sk: 12, //Integer
  defense_kick_blocks: 12, //Integer computed by defense_xpblk + defense_fgblk
  defense_puntblk: 12, //Integer
  defense_safe: 12, //Integer
  kickret_tds: 12, //Integer
  puntret_tds: 12, //Integer
  defense_tds: 12, //Integer

//Holds the data for the current week

  passing_yds_curr: 12, //Integer
  rushing_yds_curr: 12, //Integer
  receiving_yds_curr: 12, //Integer
  passing_tds_curr: 12, //Integer
  rushing_tds_curr: 12, //Integer
  receiving_tds_curr: 12, //Integer
  kicking_fgm_curr: 12, //Integer
  kicking_fgmissed_curr: 12, //Integer computed by kicking_fga - kicking_fgm
  kicking_xpmade_curr: 12, //Integer
  passing_int_curr: 12, //Integer
  fumbles_lost_curr: 12, //Integer
  defense_int_curr: 12, //Integer
  defense_ffum_curr: 12, //Integer
  defense_sk_curr: 12, //Integer
  defense_kick_blocks_curr: 12, //Integer computed by defense_xpblk + defense_fgblk
  defense_puntblk_curr: 12, //Integer
  defense_safe_curr: 12, //Integer
  kickret_tds_curr: 12, //Integer
  puntret_tds_curr: 12, //Integer
  defense_tds_curr: 12, //Integer

}
```

PLAYER_SCORE

```
{
  _id: "xxxx",
  p_id: "xxxx",
  score: [2, 3, 5], //scores by weeks
}
```

LEAGUES (leagues)

```
{
  _id: "xxxx",
  name: "Our First Leagues",
  stage: "Initial", //Initial, Draft, Game, Finish
  limit: 4,
  accounts: ["user_id1", "user_id2"],
  draft_run: 0, //draft_run are from 0 to 79 = 20(player in each team) * 4(limit) - 1
  formula: {
    td_passway_weight: 4,
  }
}
```

POOL (pool)

```
{
  _id: "....",
  league_id: "xxxx",
  player_id: "xxxx",
  account_id: "xxxx",
}
```

ARRANGEMENT (arrangement)

```
{
  _id: "....",
  fantasy_team_id: "xxxx",
  position_qb_0: "player_id_xxx",
  position_rb_0: "player_id_xxx",
  position_wr_0: "player_id_xxx",
  position_wr_1: "player_id_xxx",
  position_te_0: "player_id_xxx",
  position_k_0: "player_id_xxx",
  position_p_0: "player_id_xxx",
  position_defense_0: "player_id_xxx",
  position_defense_1: "player_id_xxx",
  position_defense_2: "player_id_xxx",
  position_defense_3: "player_id_xxx",
  position_defense_4: "player_id_xxx",
}
```

FANTASY TEAMS (fantsay_team)

```
{
  account_id: "xxxx",
  fancy_team_id: "xxx",
  league_id: "xxx",
  name: "My awesome teams"
  wins: 2,
  lose: 3,
}
```

GAME_RECORD

```
{
  record_id: "xxxxx",
  league_id: "xxx",
  week: 1,
  first_team: "xxxxx",
  second_team: "xxxxxx",
  winner: 0, // 0 mean first, 1 mean second
  first_score: 44,
  second_score: 23,
}
```

SCHEDULE (schedule)

```
{
  week_no: 1, //Integer
  first_team: "xxxxx",
  second_team: "xxxxx",
}
```