**Project Three:**

[**IDEAS AND TEAM WORKING NOTES HERE**](https://docs.google.com/document/d/13hyAeyYjg3v18FUfRP068TZdQl4Rx0xZPPvhey_kjvM/edit?usp=sharing)

|  |
| --- |
| **Team Members** |
| Alex Aguirrebena |
| Jen Jayme |
| Giovanni Noisy |
| Royce Williams |
| Stacey Yoon |

## **Project Task Schedule**

|  |  |  |
| --- | --- | --- |
| **Date** | **Task** | **Notes** |
| **08/06** | Intro to React & JSX |  |
| **08/07** | React State |  |
| **08/08** | Saturday (No Class) |  |
| **08/09** | Sunday (No Class) |  |
| **08/10** | React Forms and React Router |  |
| **08/11** | React Hooks and Context API |  |
| **08/12** | React and the Global Store |  |
| **08/13** | MERN |  |
| **08/14** | React Review |  |
| **08/15** | Saturday (No Class) |  |
| **08/16** | Sunday (No Class) |  |
| **08/17** | Project Work  (Optional CS Review) |  |
| **08/18** | Project Work  (Optional CS Review) |  |
| **08/19** | Project Work  (*Ruby & Python)*  (Optional CS Review) |  |
| **08/20** | Project Work  (*Performance*)  (Optional CS Review) |  |
| **08/21** | Project Work  (*Breakout*)  (Optional CS Review) |  |
| **02/22** | Saturday (No Class) |  |
| **02/23** | Sunday (No Class) |  |
| **03/24** | Project Work (Optional CS Review) |  |
| **03/25** | Project Work (Optional CS Review) |  |
| **03/26** | Mock Interview &  Project Work | MOCK INTERVIEW |
| **03/27** | Project Work |  |
| **03/28** | Project Work  READY MVP  MOCK PROJECT PRESENTATION | READY MVP  MOCK PROJECT PRESENTATION |
| **03/29** | Saturday (No Class) | PROJECT POLISH |
| **03/30** | Sunday (No Class) | PROJECT POLISH |
| **03/31** | Presentations &  LAST DAY OF CLASS |  |

## 

## 

## **Coding Requirements:**

* Must use ReactJS in some way (even if minimal)
* Must use a Node and Express Web Server
* Must be backed by a MySQL or MongoDB Database with a Sequelize or Mongoose ORM
* Must have both GET and POST routes for retrieving and adding new data
* Must be deployed using Heroku (with Data)
* Must utilize at least two libraries, packages, or technologies that we haven’t discussed
* Must allow for or involve the authentication of users in some way
* Must have a polished frontend / UI
* Must have folder structure that meets MVC Paradigm
* Must meet good quality coding standards (indentation, scoping, naming)

## **Expectations:**

* We expect whatever you build to have utility
* We expect you to have market or real-world research that evidences your idea has REAL value to people.
* We expect you to have done research on other web / mobile applications in your domain.
* We expect you to put serious time and thought into this.
* We expect you to report problems you are facing along the way.
* We expect you to utilize some form of project management system.
* We expect you to dig deep into documentation and external resources to learn what you need.

# **Teamwork & Expectations**

## **Team Effort**

Before anything, remember that Projects are a group effort: Working closely with your teammates is a requirement. This both helps teach real-world collaborative workflows, and enables you to tackle more difficult problems than you'd be able to working alone.

In other words, working in groups allows you to work smart and dream big. Take advantage of it!

**Note**:

* 100% participation and attendance is required in order to receive credit for the Group Project work.
* Participation in all 3 Projects are course requirements.

## **Git Workflow**

How will you organize your branches? General Workflow? What is your target number of commits?

We have established a master repo and added all team members as collaborators. Each team member has cloned the master repo and created a branch on their desktop. Team members will do all work in their branch, then “git push origin branch” so changes are incorporated in their own master, then submit a “Pull Request” to merge their changes into the repo master.

|  |
| --- |
| Git Commit # |
| 200 |

## **Stuck time ( How long until you consult a teammate?):**

* 30 minutes: Team members agree to consult a teammate after 30 minutes of stuck time.

## **Working Agreements:**

Examine these resources:

* <http://www.iliokb.com/2012/04/example-working-agreement.html>
* <http://www.payton-consulting.com/wp-content/uploads/2014/07/WorkingAgreements.jpg>

**EXAMPLE:**

During Sprint Do The Following:

|  |
| --- |
| **Your Team Working Agreements** |
| During Sprint Do The Following:   * Team huddles are important to keep up momentum and build our synergy. We aim to have them regularly and have everyone present. * If a team member has a time conflict, s/he/they updates the team in advance. Efforts may be made to shift the meeting time, depending on agenda and availability. However, we prefer meetings scheduled at least a day in advance, and prefer sticking to previously scheduled times whenever possible. * We are all expected to be working on the project nearly every day. * Every team member is committed to the value of the application over individual recognition. * Every team member will practice active listening and create space to hear the thoughts of all others. When decisions need to be made, we agree to call for a round of input from each member to create space for everyone to be heard. * Every team member will be directly engaged with the work, not answer texts or phone calls, social media or other off-topic material, except in emergencies which will be communicated to the team * We want to plan the development steps better in advance and be more intentional and planful about dividing the workload rather than scurrying in all directions. |

## **Agile Stand-Ups:**

### For software teams, the stand-up is like the team’s huddle. It’s even commonly known as the daily scrum, and reinforces “we” to keep everyone aware of the team’s landscape and progress.

A stand-up is a daily meeting that involves the core team.

This meeting’s flavor is unique to each team, but commonly we use three simple questions to generate structure:

1. What did I work on yesterday?
2. What am I working on today?
3. What issues are blocking me?

These questions highlight progress and help flag team blockers. Also, it strengthens the team when everyone shares the progress they’re contributing to the team. The daily reinforcement of sharing individual successes and plans keeps everyone excited about the team’s overall progress.

## **Agile Playbacks:**

Teams sometimes have a weekly/bi-weekly meeting called a "playback". This meeting allows team members to explain and demo completed features and the work they did during this past sprint. (normally a work week). Playbacks are important to keep project managers up to date, as well other team members to what has been accomplished during a sprint.

The playback format:

* + Tell your team what you worked on and how it went.
  + Show the progress you have made on your work.
  + Demo any finished work/features.

We encourage you to set up time for playbacks at the end of each sprint.

# **Application Info:**

|  |  |
| --- | --- |
| **Project Title:** | Bragging Rights (working title) |

## **Logo Image**



## **Project Description**

|  |
| --- |
| We aim to create a platform for gamers where a group of friends can initiate multi-game tournaments, and generate a dashboard of visualized statistics for all players over the course of the tournament.  Avid gamers have access to performance statistics and often compare their statistics, but don’t currently have a platform that enables them to create a tournament-style competition of a defined set of games, and limited number of competitors. Friends often play against others and debate which statistics are most meaningful. Our platform will enable a circle of friends to create a unified scoring rubric with some performance statistics weighted more than others. In general the platform will bring individual gaming into a more communal, specific and time-limited activity among friends.  For our MVP, we’ll start with one popular game that has a relatively user-friendly API - “Call of Duty.” Once functional, we’ll try to add another popular game (League of Legends) as a stretch goal.  When a registered player creates a tournament, they will have a chance to specify when it will start, how many games to include and how to set up their scoring rubric, guided by suggested defaults. They will invite friends to participate using a code and start the tournament once enough players have signed up. The tournament dashboard will track the number of games specified by tournament creator, and return tallied performance scores to a dashboard. |

Before you start writing any code, your group should outline the scope and purpose of your project. This helps provide direction and prevent [scope creep](https://en.wikipedia.org/wiki/Scope_creep).

Write this as a brief summary of your interests and intent, including:

* Problem / Motivation (Why you feel it is valuable)
* How it addresses the problem
* Possible source for such data

## **Repo Links**

|  |
| --- |
| Code Repo URL |
| <https://github.com/seyouny/StatsDash> |

## 

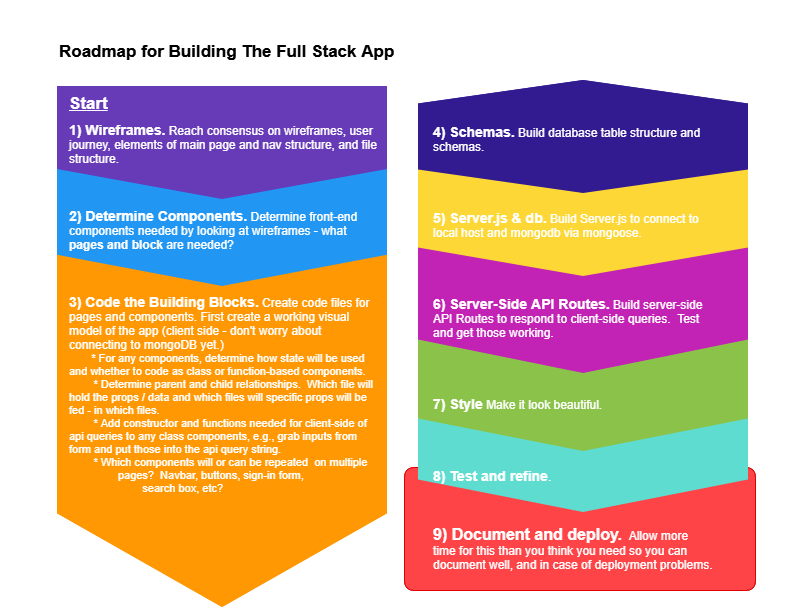
|  |
| --- |
| Deployed URL |
| <https://obscure-escarpment-21392.herokuapp.com/> |

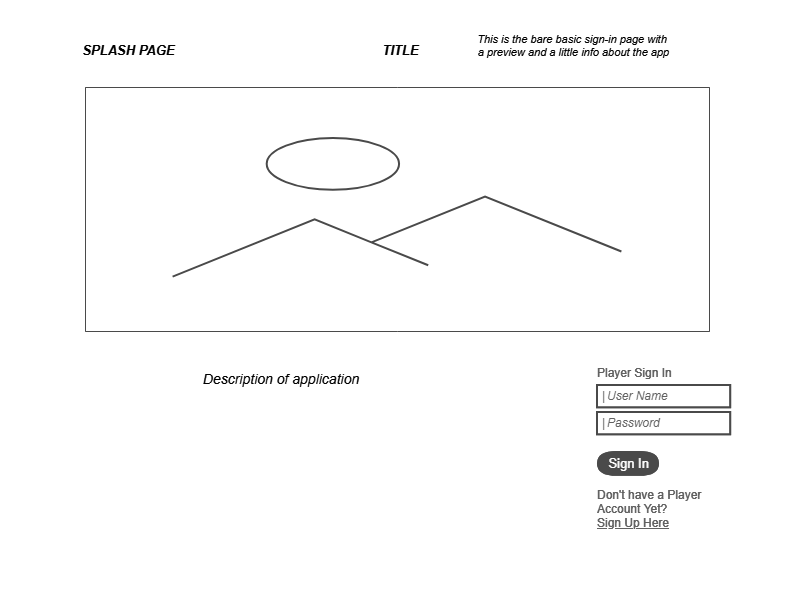
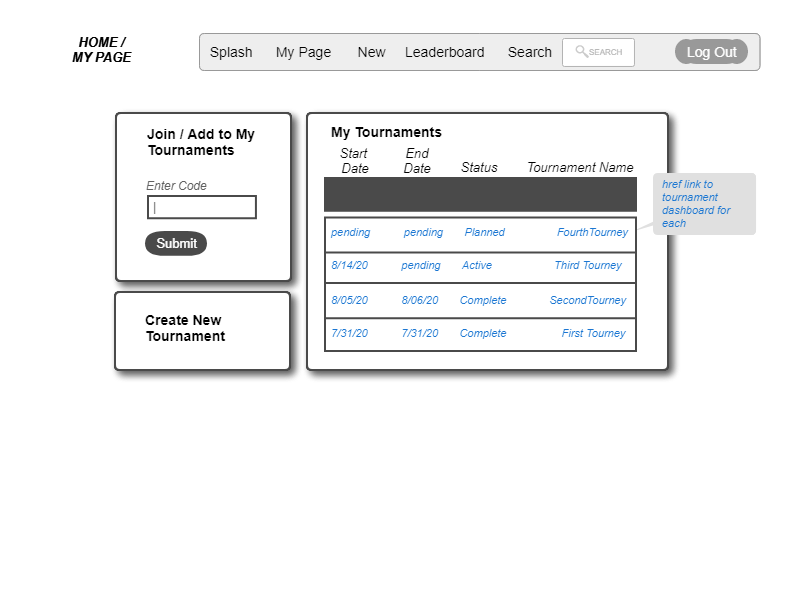
# **Planning: Design**

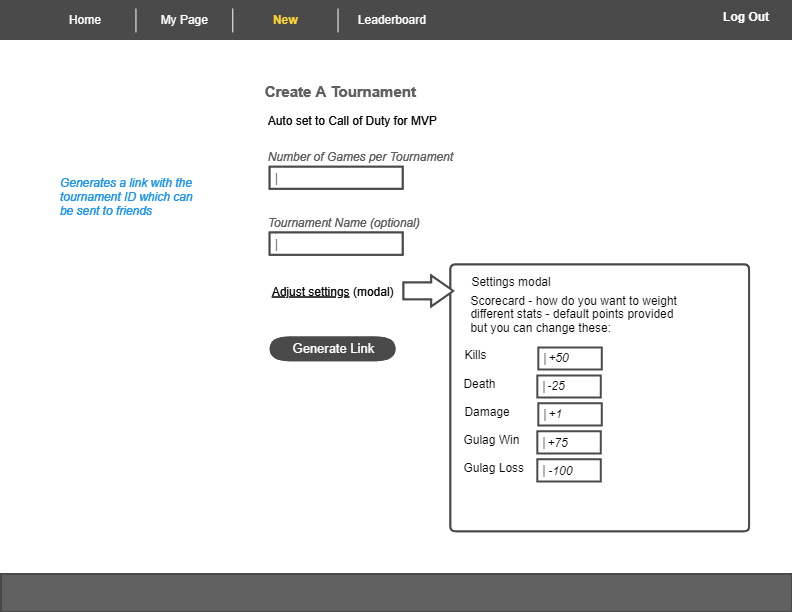
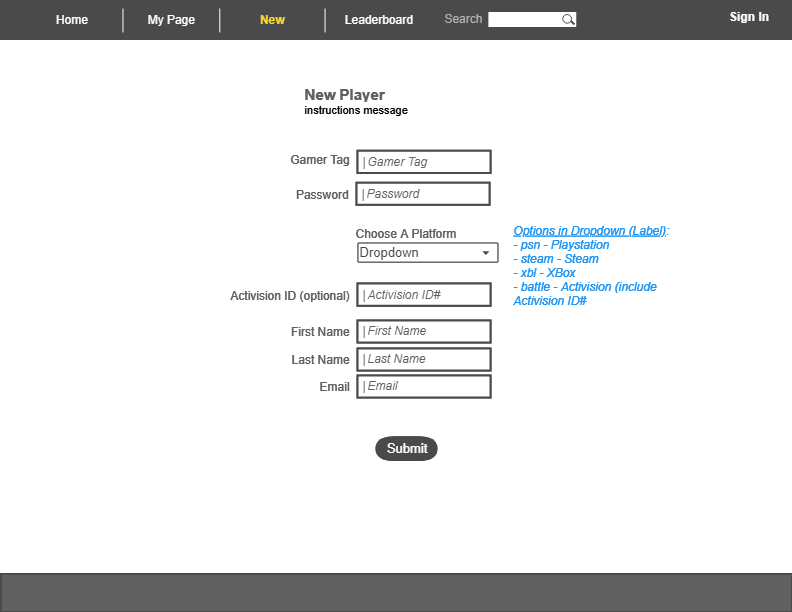
## **User Stories**

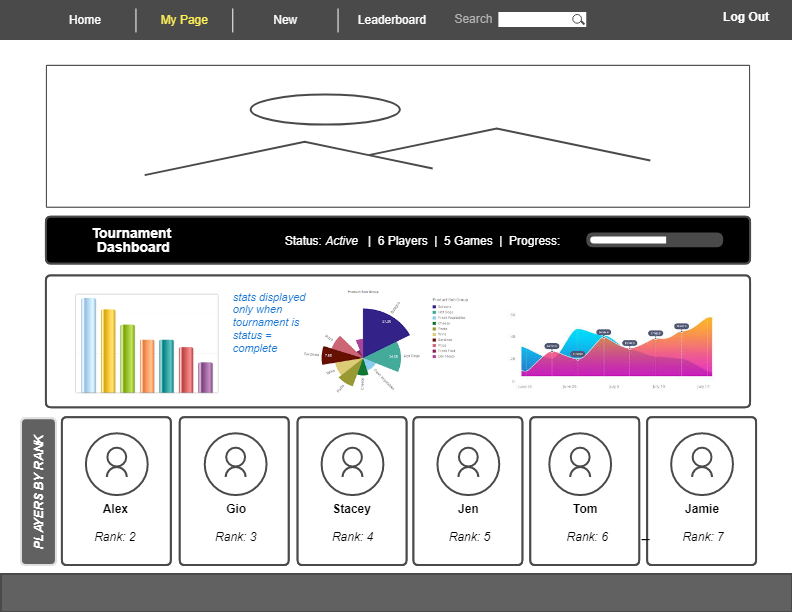
|  |  |  |
| --- | --- | --- |
| As a <role>, | I want <feature> | so that <reason>. |
| 1 Serious Gamer | To compare performance among my friends and acquaintances | We have a common rubric for rating our skill levels at a particular game. |
| 2 Hobby Gamer | To have a structured way to participate in a finite series of games with a small group of players | Our play extends across multiple games yet has an end point, and we can measure skills across a set of games. |
| 3 Recreational Gamer | To play my favorite game with others and see the results across multiple games | I can build a circle of favorite game partners and improve my skills. |

## **WireFrame Images**

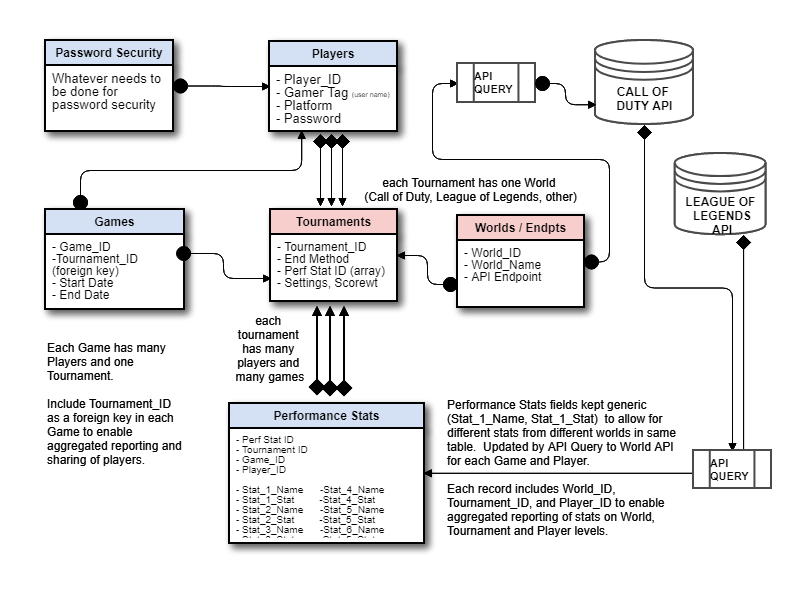






## **Models and Columns:**



|  |  |  |  |
| --- | --- | --- | --- |
| **Model Name** | **Model Attributes** | **Model validations** | **Model Associations** |
| **Main Tournament Dashboard** | Stateful component:  Has 3 states: admin, active tournament & completed tournament | 3 states - admin, active tournament & completed tournament | Gets data from performance stat table |
| **Player** | Player Context passes user info and login status throughout all pages. Stateful component to track logged in or not. | Username must be unique, maps to gamer tag, has associated password | Password hash protection.  User ID associated with each game, tourney & stat. |
| **Performance Stats Table** | Class-based component with many properties. | Rows corresponds to # of games per player | Props include player ID, tournament ID, game ID. |
| **Columns** | (Described in above visual for each table) |  |  |

## **Describe Associations (Has Many/Belongs To):**

Each PLAYER can have many TOURNAMENTS.

Each TOURNAMENT always has many PLAYERS.

Each TOURNAMENT has many GAMES;

Each GAME belongs to one TOURNAMENT;

Each PERFORMANCE STAT-SET (row/record) belongs to one PLAYER, one GAME, and one TOURNAMENT; the record has 5 to 15 “stat” columns.

The number of PERFORMANCE STAT-SETs will correspond to the number of games x the number of players; that is, for 5 players in a TOURNAMENT of 4 GAMES, there will be 20 PERFORMANCE STAT-SET records..

## **GET and POST Routes (Restful Routes):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resource** | **URL** | **HTTP Verb** | **Action** | **Used For** |
| Post (example) | /api/posts | GET | READ | Returning JSON data for ALL Posts |
| Gamer Stats | /api/stats/  :gamertag | GET | READ | Returns gamer stats |
| Tournament Stats | /api/stats/  :tournID | GET | READ | Returns tournament stats |
| Player Name | /api/new/player | POST | SAVE | Save player name & password |
| Create New Tournament | /api/new/tourn | POST | SAVE | Save new tournament |

# **Planning: Technology**

## **APIs to be Used:**

|  |  |  |  |
| --- | --- | --- | --- |
| API # | API Base URL | Paraments | What is it? |
| 1 | Call of Duty  https://call-of-duty-modern-warfare.p.rapidapi.com/warzone/{gamertag}/{platform} | gamertag  platform | Performance statistics on all players of Call of Duty |
| 2 | League of Legends  [/lol/champion-mastery/v4/champion-masteries/by-summoner/{encryptedSummonerId}/by-champion/{championId}](https://developer.riotgames.com/apis#champion-mastery-v4/GET_getChampionMastery) | championID SummonerID | Performance statistics on all players of League of Legens |
| 3 | Firebase  https://identitytoolkit.googleapis.com/v1/accounts:signInWithCustomToken?key=[API\_KEY] | customtoken | We would use the password authentication feature |

**Screenshot or code Snippet of APIs to be Used:**

var unirest = require("unirest");

var req = unirest("GET", "https://call-of-duty-modern-warfare.p.rapidapi.com/multiplayer/Chob%252321309/battle");

req.headers({

"x-rapidapi-host": "call-of-duty-modern-warfare.p.rapidapi.com",

"x-rapidapi-key": "6666346643msh3652df8799d7cf8p1b61b8jsn7d9d2e61466e",

"useQueryString": true

});

req.end(function (res) {

if (res.error) throw new Error(res.error);

console.log(res.body);

});

**Screenshot or code Snippet of RESPONSE from APIs to be Used:**

***{ 19 items***

**"engagement":NULL**

**"level":78**

**"levelXpGained":7000**

**"levelXpRemainder":3000**

**"lifetime":{*6 items***

**"accoladeData":{...}*1 item***

**"all":{...}*1 item***

**"itemData":{...}*12 items***

**"map":{}*0 items***

**"mode":{...}*16 items***

**"scorestreakData":{...}*2 items***

**}**

**"maxLevel":0**

**"maxPrestige":0**

**"p":1**

**"paragonId":0**

**"paragonRank":0**

**"platform":"battle"**

**"prestige":0**

**"prestigeId":0**

**"s":1**

**"title":"mw"**

**"totalXp":960000**

**"type":"mp"**

**"username":"Lierrmm#2364"**

**"weekly":{*3 items***

**"all":{...}*1 item***

**"map":{}*0 items***

**"mode":{}*0 items***

**}**

**}**

## **Libraries to be Used:**

|  |  |  |  |
| --- | --- | --- | --- |
| Library # | Doc Link | What does it do? | How did you use it? |
| 1 | jQuery <https://jquery.com/> | Target DOM elements on client sde |  |
| 2 | React-Bootstrap  <https://react-bootstrap.github.io/> | CSS Framework specifically for React | Page structure |
| 3 | React Style-Guidist <https://react-styleguidist.js.org/> | React development environment | Helps you develop and visualize react components in isolation |
| 4 | ObservableHQ <https://observablehq.com/> | Data visualizations / charts | Data visualizations / charts |

## **Packages Required:**

|  |  |  |  |
| --- | --- | --- | --- |
| Package # | Package Link | What does it do? | How did you use it? |
| 1 | MySql  <https://www.mysql.com/> | Relational database platform | For saving player, tournament and stats info |
| 2 | Sequelize <https://sequelize.org/> | Object-Relational Mapper | To assist in constructing tables |
| 3 | Express <https://expressjs.com/> | Server-side routing | To support routing of get and post requests to database |
| 4 | Axios <https://www.npmjs.com/package/axios> | Client-side API queries | To support queries to Call of Duty API |

## **Migration Strategy:**

* Seeds.sql
* Schema.sql

## **Authentication of Users:**

We’ll use Firebase’s drop-in authentication solution to validate the user's email address and password.

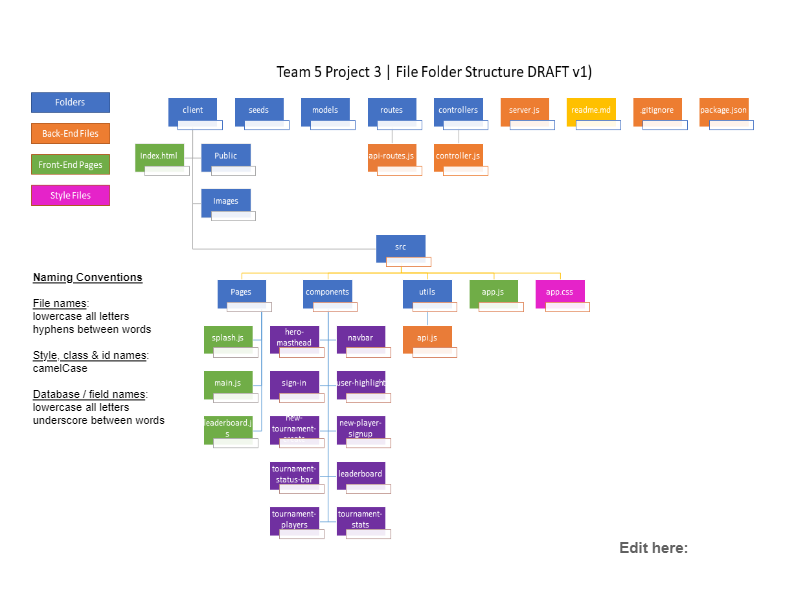
## **Validations:**

Password given matches password associated with email address.

## **React Components**

To include the following:

* Player (User) Context
* Main Dashboard
* Navbar
* Admin Buttons
* New Player Signup
* New Tournament Create



# **Project Reflection**

## **Retrospective Notes (Due Date):**

**Team Reflection**:

* What Went Well
* What Did not go well
* What can **WE** improve for next time

**Team Member Feedback (directed at your team member(s))**:

* What Went Well
* What Did not go well
* What can **YOU** improve for next time

|  |
| --- |
| **Team Reflection & Team Member Feedback** Google Doc Link |
|  |

**Self Reflection**:

* What Went Well
* What Did not go well
* What can **I** improve for next time

|  |  |  |
| --- | --- | --- |
| **Self Reflection** Link |  |  |
| Name |  |  |
| Name |  |  |
| Name |  |  |
| Name |  |  |

## 

## **Blog / Medium Post (Due Date):**

|  |  |
| --- | --- |
| **Blog / Medium**  Link |  |
| Team Member #1 |  |
| Team Member # 2 |  |
| Team Member # 3 |  |
| Team Member # 4 |  |

# **Notes:**