CodingLab
Powered by RBK

Bootstrap

Data Modeling

Part I: Modeling a Single Entity

An introduction to data modeling

Duration: 30 minutes
Q&A: 5 minutes by the end of the lecture

What is Data Modeling?

We often write programs that aim to recreate some aspect of the real world: people, places, and things; and the properties and behaviors that pertain to them.

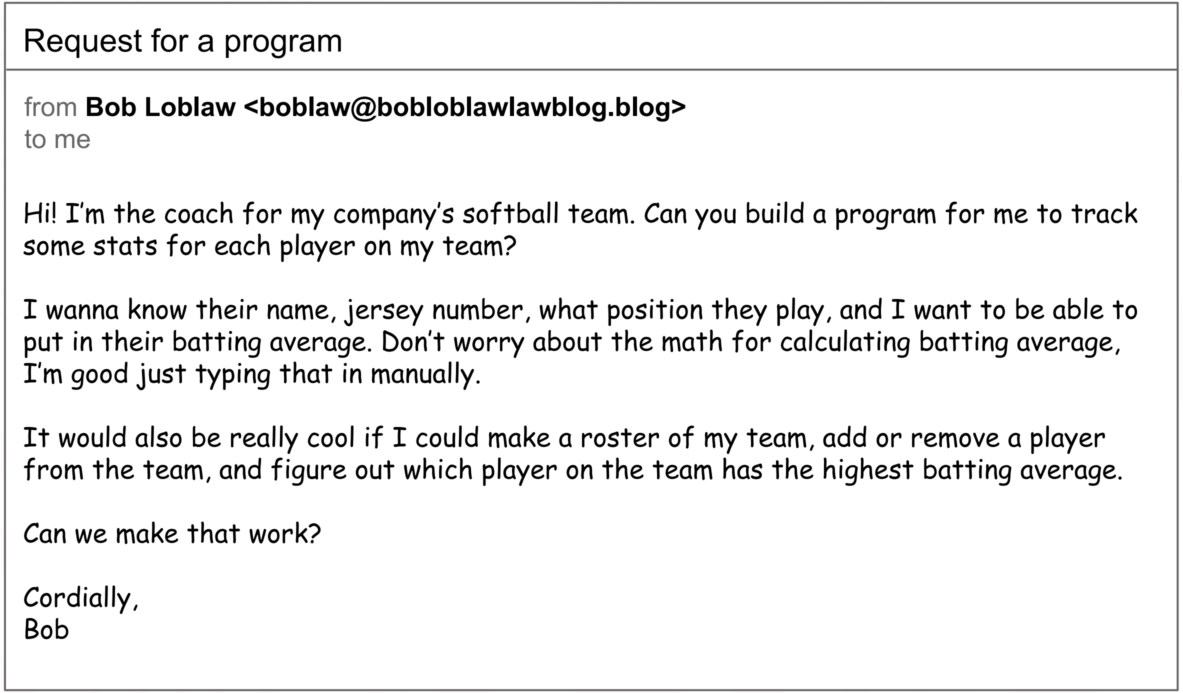
We can use code to create **models** of these people, places, or things. In JavaScript, objects can help us store related data while functions can help us simulate behaviors.

Data Modeling is the process by which we arrive at these simulated objects and behaviors.

Getting Familiar with Data Modeling

In this lesson, we'll demonstrate the data modeling process by simulating the experience of writing code for a client.

The client has sent us an email specifying what they need our program to do...



Request for a program

from **Bob Loblaw <boblaw@bobloblawlawblog.blog>**
to me

Hi! I'm the coach for my company's softball team. Can you build a program for me to track some stats for each player on my team?

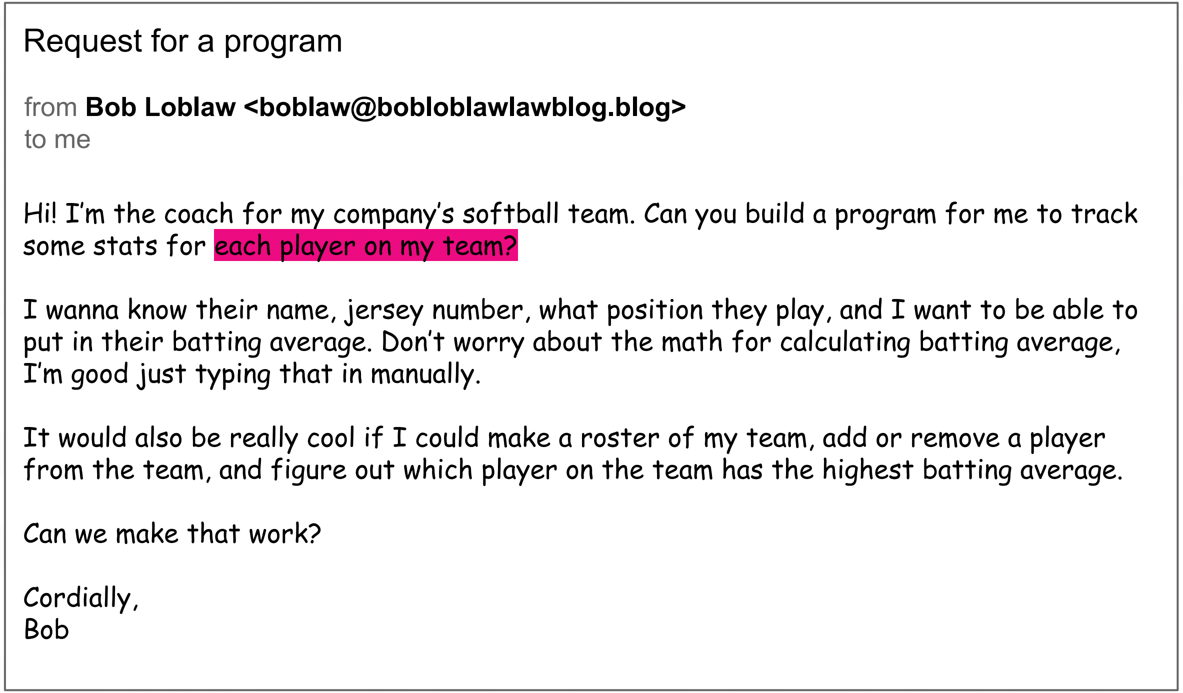
I wanna know their name, jersey number, what position they play, and I want to be able to put in their batting average. Don't worry about the math for calculating batting average, I'm good just typing that in manually.

It would also be really cool if I could make a roster of my team, add or remove a player from the team, and figure out which player on the team has the highest batting average.

Can we make that work?

Cordially,
Bob

Let's analyze Bob's request and see what the requirements of our problem are.



Request for a program

from **Bob Loblaw <boblaw@bobloblawlawblog.blog>**
to me

Hi! I'm the coach for my company's softball team. Can you build a program for me to track some stats for each player on my team?

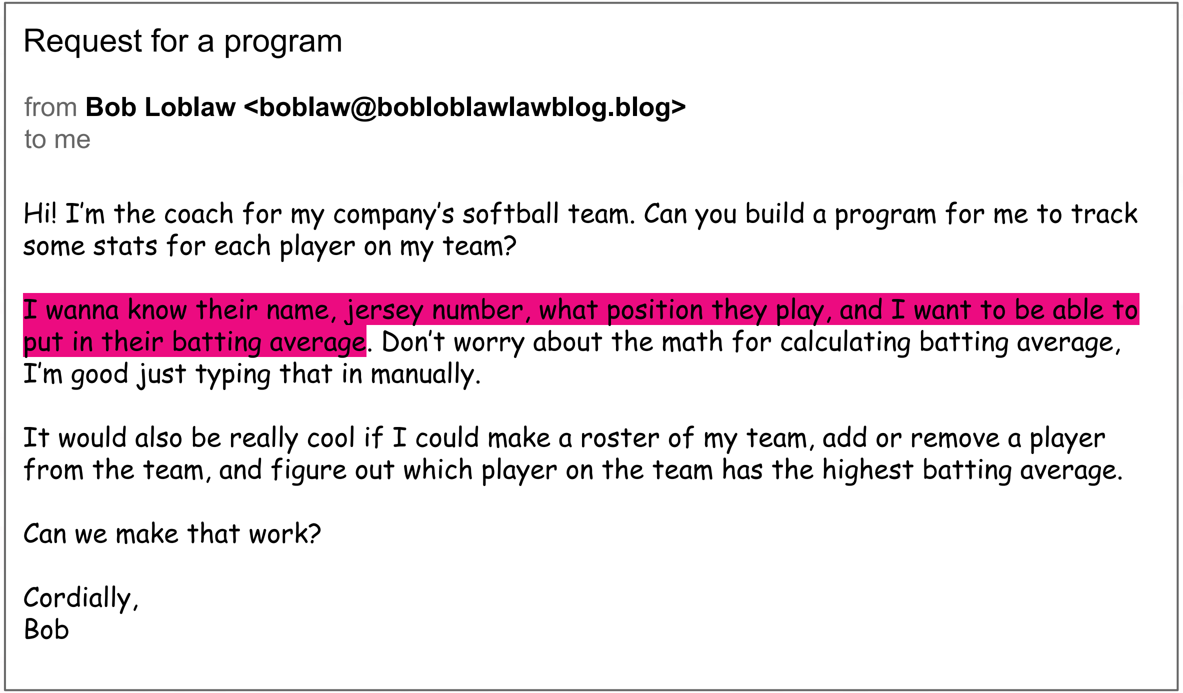
I wanna know their name, jersey number, what position they play, and I want to be able to put in their batting average. Don't worry about the math for calculating batting average, I'm good just typing that in manually.

It would also be really cool if I could make a roster of my team, add or remove a player from the team, and figure out which player on the team has the highest batting average.

Can we make that work?

Cordially,
Bob

For this program, we'll be tracking information related to a softball player. We might to refer to each softball player as an **entity** in our system - a distinct unit of information.



Request for a program

from **Bob Loblaw <boblaw@bobloblawlawblog.blog>**
to me

Hi! I'm the coach for my company's softball team. Can you build a program for me to track some stats for each player on my team?

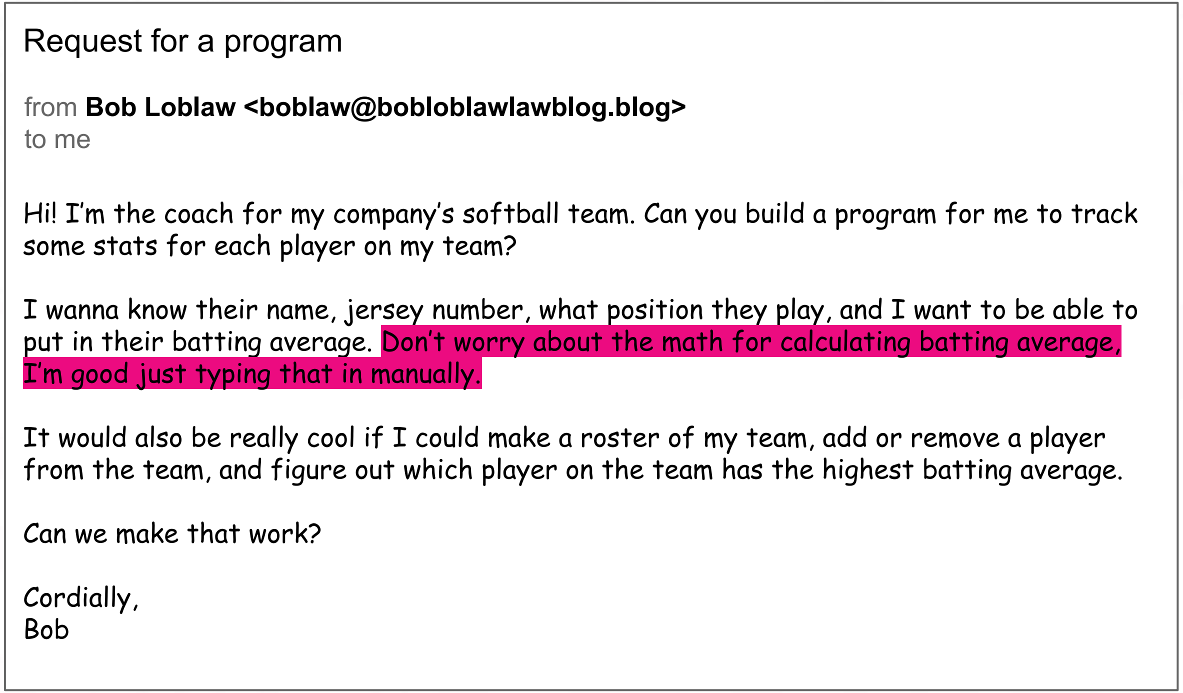
I wanna know their name, jersey number, what position they play, and I want to be able to put in their batting average. Don't worry about the math for calculating batting average, I'm good just typing that in manually.

It would also be really cool if I could make a roster of my team, add or remove a player from the team, and figure out which player on the team has the highest batting average.

Can we make that work?

Cordially,
Bob

Here, the client has spelled out what **properties** we need to simulate for each player on the team.



Request for a program

from **Bob Loblaw <boblaw@bobloblawlawblog.blog>**
to me

Hi! I'm the coach for my company's softball team. Can you build a program for me to track some stats for each player on my team?

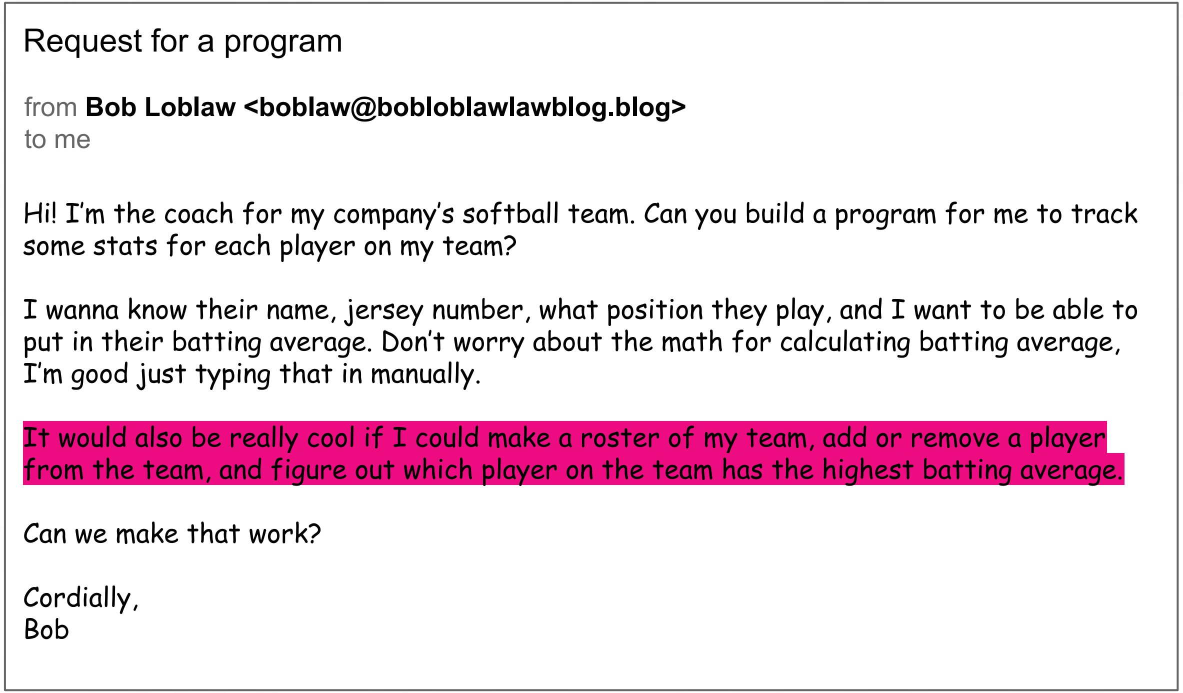
I wanna know their name, jersey number, what position they play, and I want to be able to put in their batting average. Don't worry about the math for calculating batting average, I'm good just typing that in manually.

It would also be really cool if I could make a roster of my team, add or remove a player from the team, and figure out which player on the team has the highest batting average.

Can we make that work?

Cordially,
Bob

It looks like we'll need to provide some way for the client to update the batting average, but the client has some pretty simple expectations for this part of our program.



Request for a program

from **Bob Loblaw <boblaw@bobloblawlawblog.blog>**
to me

Hi! I'm the coach for my company's softball team. Can you build a program for me to track some stats for each player on my team?

I wanna know their name, jersey number, what position they play, and I want to be able to put in their batting average. Don't worry about the math for calculating batting average, I'm good just typing that in manually.

It would also be really cool if I could make a roster of my team, add or remove a player from the team, and figure out which player on the team has the highest batting average.

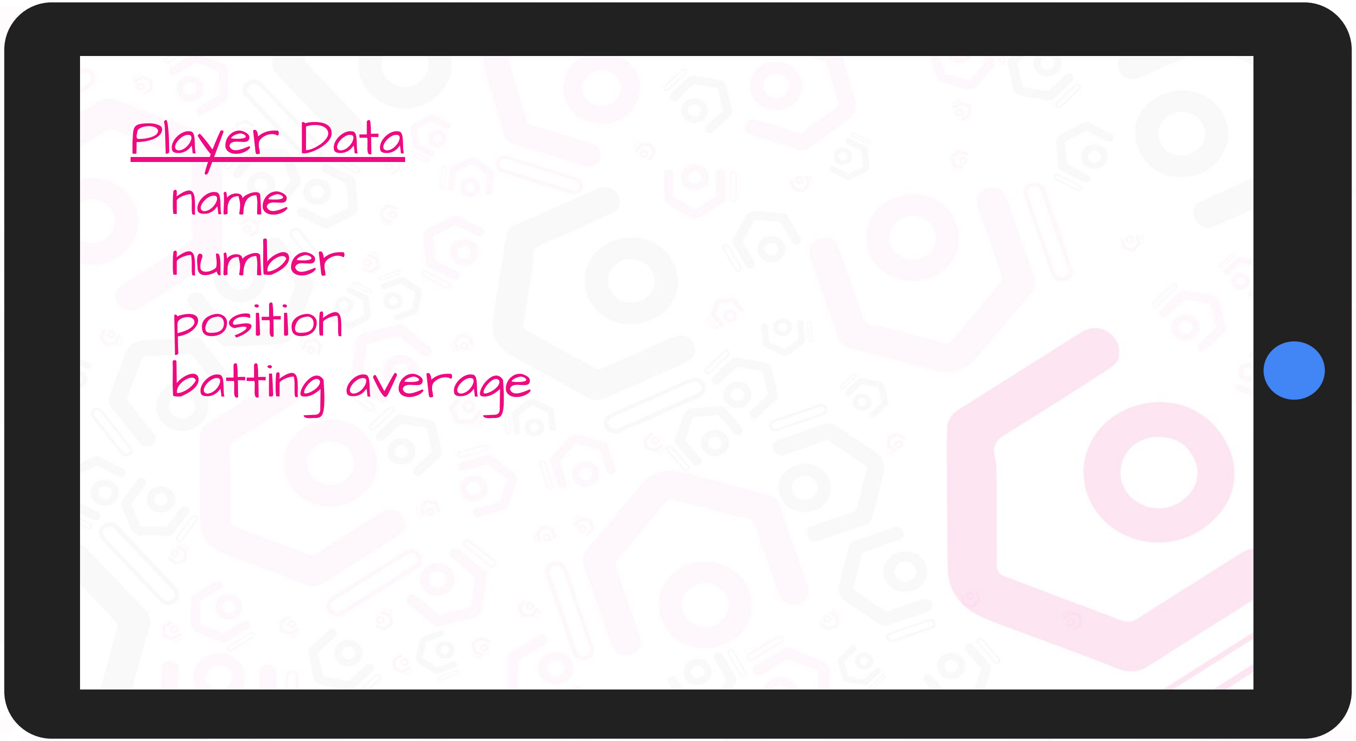
Can we make that work?

Cordially,
Bob

To fulfill the requests in this paragraph, we'll need to add in some behavior that considers an entire team of softball players, and does some work accordingly.

Data Modeling

Let’s consider what we’ve just learned from analyzing our client’s request. We’ll make some notes on the whiteboard to help us arrive at a clear mental picture of what we want before we begin writing code.

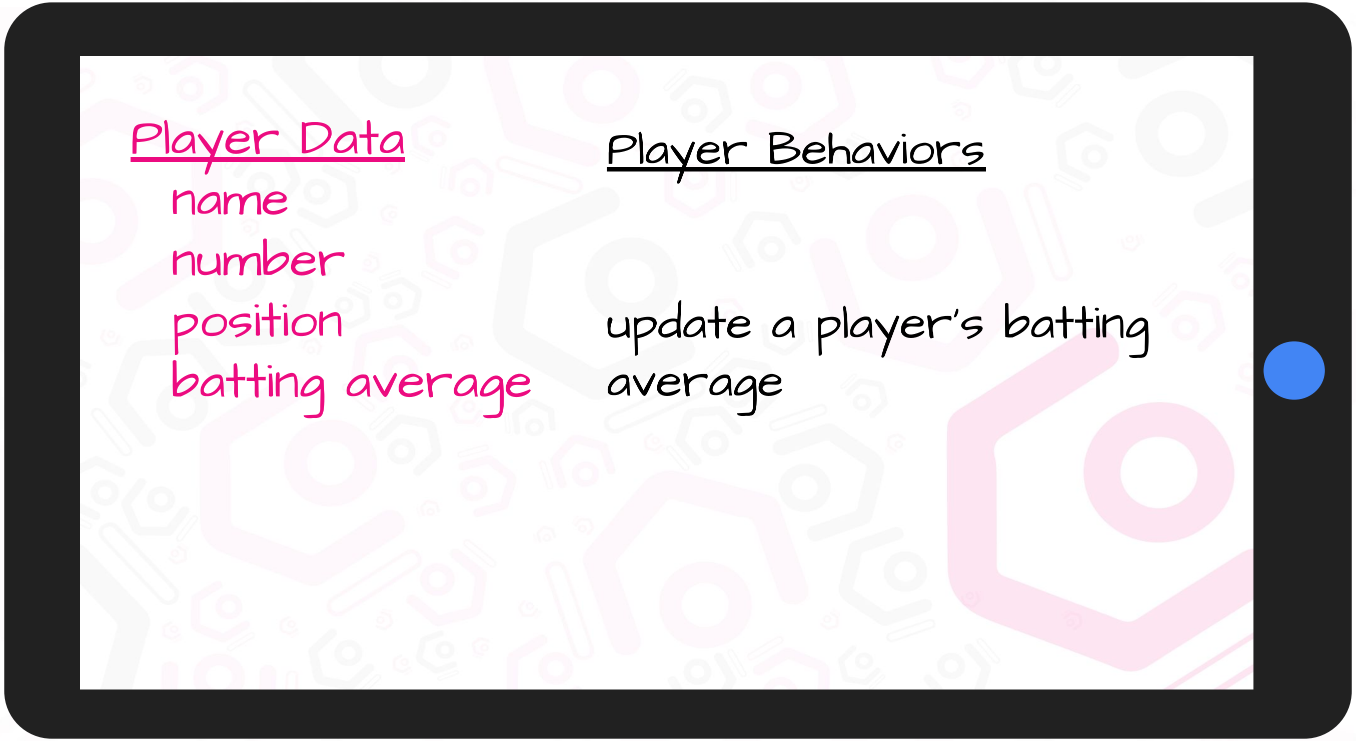


Player Data

- name
- number
- position
- batting average

Players are our primary entity. We'll make a list of the different characteristics that we'll track for each player.

Data Modeling



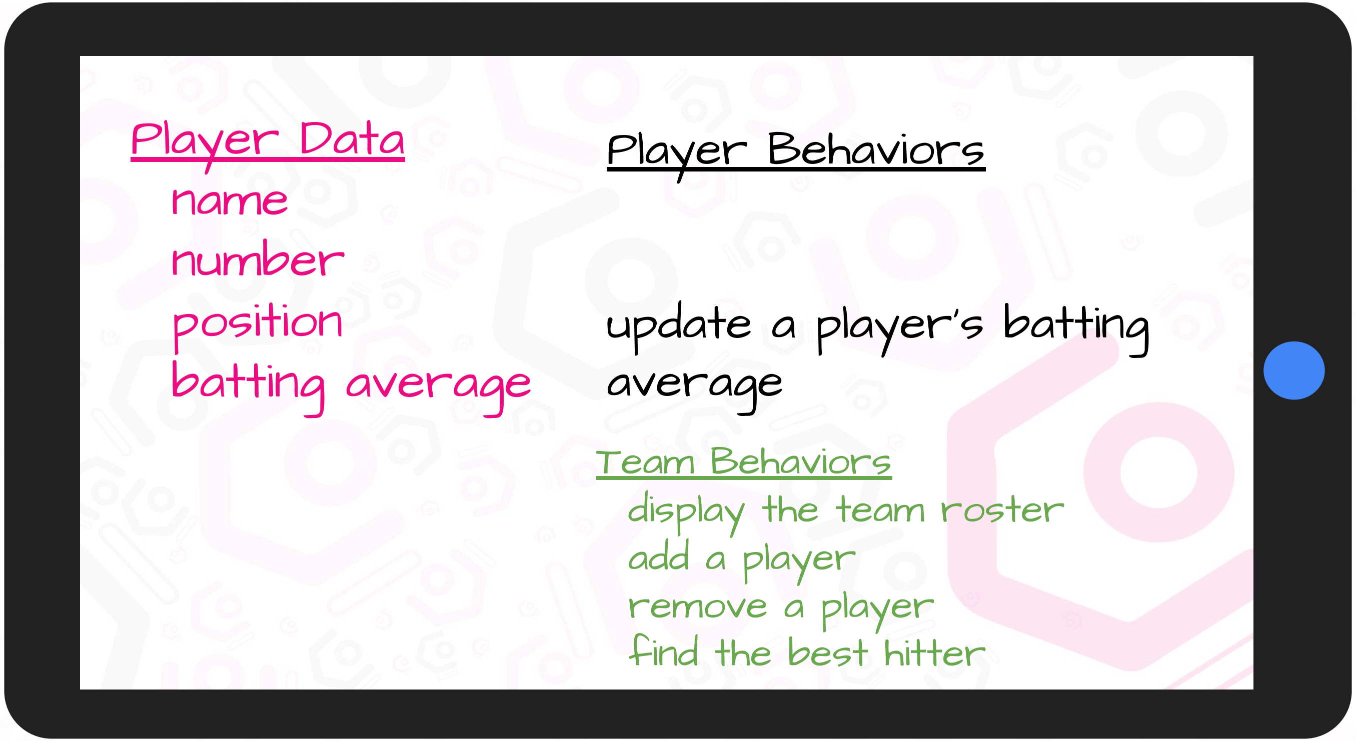
Player Data

- name
- number
- position
- batting average

Player Behaviors

update a player's batting average

The client only specified one kind of behavior for a player - updating their batting average. We may discover the need for more actions later, so we'll leave a little room just in case.



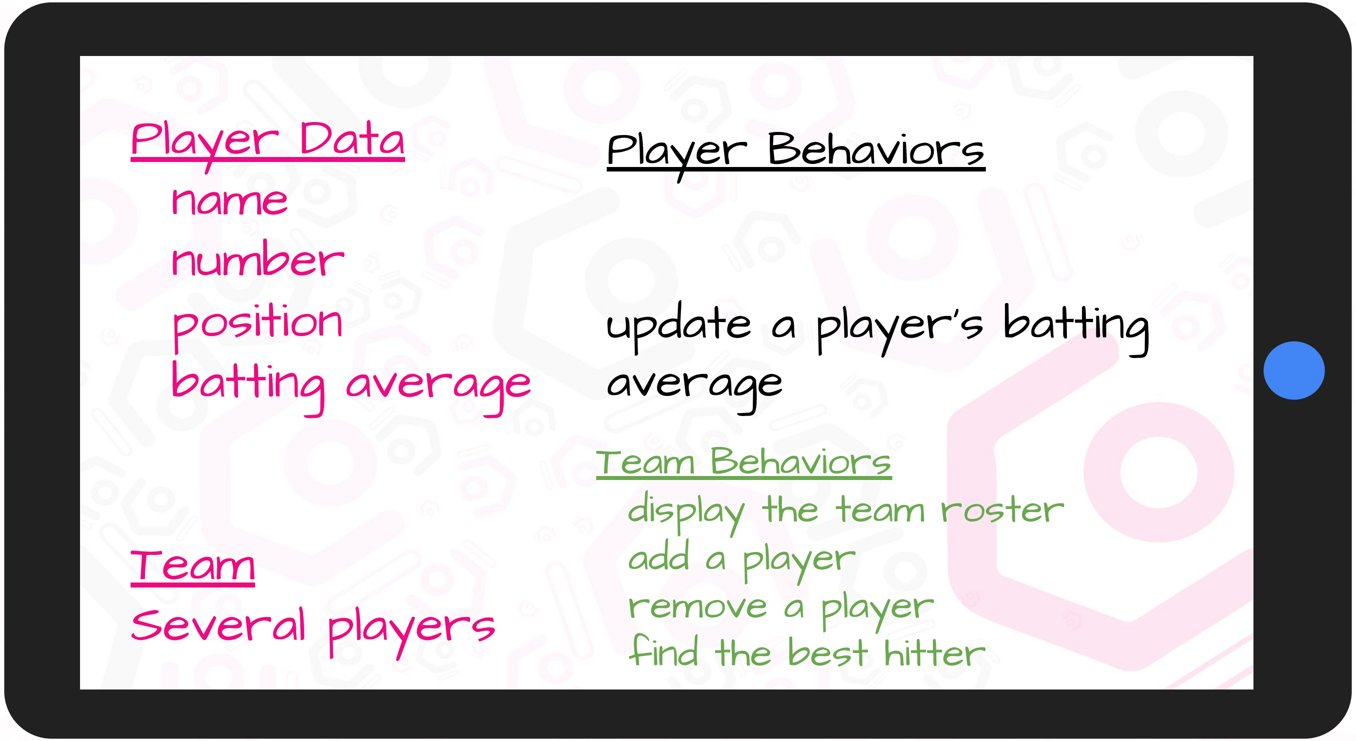
Player Data
- name
- number
- position
- batting average

Player Behaviors
- update a player's batting average

Team Behaviors
- display the team roster
- add a player
- remove a player
- find the best hitter

The client's request included a number of behaviors related to the team, so let's make a list of those.

Data Modeling



Player Data

name

number

position

batting average

Team

Several players

Player Behaviors

update a player's batting average

Team Behaviors

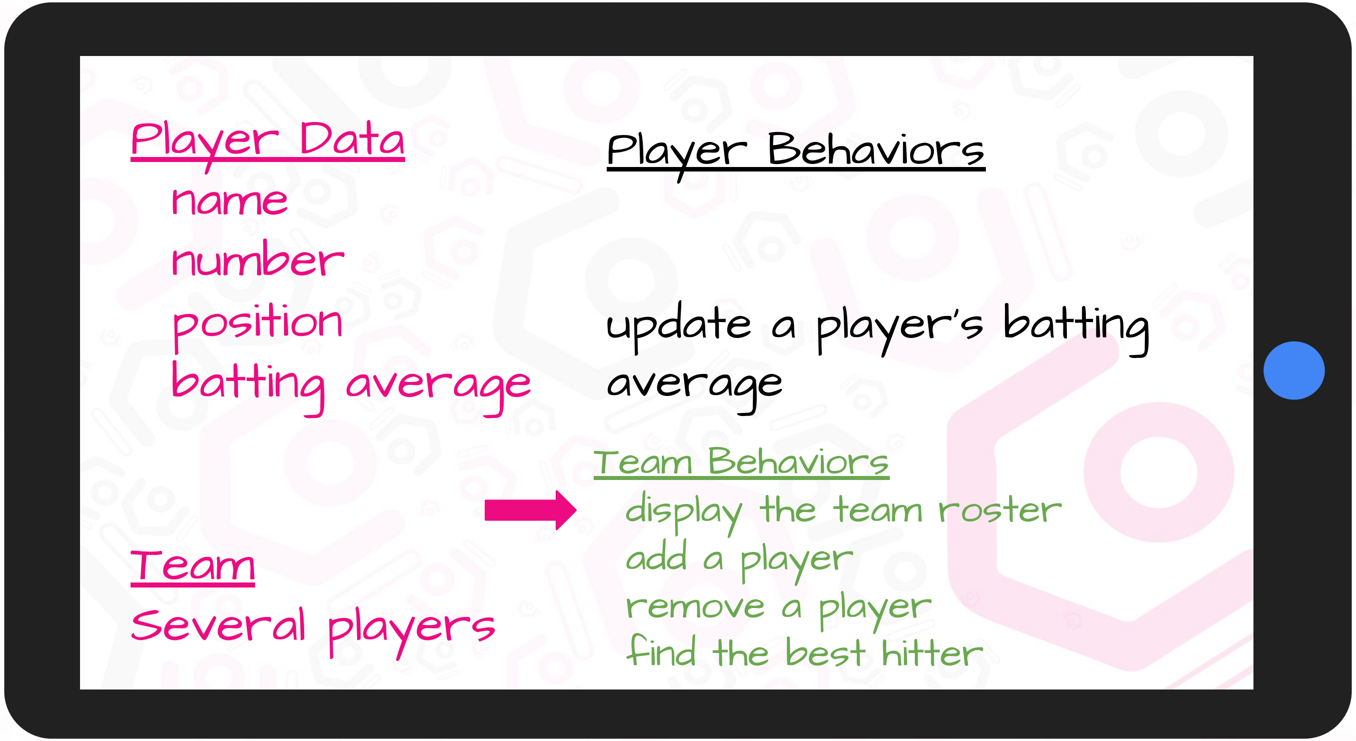
display the team roster

add a player

remove a player

find the best hitter

Because we have actions meant to be taken on a collection of players, let's have a way to store many individual players in one team.



Player Data
name
number
position
batting average

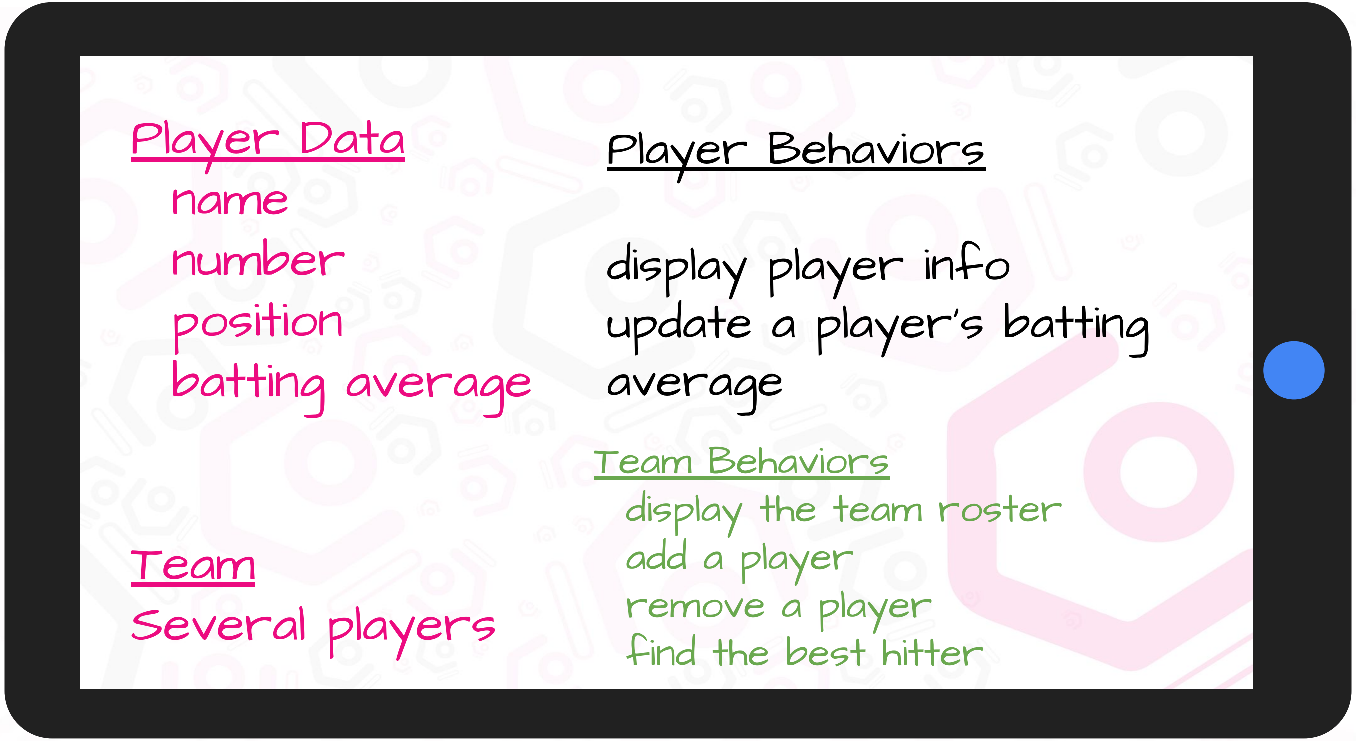
Player Behaviors

update a player's batting average

Team
Several players

Team Behaviors
display the team roster
add a player
remove a player
find the best hitter

This request involves displaying information about each individual player on the team. What behavior can we add to the player that would help us to more easily implement this feature?



Player Data

name

number

position

batting average

Team

Several players

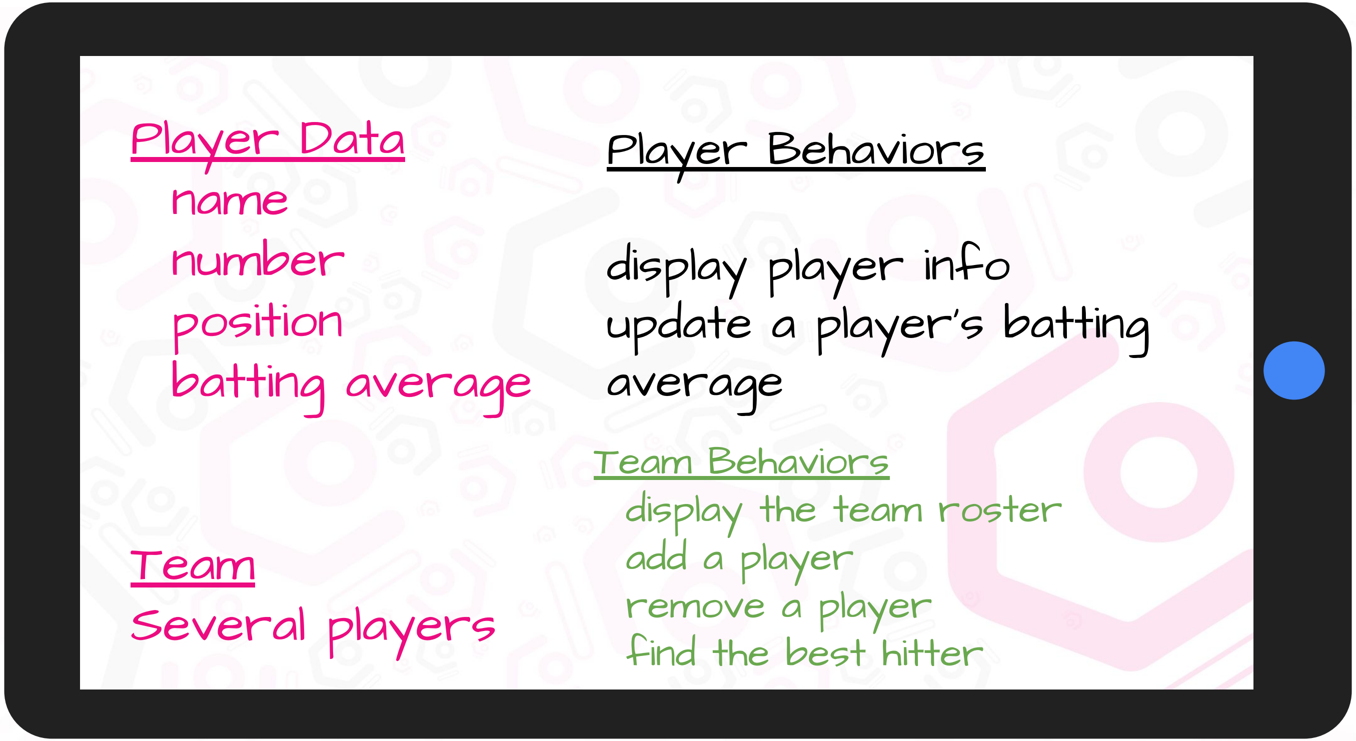
Player Behaviors

display player info
update a player's batting
average

Team Behaviors

display the team roster
add a player
remove a player
find the best hitter

We could make a behavior that displays info about an individual player.



Player Data

name

number

position

batting average

Team

Several players

Player Behaviors

display player info

update a player's batting average

Team Behaviors

display the team roster

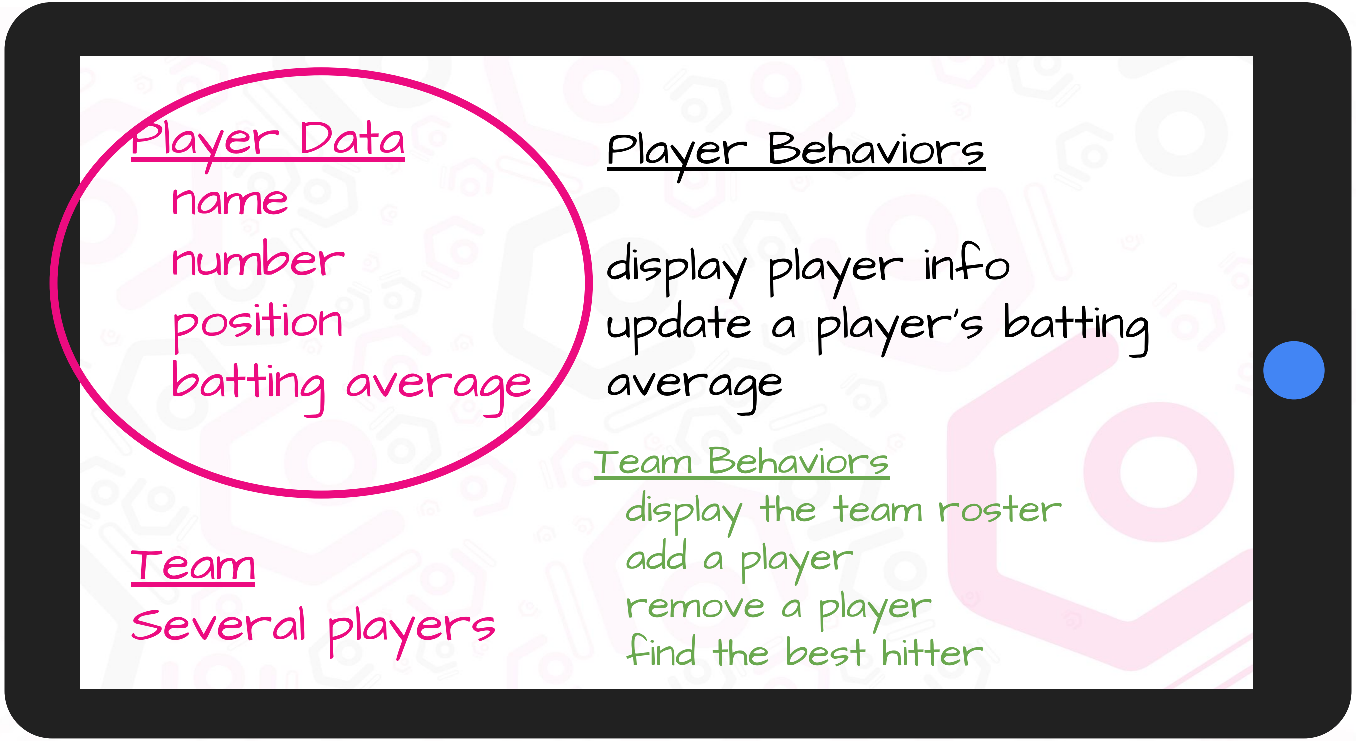
add a player

remove a player

find the best hitter

These notes cover the requests from our client. Let's think about how these concepts can translate into code.

Data Modeling



Player Data
- name
- number
- position
- batting average

Team

Several players

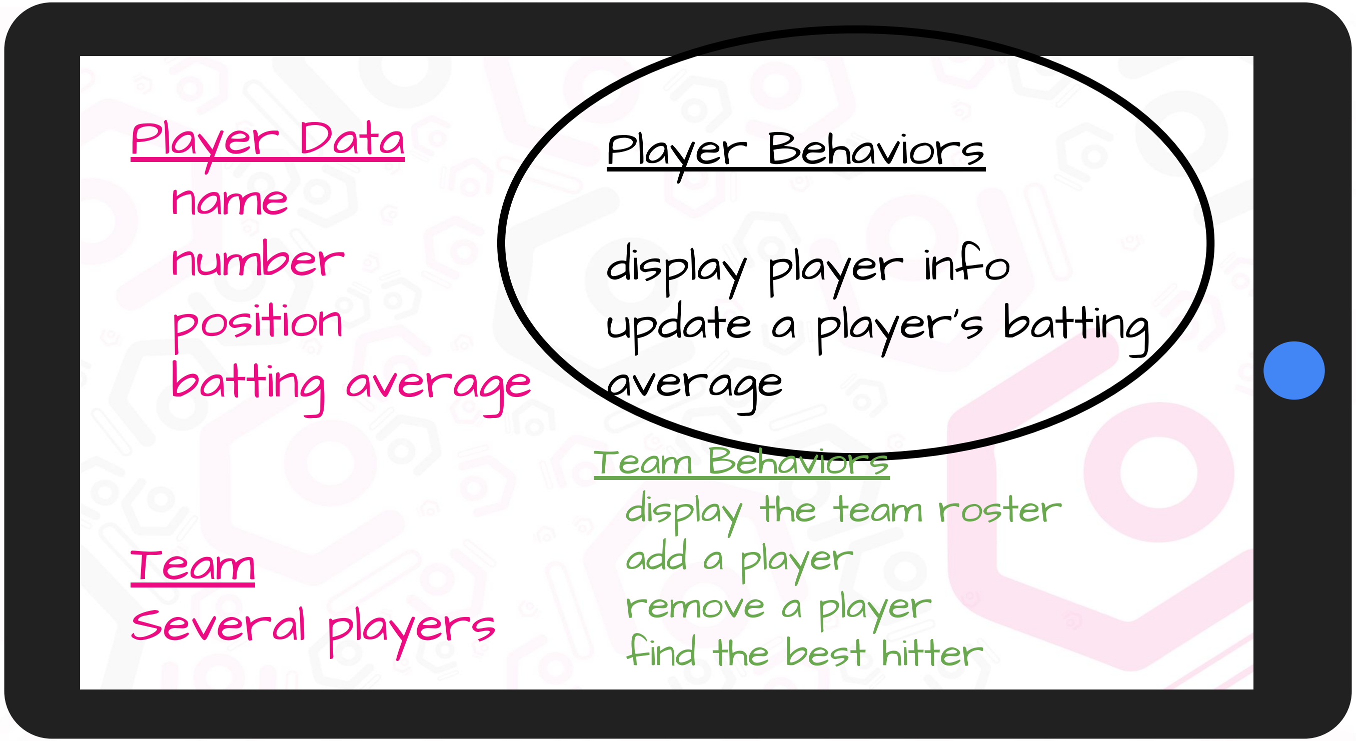
Player Behaviors

display player info
update a player's batting average

Team Behaviors
- display the team roster
- add a player
- remove a player
- find the best hitter

We might imagine these characteristics as properties of a `player` object.

Data Modeling



Player Data

name

number

position

batting average

Player Behaviors

display player info

update a player's batting average

Team

Several players

Team Behaviors

display the team roster

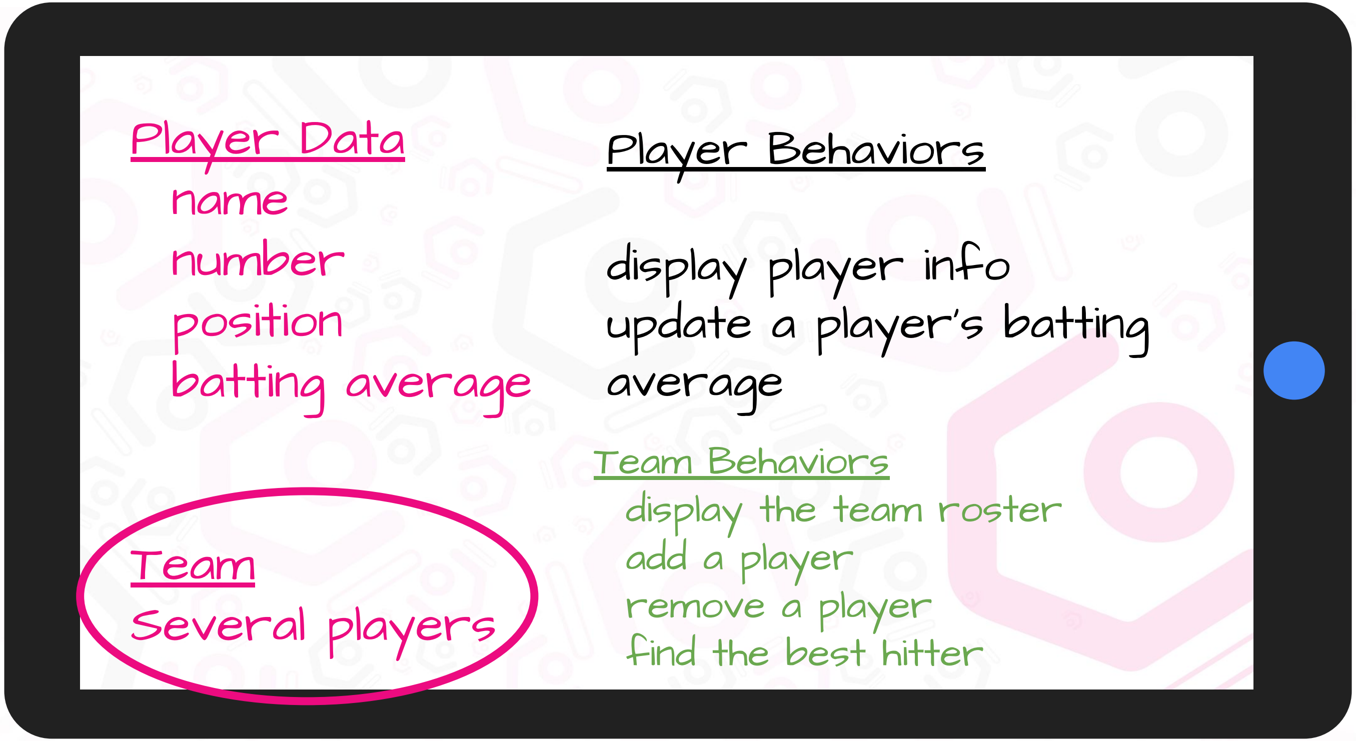
add a player

remove a player

find the best hitter

Displaying info for a player and updating batting average could be functions which operate on a player object.

Data Modeling



Player Data
- name
- number
- position
- batting average

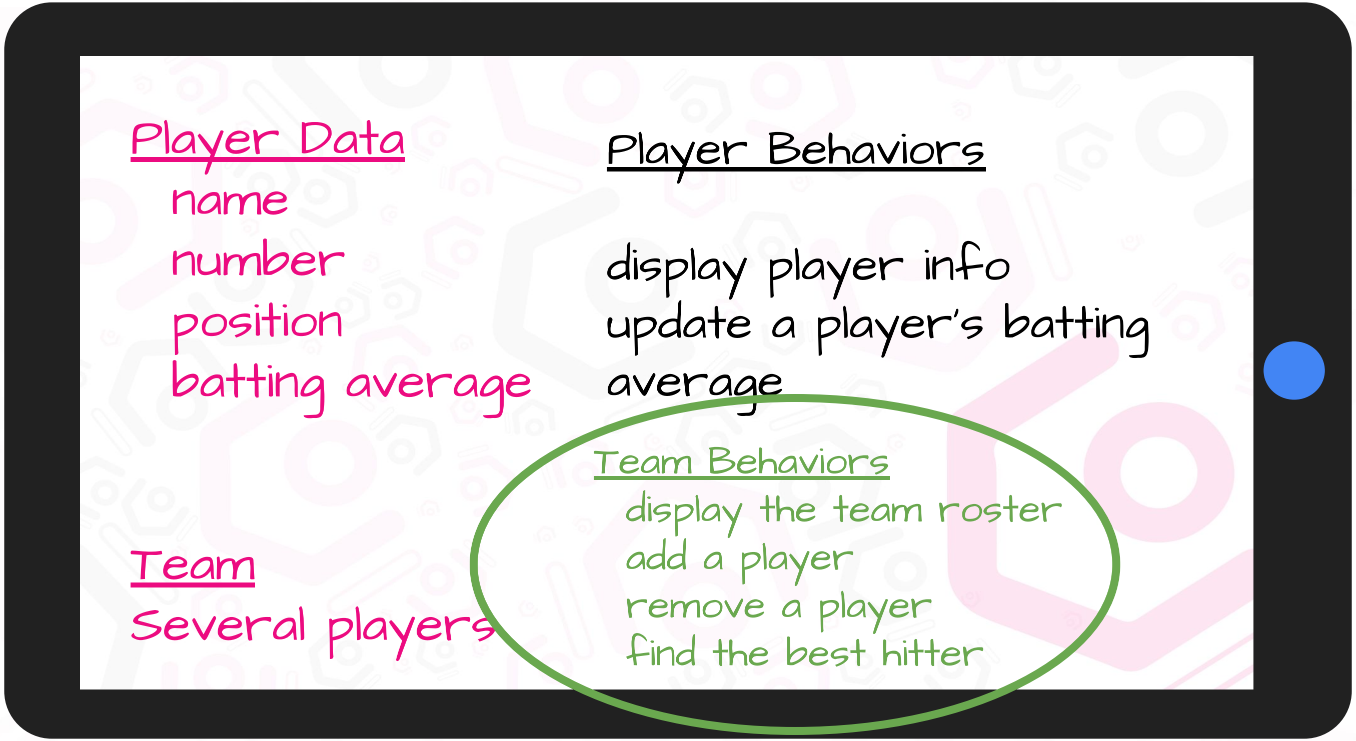
Player Behaviors

display player info
update a player's batting average

Team Behaviors
- display the team roster
- add a player
- remove a player
- find the best hitter

Team
Several players

To represent our team, we could create an array of `player` objects.



Player Data

name

number

position

batting average

Player Behaviors

display player info

update a player's batting average

Team Behaviors

display the team roster

add a player

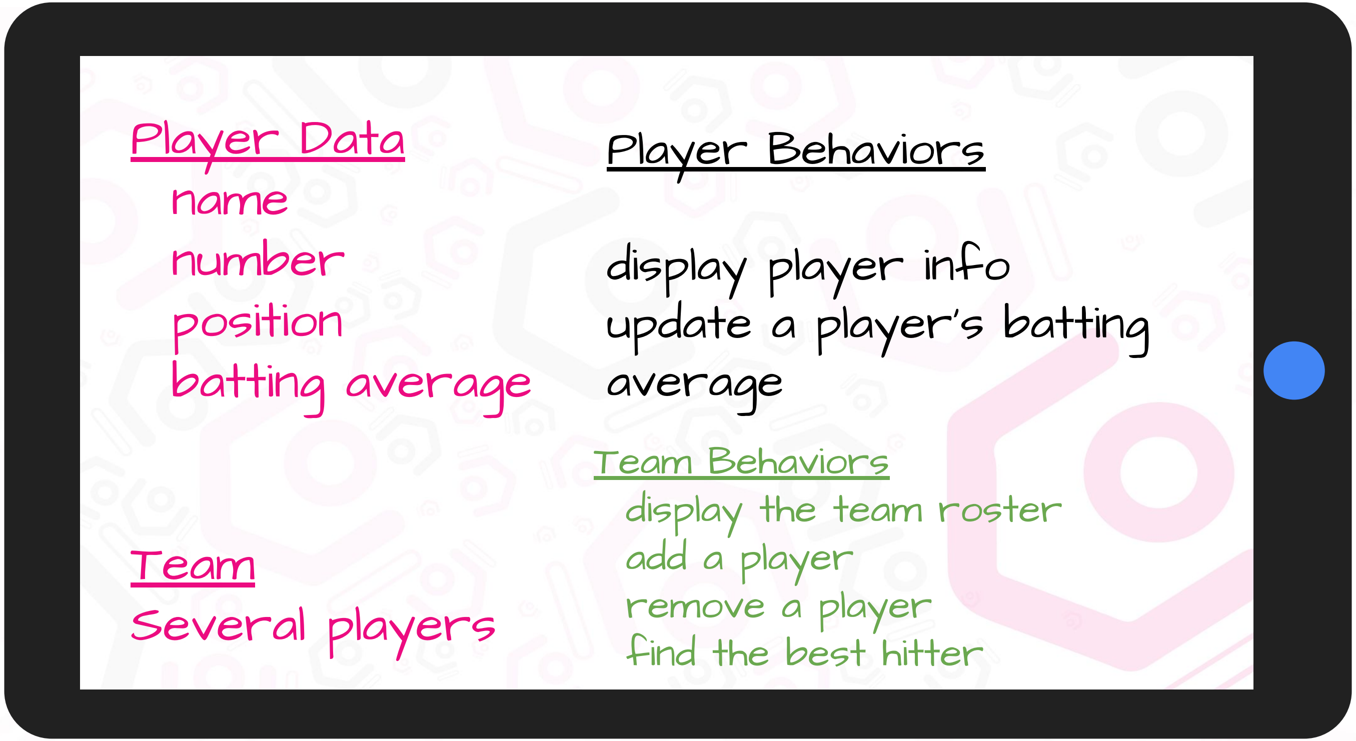
remove a player

find the best hitter

Team

Several players

And these team-related behaviors can become functions which operate on that `team` array.



Player Data
- name
- number
- position
- batting average

Player Behaviors

display player info
update a player's batting average

Team
Several players

Team Behaviors
- display the team roster
- add a player
- remove a player
- find the best hitter

We have a plan in mind! Now let's turn our plan into code!

Data Modeling

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};
```

Let’s start by creating a `player` object.

Data Modeling

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};

var player2 = {
  name: ‘Nick’,
  number: 12,
  position: ‘CF’,
  battingAvg: .282
};

var player3 = {
  name: ‘Taehyung’,
  number: 44,
  position: ‘SS’,
  battingAvg: .318
};
```

Let’s start by Every time we add a player on our team, we’ll need to type out the code necessary to create a new object. This means a lot of repetitive code! **What can we do to save ourselves from typing this repetitive code?**a `player` object.

Data Modeling

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};

var player2 = {
  name: ‘Nick’,
  number: 12,
  position: ‘CF’,
  battingAvg: .282
};

var player3 = {
  name: ‘Taehyung’,
  number: 44,
  position: ‘SS’,
  battingAvg: .318
};
```

```
function makePlayer(                         )
{







}
```

What if we had a function designed to create `player` objects? Let’s put our repetitive object code aside and build out that function.

Data Modeling

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};

var player2 = {
  name: ‘Nick’,
  number: 12,
  position: ‘CF’,
  battingAvg: .282
};

var player3 = {
  name: ‘Taehyung’,
  number: 44,
  position: ‘SS’,
  battingAvg: .318
};
```

```
function makePlayer(name, number, position, battingAvg)
{







}
```

That function could take in all the values related to a particular player…

CodingLab Powered by RBK

Data Modeling

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};

var player2 = {
  name: ‘Nick’,
  number: 12,
  position: ‘CF’,
  battingAvg: .282
};

var player3 = {
  name: ‘Taehyung’,
  number: 44,
  position: ‘SS’,
  battingAvg: .318
};
```

```
function makePlayer(name, number, position, battingAvg)
{
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}
```

...and return an an object with our values mapped to the appropriate keys.

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};

var player2 = {
  name: ‘Nick’,
  number: 12,
  position: ‘CF’,
  battingAvg: .282
};

var player3 = {
  name: ‘Taehyung’,
  number: 44,
  position: ‘SS’,
  battingAvg: .318
};
```

```
function makePlayer(name, number, position, battingAvg)
{
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}
```

The function we wrote for this pattern is called a **factory function**. It is designed to make instances of an object that have the same properties, but different values.

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};

var player2 = {
  name: ‘Nick’,
  number: 12,
  position: ‘CF’,
  battingAvg: .282
};

var player3 = {
  name: ‘Taehyung’,
  number: 44,
  position: ‘SS’,
  battingAvg: .318
};
```

```
function makePlayer(name, number, position, battingAvg)
{
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}

var player1 = makePlayer(‘Karen’, 10, ‘1B’, .204);
```

Now when we want to create a new player, we can use our factory function instead of having to type out a new object every single time.

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};

var player2 = {
  name: ‘Nick’,
  number: 12,
  position: ‘CF’,
  battingAvg: .282
};

var player3 = {
  name: ‘Taehyung’,
  number: 44,
  position: ‘SS’,
  battingAvg: .318
};
```

```
function makePlayer(name, number, position, battingAvg)
{
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}

var player1 = makePlayer(‘Karen’, 10, ‘1B’, .204);
var player2 = makePlayer(‘Nick’, 12, ‘CF’, .282);
```

Now when we want to create a new player, we can use our factory function instead of having to type out a new object every single time.

CodingLab Powered by RBK

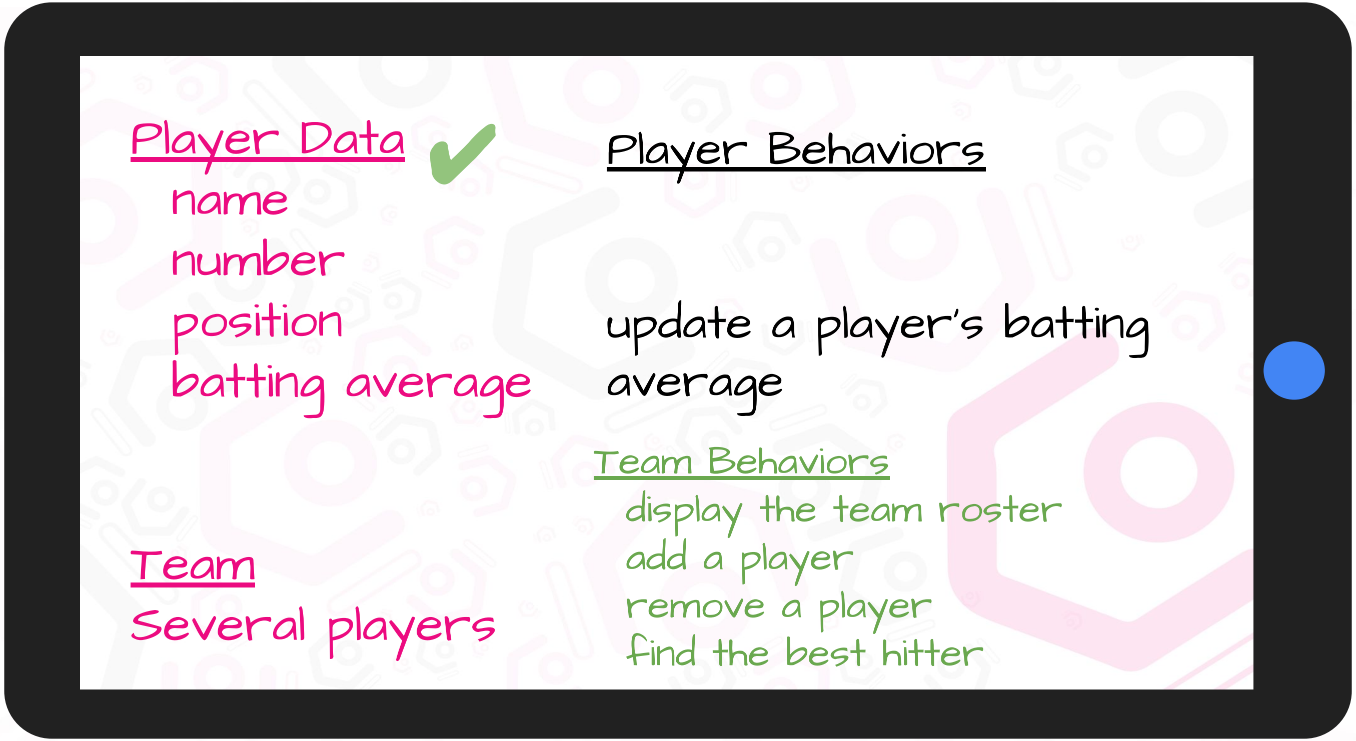
Data Modeling

```
var player1 = {
  name: ‘Karen’,
  number: 10,
  position: ‘1B’,
  battingAvg: .204
};

var player2 = {
  name: ‘Nick’,
  number: 12,
  position: ‘CF’,
  battingAvg: .282
};

var player3 = {
  name: ‘Taehyung’,
  number: 44,
  position: ‘SS’,
  battingAvg: .318
};
```

```
function makePlayer(name, number, position, battingAvg)
{
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}

var player1 = makePlayer(‘Karen’, 10, ‘1B’, .204);
var player2 = makePlayer(‘Nick’, 12, ‘CF’, .282);
var player3 = makePlayer(‘Taehyung’, 44, ‘SS’, .318);
```

Now when we want to create a new player, we can use our factory function instead of having to type out a new object every single time.

Data Modeling



Player Data ✓

name

number

position

batting average

Team

Several players

Player Behaviors

update a player's batting average

Team Behaviors

display the team roster

add a player

remove a player

find the best hitter

We've accounted for our player data. We've also made a new behavior: the factory function which creates our `player` objects. Let's write the other two functions that operate on `player` objects.

Data Modeling

```
function makePlayer(name, number, position, battingAvg) {
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}

function displayPlayer(player) {

}
```

To display our player information, we'll just write a function that accepts a player as an argument...

Data Modeling

```
function makePlayer(name, number, position, battingAvg) {
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}

function displayPlayer(player) {
  return player.number + ' ' + player.position + ' ' + player.name;
}
```

...and returns a string that references properties on that player.

Data Modeling

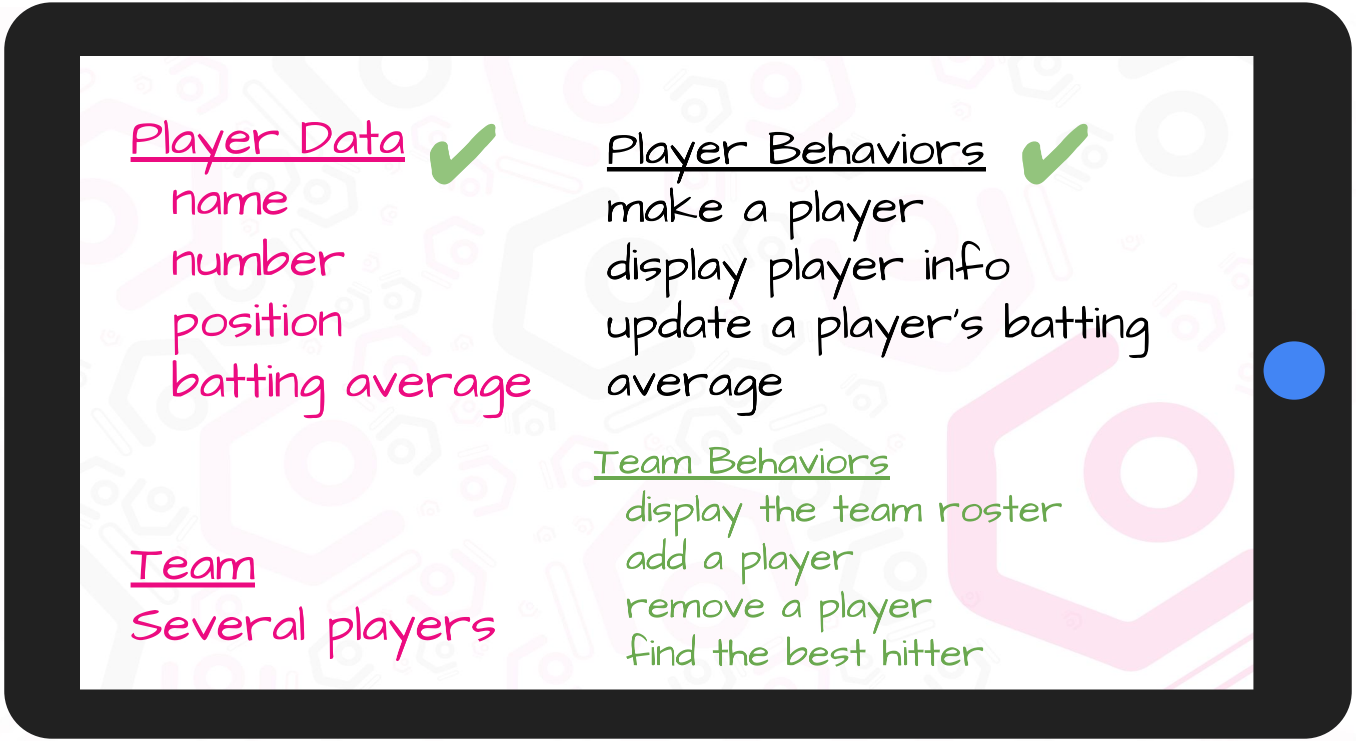
```
function makePlayer(name, number, position, battingAvg) {
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}

function displayPlayer(player) {
  return player.number + ' ' + player.position + ' ' + player.name;
}

function updateBattingAvg(player, newAvg) {

}
```

Because our user asked for a way to type in a new batting average, let’s provide a function to handle that.

```
function makePlayer(name, number, position, battingAvg) {
  return {
    name: name,
    number: number,
    position: position,
    battingAvg: battingAvg
  };
}

function displayPlayer(player) {
  return player.number + ' ' + player.position + ' ' + player.name;
}

function updateBattingAvg(player, newAvg) {
  player.battingAvg = newAvg;
}
```

We’ll reassign the batting average property of a given player object with a supplied new value.

Data Modeling



Player Data ✓

name

number

position

batting average

Team

Several players

Player Behaviors ✓

make a player

display player info

update a player's batting average

Team Behaviors

display the team roster

add a player

remove a player

find the best hitter

This accounts for all our expected functions which operate on a player object. In Part II, we'll turn our attention to the work that needs to be done on teams, which are collections of player objects.

CodingLab
Powered by RBK

That’s it

Data Modeling