

# **SPORTCRED - Design Document**

**Team TODO**

**Sprint 4**

Note: Changes marked with this symbol:



## Table of Contents

System Boundary Diagram	1
MongoDB Documents / Mongoose Models	2 - 4
REST API UML Diagrams	5 - 9
ReactJS DOM Diagram	10

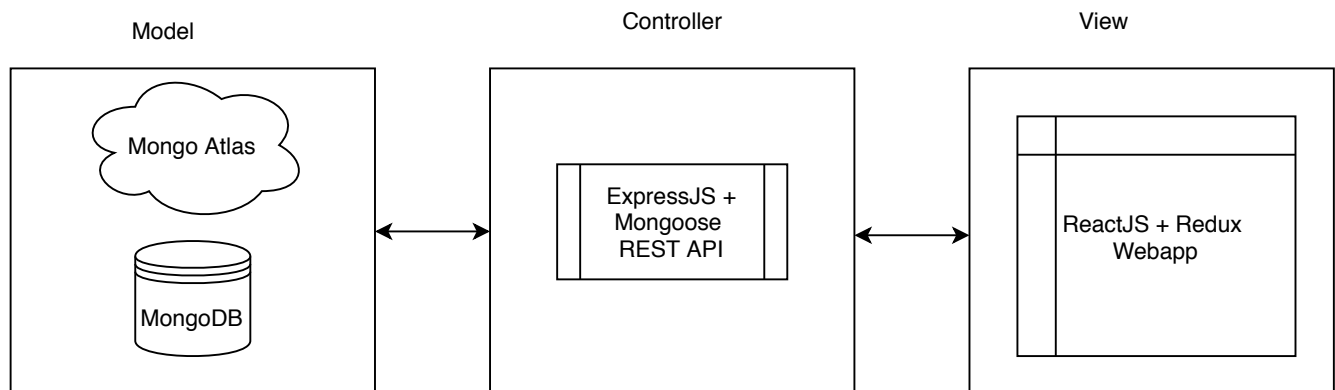
## System Boundary Diagram

Technically, redux is not an exact MVC architecture, but is similar to one. We have our webapp's state stored inside a "store" created and managed by redux. Each interaction to our components on the front end emits an action that will either (or do both) 1) update state in the frontend, or 2) update state in the front end, trigger a side effect (e.g. API call) and then once again update state in the front end when the side effect is resolved.

We have our react app subscribed to this state store and updating accordingly. Redux can be kind of thought as a controller, but the interaction with our Model (mongoDB) is truly done through API calls to our REST endpoint hosted using ExpressJS and Mongoose (mongoDB driver).

The reason this is different is because the traditional MVC pattern has the model directly affecting and triggering updates of the View, but here we have Redux triggering updates and Redux will only trigger updates when the controller returns with information. The View is still dependent on the model, but it's just not directly subscribed to the model

Related resource: <https://www.clariontech.com/blog/mvc-vs-flux-vs-redux-the-real-differences>



# MongoDB Documents / Monoose Models

CHANGED

**Class Name: User**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Represent all information tied to a user

Knows username  
Knows password  
Knows "bio" information  
Knows demographic information  
Knows ACS score  
Knows active trivia games  
Knows status for daily debate question  
Knows current picks and predictions for the user  
Knows profile picture

**Collaborators:**

Post  
DebateAnswer  
TriviaGame  
PAndP  
Acs

CHANGED

**Class Name: Post**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Represent all information of a post on open court.

Knows text content of post  
Knows picture content of post  
Knows owner of post  
Knows date and time posted  
Knows comments on post

**Collaborators:**

User  
Comment  
User-profile

CHANGED

**Class Name: Comment**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Represent all information of comment

Knows text content of comment  
Knows owner of comment  
Knows date and time of comment

**Collaborators:**

Post  
User-profile

**Class Name: DebateQuestion**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Represent all information tied to a daily debate question

Knows text content of debate question  
Knows targeted tier or debate question

**Collaborators:**

Debate  
Response

**Class Name: Response**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Represents a user's response to a debate topic

Knows who made the response and what the response was  
Knows how many times this response was distributed to be evaluated  
Knows when the response was made  
Knows what the ratings the response received from other users

**Collaborators:**

User  
Debate

**Class Name: TriviaQuestion**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Holds questions related to a trivia game

Knows a trivia question  
Knows correct answers to question bank in trivia game  
Knows all possible Responses to question

**Collaborators:**

TriviaAnswer

**Class Name: TriviaAnswer**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Holds one potential response to a trivia question

Knows value of the response  
Knows whether the response is correct or incorrect

**Collaborators:**

None

**Class Name: user-picks**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Represent the picks that the user has made about future awards

Knows the user who has made this pick  
Knows the year for which this pick is relevant  
Knows the picks that the user has created for this year  
Knows the results of the picks that the user has chosen

**Collaborators:**

User

**Class Name: game-picks**

**Parent Classes: None**

**Subclasses: None**

**Responsibilities:**

Represent game pick for the daily picks feature.

Knows the user that created the pick  
Knows map containing the game and pick that the user has chosen  
Knows whether or not this pick has been evaluated or not

**Collaborators:**

User  
Game

<b>Class Name: games</b> <b>Parent Classes: None</b> <b>Subclasses: None</b>
<b>Responsibilities:</b> Represent a game that has happened or will happen.  Knows the date of the game Knows the image of the away and home teams Knows the names of the home and away teams Knows the winner of the game
<b>Collaborators:</b> game-picks

<b>Class Name: bracket</b> <b>Parent Classes: None</b> <b>Subclasses: None</b>
<b>Responsibilities:</b> Represent playoff bracket choices of the user.  Knows the user that created the bracket Knows map containing the bracket choice and the scores user has chosen Knows whether or not this pick has been evaluated or not
<b>Collaborators:</b> User

<b>Class Name: Debate</b> <b>Parent Classes: None</b> <b>Subclasses: None</b>
<b>Responsibilities:</b> Represents an active debate between a group of users.  Knows the users involved in the debate Knows what the debate question is Knows which tier the debate is apart of Knows when the debate happened Knows which responses were made for the debate
<b>Collaborators:</b> User Response

<b>Class Name: HHTriviaGames</b> <b>Parent Classes: None</b> <b>Subclasses: None</b>
<b>Responsibilities:</b> Represents a head to head trivia game  Knows the players of the game Knows the state of the game (finished, still going on, player 1 won, draw, etc) Knows the trivia questions to give players
<b>Collaborators:</b> Trivia Question

CHANGED

<b>Class Name: Acs</b> <b>Parent Classes: None</b> <b>Subclasses: None</b>
<b>Responsibilities:</b> Represent all information tied to a user  Knows games acs score Knows history acs score Knows predictions acs score Knows analysis acs score
<b>Collaborators:</b> None

CHANGED

<b>Class Name: user-profile</b> <b>Parent Classes: None</b> <b>Subclasses: None</b>
<b>Responsibilities:</b> Represent a user by displaying their username, acs, bio, and profile pic.  Knows the user that is going to be displayed
<b>Collaborators:</b> User

CHANGED

## REST API UML Diagrams

GET /users
+ List of User objects

GET /users/:ids
+ List of User objects

POST /users
+ username: string + password: string + age: integer + gender: string + acs: integer

DELETE /users/:id
+ User object

PUT /users/:id
+ username: string (opt) + password: string (opt) + age: integer (opt) + gender: string (opt) + acs: object (opt) + triviaGames: [string] (opt) + pAndP: [string] (opt)
+ User object

CHANGED

PUT /users/:id/profilepic
+ picture: string

GET /ocposts
+ List of OpenCourtPost objects

GET /ocposts/:id
+ OpenCourtPost object

POST /ocposts
+ content: string + origPoster: string + picture: string

DELETE /ocposts/:id
+ OpenCourtPost object

PUT /ocposts/:id
+ body: string (opt) + owner: string (opt)
+ OpenCourtPost object

DELETE /ocposts/:id/comments/:cid
+ Comment object

GET /ocposts/:id/comments
+ [Comment object]

GET /ocposts/:id/comments/:cid
+ Comment object

POST /ocposts/:id/comments
+ content: string + origPoster: string

GET /debate-questions
+ [DebateQuestion object]

GET /debate-questions/get-by-tier/:tier
+ tier: String
+ [DebateQuestion object]

GET /debate
+ [Debate object]

GET /debate/:id
+ id: String
+ Debate object

GET /debate/get-by-date/:date
+ date: Date
+ [Debate object]

GET /debate/get-by-userid/:userid
+ userid: String
+ [Debate object]

GET /debate/get-by-tier/:tier
+ tier: String
+ [Debate object]

POST /debate
+ tier: String + debaterIds: List of ObjectIds + responselds: List of ObjectIds + question: String + date: Date + isEvaluated: Boolean
+ newDebate: Debate object

PUT /debate/:id
+ id: String + tier: String + debaterIds: List of ObjectIds + responselds: List of ObjectIds + question: String + date: Date + isEvaluated: Boolean
+ newDebate: Debate object

GET /trivia/questions
List of TriviaQuestion objects

GET /trivia/:id
+ body: string
+ user:number

POST /trivia
+ question: string +responses: List of trivia-answer object
+ newQuestion: trivia-question object

PUT /user-picks/:uid
+ year: Number + user: uid +picks: user-pick Object +results: Object +isEvaluated: Boolean
+ user-pick object

POST /user-picks/
+ year: Number + user: uid +picks: user-pick Object +results: Object +isEvaluated: Boolean
+ user-pick object

GET /games/
+ Array of games object

GET /user-picks/:id
+ id: String
+ user-pick object

GET /team/
+ Array of team object

GET /game-picks/:id
+ id: String
+ game-pick object

GET /team/:uid
+ team object

PUT /game-picks/:uid
+ user: uid + picks: user-pick Object
+ game-pick object

GET /winners/:year
+ winner object

POST /game-picks/
+ user: uid + picks: user-pick Object
+ game-pick object

GET /bracket/:year
+ bracket object

POST /bracket/bracketChoice/
+ teamOne: String + teamTwo: String + winnerID: String + resultForWinner: String + userID: String + isFirstMatch: Boolean + winnerScore: Number + loserScore: Number
+ bracketChoice object

PUT /user-picks/:uid
+ year: Number + user: uid + picks: user-pick Object + results: Object + isEvaluated: Boolean
+ user-pick object

GET /bracket/bracketChoice/userID/:id
+ bracketChoice object

PUT /bracket/bracketChoice/:id
+ teamOne: String + teamTwo: String + winnerID: String + resultForWinner: String + userID: String + isFirstMatch: Boolean + winnerScore: Number + loserScore: Number
+ bracketChoice object

GET /debate-responses
+ [Response object]

GET /debate-responses/get-from-list-of-ids
+ responseids: [String]
+ [Response object]



GET /debate-responses/get-assigned-responses
+ id: String
+ [String]

GET /debate-responses/:id
+ id: String
+ [Response object]

POST /debate-responses/
+ user: String + content: String + count: Number +ratings: Map +date: Date
+ Response object

PUT/debate-responses/update-count/:id
+ id: String + count: Number
+ Response object

PUT /debate-responses/put-rating/:id
+ raterId: String + rating: Number + id: id
+ Response object

CHANGED

PUT /hhtrivia/:id/start/:player
+ id: string + player: integer
+ HHTriviaGame object

CHANGED

PUT /hhtrivia/:id/acsChange/:player
+ id: string + player: integer + acsChange: integer
+ HHTriviaGame object

CHANGED

CHANGED

PUT /hhtrivia/:id/evaluate
+ id: string
+ HHTriviaGame object

PUT /hhtrivia/:id/increment-correct/:player
+ id: string + player: integer
+ HHTriviaGame object

CHANGED

POST /hhtrivia/create-game
+ userId: string
+ HHTriviaGame object

CHANGED

PUT /hhtrivia/join-game
+ userId: string
+ HHTriviaGame object

CHANGED

DELETE /hhtrivia/:id

+ id: string

+ HHTriviaGame object

CHANGED

GET /hhtrivia/user-games/:id

+ id: string

+ HHTriviaGame object

CHANGED

GET /hhtrivia/:id

+ id: string

+ HHTriviaGame object

CHANGED

GET /acs

+ id: string

+ Acs object

CHANGED

Put /acs

+ id: string  
+ type: string  
+ updatedAcs: integer

+ Acs object

## **ReactJS DOM Diagram (changed) :**

**Link for draw.io file download:**

**<https://drive.google.com/file/d/1qf18acrGXX0H2jrRnTPxAIV4UYIxtJQd/view?usp=sharing>**

**Png was too large!**