

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 - 3
REST API UML Diagrams	4 - 8
ReactJS DOM Diagram	9

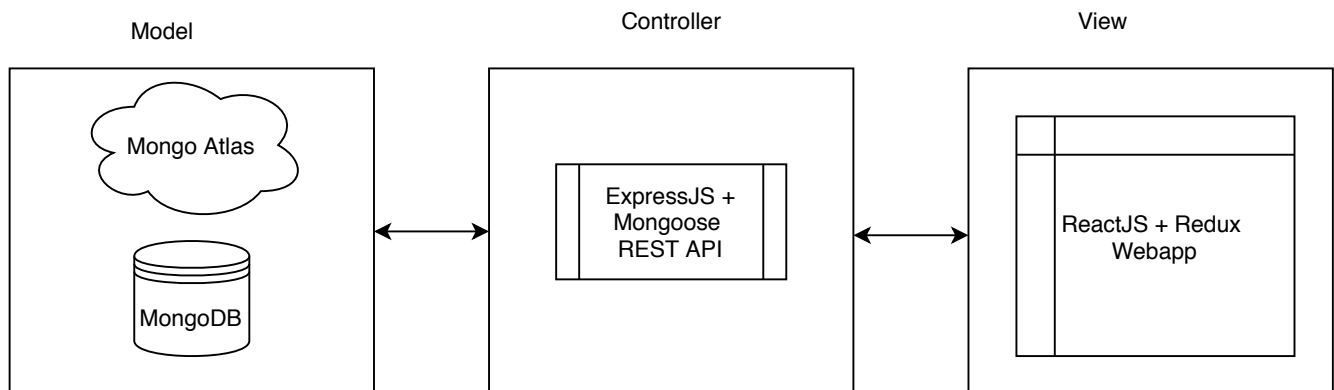
System Boundary Diagram

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 / Monooose 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

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: 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

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

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: 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

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: 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: 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

Class Name: games Parent Classes: None Subclasses: None
Responsibilities: 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
Collaborators: game-picks

Class Name: HHTriviaGame Parent Classes: None Subclasses: None
Responsibilities: 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
Collaborators: Trivia Question

Class Name: Team Parent Classes: None Subclasses: None
Responsibilities: Represents teams from the NBA Knows the team name, and players. Also has the team logo
Collaborators: user-picks

Class Name: Winner Parent Classes: None Subclasses: None
Responsibilities: Represents the winners of end-of-season awards in a given year Knows the year and the names of the winners of the awards
Collaborators: user-picks

Class Name: bracket Parent Classes: None Subclasses: None
Responsibilities: Represent playoff bracket matchups Knows the score and winner of each match
Collaborators: User

Class Name: bracketChoices Parent Classes: None Subclasses: None
Responsibilities: 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
Collaborators: User Bracket

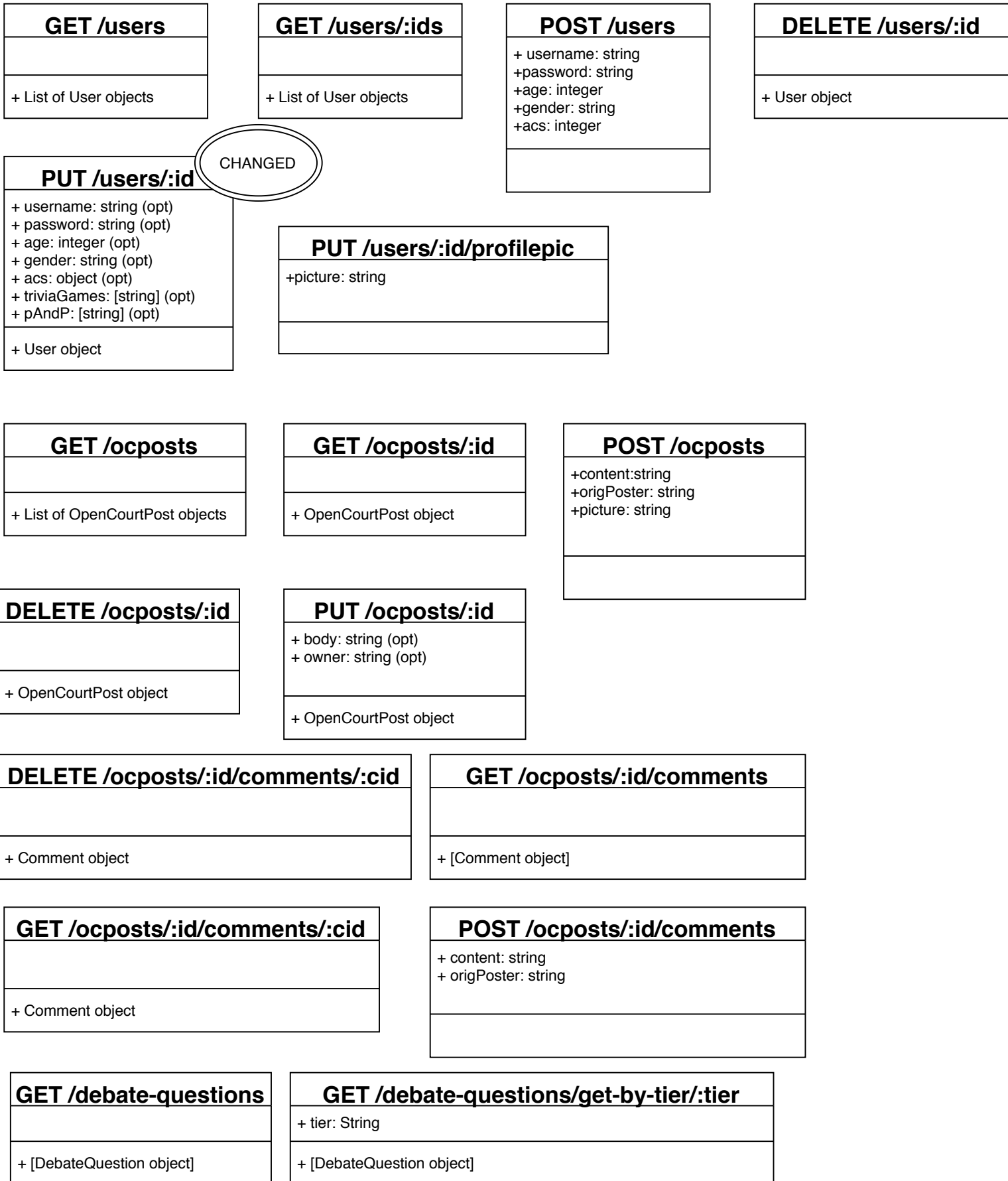
Class Name: Acs Parent Classes: None Subclasses: None
Responsibilities: Represent all information tied to a user Knows games acs score Knows history acs score Knows predictions acs score Knows analysis acs score
Collaborators: None

Class Name: Debate Parent Classes: None Subclasses: None
Responsibilities: 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
Collaborators: User Response

Class Name: Player Parent Classes: None Subclasses: None
Responsibilities: Represents players from the NBA. Knows the player name, team and whether they are a rookie
Collaborators: user-picks

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

REST API UML Diagrams



GET /debate
+ [Debate object]

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

GET /debate/:id
+ id: String
+ 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

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

CHANGED

CHANGED

CHANGED

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

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

CHANGED

CHANGED

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

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!