SPORTCRED - Design Document

Team TODO

Table of Contents

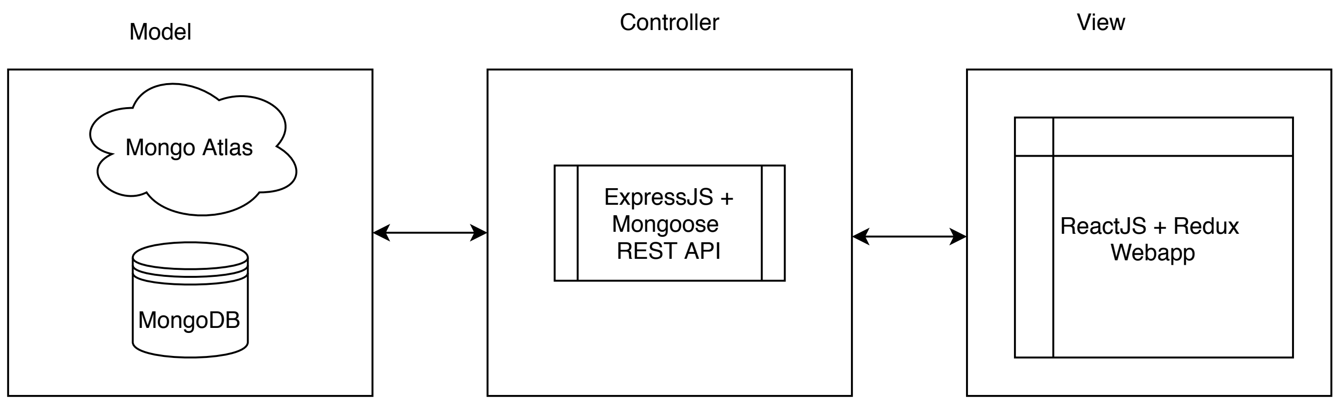
System Boundary Diagram

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 / Mongoose Models

Class Name: User
Parent Classes: None
Subclasses: None

Responsibilities:
Represent all information tied to a user

Knows username
Knows password
Knows "about me" 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

Collaborators:
OpenCourtPost
DebateAnswer
TriviaGame
PAndP

Class Name: OpenCourtPost
Parent Classes: None
Subclasses: None

Responsibilities:
Represent all information of a post on open court.

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

Collaborators:
User
Comment

Class Name: Comment
Parent Classes: None
Subclasses: None

Responsibilities:
Represent all information of comment

Knows original OpenCourtPost
Knows text content of comment
Knows owner of comment
Knows date and time of comment
Knows replies to comment

Collaborators:
OpenCourtPost

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 debate answers to debate question
Knows date and time of debate question creation
Knows targeted tier or debate question

Collaborators:
DebateAnswer

Class Name: DebateAnswer
Parent Classes: None
Subclasses: None

Responsibilities:
Represent all information tied to an answer to a daily debate question

Knows original debate question
Knows text content of answer to debate question

Collaborators:
DebateQuestion
User

Class Name: TriviaGame
Parent Classes: None
Subclasses: None

Responsibilities:
Holds questions related to a trivia game

Knows question bank of trivia game
Knows correct answers to question bank in trivia game
Knows player list for trivia game
Knows answers each player has given for trivia game

Collaborators:
User

Class Name: PAndP
Parent Classes: None
Subclasses: None

Responsibilities:
Represent picks and prediction a user has made

Knows current season picks and predictions
Knows playoff bracket picks and predictions

Collaborators:
User

REST API UML Diagrams

GET /users

+ List of User objects

GET /users/:id

+ User object

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: integer (opt)
+ triviaGames: [string] (opt)
+ pAndP: [string] (opt)

+ User object

GET /ocposts

+ List of OpenCourtPost objects

GET /ocposts/:id

+ OpenCourtPost object

POST /ocposts

+ body: string
+ owner: 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

+ body: string
+ owner: string

POST /ocposts/:id/comments/:cid/replies

+ body: string
+ owner: string

GET /ocposts/:id/comments/:cid/replies

+ [Comment object]

GET /ocposts/:id/comments/:cid/replies/:rid

+ Comment object

GET /debatequestions

+ [DebateQuestion object]

GET /debatequestions/:tier

+ [DebateQuestion object]

POST /debatequestions/:tier

+ body: string

DELETE /debatequestions/:tier/:id

+ DebateQuestion object

PUT /debatequestions/:tier/:id

+ body: string (opt)

+ DebateQuestion object

GET /debatequestions/:tier/:id

+ DebateQuestion object

POST /debatequestions/:tier/:id/replies

+ body: string
+ owner: string

GET /debatequestions/:tier/:id/replies

+ [DebateAnswer object]

GET /debatequestions/:tier/:id/replies/:id

+ DebateAnswer object

GET /triviagames

+ [TriviaGame object]

GET /triviagames/:id

+ TriviaGame object

POST /triviagames

+ players: [string]
+ questions: dict

POST /triviagames/:id/questions/:qid/answers/:uid

+ choice: string

GET /triviagames/:id/questions

+ [string]

GET /triviagames/:id/questions/:uid

+ string

GET /triviagames/:id/questions/:qid/answers/:uid

+ string

GET /pandps/:uid

+ pAndP object

POST /pandps/:uid

+ picks: pAndP object

DELETE /pandps/:uid

+ pAndPObject

PUT /pandps/:uid

+ picks: pAndP object (opt)

+ pAndP object

ReactJS DOM Diagram:

Link for draw.io file download:
https://drive.google.com/file/d/1UCoVQJUcR9--3jiEZcmtqHVc56nx7LXQ/view?usp=sharing

Or please see the DOMDesign.png file in the repo as the diagram was too large to fit in this pdf