**Client**

* Client code goes in client/src

**React**

* Create a function component that list out a collection of your models (ListOfUsers)
  + Use prop types to define what props the component needs
  + It should probably need a prop named after the plural of your model, and it should probably be an array
  + Probably want to map over this array and create some divs or li’s showing 3 of the properties of the item
  + Create a Link on each item so you can navigate to the detail page
  + Create a button on each item so you can delete the item
* Create a class component that makes a new model (redux fetch practice)
  + Create a form and a button
  + Put a label and an input for each property your model has
  + Bonus - use a drop down if there are a limited number of values for a property
  + Register onChange for each input and use setState to store the information typed in
  + Register onSubmit for the form.
  + onSubmit call an action called createThing(change it to make sense for your model)
* Create a function component that shows details of one thing (UserDetail)
  + Use a prop that has the array of your models
  + Use a parameter from the route path /:id to find the model to show
  + Use whatever html you want to show all of the properties of your thing
* App.js
  + componentDidMount, call loadThings

**React Router**

* Import necessary components from react-router-dom into App.js
* Make sure to wrap everything in BrowserRouter
* Create a route to show the list container
* Create a route to show the create container
* Create a route to show the detail container. Make sure this route has a variable in it
* Make sure to wrap all routes in Switch
* Create links to the list and create routes, put them anywhere in App.js outside of the Switch.

**Redux**

\* The only state you need is an array of your models

\* state.js

\* Create actions for loading your models and models loaded

\* loadThings() - do a fetch get to “/things”

\* thingsLoaded(things) - THINGS\_LOADED

\* Create an action for saving a new model

\* createThing(thing) - do a fetch post to “/things”

\* when the fetch is complete, dispatch to loadThings

\* Create an action for deleting an item

\* deleteThing(id) - do a fetch delete to “/things/” + id

\* when the fetch is complete, dispatch to loadThings

\* Create reducer for the state

\* care about the action THINGS\_LOADED

\* Create containers for all of you components

\* mapStateToProps and mapDispatchToProps

\* The list component container should mapStateToProps for the array of things

\* The list component container should mapDispatchToProps for the deleteThing action

\* The new thing component container should mapDispatchToProps for the saveThingAction

\* AppContainer should mapDispatchToProps for loadThings.

**Server - use advanced-express-practice as an example**

* The code for this goes in the /server folder
* Use express to create a server listening on port 3001
* Use mongoose to connect to a MongoDB database called “checkpoint2”
* Create a Model for your thing
* Create a Route and Controller for your thing
* In the Route, create routes for getting all things, getting one thing by id, deleting one thing, updating one thing, and creating one thing
* In the Controller, create functions for list,show,create,update,remove

**Points**

* When I yarn start, the web page loads with no errors - 10pts
* I can navigate to the list and create screen from links - 10pts
* I can enter information into the create screen and click save. If I go back to the list screen, that new information is there. - 20pts
* If I go to the list screen it will show all current existing information - 20pts
* The list screen has a view link for each item. If I click that link the detail screen appears. The details of the item are shown. - 20pts
* If I click the delete button on the list, the item I clicked will be removed from the list - 20pts