Touchable Components Lab

We've been hearing a complaint from the users of our app. They say that they can reserve seats just fine but that they can't envision where the seats are in the theater. Some peole like to sit close to the screen. Others want to back way off. Some like to sit on the sides or in the back where they don't feel so cramped. Still others want to sit right in the center.

So they're asking for a seat map when they reserve their seats. Let's see if we can do that.

1. Open a browser or some other tool where you can make GET requests and make one to http://localhost:5000/api/theaters/:theater_id/tables/:table_id/seats with a valid theater_id and table_id. You'll notice that there is an x and a y location for each table.

This number is what we'll use to place the seats.

2. Create a new component called SeatMap.js. Feel free to copy its contents directly from PickSeats.js since it is going to have the same basic functionality -- the user will be able to add seats to their cart.

We will need to change the JSX obviously.

3. Leave the headers at the top and the checkout button at the bottom. But replace all the stuff between them with a <View> that takes up 100% of the remaining space (hint: flex: 1).

Pinch-to-Zoom

Remember that <ScrollView> will allow us to pinch-zoom if we set it up just right, but it only works on iOS. So let's install a component that will work cross-platform.

4. Do this:

```
npm install react-native-pinch-zoom-view
```

Feel free to read up on this thing to make sure you understand how it works. But it will expose a new React Native component called <PinchZoomView>.

5. Go ahead and place one of those inside the <View> you just added.

Making it absolutely positioned

6. Put a <View> inside your <PinchZoomView>. Make this view have a style of position: 'absolute'.
7. Put some placeholder text and/or images inside of it to make sure you like how it works to start.
8. Move your placeholder text and images into different locations by setting the left and top properties of their container view.
9. Run and test several times, adjusting top and left until you get a feel for how to position things in the <View>.

Adding the tables back in

10. In the JSX of SeatMap, remove your placeholder text/images and do this instead:

```
tables.map(table => <Table />)
```

Note: in one of the prior labs there was a bonus step to create a <Table> component. If you didn't do that before, go ahead and do it now. It can just render a <Text> whose title is the table number.

11. In addition to the props you were passing in before, make sure that the table's x and y position are being passed in as well. (Hint: They may already be being passed in if you just sent the whole table object down somehow).

12. Use these x and y values to position your <Table>s. The x values are from the left and y values are from the top. You may need to do some math to spread them out a across your device. Don't worry about getting them perfect just yet. We'll adjust later.
13. Run and test to make sure the <Table>s appear with some room between them.

Placing the seats

Remember that each table has one to four seats. Let's place those just like you did with the <Table>.

14. As before, if you don't already have a <Seat> component, create that now. It can render a <View> that has the seat number in it.
15. And again, using the x and y positions of the table, do a little math to position the seats next to the table. Don't worry about getting them perfectly placed just yet.
16. Run and test. Once you can see the right amount of seats next to each table, you can move on.

Making them look right

Let's make the seats look like seats.

17. Take a look in the project's starters. You'll find a file called seat.png. Copy this to your project's folder so it'll be compiled into the app.
18. Add it to the <Seat> as an <Image>. Resize it and place it as needed.
19. Now is where you can adjust the tables and the chairs. Change their sizes and locations to fit within your <PinchZoomView> control.

Making it Touchable

We want the user to be able to press a seat to reserve it.

20. Wrap each seat in a TouchableHighlight. Put an onPress event on it to dispatch a Redux action to "SELECT_SEAT" with a payload of the seat number and showing. Also change the current seat's status to "isSelected".
21. Run and test in the debugger. Just to make sure that your touchable is indeed dispatching the right action and has a seat_number and showing object.

As a bonus, you'll also notice that as soon as you tap your seat it changes to an orange color. This happened because in the Styles Lab, we added that conditional style.

22. Next, you'll probably want to add the "SELECT_SEAT" action case to your reducer. It will add this seat to state.cart which is an array of seats.
23. Go ahead and test again to make sure that cart is getting populated.

24. Bonus!! If the seat has a status of "seatIsTaken" when the user taps on it, return from the event handler without doing anything. We can't reserve a seat that has already been sold, right?
25. Extra Bonus!! If the seat has a status of "isSelected", remove the status and dispatch an action to "UNSELECT_SEAT". Make that action remove the seat from the cart.