Single-value Components

We've already accomplished some of the hardest parts of our app. It may be a little surprising that we've waited this long to get to one of the most fundamental features of React Native; components that hold a single value. In this lab we're going to display the data we worked so hard to get into our app. We'll display that data in <Text>s, <Image>s, and <TextInput>s. Are you ready? Let's go!

The Landing component

1. Images in your project must be local, probably in a folder called assets. We've provided some image files for you in the starters folder. Copy them into your project in the assets folder.

Our main view will hold a header and a list of films we're currently showing.

2. At the top of Landing.js, make sure that you have some import statements:

```
import { fetchFilms, getBaseUrl } from '../utils/repository';
import { Image, View, Text } from 'react-native';
```

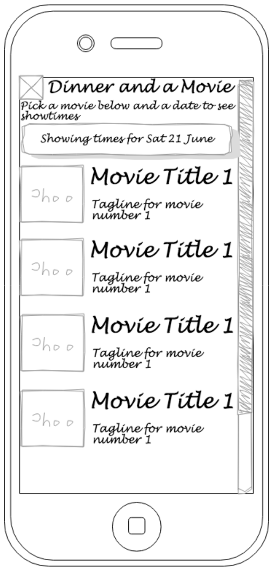
3. Wrap the component with a <View> or React fragment.

4. In that <View>, add an <Image> pointing to the logo of our business:

```
<Image source={require('../assets/daam.png')} style={{ height:
100, width: 100, }} />
```

5. Add a <Text> with the name of our business, "Dinner And A Movie".

6. Add a <Text> to tell the user to "Tap on a film to see its details and pick a date to see showtimes".


Dinner and a Movie
Pick a movie below and a date to see showtimes
Showing times for Sat 21 June
Movie Title 1
Tagline for movie number 1
Movie Title 1
Tagline for movie number 1
Movie Title 1
Tagline for movie number 1
Movie Title 1
Tagline for movie number 1

Note: The things you add will not look like our drawings yet; the contents may not all fit on the screen and the text won't be bolded or centered. Don't worry about that. We'll fix all of those things in a future lab.

7. Run and test. Do you see them so far? Awesome. Let's make it a little more challenging.

We want to show a bunch of films. But how do you show multiple <Image>s and <Text>s? As all React developers know, you'll iterate the films using .map().

8. Go ahead and .map() through the films:

```
{films.map(film => <Text key={film.id}>{film.title}</Text>)}
```

This will show each film title.

9. Add each film's tagline.

10. Add an <Image> above each title to show the poster. Notice that the poster is not bundled from our assets folder but is served live from our data server. This makes sense because we don't want our customers to have to download and install a new copy of our app when we start showing a new movie in one of our theaters, right?

```
<Image source={{ uri: `${host}/${film.poster_path}` }}
style={{ height: 100, width: 100, resizeMode: "contain" }} />
```

11. Run your app. Do you see your component? Great! Let's do some more.

Adding the ShowingTimes

Under each film, we'd like to show the user what times the movie is showing. And we have a component already that can do that; <ShowingTimes />

12. For each film, insert an instance of the <ShowingTimes> compononent. (Hint: Remember to pass the film and the selectedDate as props).

ShowingTimes isn't actually showing anything currently. Let's fix that.

13. Add a <Text>:

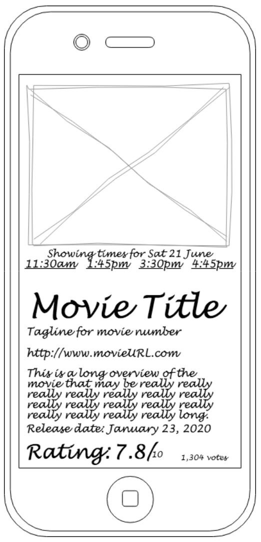
```
<Text>Showing Times for {new Date(selectedDate).toShowingDateString()}</Text>
```

14. Using what you've just learned, have ShowingTimes display each time in a <Text>.
    Hints:
    - Since there are multiple times, you'll use {showings.map(...)} to iterate.
    - To format the displayed time, do this:

```
<Text>{new Date(showing.showing_time).toShowingTimeString()}</Text>
```

15. Run and test.

Hey, you have six different films with different titles, posters, taglines, and showing times. That's great! Let's move on to another component.


Showing times for Sat 21 June
11:30am 1:45pm 3:30pm 4:45pm
Movie Title
Tagline for movie number
http://www.movieURL.com
This is a long overview of the movie that may be really really really really really really really really really really really really really really really really really long.
Release date: January 23, 2020
Rating: 7.8/10
1,304 votes

FilmDetails

16. Let's start with an image. We want to see the proper poster image just like we did on the Landing scene but now we have a little more room. Feel free to make the image larger.

17. Underneath the image, place an instance of the ShowingTimes component.

18. Add the movie title, tag line, and web site.

19. Add the overview of the movie. Note that it will be multi-line.

20. Show the release date.

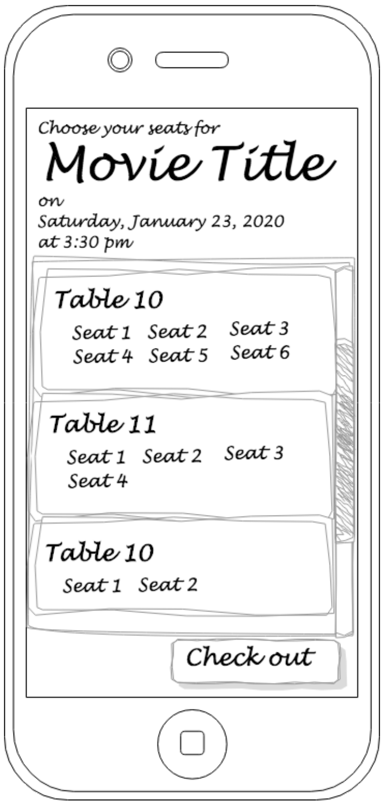
21. Show the runtime.

22. Show the rating like this "Rating: X/10 Y votes" where X is the vote_average

and Y is the vote_count.

Got it? Great! Moving on to PickSeats.

PickSeats



23. Add a <Text> saying "Choose your seats for"

24. Add the showing date and time.

25. Run and test.

26. .map() through the theater.tables and add the table number.

27. Within each table, .map() through the table.seats and add the seat number.

28. Lastly, add a <Button>:
```
<Button title="Check out"></Button>
```

29. Run and test.

30. Bonus! If you are way ahead, fetch the film from the API server and put the film title at the top of the page.

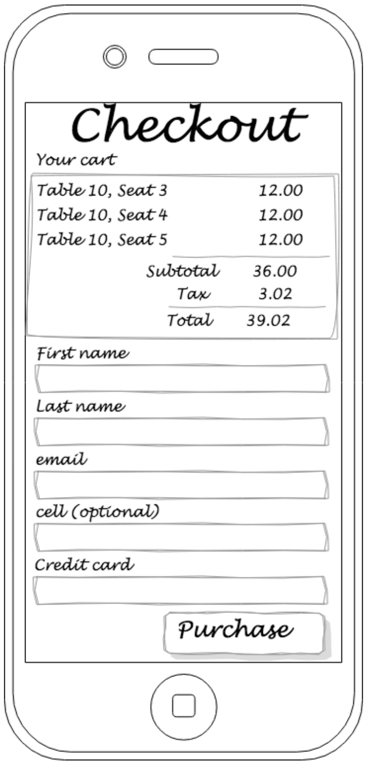
Let's do one last scene, Checkout.

Checkout

This component's purpose is to allow the user to pay for the seats they've selected in PickSeats. They'll add the seats to their cart in PickSeats and check out in Checkout. We need to gather some information from them.

31. First, make sure that Checkout is receiving two props: the cart and the setCart function.

32. Add a title, "Checkout"



33. Add "Your cart".

Don't worry about the cart section just yet.

34. Add a <Text> "First name". This will be the label for our TextInput.

35. Add a <TextInput> for that first name:

```
<TextInput value={firstName} onChangeText={setFirstName} />
```

36. Create a firstName state variable and make your TextInput set that variable with a setFirstName() function.

37. Run and test. Use console.log()s or whatever you choose to make sure it's reading the value typed in.

38. Got it working? Great! Now do the same for:

- Last Name
- Email (optional)
- Cell (optional)
- Credit Card number (aka. "PAN")
- Expiry month
- Expiry year
- CVV

Note: None of these have to actually work. We just want to capture the data for now.

39. Add a button

```
<Button title="Purchase" />
```