VIETNAM NATIONAL UNIVERSITY - HO CHI MINH
UNIVERSITY OF SCIENCE

Coursera - React Native of Meta

Module 2

Student:
23127438 - Dang Truong Nguyen

Academic Supervisors:
Bui Duy Dang
Huynh Lam Hai Dang
Tran Trung Kien

Semester: I
Academic Year: 2025 - 2026

Ho Chi Minh City, October 30, 2025

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Render large list with FlatList

1.1. Video: Rendering large lists using FlatList component

1.1.1. Why FlatList?

- While ScrollView provides a very useful functionality, it does also introduce the issue of slow rendering times for large lists.
- FlatList is another approach that can enable faster performance.
- Rendreing time with FlatList is not affected by large lists.
- For example, using ScrollView with text and images can take a lot of time waiting for rendering because user have to wait for all these images load before they can experience the app.

1.1.2. FlatList - Lazy rendering

- Instead of rendering the items all at once, FlatList render as the items appear.
- Items are only rendered when they are about to appear on the screen and are removed when the user scrolls away from them.
- Result in faster rendering and superior performance.
- **Practice Question:** The `FlatList` component uses lazy rendering of component items. Which of the following are characteristics of lazy rendering? Choose all that apply. $\rightarrow$ Items are rendered as the user encounters them in the app, Rendered items are removed when the user scrolls away from them.

1.1.3. Syntax for FlatList

- Basic use:

```
<FlatList
    data={items}
    renderItem={renderItem}
/>
```

- Props:

```
renderItem({ item, index, separators })
```

  - item: single from array.
  - index: unique index from the data array.
  - separators: optional, highligh or unhighligh items.

1.2. Video: Using the FlatList component

- How to use:

```
<FlatList
    data={menuItemsToDisplay}
    renderItem={renderItem}
/>
```

- Props:

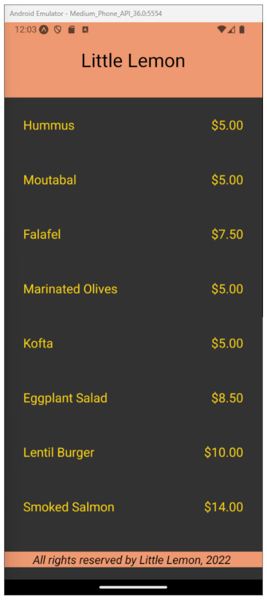
```
// Array of item
menuItemsToDisplay;

// Return a HTML tag, custom tag in this case
const renderItem = ({ item }) => {
    return <Item name={item.name} />
}
```

- **Practice Question:** Which properties are required when using the `FlatList` component? Select all that apply. → RenderItem, Data

1.3. Reading: Explore the FlatList Component

This part is to pratice using FlatList as the same above part's work.

1.4. Reading: Exercise: Render a large list using FlatList



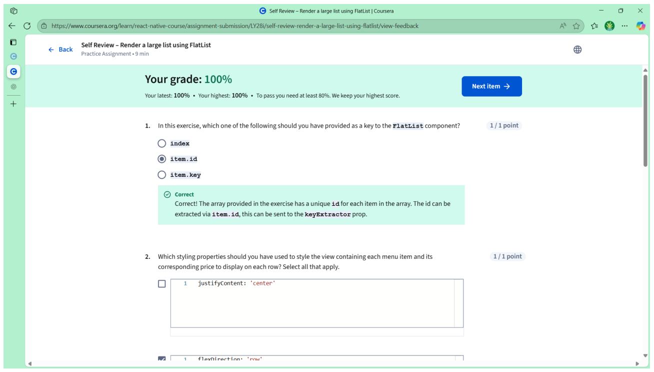
Little Lemon

Hummus	$5.00
Moutabal	$5.00
Falafel	$7.50
Marinated Olives	$5.00
Kofta	$5.00
Eggplant Salad	$8.50
Lentil Burger	$10.00
Smoked Salmon	$14.00

All rights reserved by Little Lemon, 2022

Figure 1.1: Exercise: Render a large list using FlatList

1.5. Self review: Render a large list uisng FlatList


Self Review – Render a large list using FlatList | Coursera
https://www.coursera.org/learn/react-native-course/assignment-submission/LY28i/self-review-render-a-large-list-using-flatlist/view-feedback
Back
Next item

Self Review – Render a large list using FlatList
Practice Assignment • 9 min

Your grade: 100%
Your latest: 100% • Your highest: 100% • To pass you need at least 80%. We keep your highest score.

1. In this exercise, which one of the following should you have provided as a key to the `FlatList` component? 1 / 1 point

- ◯ `index`
- [✓ `item.id`]
- ◯ `item.key`

Correct
Correct! The array provided in the exercise has a unique `id` for each item in the array. The id can be extracted via `item.id`, this can be sent to the `keyExtractor` prop.

2. Which styling properties should you have used to style the view containing each menu item and its corresponding price to display on each row? Select all that apply. 1 / 1 point

- ☐
```
justifyContent: 'center'
```
- ☑
```
flexDirection: 'row'
```

Figure 1.2: Self review: Render a large list using FlatList

1.6. Video: FlatList Methods

1.6.1. keyExtractor

- Using `keyExtractor` for caching and as the react key to track item recording.
- Using at the same as using `map()` in react, we need to pass a key props.

1.6.2. ItemSeperatorComponent

- Using:

```
const Seperator = () => <View
   style={menuItemsStyle.seperator} />


<FlatList
    data={menuItemsToDisplay}
    keyExtractor={(item) => item.id}
    renderItem={renderItem}
    ItemSeparatorComponent={Seperator}
/>
```

1.6.3. ListHeaderComponent

- Using:

```
const Header = () => <Text
  style={menuItemsStyle.headerText}>View Menu</Text>

<FlatList
    data={menuItemsToDisplay}
    keyExtractor={(item) => item.id}
    renderItem={renderItem}
    ItemSeparatorComponent={Seperator}
    ListHeaderComponent={Header}
/>
```

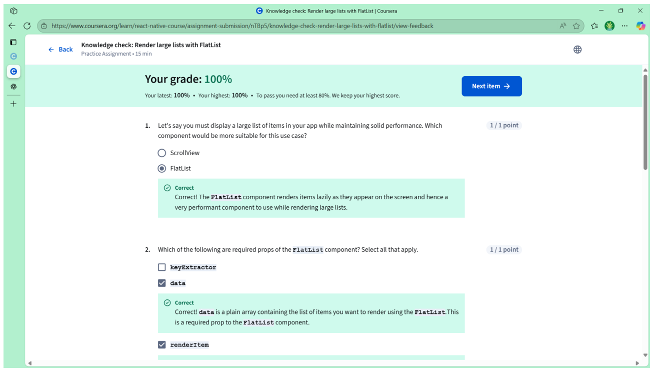
- **Practice Question:** In your app, you have written a header component, and then rendered that component inside of `FlatList` and passed it to the `ListHeaderComponent` prop. When the user scrolls through the list in the app, this header will remain fixed in place. True or false? $\rightarrow$ False

1.6.4. ListFooterComponent

- Using:

```
const Footer = () => <Text
  style={menuItemsStyle.footerText}>Footer</Text>

<FlatList
    data={menuItemsToDisplay}
    keyExtractor={(item) => item.id}
    renderItem={renderItem}
    ItemSeparatorComponent={Seperator}
    ListHeaderComponent={Header}
    ListFooterComponent={Footer}
/>
```

1.7. Practice Assignment: Knowledge Check: Render large lists with FlatList


Knowledge check: Render large lists with FlatList | Coursera
https://www.coursera.org/learn/react-native-course/assignment-submission/nTBp5/knowledge-check-render-large-lists-with-flatlist/view-feedback
Back
Next item

Knowledge check: Render large lists with FlatList
Practice Assignment • 15 min

Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 80%. We keep your highest score.

1. Let's say you must display a large list of items in your app while maintaining solid performance. Which component would be more suitable for this use case? — 1 / 1 point

   - ScrollView
   - [✓ FlatList]

   Correct
   Correct! The `FlatList` component renders items lazily as they appear on the screen and hence a very performant component to use while rendering large lists.

2. Which of the following are required props of the `FlatList` component? Select all that apply. — 1 / 1 point

   - `keyExtractor`
   - [✓ `data`]

   Correct
   Correct! `data` is a plain array containing the list of items you want to render using the `FlatList`.This is a required prop to the `FlatList` component.

   - [✓ `renderItem`]

Figure 1.3: Knowledge Check: Render large lists with FlatList

CHAPTER 2

Render Large Lists with SectionList

2.1. Video: Render large lists by sections using SectionList

2.1.1. Why SectionList?

- The entire menu rendered with FlatList is presented as one long uninterrupted list.
- To make things tidy, divide the list into sections.
- SectionLists has the same features found in FlatList component, but it also supports the separation of items in the lists.

2.1.2. Comparision between FlatList and SectionList components

- FlatList and SectionList use the same lazy rendering and inherits the props of the ScrollView.
- SectionList support break the layout into smaller chunks and add appropriate child-headers.
- **Practice Question:** Which of the following are characteristics of the `SectionList` component? Select all that apply. → It uses lazy rendering, It supports section headers and separators, It inherits props of the `ScrollView` component.

→ SectionList improves performance and enables to present the menu items in a more pleasant and easy to read.

2.1.3. Syntax

SectionList component syntax

```
<SectionList
    sections={items} // Array of list sections
```

```
        renderItem={renderItem} // Default renderer
    />
```

- The sections must follow this structure:

```
        [
            {
                title: '',
                data: []
            },
            ...
        ]
```

- Then `renderItem` uses the below props, which the item is each item of `data` array.
- The `renderSectionHeader` need to pass the single section, then extract the title from those section.

renderItem syntax

```
    renderItem({ item, index, section, separators });

    /*
        item: section data key
        index: item position in section
        section: section object
        separators: contains special functions
    */
```

2.1.4. Some props could be passed into SectionList

- Header
- Footer
- Separators
- ...

2.2. Video: Using the SectionList component

- **Practice Question:** You are creating a 'Contact' page for your React Native app that separates contact details into sections such as email, phone, and social media. In your `SectionList` component, which property would you use to extract the data from the array to render these section headers? → renderSectionHeader
- The iOS's Header of each section sticks to the top of the SectionList.
- Its not applied for Android.

2.3. Reading: Explore the SectionList component

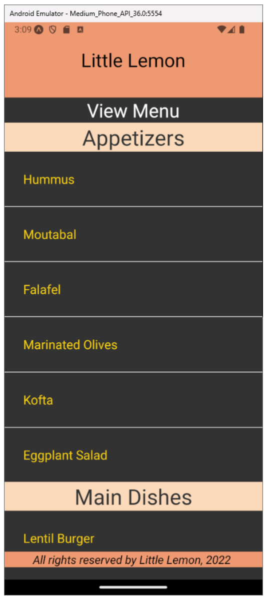
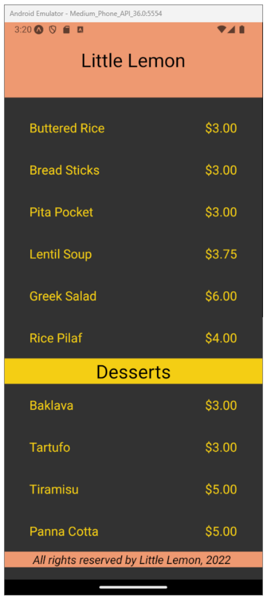


Figure 2.1: Exercise: Explore the SectionList component

2.4. Reading: Exercise: Render a large list using SectionList



Android Emulator - Medium_Phone_API_36.0:5554

3:20

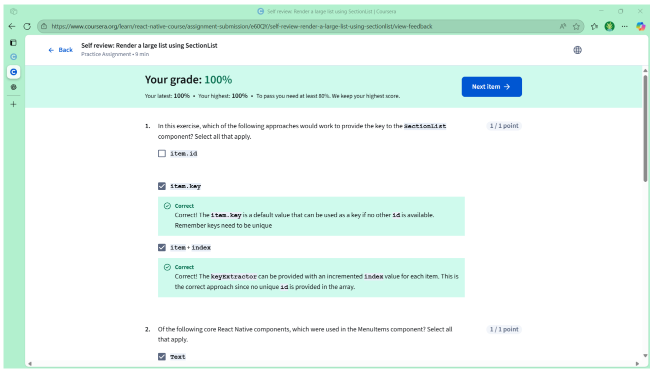
Little Lemon

Buttered Rice	$3.00
Bread Sticks	$3.00
Pita Pocket	$3.00
Lentil Soup	$3.75
Greek Salad	$6.00
Rice Pilaf	$4.00
Desserts	
Baklava	$3.00
Tartufo	$3.00
Tiramisu	$5.00
Panna Cotta	$5.00

All rights reserved by Little Lemon, 2022

Figure 2.2: Exercise: Render a large list using SectionList

2.5. Practice Assignment: Self review: Render a large list using SectionList



Self review: Render a large list using SectionList | Coursera

https://www.coursera.org/learn/react-native-course/assignment-submission/e60QY/self-review-render-a-large-list-using-sectionlist/view-feedback

Back

Self review: Render a large list using SectionList
Practice Assignment • 9 min

Your grade: 100%

Your latest: 100% • Your highest: 100% • To pass you need at least 80%. We keep your highest score.

Next item

1. In this exercise, which of the following approaches would work to provide the key to the `SectionList` component? Select all that apply. 1 / 1 point

- [ ] `item.id`
- [x] `item.key`

  Correct
  Correct! The `item.key` is a default value that can be used as a key if no other `id` is available. Remember keys need to be unique

- [x] `item` + `index`

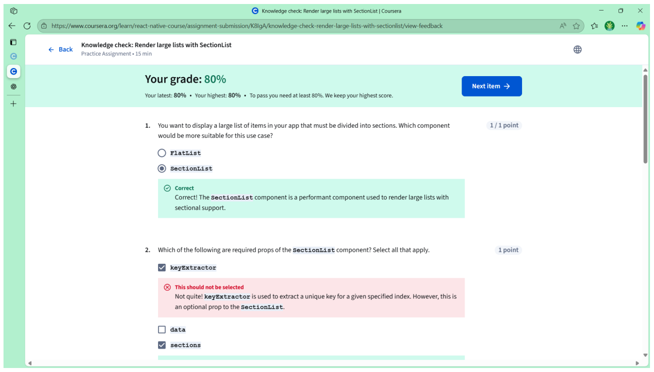
  Correct
  Correct! The `keyExtractor` can be provided with an incremented `index` value for each item. This is the correct approach since no unique `id` is provided in the array.

2. Of the following core React Native components, which were used in the MenuItems component? Select all that apply. 1 / 1 point

- [x] `Text`

Figure 2.3: Self review: Render a large list using SectionList

2.6. Practice Assignment: Knowledge checking: Render a large list using SectionList


Knowledge check: Render large lists with SectionList | Coursera
https://www.coursera.org/learn/react-native-course/assignment-submission/K8IgA/knowledge-check-render-large-lists-with-sectionlist/view-feedback
Back
Next item

Knowledge check: Render large lists with SectionList
Practice Assignment • 15 min

Your grade: 80%

Your latest: **80%** • Your highest: **80%** • To pass you need at least 80%. We keep your highest score.

1. You want to display a large list of items in your app that must be divided into sections. Which component would be more suitable for this use case? — 1 / 1 point

   - ○ `FlatList`
   - ● `SectionList`

   Correct
   Correct! The `SectionList` component is a performant component used to render large lists with sectional support.

2. Which of the following are required props of the `SectionList` component? Select all that apply. — 1 point

   - ☑ `keyExtractor`

   This should not be selected
   Not quite! `keyExtractor` is used to extract a unique key for a given specified index. However, this is an optional prop to the `SectionList`.

   - ☐ `data`
   - ☑ `sections`

Figure 2.4: Knowledge Check: Render a large list using SectionList

CHAPTER 3

Accept user input

3.1. Video: What is the TextInput component?

3.1.1. Use case in this project

- Little Lemon App want to get the feedback from the customer.
- They allow users to type in their name, contact details and custom messages.

3.1.2. TextInput

- A native component that will open up the native iOS or Android keyboard depending on the device you are using.
- The same code, users have different experiences across platform.
- Benefits:
    - Auto correction
    - Placeholder text
    - Support for different keyboard types
    - Auto capitalization

3.1.3. Syntax

- TextInput syntax

```
    <TextInput
        // Styles
        style={styles.input}
        onChangeText={onChangeFirstName}
        // Default text in input box
        placeholder='First name'
        // Value displayed in the input box
        value={firstName}
    />
```

- `onChangeText`
  - Its a callback.
  - Use with `useState()`

    ```
                const [firstName, onChangeFirstName] = useState('')
    ```

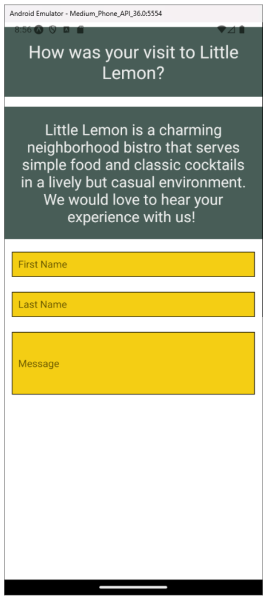
- **Practice Question:** In the Little Lemon mobile app, you'd like a to add a text input box that displays the text "Email address" before the user types anything inside of it. Which prop would you pass in to accomplish this? → placeholder

- More props:
  - `keyboardType`: "default","number pad", "decimal pad", "numeric", "email address", "phone pad", ...

3.2. Video: Configure the TextInput Component

- **Practice Question:** You are adding a text input box to your app for the user to enter their email address. You will need to configure at least two separate `TextInput` components to do this – one for iOS devices, and another for Android devices. True or false? → False

3.3. Reading: TextInput Component and its Feature



Android Emulator - Medium_Phone_API_36.0:5554

How was your visit to Little Lemon?

Little Lemon is a charming neighborhood bistro that serves simple food and classic cocktails in a lively but casual environment. We would love to hear your experience with us!

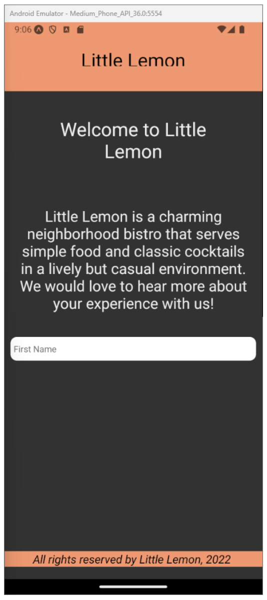
First Name

Last Name

Message

Figure 3.1: TextInput Component

3.4. Reading: Exercise: Create a TextInput Component



Android Emulator - Medium_Phone_API_36.0:5554

9:06

Little Lemon

Welcome to Little Lemon

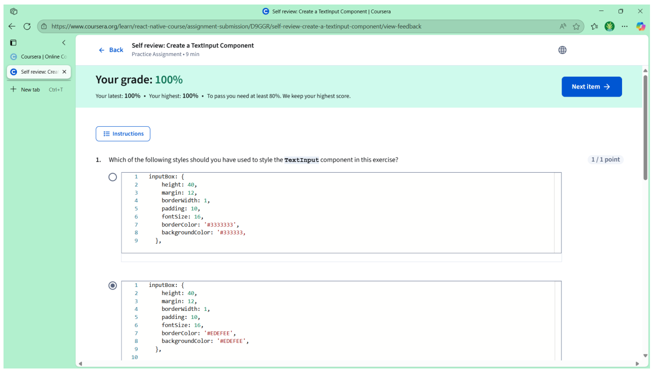
Little Lemon is a charming neighborhood bistro that serves simple food and classic cocktails in a lively but casual environment. We would love to hear more about your experience with us!

First Name

All rights reserved by Little Lemon, 2022

Figure 3.2: Exercise: Create a TextInput Component

3.5. Practice Assignment: Self review: Create a TextInput Component


Self review: Create a TextInput Component | Coursera

https://www.coursera.org/learn/react-native-course/assignment-submission/D9GGR/self-review-create-a-textinput-component/view-feedback

Self review: Create a TextInput Component
Practice Assignment • 9 min

Your grade: 100%

Your latest: 100% • Your highest: 100% • To pass you need at least 80%. We keep your highest score.

1. Which of the following styles should you have used to style the `TextInput` component in this exercise? 1 / 1 point

```
inputBox: {
    height: 40,
    margin: 12,
    borderWidth: 1,
    padding: 10,
    fontSize: 16,
    borderColor: '#3333333',
    backgroundColor: '#333333,
  },
```

```
inputBox: {
    height: 40,
    margin: 12,
    borderWidth: 1,
    padding: 10,
    fontSize: 16,
    borderColor: '#EDEFEE',
    backgroundColor: '#EDEFEE',
  },
```

Figure 3.3: Self review: Create a TextInput Component

3.6. Video: Virtual Keyboard on Mobiles App

3.6.1. Problem about keyboard

- Open the virtual keyboard in an emulator:
- Ctrl/Cmd + Shift + K
- When the keyboard opens, it may cover important parts of your app.

3.6.2. React Native Props

- `keyboardDismissMode` property
  - The keyboard will be dismissed when user scrolls the view.

    ```
    <ScrollView keyboardDismissMode='on-drag'/>
    ```

  - Set to none if you want the keyboard to stay open even when the user scrolls the view.
- `KeyboardAvoidingView` component
  - Automatically adjust its height, position, or padding to avoid conflict with other views.
  - Push the view on the top of the keyboard when it appears.
  - This component inherit all the props of the View component.

```
            <KeyboardAvoidingView
                behavior={Platform.OS === 'ios' ? 'padding' :
                   'position'}
            >
                <TextInput
                    style={styles.input}
                    onChangeText={onChangeFirstName}
                    placeholder='First name'
                    value={firstName}
                />
            </KeyboardAvoidingView>
```

- **Practice Question:** Your app has text input boxes that open the virtual keyboard when tapped. For the `keyboardDismissMode` property of the `ScrollView`, you've set the value to `on-drag`. What does this do? → The virtual keyboard will be dismissed when the user scrolls.

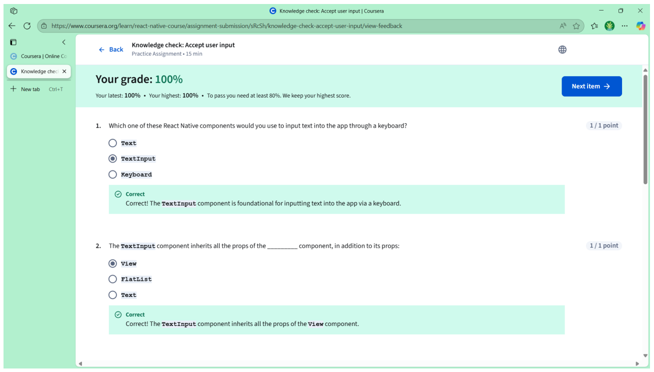
3.7. Reading: Tips and Tricks to handle virtual keyboard

This part is the same as in the previous section about virtual keyboard.

3.8. Video: Handling the Virtual Keyboard

- **Practice Question:** In your mobile app, you want to set the virtual keyboard to be dismissed as soon as the user scrolls. Which value of `keyboardDismissMode` would you assign to set this behavior? → on-drag

3.9. Practice Assignment: Knowledge Check: Accept user input


Knowledge check: Accept user input | Coursera
https://www.coursera.org/learn/react-native-course/assignment-submission/sRcSh/knowledge-check-accept-user-input/view-feedback
Coursera | Online Co
Knowledge chec
New tab Ctrl+T
Back
Next item

Knowledge check: Accept user input
Practice Assignment • 15 min

Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 80%. We keep your highest score.

1. Which one of these React Native components would you use to input text into the app through a keyboard? 1 / 1 point

   - ○ `Text`
   - ● `TextInput`
   - ○ `Keyboard`

   Correct
   Correct! The `TextInput` component is foundational for inputting text into the app via a keyboard.

2. The `TextInput` component inherits all the props of the _________ component, in addition to its props: 1 / 1 point

   - ● `View`
   - ○ `FlatList`
   - ○ `Text`

   Correct
   Correct! The `TextInput` component inherits all the props of the `View` component.

Figure 3.4: Knowledge Check: Accept user input

CHAPTER 4

Working with TextInput components

4.1. Reading: Common props of TextInput component

4.1.1. placeholder

- The placeholder string displayed before the user enters anything into the text input box.
- Syntax:

```
<TextInput
    style={styles.input}
    value={firstName}
    onChangeText={onChangeFirstName}
    placeholder={'First Name'}
/>
```

4.1.2. multiline

- Boolean. When set to true, the text input can be muliple lines.
- By default, it is set to false, all the text that cannot fit within a single line is truncated and not visible to user.
- Syntax:

```
<TextInput
    style={styles.messageInput}
    value={message}
    onChangeText={onChangeMessage}
    placeholder={'Please leave feedback'}
```

```
        multiline={true}
    />
```

4.1.3. maxLength

- This props limits the maximum number of characters that can be entered.
- This props accepts a number as the value.
- Syntax:

```
    <TextInput
        style={styles.messageInput}
        value={message}
        onChangeText={onChangeMessage}
        placeholder={'Please leave feedback'}
        multiline={true}
        maxLength={250}
    />
```

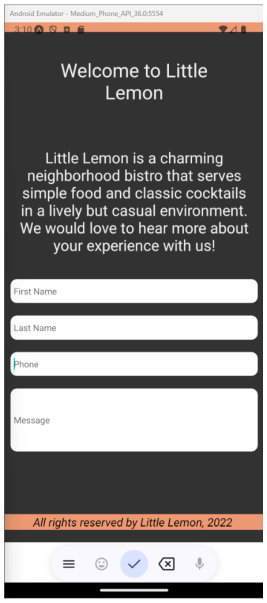
4.1.4. keyboardType

- React Native's support keyboard.
- number-pad, decimal-pad, numeric, email-address, phone-pad, url, ...
- Syntax:

```
    <TextInput
        style={styles.input}
        value={phoneNumber}
        onChangeText={onChangePhoneNumber}
        placeholder={'Phone Number'}
        keyboardType={'phone-pad'}
    />
```

4.2. Video: Passing props to TextInput Component

- Use `secureTextEntry=true` to hide the input text from user (Use for password).
- **Practice Question:** You would like to create a text input box in your app for users to enter their password. To make this more secure, you would like for the characters entered in the box to be hidden and for the box to have a limit of 14 characters. Which `TextInput` props would you pass in to accomplish this? Choose all that apply. → secureTextEntry, maxLength



Welcome to Little Lemon

Little Lemon is a charming neighborhood bistro that serves simple food and classic cocktails in a lively but casual environment. We would love to hear more about your experience with us!

All rights reserved by Little Lemon, 2022

Figure 4.1: Passing props to TextInput Component

4.3. Reading: Exercise: Create a login page

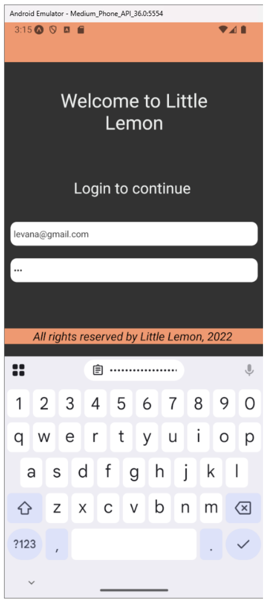
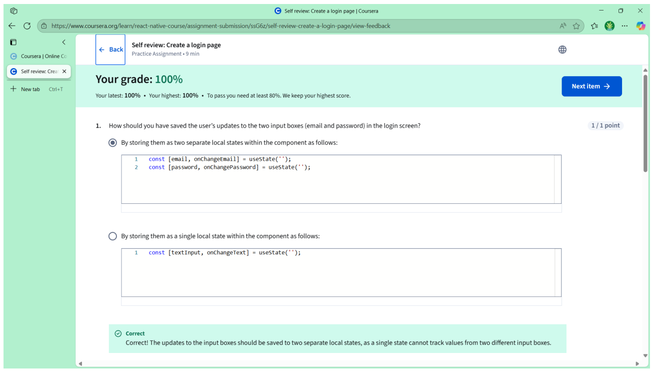

Android Emulator - Medium_Phone_API_36.0:5554
Welcome to Little Lemon
Login to continue
levana@gmail.com
All rights reserved by Little Lemon, 2022

Figure 4.2: Exercise: Create a login page

4.4. Practice Assignment: Self review: Create a login page


Self review: Create a login page | Coursera
https://www.coursera.org/learn/react-native-course/assignment-submission/ssG6z/self-review-create-a-login-page/view-feedback
Back
Next item

Self review: Create a login page
Practice Assignment • 9 min

Your grade: 100%

Your latest: 100% • Your highest: 100% • To pass you need at least 80%. We keep your highest score.

1. How should you have saved the user's updates to the two input boxes (email and password) in the login screen? 1 / 1 point

[✓ By storing them as two separate local states within the component as follows:]

```
const [email, onChangeEmail] = useState('');
const [password, onChangePassword] = useState('');
```

By storing them as a single local state within the component as follows:

```
const [textInput, onChangeText] = useState('');
```

Correct
Correct! The updates to the input boxes should be saved to two separate local states, as a single state cannot track values from two different input boxes.

Figure 4.3: Self review: Create a login page

4.5. Video: Using TextInput Methods

4.5.1. onFocus

- This is a callback medthod that is called when the text input is focused.

- $onFocus(() => \{\})$

4.5.2. onBlur

- This is also a callback method, called when the text input it remove from being focused.
- $onBlur(() => \{\})$

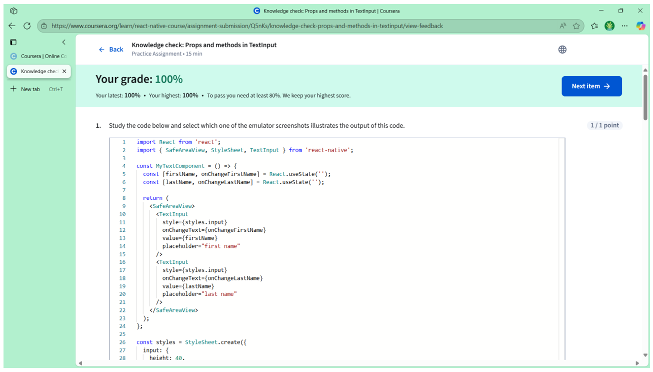
4.6. clearButtonMode

- A clear button will appear on the right side of the text box.
- This button allow to clear all the input text.
- Only support for iOS.
- Dont work if we have `multiline` props.
- clearButtonMode='always'. The default is 'never'
- **Practice Question:** You are creating a form for your app and would like it to display an alert whenever a user exits a certain text input box. Which prop of the `TextInput` component would you use to achieve this? → onBlur

4.7. Reading: TextInput Methods

This part is the same content with above part which I have taken note clearly.

4.8. Practice Assignment: Knowledge Check: Props and methods in TextInput


Knowledge check: Props and methods in TextInput | Coursera
https://www.coursera.org/learn/react-native-course/assignment-submission/Q5nKs/knowledge-check-props-and-methods-in-textinput/view-feedback
Coursera | Online Co
Knowledge chec
New tab Ctrl+T
Back
Next item

Knowledge check: Props and methods in TextInput
Practice Assignment • 15 min

Your grade: 100%

Your latest: 100% • Your highest: 100% • To pass you need at least 80%. We keep your highest score.

1. Study the code below and select which one of the emulator screenshots illustrates the output of this code. 1 / 1 point

```
import React from 'react';
import { SafeAreaView, StyleSheet, TextInput } from 'react-native';

const MyTextComponent = () => {
  const [firstName, onChangeFirstName] = React.useState('');
  const [lastName, onChangeLastName] = React.useState('');

  return (
    <SafeAreaView>
      <TextInput
        style={styles.input}
        onChangeText={onChangeFirstName}
        value={firstName}
        placeholder="first name"
      />
      <TextInput
        style={styles.input}
        onChangeText={onChangeLastName}
        value={lastName}
        placeholder="last name"
      />
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  input: {
    height: 40,
```

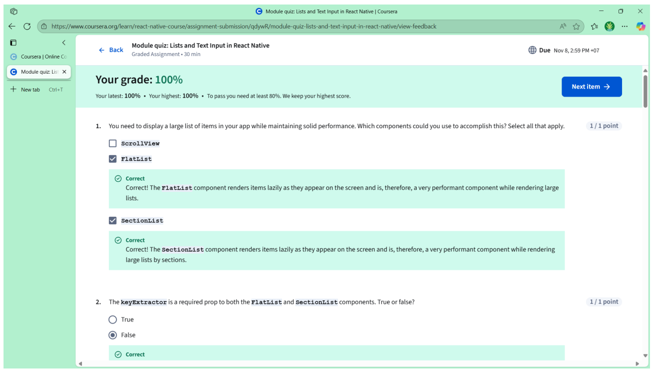
Figure 4.4: Knowledge Check: Props and methods in TextInput

4.9. Video: Module summary: Lists and Text Input in React Native

- FlatList component
  - Lazy rendering
  - FlatList syntax
  - Render large lists
- FlatList methods
  - keyExtractor
  - ItemSeperatorComponent
  - ListHeaderComponent
  - ListFooterComponent
- SectionList component
  - Differences from FlatList
  - SectionList syntax

  - Render large list with sections
- TextInput component
  - Create text input boxes
  - Configure TextInput
  - Create TextInput components in ScrollView
- Keyboard Handling
  - keyboardDismissMode
  - KeyboardAvoidingView
- TextInput props
  - Enable additional TextInput functionalities
  - Set responses to actions in text box

4.10. Graded Assignment: Module quiz: Lists and Text Input in React Native


Module quiz: Lists and Text Input in React Native | Coursera
https://www.coursera.org/learn/react-native-course/assignment-submission/qdywR/module-quiz-lists-and-text-input-in-react-native/view-feedback
Coursera | Online Co
Module quiz: Lis
New tab Ctrl+T
Back

Module quiz: Lists and Text Input in React Native
Graded Assignment • 30 min

Due Nov 8, 2:59 PM +07

Your grade: 100%

Your latest: 100% • Your highest: 100% • To pass you need at least 80%. We keep your highest score.

1. You need to display a large list of items in your app while maintaining solid performance. Which components could you use to accomplish this? Select all that apply. 1 / 1 point

- [ ] `ScrollView`
- [x] `FlatList`

Correct
Correct! The `FlatList` component renders items lazily as they appear on the screen and is, therefore, a very performant component while rendering large lists.

- [x] `SectionList`

Correct
Correct! The `SectionList` component renders items lazily as they appear on the screen and is, therefore, a very performant component while rendering large lists by sections.

2. The `keyExtractor` is a required prop to both the `FlatList` and `SectionList` components. True or false? 1 / 1 point

True [✓ False]

Correct

Figure 4.5: Module quiz: Lists and Text Input in React Native