

**VIETNAM NATIONAL UNIVERSITY - HO CHI MINH  
UNIVERSITY OF SCIENCE**

**Coursera - React Native of Meta**

**Module 3**

**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 31, 2025**

# TABLE OF CONTENTS

<b>TABLE OF CONTENTS</b>	<b>1</b>
<b>LIST OF FIGURES</b>	<b>3</b>
<b>1 Pressable Component</b>	<b>4</b>
1.1 Video: What is the Pressable Component? . . . . .	4
1.1.1 Pressable . . . . .	4
1.1.2 Pressable Methods . . . . .	4
1.1.3 HitRect . . . . .	5
1.2 Video: Using Pressable Component . . . . .	5
1.3 Reading: Exploring Pressable . . . . .	5
1.4 Practice Assignment: Self review: Create a clickable text area with Pressable . .	6
1.5 Practice Assignment: Knowledge Check: Pressable Component . . . . .	7
<b>2 Images Component</b>	<b>8</b>
2.1 Video: Displaying Images in React Native . . . . .	8
2.1.1 Images Component . . . . .	8
2.1.2 Using . . . . .	8
2.1.3 Support images type . . . . .	9
2.2 Video: Using Image Component . . . . .	10
2.3 Video: Styling an Image within the app . . . . .	10
2.4 Video: Passing props to the Image Component . . . . .	10
2.5 Reading: Exploring props to the Image Component . . . . .	11
2.6 Video: Setting Background Images . . . . .	11

2.7	Reading: Exercise: Displaying Images in your app . . . . .	12
2.8	Practice Assignment: Self review: Displaying Images in your app . . . . .	12
2.9	Practice Assignment: Knowledge Check: Images in React Native . . . . .	13
<b>3</b>	<b>Hooks</b>	<b>14</b>
3.1	Video: What are Hooks? . . . . .	14
3.2	Video: Using the useColorScheme hook . . . . .	14
3.3	Reading: Exploring the useColorScheme hook . . . . .	15
3.4	Using useWindowDimensions hook . . . . .	15
3.5	Reading: Explore the useWindowDimensions hook . . . . .	16
3.6	Video: Using other community hooks . . . . .	16
3.7	Reading: Exploring other community hooks . . . . .	16
3.7.1	useDeviceOrientation Hook . . . . .	16
3.7.2	useAppState Hook . . . . .	17
3.7.3	useClipboard Hook . . . . .	17
3.8	Reading: Exercise: Hooks in React Native . . . . .	17
3.9	Practice Assignment: Self review: Hooks in React Native . . . . .	18
3.10	Practice Assignment: Knowledge Check: Hooks in React Native . . . . .	18
3.11	Module summary: Pressable, Images and Hooks in React Native . . . . .	18
3.12	Practice Assignment: Module quiz: Pressable, Images and Hooks in React Native	19

## LIST OF FIGURES

Figure 1.1	Screen before pressing the View Menu Button . . . . .	5
Figure 1.2	Screen after pressing the View Menu Button . . . . .	5
Figure 1.3	Before pressing the Login Button . . . . .	6
Figure 1.4	After pressing without filling the form . . . . .	6
Figure 1.5	After pressing with the form filled . . . . .	6
Figure 1.6	Self review: Create a clickable text area with Pressable . . . . .	6
Figure 1.7	Knowledge Check: Pressable Component . . . . .	7
Figure 2.1	Using Image Component . . . . .	10
Figure 2.2	resizeMode contain . . . . .	11
Figure 2.3	resizeMode cover . . . . .	11
Figure 2.4	resizeMode stretch . . . . .	11
Figure 2.5	resizeMode center . . . . .	11
Figure 2.6	resizeMode repeat . . . . .	11
Figure 2.7	Exercise: Displaying Images in your app . . . . .	12
Figure 2.8	Self review: Displaying Images in your app . . . . .	12
Figure 2.9	Knowledge Check: Images in React Native . . . . .	13
Figure 3.1	UI with light theme . . . . .	15
Figure 3.2	The window dimension of my virtual phone . . . . .	16
Figure 3.3	Self review: Hooks in React Native . . . . .	18
Figure 3.4	Knowledge Check: Hooks in React Native . . . . .	18
Figure 3.5	Module quiz: Pressable, Images and Hooks in React Native . . . . .	19

# CHAPTER 1

## Pressable Component

### 1.1. Video: What is the Pressable Component?

#### 1.1.1. Pressable

- A core component which was recently added to React Native.
- It can detect various stages of press interactions from the users on any child of components.
- It is wrapped around its child elements which then become pressable.
- For example:

```
<Pressable onPress={onPressFunction}>  
  <Text>Button</Text>  
</Pressable>
```

#### 1.1.2. Pressable Methods

- `onPressIn`: called when a press is activated
- `onPressOut`: called when a press is inactivated, user move away from the element.
- `onLongPress`: 500 milisecond and longer.
- `delayLongPress`: customized `onLongPress` timing.
- **Practice Question:** A user's press gesture is deactivated on the Little Lemon app. Which method is called when this action occurs? → `onPressOut`

### 1.1.3. HitRect

- Define pressable range for an element.
- Finger are not always precise, especially on small screen, so this option solve this problem quite lean.

## 1.2. Video: Using Pressable Component

- **Practice Question:** The `Pressable` component only accepts `Text` components as children. True or false? → False

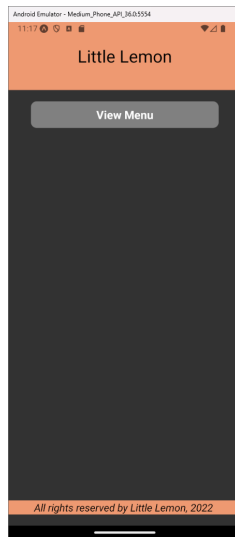


Figure 1.1: Screen before pressing the View Menu Button

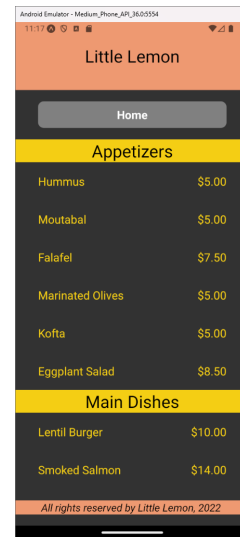


Figure 1.2: Screen after pressing the View Menu Button

### 1.3. Reading: Exploring Pressable

This part is the same progress with the above section about render state base on the state of View Menu Button.



Figure 1.3: Before pressing the Login Button

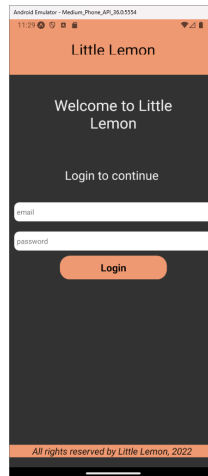


Figure 1.4: After pressing without filling the form

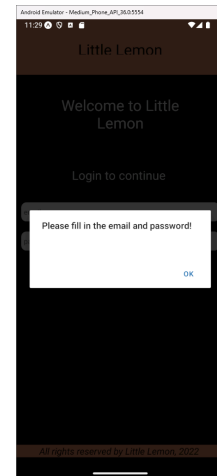


Figure 1.5: After pressing with the form filled

## 1.4. Practice Assignment: Self review: Create a clickable text area with Pressable

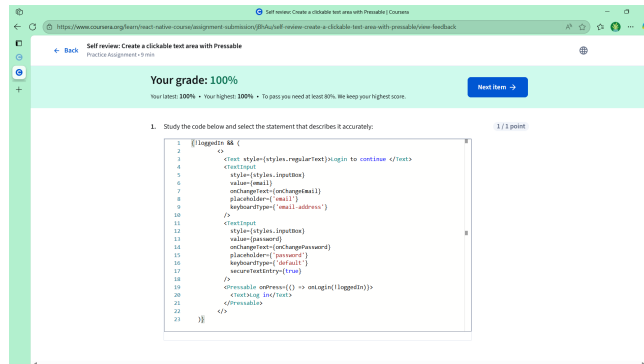


Figure 1.6: Self review: Create a clickable text area with Pressable

## 1.5. Practice Assignment: Knowledge Check: Pressable Component

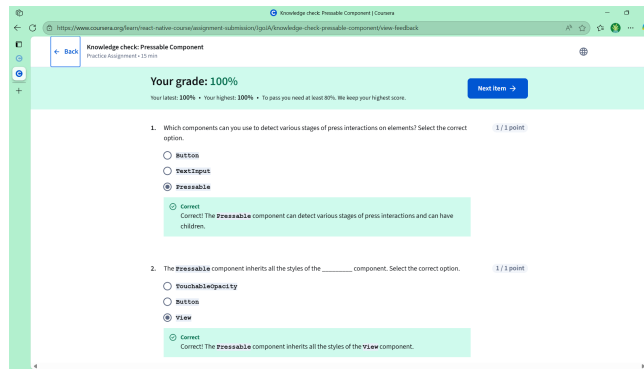


Figure 1.7: Knowledge Check: Pressable Component



## CHAPTER 2

### Images Component

#### 2.1. Video: Displaying Images in React Native

##### 2.1.1. Images Component

- It is a core component used to displaying different types of images.
- Display:
  - Static images from Resources
  - Temporary local images
  - Images from local disk
  - Network images

##### 2.1.2. Using

- Create a img folder within the project folder.
- Syntax:

```
<View>
  <Image
    style={}
    source={require('url')}
  />
</View>
```

- New props to pass in the style: **resizeMode**, which determines how to resize the image when the frame doesn't match the image's dimensions.

- Option: 'stretch', 'repeat', 'cover', 'center'
- If the images is hosted on network:

```
<View>
  <Image
    style={}
    source={{
      uri: 'url'
    }}
  />
</View>
```

- React Native also provides `loadingIndicatorSource`, which represents resources used to render the loading indicator, this is displayed until the resources is fully loaded.

- **Practice Question:** You would like to add some images to your mobile app using the `Image` component. Which sources can you retrieve images from? Choose all that apply.

→ Answer:

- ☐ A local drive on your computer
- ☐ A web URL
- ☐ Temporary Images
- ☐ Resources

### 2.1.3. Support images type

- png
- jpg
- gif
- webp
- These type are supported for both iOS and Android.
- In addition, iOS supports for psd, raw and some other uncompressed image format.

## 2.2. Video: Using Image Component

- **Practice Question:** In your project folder, you have a folder called `img` inside of which you have uploaded an image file titled `header.png`. How would you render this image using an `Image` component? Choose the appropriate line of code from the following. →

```
<Image source={require('img/header.png')} />
```

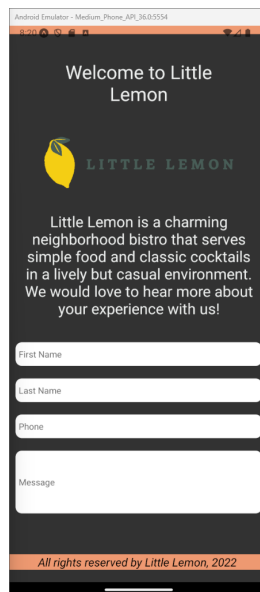


Figure 2.1: Using Image Component

## 2.3. Video: Styling an Image within the app

- This part is to use the `resizeMode` method that I have already used in the above part.
- **Practice Question:** Which one of the resize modes is described in the following statement: It will scale the image uniformly so that both the dimensions – the width and the height of the image will be equal to or less than the corresponding dimension of the view. → Contain

## 2.4. Video: Passing props to the Image Component

- Using `accessible` props, a Boolean.

- When the Boolean is true, it indicates that the image is an accessibility element.
- **Practice Question:** If you want an image to fill the entire box, which of the following resize mode props will you use? → Cover

## 2.5. Reading: Exploring props to the Image Component

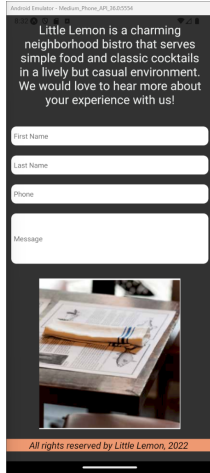


Figure 2.2: resizeMode contain

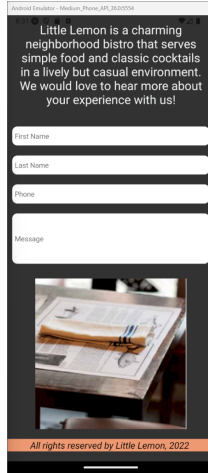


Figure 2.3: resizeMode cover

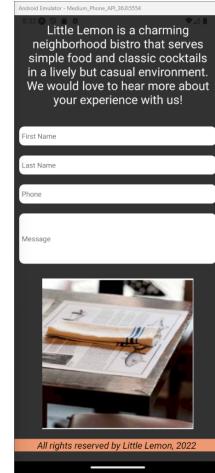


Figure 2.4: resizeMode stretch



Figure 2.5: resizeMode center

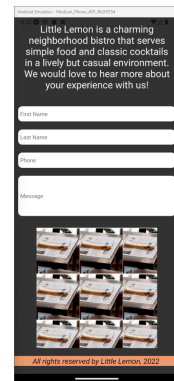


Figure 2.6: resizeMode repeat

## 2.6. Video: Setting Background Images

- `imageBackground` component, which can utilize display background images.
- This component support for both iOS and Android.

- Its inherit all props from the Images.
- **Practice Question:** When an image is rendered as a background image, it implies that you can have text and other content on top of it. True or false?
- → True

## 2.7. Reading: Exercise: Displaying Images in your app



Figure 2.7: Exercise: Displaying Images in your app

## 2.8. Practice Assignment: Self review: Displaying Images in your app

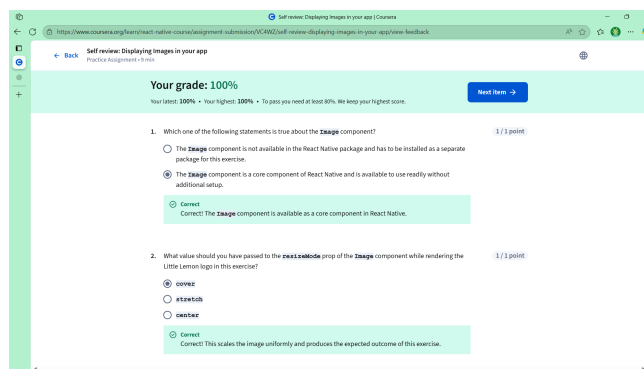


Figure 2.8: Self review: Displaying Images in your app

## 2.9. Practice Assignment: Knowledge Check: Images in React Native

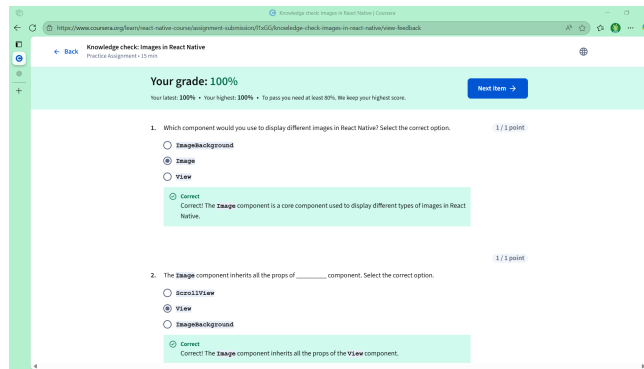


Figure 2.9: Knowledge Check: Images in React Native

## CHAPTER 3

### Hooks

#### 3.1. Video: What are Hooks?

- useState hook is used to manage the state within a component and to keep track of this.
- **Practice Question:** Observe the following line of code which sets up the useState hook:

```
const [showMenu, setShowMenu] = useState(false);
```

Which element in this code represents the state variable? → showMenu

- In addition to the hooks that are built-in to React, you can also build your own hooks which will let you extract custom component logic into reusable functions
- The biggest benefit of hooks overall is the readability and simplicity that they provide for your code.

#### 3.2. Video: Using the useColorScheme hook

- Provides and subscribes to color scheme updates from the appearance module in React Native.
- Practical use: Apply styling changes in the app based on whether the user's device is set to light theme or dark theme.
- Value: light, dark, null **Practice Question:** You want to apply the useColorScheme hook to adjust your app's color scheme based on the device's color theme setting. Which color theme options are available to you? Select all that apply. → null, light, dark

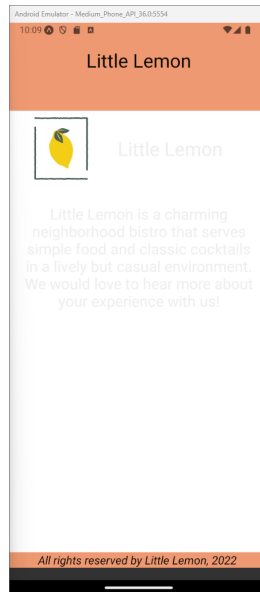


Figure 3.1: UI with light theme

### 3.3. Reading: Exploring the `useColorScheme` hook

This part is the same progress with above part.

### 3.4. Using `useWindowDimensions` hook

- This hook provides information about the window size.
- From this information, we can determine how elements could appear on Windows of different sizes and make more informed decisions on how to adjust size scale.
- Using:

```
const {width, height, fontScale} = useWindowDimensions();
```

**Practice Question:** You're creating an app using React Native, and you need to make some design decisions. For what purpose would you use the `useWindowDimensions` hook? Check all that apply. → to determine the window height, width, font scale.



### 3.5. Reading: Explore the useWindowDimensions hook

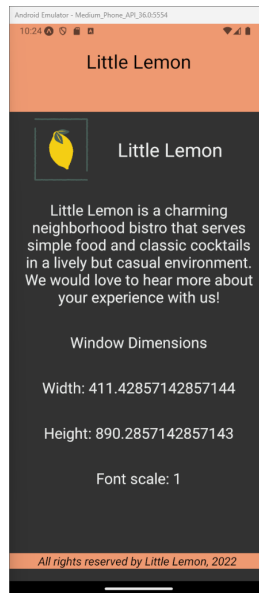


Figure 3.2: The window dimension of my virtual phone

### 3.6. Video: Using other community hooks

- React Native is an open source, meaning that the developer community can contribute new hooks and extend React Native.
- To access, first we need to access the web page at [Github.com/react-native-community/hooks](https://github.com/react-native-community/hooks).
- Installation:

```
npm install @react-native-community/hooks
```

**Practice Question:** You have a React Native app that utilizes location tracking, but you would like this feature to be disabled when a user pushes the app to the background. Which community hook could you use to set up this functionality? → `useAppState`

### 3.7. Reading: Exploring other community hooks

#### 3.7.1. useDeviceOrientation Hook

- This hook can determine if the user's mobile device is viewed in landscape or portrait mode.

- You can utilize this hook to determine the orientation if your app needs to support both modes.

```
import { useDeviceOrientation } from
  '@react-native-community/hooks'

const orientation = useDeviceOrientation()

console.log(orientation.portrait) // Boolean
console.log(orientation.landscape) // Boolean
```

### 3.7.2. useAppState Hook

- This hook is used to determine the current app state.
- It can be active, background or inactive (iOS only).

### 3.7.3. useClipboard Hook

- Store text from the clipboard within the app.
- Using:

```
import { useClipboard } from
  '@react-native-community/hooks'

const [data, setString] = useClipboard()

<Text>{data}</Text>

<Button title='Update Clipboard' onPress={() =>
  setString('new clipboard data')}>Set Clipboard</Button>
```

## 3.8. Reading: Exercise: Hooks in React Native

This part is to implement the light/dark mode that I have done in the section 3.2

### 3.9. Practice Assignment: Self review: Hooks in React Native

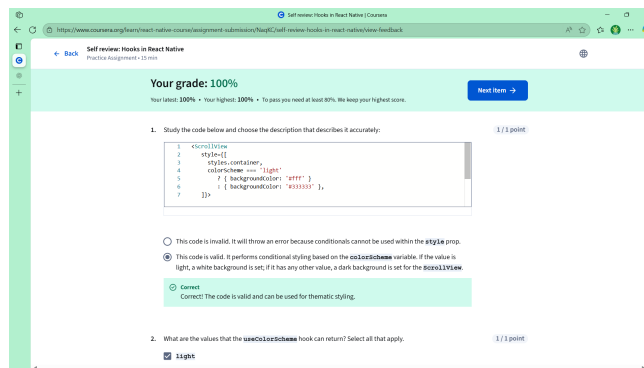


Figure 3.3: Self review: Hooks in React Native

### 3.10. Practice Assignment: Knowledge Check: Hooks in React Native

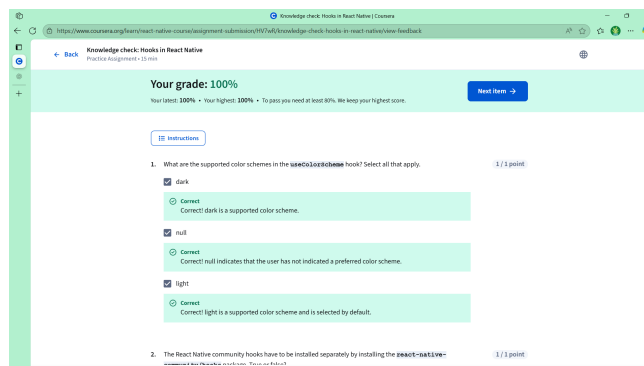


Figure 3.4: Knowledge Check: Hooks in React Native

### 3.11. Module summary: Pressable, Images and Hooks in React Native

- Pressable components
  - Explain Pressable component
  - Identify Pressable methods
  - Explain how to represent a Pressable component
  - Create Pressable that responds to press

- Track Pressable state
- Images components
  - Explain Image component
  - Identify images sources
  - Identify supported Image types
  - Create Image component
  - Style an Image
- Hooks in React Native
  - Describe uses for hooks
  - Describe hook structure
  - useColorScheme hook
  - useWindowDimensions hook
  - Community hooks

### 3.12. Practice Assignment: Module quiz: Pressable, Images and Hooks in React Native

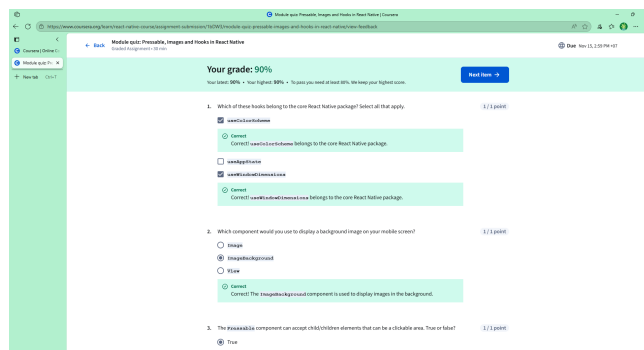


Figure 3.5: Module quiz: Pressable, Images and Hooks in React Native