

Lập trình Mobile

Ths. Đỗ Phúc Thịnh

Tóm tắt bài trước



1. ECMA Script

2. StyleSheet

→ Bài này sẽ học

1. Component

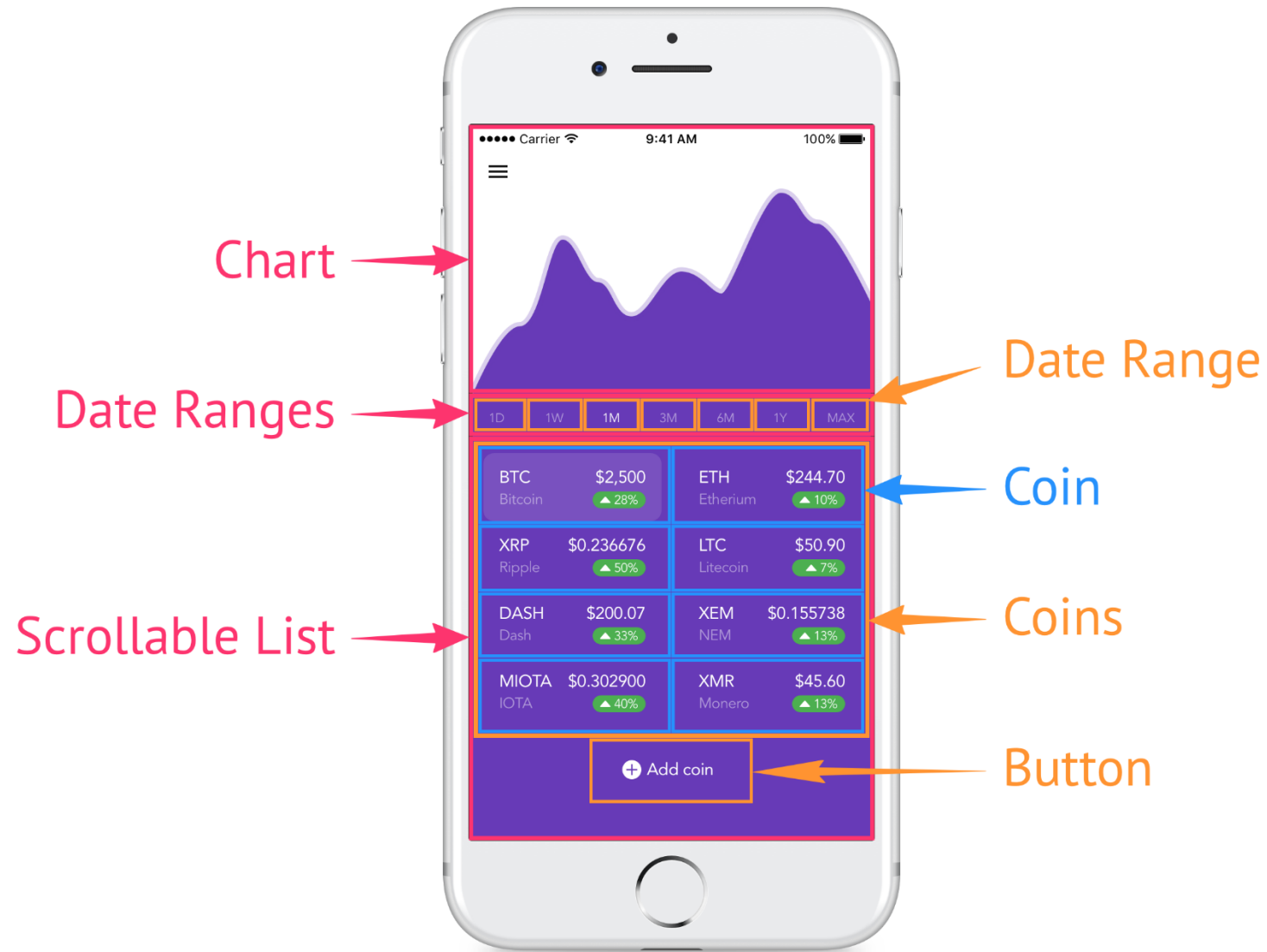
2. Vòng đời của Component

3. Các Component cơ bản

Component



- Ví dụ



- Component là một **thực thể độc lập**, không phụ thuộc lẫn nhau và có thể tái sử dụng.
- Component **mô tả một phần giao diện** của ứng dụng.
- Mỗi component sẽ **thực hiện các nhiệm vụ riêng** mà không cần quan tâm nó ảnh hưởng tới các thành phần khác như nào.
- *Import:*

```
import {Component} from 'react';
```

- Có 2 loại Component:
Function Component

```
export default function Hello() {  
  return (  
    <View>  
      <Text> Hello </Text>  
    </View>  
  );  
}
```

Class Component

```
import React, { Component } from 'react';  
import { View, Text, StyleSheet } from 'react-native';  
  
export default class Hello extends Component {  
  render() {  
    return (  
      <View>  
        <Text> Hello </Text>  
      </View>  
    );  
  }  
}
```

Vòng đời của component

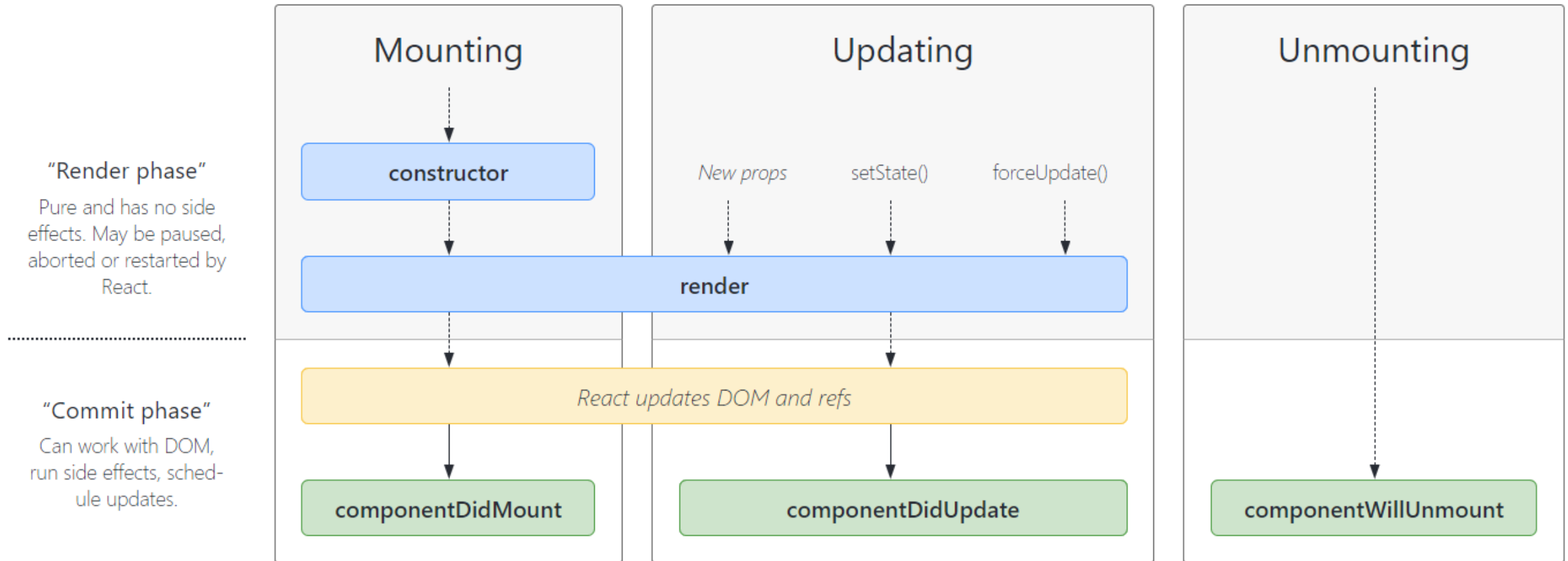


- **Mỗi Component có một vòng đời riêng** trong quá trình người dùng tương tác với ứng dụng (**Class Component**)
- Một vòng đời của Component gồm:
 - **Mounting**
 - **Updating**
 - **UnMounting**

- **Mounting**: Được gọi khi Component được **khởi tạo và thêm vào DOM**, gồm các phương thức **chính**:
 - **constructor()**: Phương thức khởi tạo
 - **render()**: Hiển thị giao diện của Component
 - **componentDidMount()**: Được gọi sau khi render

- **Updating**: Được gọi khi **Component** được **cập nhật**, gồm các phương thức **chính**:
 - **shouldComponentUpdate()**: Kiểm tra Component có cập nhật không
 - **render()**: Hiển thị lại giao diện của Component nếu **shouldComponentUpdate()** cho kết quả là **true**
 - **componentDidUpdate()**: Được gọi khi **render()** xảy ra

- **Unmounting:** Được gọi khi **Component bị xóa khỏi DOM**, gồm phương thức:
 - **componentWillUnmount():** Được gọi khi ứng dụng bị đóng hoàn toàn



- Xem thêm ở <https://reactjs.org/docs/react-component.html>

- Để áp dụng cho **Function Component**, người ta sẽ dùng **Hooks**
- Ví dụ: `useEffect()`, `useCallback()`, `useLayoutEffect()`...

1. Xây dựng một Class Component, tiến hành gọi các phương thức như hình và đưa ra nhận xét

```
class Hello extends Component {  
  constructor(){  
    super();  
    console.log('constructor')  
  }  
  componentDidMount(){  
    console.log('componentDidMount')  
  }  
  render() {  
    console.log('render')  
    return (  
      <View>  
        <Text> Hello </Text>  
      </View>  
    );  
  }  
}
```

Các Component cơ bản



- Ngoài **<View>** và **<Text>** đã học được ở bài trước, trong React

Native còn có những component khác như:

- **Button**
 - **Image**
 - **TextInput**
 - **FlatList**
 - ...
- Xem thêm ở: <https://reactnative.dev/docs/components-and-apis>

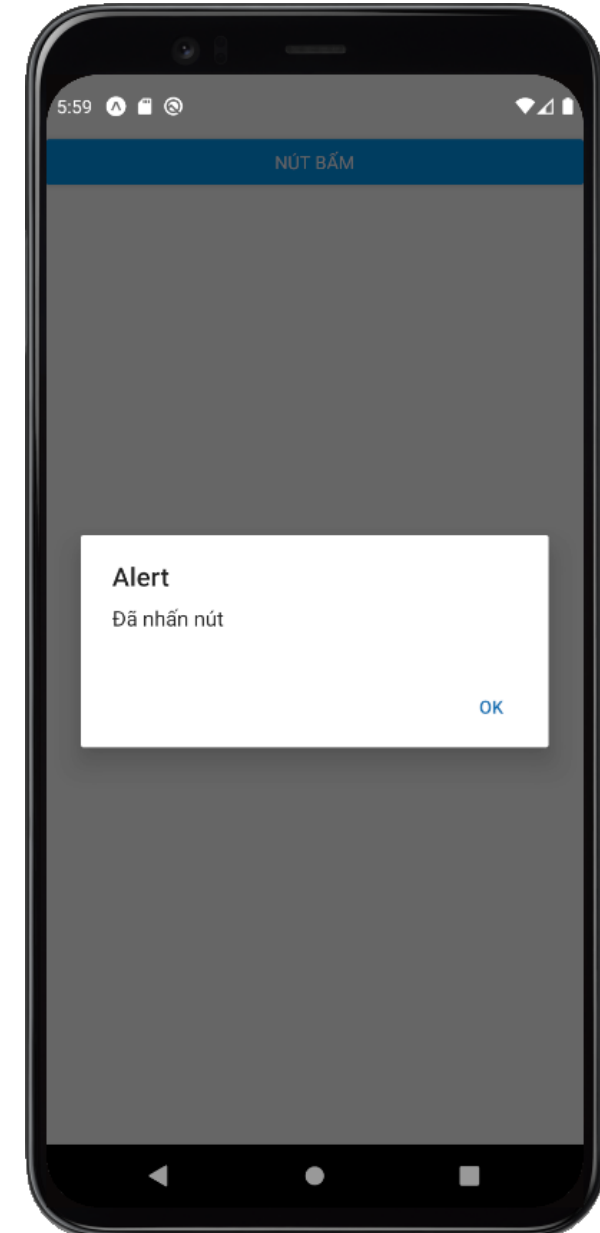
- Có hai loại: **Button** và **Touchable**
 - **Button**: Hiển thị nút bấm cơ bản
 - **TouchableOpacity**: Một dạng nút bấm custom, có thể chỉnh sửa theo ý mình,
 - Ngoài ra còn có **TouchableWithoutFeedback**, **TouchableHighlight**...
- Cú pháp khai báo:

```
import { Button, TouchableOpacity }  
      from 'react-native';
```

- Các thuộc tính bắt buộc của Button:
 - **title**: Chữ hiện trên Button
 - **onPress**: Gọi hàm khi người dùng nhấn vào Button
- Một số thuộc tính khác:
 - **color**: Màu của Button
 - **disabled**: Vô hiệu hóa Button

- Ví dụ

```
export default function App() {  
  return (  
    <View style={{marginTop: 50}}>  
      <Button  
        title='Nút bấm'  
        onPress={() => alert('Đã nhấn nút')}  
      />  
    </View>  
  );  
}
```



- Ví dụ

```
export default function App() {  
  return (  
    <View style={{marginTop: 50}}>  
      <TouchableOpacity onPress={() => alert('Đã nhấn')}>  
        <Text style={{fontSize:50}}>Touchable 1</Text>  
        <Text>Touchable 2</Text>  
      </TouchableOpacity>  
    </View>  
  );  
}
```

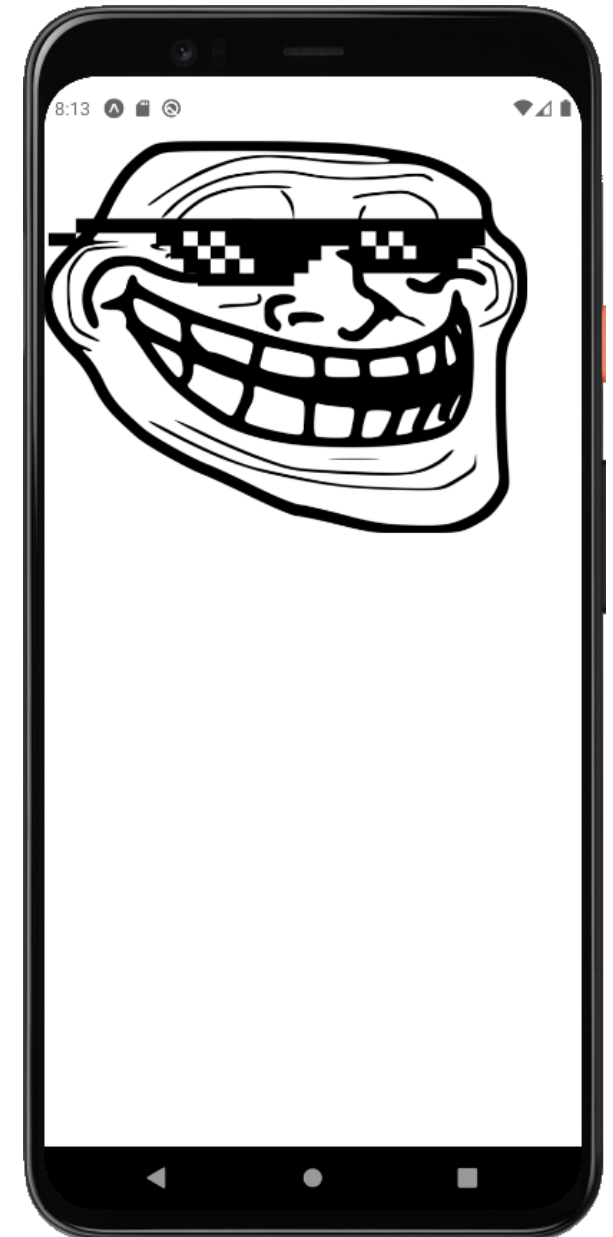


- **Image** dùng để hiển thị hình ảnh
- Thuộc tính cơ bản của **Image**:
 - **source**: Đường dẫn đến ảnh
 - **style**: Định dạng cho ảnh
- Cú pháp khai báo:

```
import { Image } from 'react-native';
```

- Ví dụ 1

```
export default function App() {  
  return (  
    <View style={{marginTop: 50}}>  
      <Image  
        style={{width: 370, height: 300}}  
        source={require('./image.png')}  
      />  
    </View>  
  );  
}
```



- Ví dụ 2

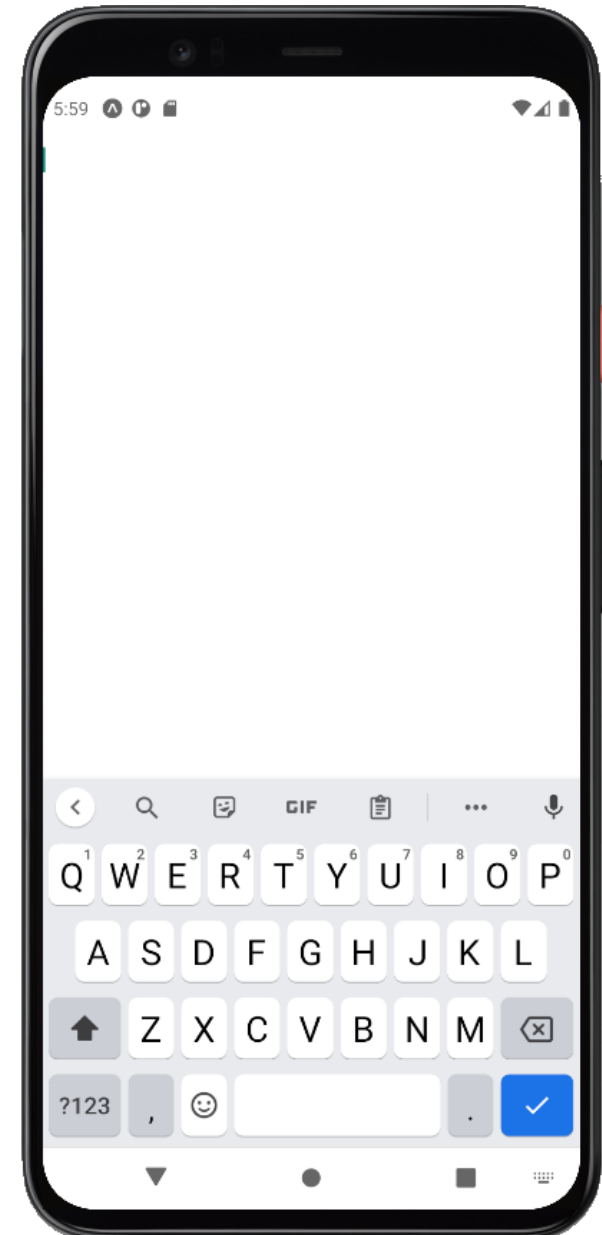
```
<Image
  source={{
    uri: 'https://reactnative.dev/img/tiny_logo.png',
  }}
/>
```


- **TextInput** dùng để hiển thị ô trống cho người dùng nhập vào
- Thuộc tính cơ bản của **TextInput**:
 - autoComplete: Tự động hoàn chỉnh chữ nhập vào
 - autoCapitalize: Tự động viết hoa ký tự đầu
 - style: Định dạng cho TextInput, **mặc định TextInput không có định dạng nên không nhìn thấy trên màn hình**
- Cú pháp khai báo:

```
import { TextInput } from 'react-native';
```

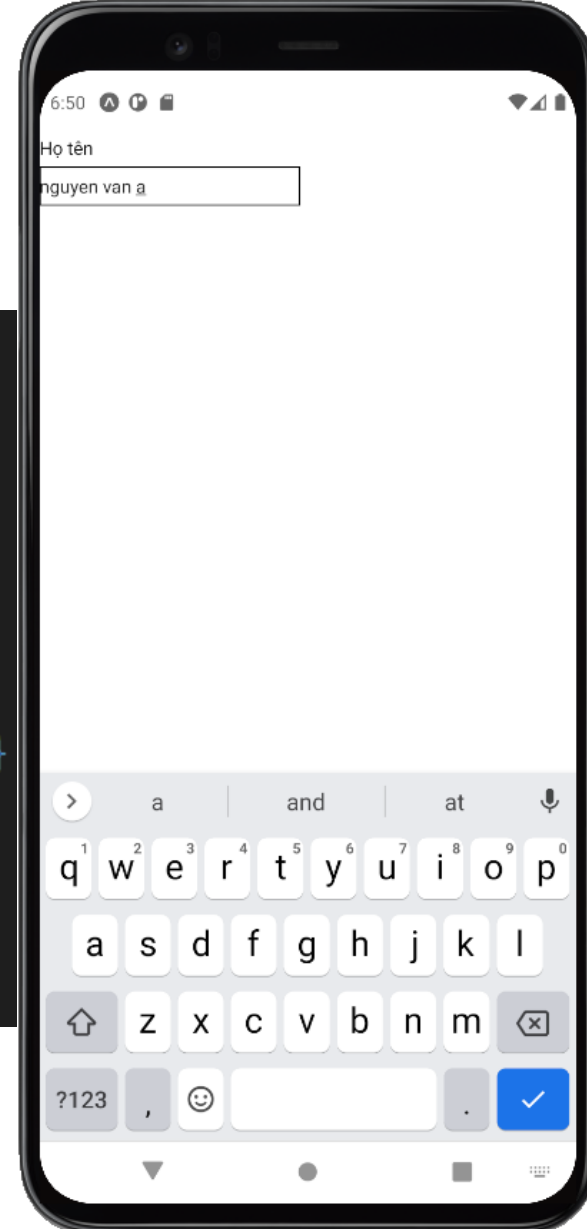
- Ví dụ 1

```
export default function App() {  
  return (  
    <View style={{marginTop: 50}}>  
      <TextInput  
        autoComplete={true}  
        autoCapitalize='none'  
      />  
    </View>  
  );  
}
```



- Ví dụ 2

```
export default function App() {  
  return (  
    <View style={{marginTop: 50}}>  
      <Text>Họ tên</Text>  
      <TextInput  
        autoComplete={true}  
        autoCapitalize='none'  
        style={{marginTop: 5, borderWidth: 1, borderColor: 'black', width: 200}}  
      />  
    </View>  
  );  
}
```

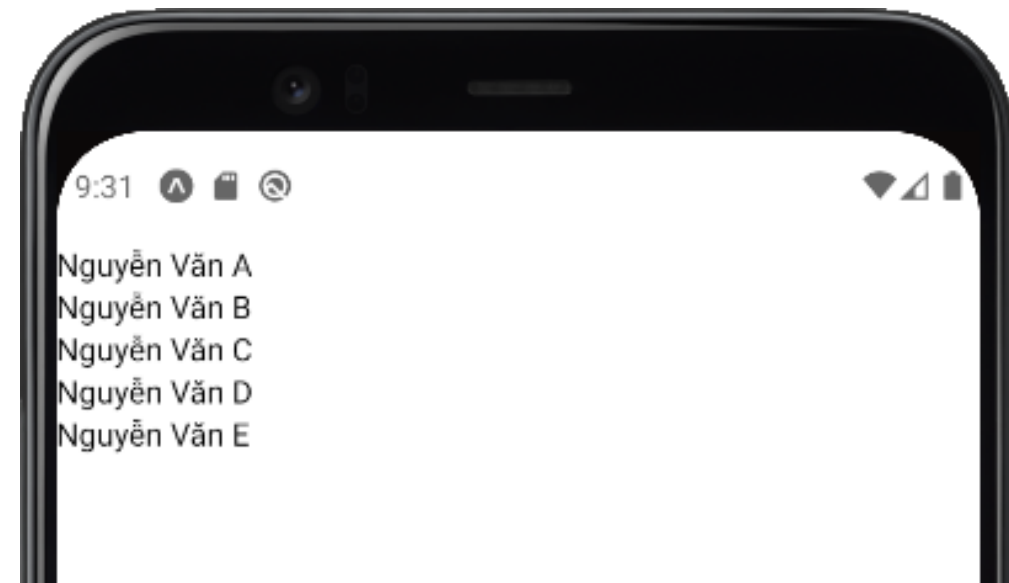


- **FlatList** dùng để hiển thị một danh sách
- Thuộc tính cơ bản của **FlatList** :
 - **data**: Dữ liệu dùng cho **FlatList**
 - **renderItem**: Trả về giao diện của từng **Item** trong **FlatList**
 - **keyExtractor**: Trích xuất khóa để theo dõi từng item
 - **style**: Định dạng cho ảnh
- Cú pháp khai báo:

```
import { FlatList } from 'react-native';
```

- Ví dụ

```
export default function App() {  
  const data = [  
    { ten: 'Nguyễn Văn A' },  
    { ten: 'Nguyễn Văn B' },  
    { ten: 'Nguyễn Văn C' },  
    { ten: 'Nguyễn Văn D' },  
    { ten: 'Nguyễn Văn E' },  
  ]  
  return (  
    <View style={{marginTop: 50}}>  
      <FlatList  
        data={data}  
        renderItem={({item})=><Text>{item.ten}</Text>}  
      />  
    </View>  
  );  
}
```



- Warning khi **thiếu key** cho mỗi item

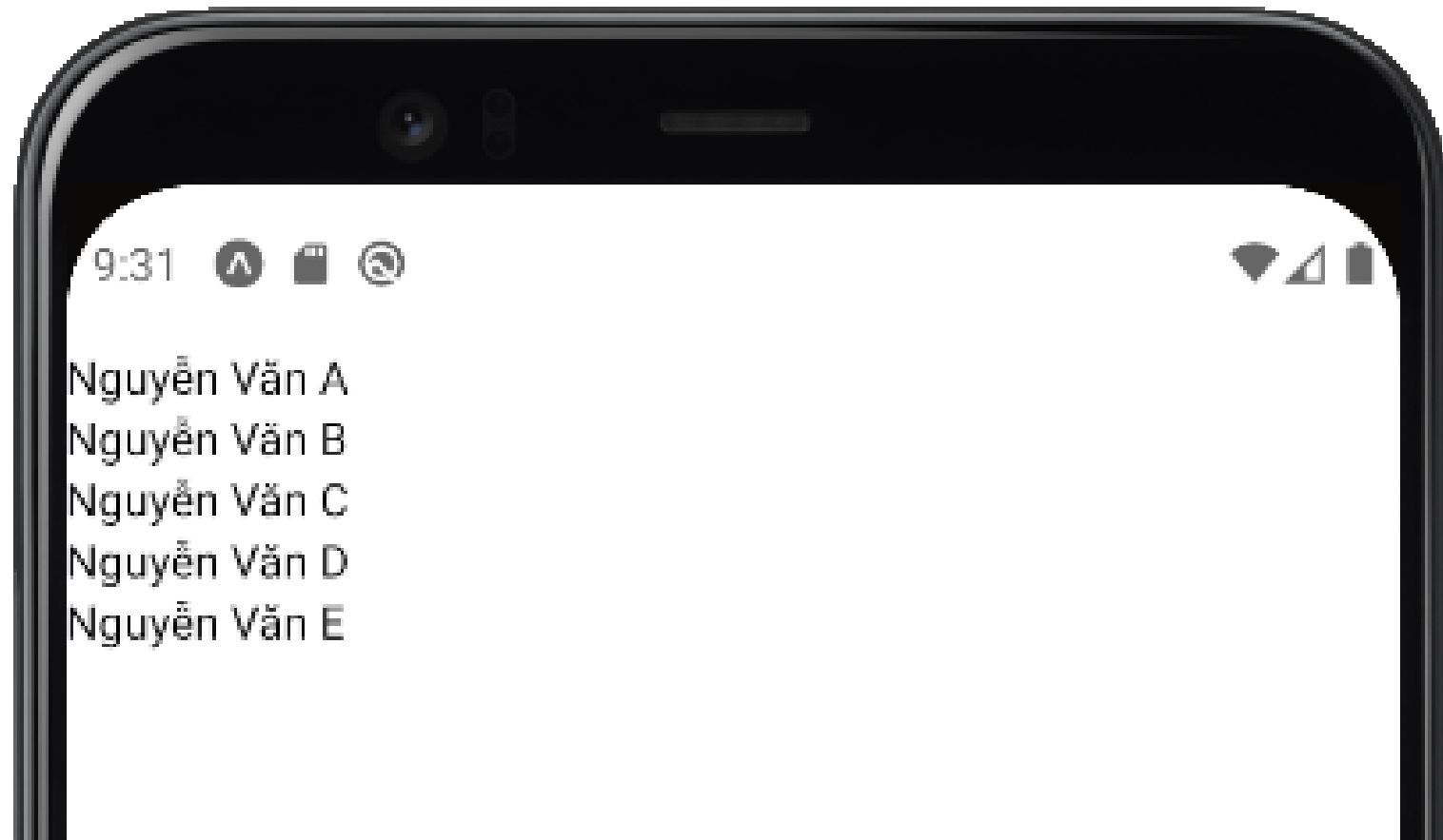
```
npm
Finished building JavaScript bundle in 61ms.
Running application on Android SDK built for x86.

VirtualizedList: missing keys for items, make sure to specify a key or id property on each item or provide a custom keyExtractor.,
- node_modules\react-native\Libraries\LogBox\LogBox.js:117:10 in registerWarning
- node_modules\react-native\Libraries\LogBox\LogBox.js:63:8 in warnImpl
- node_modules\react-native\Libraries\LogBox\LogBox.js:36:4 in console.warn
- node_modules\expo\build\environment\react-native-logs.fx.js:18:4 in warn
- node_modules\react-native\Libraries\Lists\VirtualizedList.js:1075:8 in render
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:12478:21 in finishClassComponent
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:12403:43 in updateClassComponent
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:19181:22 in beginWork$1
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:18085:22 in performUnitOfWork
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:18013:38 in workLoopSync
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:17977:18 in renderRootSync
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:17674:33 in performSyncWorkOnRoot
* [native code]:null in performSyncWorkOnRoot
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:5321:31 in runWithPriority$argument_1
- node_modules\scheduler\cjs\scheduler.development.js:653:23 in unstable_runWithPriority
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:5316:21 in flushSyncCallbackQueueImpl
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:5304:28 in flushSyncCallbackQueue
- node_modules\react-native\Libraries\Renderer\implementations\ReactNativeRenderer-dev.js:17125:30 in scheduleUpdateOnFi
```

- Sửa lại như sau

```
export default function App() {  
  const data = [  
    { id: '1', ten: 'Nguyễn Văn A' },  
    { id: '2', ten: 'Nguyễn Văn B' },  
    { id: '3', ten: 'Nguyễn Văn C' },  
    { id: '4', ten: 'Nguyễn Văn D' },  
    { id: '5', ten: 'Nguyễn Văn E' },  
  ]  
  return (  
    <View style={{marginTop: 50}}>  
      <FlatList  
        data={data}  
        renderItem={({item})=><Text>{item.ten}</Text>  
        keyExtractor={item => item.id}  
      />  
    </View>  
  );  
}
```

- Kết quả



Đề tài giữa kỳ



- Tìm hiểu một thư viện **React Native Component**
 - **Cách cài đặt và sử dụng thư viện**
 - **Demo một ứng dụng nhỏ sử dụng thư viện đó**
- Ví dụ: React base, React Native Elements, UI Kitten, React Native Material UI, React Native Material Kit, Nachos UI, React Native UI Library, React Native Paper, Teaset...
- Có thể tìm hiểu thêm các thư viện khác

Bài tập thực hành

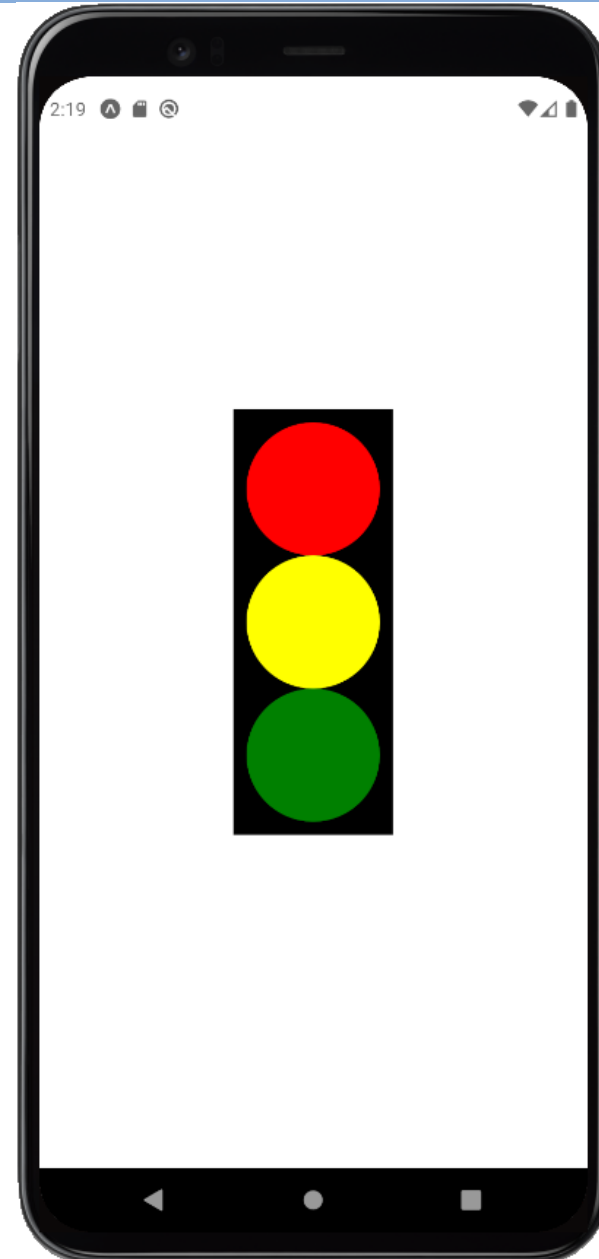


1. Xây dựng một Class Component, tiến hành gọi các phương thức như hình và đưa ra nhận xét

```
class Hello extends Component {
  constructor(){
    super();
    console.log('constructor')
  }
  componentDidMount(){
    console.log('componentDidMount')
  }
  render() {
    console.log('render')
    return (
      <View>
        <Text> Hello </Text>
      </View>
    );
  }
}
```

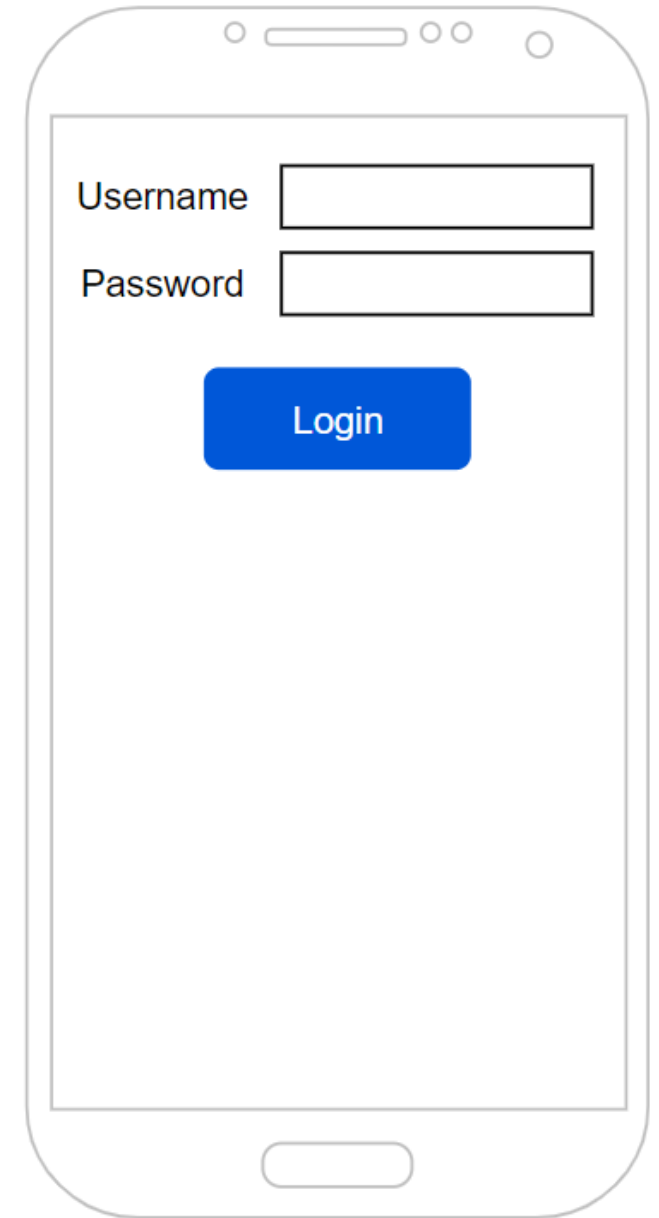
2*. Tạo giao diện như sau

Yêu cầu: Mỗi màu là một Component



3. Làm giao diện như hình

- Sử dụng **View**
- Sử dụng **Text**
- Sử dụng **TextInput**
- Sử dụng **Button** hoặc **Touchable**
- Khi nhấn vào Login sẽ thông báo đăng nhập thành công



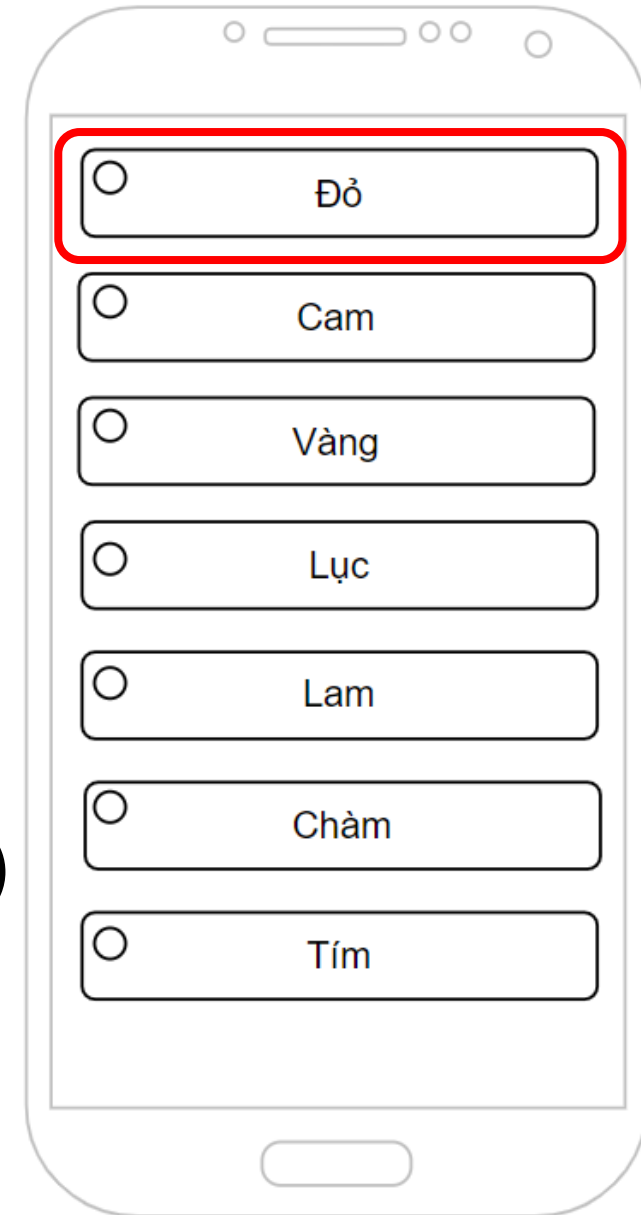
A mockup of a mobile application login screen. It features a white background with rounded corners, framed by a light gray border. At the top, there are three small circles representing status bar indicators. The main content area contains a 'Username' label followed by a text input field, a 'Password' label followed by another text input field, and a blue 'Login' button with white text. At the bottom, there is a light gray oval shape representing a home indicator.

4. Sử dụng **FlatList** làm giao diện như hình

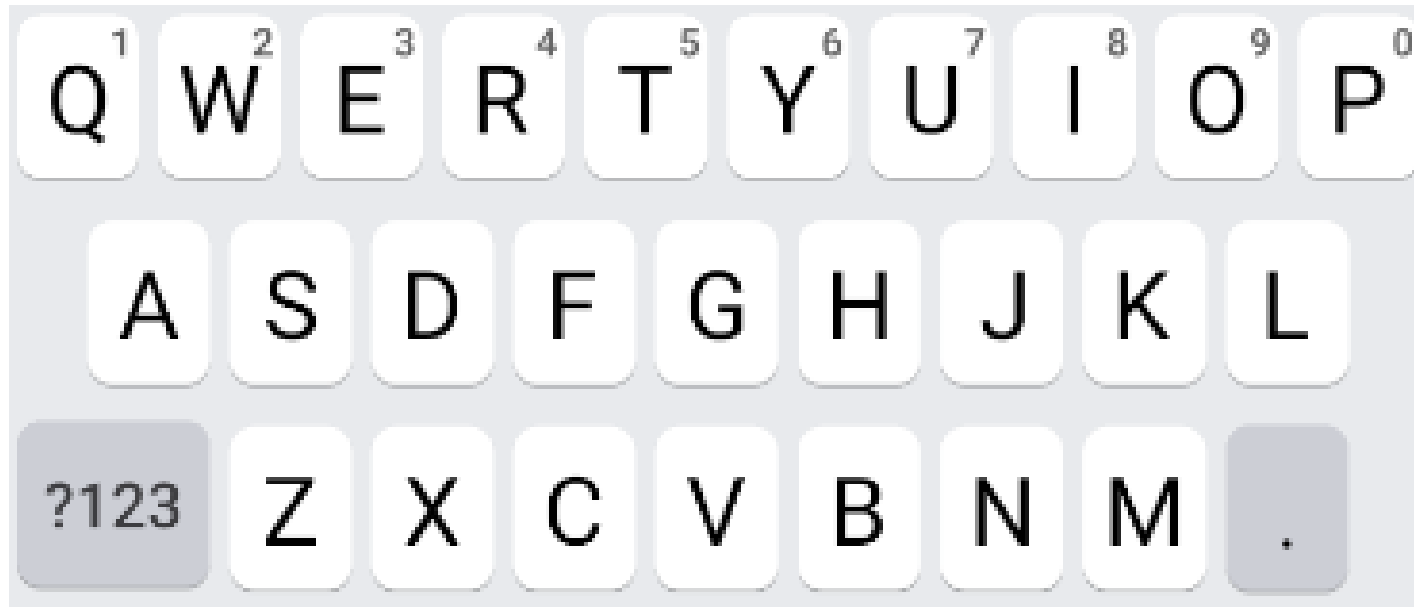
- Sử dụng **View**
- Sử dụng **Text**

Gợi ý: - Tạo mảng dữ liệu gồm màu, id
[`{id: '1', mau: 'Đỏ'}`,...]

- Đưa dữ liệu vào FlatList dùng thuộc tính **data**
- **renderItem** trả về **1 thẻ View** (như khoanh đỏ)



5*. Làm giao diện như hình. Yêu cầu: mỗi nút là một Component



- **Chụp hình, mô tả** quá trình làm và kết quả đạt được vào trong file word
- Lưu và đặt tên file word theo dạng: **MSSV.doc** hoặc **MSSV.docx**
- Nộp file word lên hệ thống **Canvas** theo tuần