

LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT

BÀI 8: XÂY DỰNG ỨNG DỤNG HOÀN CHỈNH

PHẦN 1: XÂY DỰNG CÁC THÀNH PHẦN GIAO DIỆN CƠ BẢN

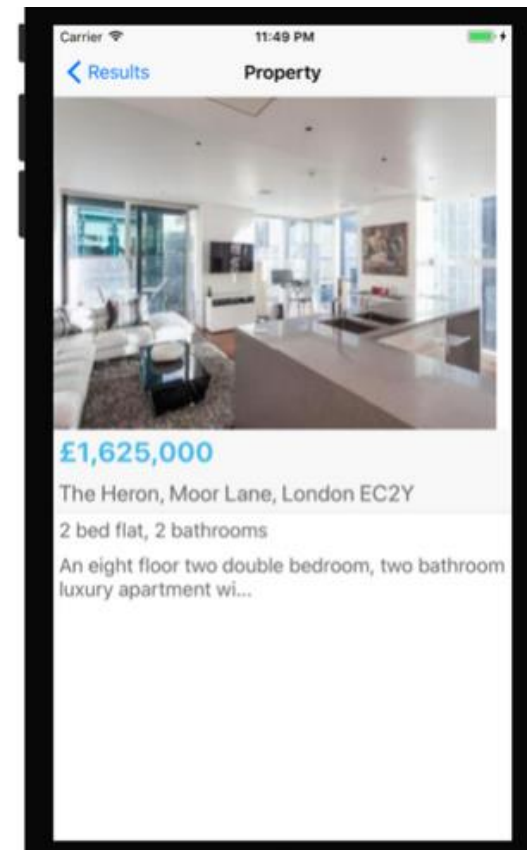
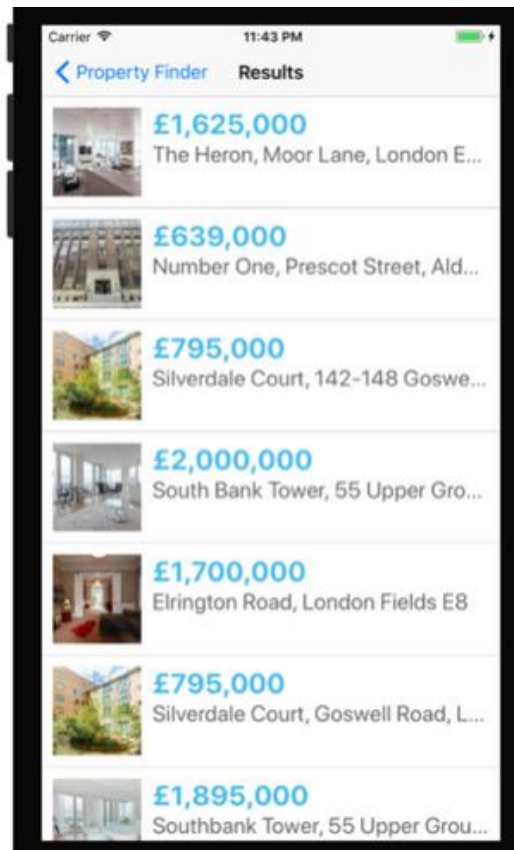
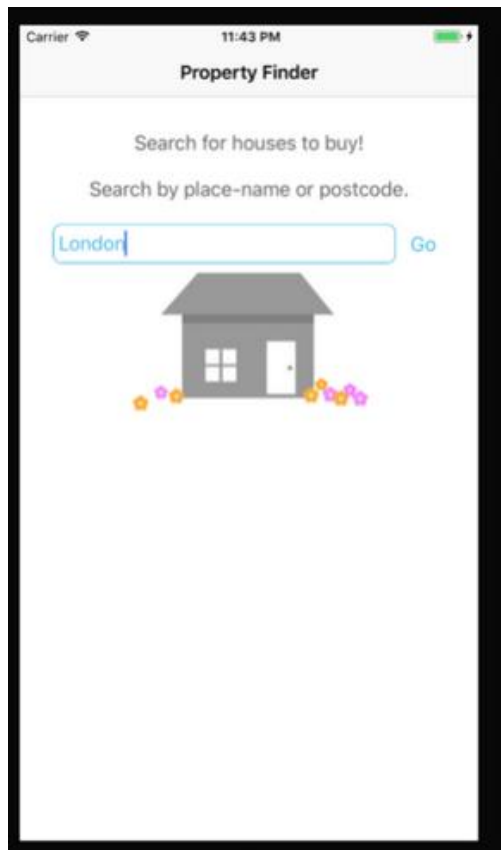
- ❑ Điểm qua quy trình xây dựng ứng dụng trên React Native
- ❑ Áp dụng các kiến thức đã học nhằm xây dựng ứng dụng React Native hoàn chỉnh



- ❑ **React Native:** Sử dụng JavaScript để viết code nhưng giao diện (UI) lại hoàn toàn Native
- ❑ Nó không có những nhược điểm thường liên quan đến một ứng dụng HTML5 lai.

- ❑ React giới thiệu một cách tiếp cận mới, cấp tiến và có chức năng cao để xây dựng các giao diện người dùng.
- ❑ UI trong React chỉ đơn giản là một chức năng của trạng thái ứng dụng hiện tại.

□ Giao diện của ứng dụng:



- ❑ Cài đặt NodeJS: Sử dụng <https://brew.sh/>
 - ❖ `brew install node`
- ❑ Sử dụng homebrew để install watchman
 - ❖ `brew install watchman`
- ❑ Sử dụng npm để cài đặt React Native Command Line Interface (CLI) tool:
 - ❖ `npm install -g react-native-cli`

NHẮC LẠI VỀ CÀI ĐẶT BẰNG TERMINAL WINDOW

- ❑ Truy nhập vào folder mà bạn muốn xây dựng ứng dụng, sau đó chạy câu lệnh sau trong Terminal
 - ❖ `react-native init TenProject`
- ❑ Truy nhập vào thư mục ứng dụng
 - ❖ `cd TenProject`

- ❑ ***node_modules*** là folder chứa React Native framework
- ❑ ***index.android.js*** là ứng dụng gốc được tạo bởi CLI tool
- ❑ ***android*** là một folder chứa dự án trên Android Studio, yêu cầu có ***bootstrap*** trong ứng dụng

- ❑ Bạn mở ***index.android.js*** và thực hiện xây dựng ứng dụng theo các bước sau:
- ❖ Import các module được yêu cầu
 - ❖ Định nghĩa các thành phần present trong UI
 - ❖ Tạo kiểu (style) cho đối tượng và viết phần điều khiển việc hiển thị layout
 - ❖ Đăng ký các thành phần với ứng dụng

- ❑ Đoạn code import này sẽ load module react và gán cho một biến có tên là React

```
1 import React, { Component } from 'react';
```

- ❑ Đoạn code export sau định nghĩa một lớp mở rộng của React Component. Export lớp này sẽ cho phép sử dụng public ở các file khác

```
export default class PropertyFinder extends Component
```

- ❑ Trong `index.android.js`, thêm phần sau vào đầu file, ngay trước các câu lệnh `import`:
- ❑ `'use strict';`
- ❑ Strict Mode: cho phép xử lý lỗi, vô hiệu hóa một số tính năng chưa lý tưởng trong JavaScript. Nó làm cho việc lập trình trên JavaScript trở nên tốt hơn!

- ❑ Bên trong lớp TenProject, ta thay thế render () n hư sau:

```
render() {  
  return React.createElement(Text, {style: styles.description},  
}
```

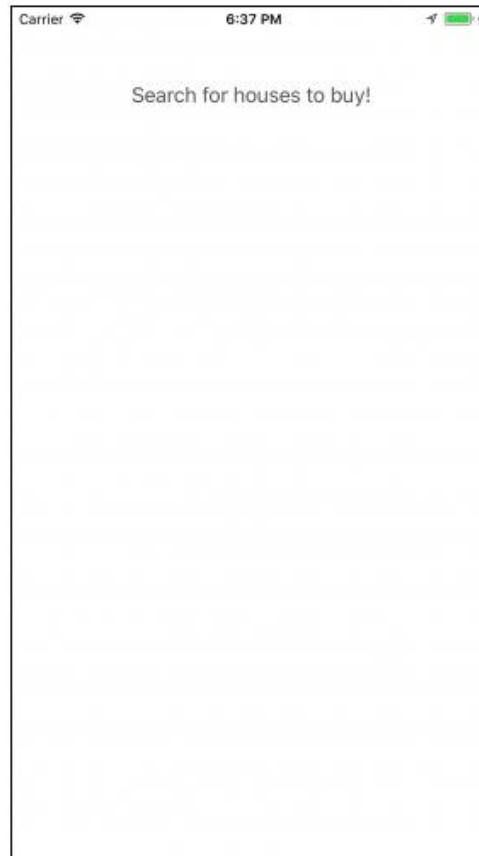
- ❑ TenProject exrtend từ React.Component, khối xây dựng cơ bản của giao diện React.
- ❑ Các thành phần chứa các thuộc tính static, các biến trạng thái, các phương thức vẽ.
- ❑ Ứng dụng hiện tại khá đơn giản và chỉ cần một phương thức render.

- ❑ Bước tiếp theo ta thay thế **const styles** bằng đoạn code sau:

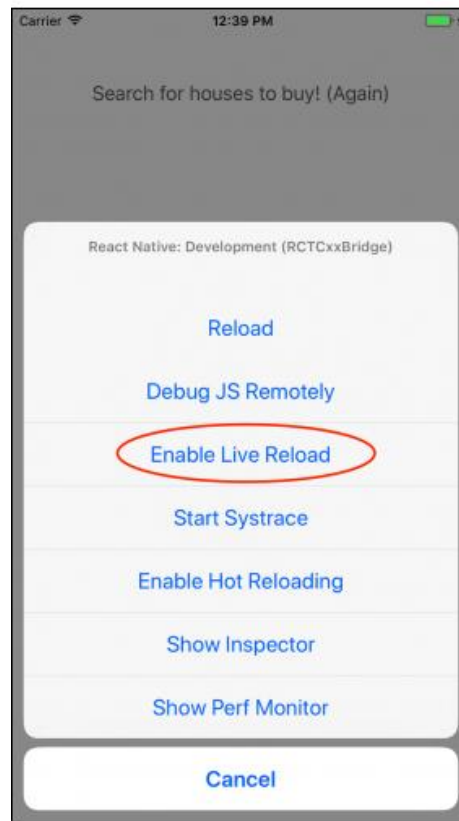
```
1  const styles = StyleSheet.create({  
2    description: {  
3      fontSize: 18,  
4      textAlign: 'center',  
5      color: '#656565',  
6      marginTop: 65,  
7    },  
8  });
```

- ❑ Điều này xác định một style đơn lẻ mà bạn đã áp dụng cho phần văn bản được mô tả

□ Khi chạy chương trình sẽ có kết quả sau:



❑ Chú ý: Để refresh app, chúng ta nhấn ***Cmd+D*** sau đó chọn ***Enable Live Reload***

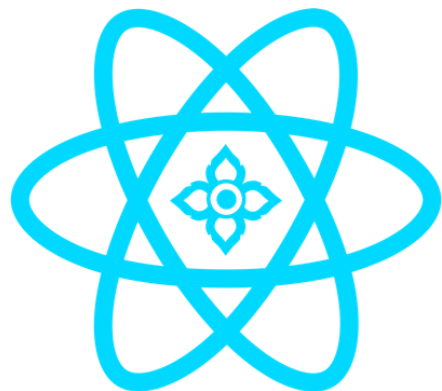


□ Ta có thể định dạng chuỗi trực tiếp (inline) như sau:

```
return <Text style={styles.description}>Search for houses to buy!</Text>;
```


□ Ta định nghĩa thành phần **SearchPage** như sau:

```
class PropertyFinder extends Component {  
  render() {  
    return (  
      <NavigatorAndroid  
        style={styles.container}  
        initialRoute={{  
          title: 'Property Finder',  
          component: SearchPage,  
        }}/>  
    );  
  }  
}
```

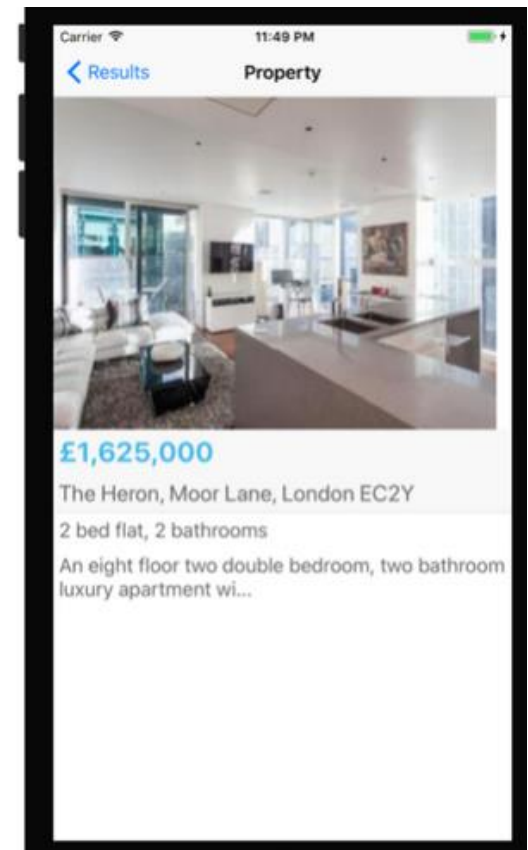
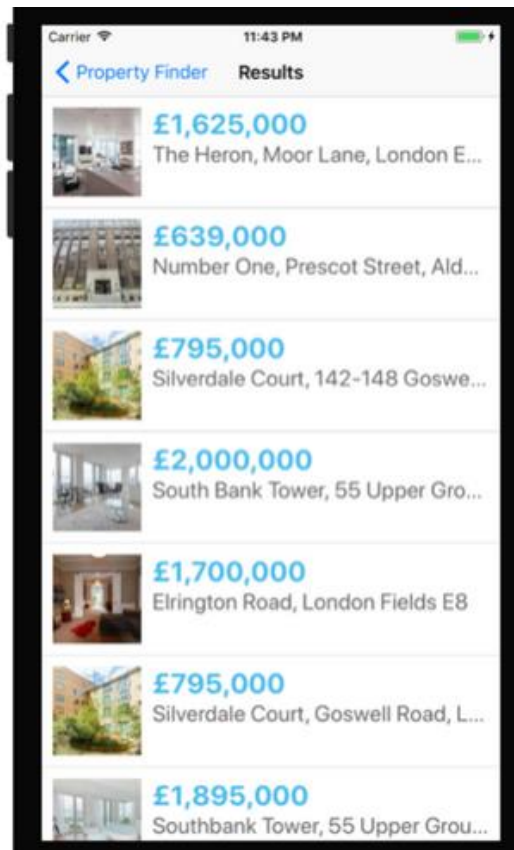
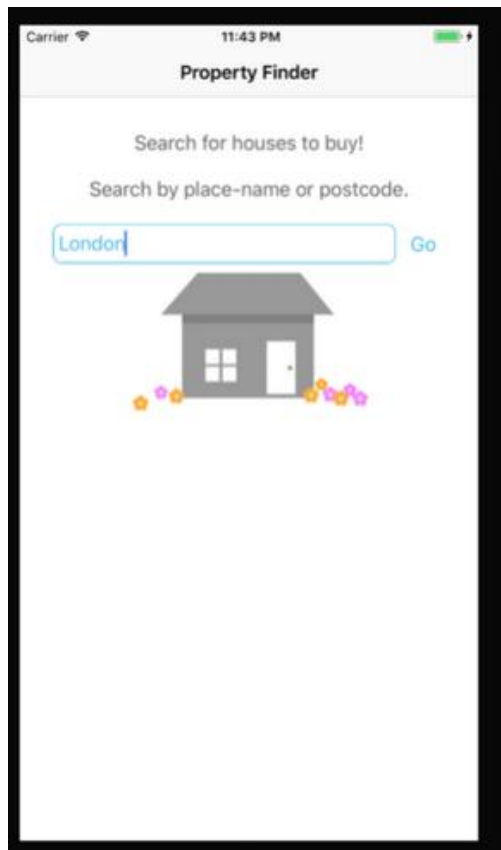


LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT

BÀI 8: XÂY DỰNG ỨNG DỤNG HOÀN CHỈNH

PHẦN 2: XÂY DỰNG CHỨC NĂNG CHÍNH CỦA HỆ THỐNG

□ Giao diện của ứng dụng:



□ Ta định nghĩa thành phần **SearchPage** như sau:

```
class PropertyFinder extends Component {  
  render() {  
    return (  
      <NavigatorAndroid  
        style={styles.container}  
        initialRoute={{  
          title: 'Property Finder',  
          component: SearchPage,  
        }}/>  
    );  
  }  
}
```

❑ Import các thành phần cần sử dụng

```
1  'use strict';  
2  
3  import React, { Component } from 'react';  
4  import {  
5    StyleSheet,  
6    Text,  
7    TextInput,  
8    View,  
9    Button,  
10   ActivityIndicator,  
11   Image,  
12 } from 'react-native';
```

❑ Class SearchPage.js

```
13 export default class SearchPage extends Component {  
14   render() {  
15     return (  
16       <View style={styles.container}>  
17         <Text style={styles.description}>  
18           Search for houses to buy!  
19         </Text>  
20         <Text style={styles.description}>  
21           Search by place-name or postcode.  
22         </Text>  
23       </View>  
24     );  
25   }  
26 }
```

□ Tiếp theo ở cuối file ta thêm đoạn code định

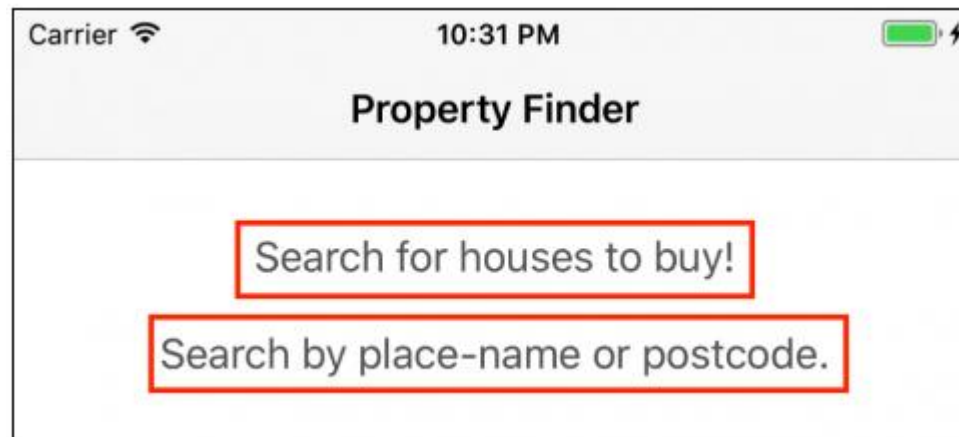
```
28 const styles = StyleSheet.create({  
29   description: {  
30     marginBottom: 20,  
31     fontSize: 18,  
32     textAlign: 'center',  
33     color: '#656565'  
34   },  
35   container: {  
36     padding: 30,  
37     marginTop: 65,  
38     alignItems: 'center'  
39   },  
40 });
```

- ❑ Để sử dụng trang vừa tạo, bạn cần import vào index.android.js như sau:

```
import SearchPage from './SearchPage';|
```

- ❑ Import SearchPage từ tệp bạn vừa tạo - sẽ tạm thời phá vỡ ứng dụng của bạn!
- ❑ Điều này là do SearchPage bây giờ được khai báo hai lần - một lần trong SearchPage.js, và một lần nữa trong tệp hiện tại.
- ❑ Bây giờ Xóa lớp SearchPage và kiểu mô tả liên quan của nó khỏi index.android.js thì chương trình không báo lỗi

❑ Ta sẽ cần định dạng chương trình trên như sau:



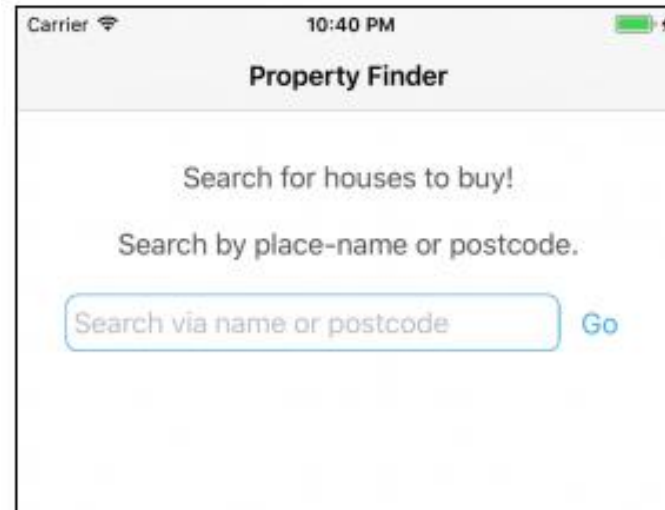
- ❑ Muốn vậy ta mở ***SearchPage.js*** và thêm đoạn code sau closing tag của thành phần second Text

```
1 <View style={styles.flowRight}>
2   <TextInput
3     style={styles.searchInput}
4     placeholder='Search via name or postcode' />
5   <Button
6     onPress={() => {}}
7     color='#48BBEC'
8     title='Go'
9   />
10 </View>
```

□ Tiếp theo ta định nghĩa Styling cho flowRight và searchInput như sau:

```
11 flowRight: {  
12   flexDirection: 'row',  
13   alignItems: 'center',  
14   alignSelf: 'stretch',  
15 },  
16 searchInput: {  
17   height: 36,  
18   padding: 4,  
19   marginRight: 5,  
20   flexGrow: 1,  
21   fontSize: 18,  
22   borderWidth: 1,  
23   borderColor: '#48BBEC',  
24   borderRadius: 8,  
25   color: '#48BBEC',  
26 },
```

- ❑ Lưu project và chạy chương trình ta sẽ có kết quả:



❑ Sử dụng tài nguyên được cung cấp và import

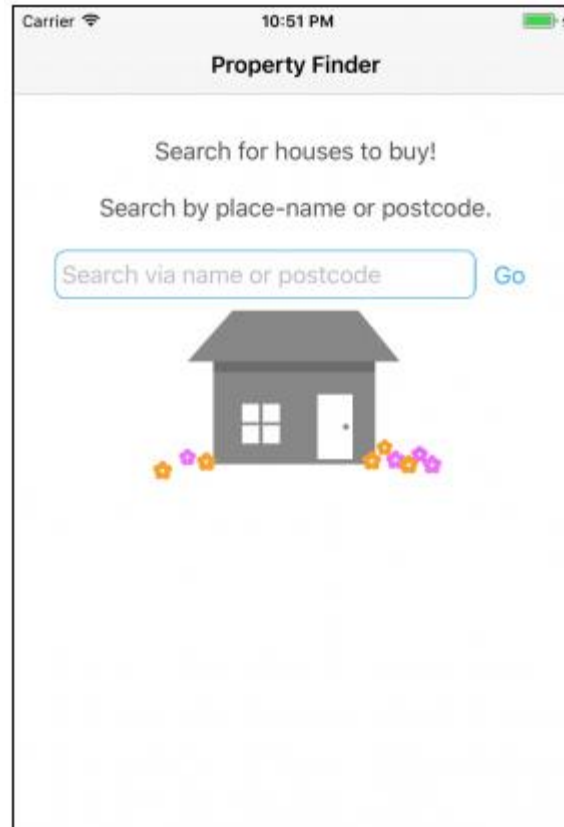
```
<Image source={require('./Resources/house.png')} style={styles.image}/>
```

❑

```
1 image: {  
2   width: 217,  
3   height: 138,  
4 },|
```

như sau:

❑ Chạy chương trình cho kết quả sau:



- ❑ React Native có thể quản lý trạng thái nội bộ của nó thông qua một đối tượng được gọi thông qua trạng thái.
- ❑ Bất cứ khi nào thay đổi trạng thái của một thành phần, `render ()` sẽ được gọi.
- ❑ Trong file `App.js`, thêm đoạn code sau trước

```
1 constructor(props) {  
2   super(props);  
3   this.state = {  
4     searchString: 'london'  
5   };  
6 }
```

❑ Trong hàm render, ta thay đổi TextInput như sau:

```
1 <TextInput
2   style={styles.searchInput}
3   value={this.state.searchString}
4   placeholder='Search via name or postcode' />
```


- ❑ Trong ***SearchPage.js*** ta thêm phương thức khởi tạo:

```
1  this.state = {  
2    searchString: 'london',  
3    isLoading: false,  
4  };|
```

- ❑ Xây dựng các phương thức trong SearchPage như sau:

□ Lấy dữ liệu:

```

1 fetch(query)
2   .then(response => response.json())
3   .then(json => this._handleResponse(json.response))
4   .catch(error =>
5     this.setState({
6       isLoading: false,
7       message: 'Something bad happened ' + error
8     }));
  
```

```

_handleResponse = (response) => {
  this.setState({ isLoading: false , message: '' });
  if (response.application_response_code.substr(0, 1) === '1') {
    console.log('Properties found: ' + response.listings.length);
  } else {
    this.setState({ message: 'Location not recognized; please try again.'});
  }
};
  
```

- ❑ Xây dựng các phương thức trong SearchPage như sau:

```
1 _executeQuery = (query) => {  
2   console.log(query);  
3   this.setState({ isLoading: true });  
4 };  
5  
6 _onSearchPressed = () => {  
7   const query = urlForQueryAndPage('place_name', this.state.searchString, 1);  
8   this._executeQuery(query);  
9 };
```

- ❑ `_executeQuery ()` cuối cùng sẽ chạy truy vấn, nhưng bây giờ nó chỉ đơn giản là ghi một tin nhắn đến bàn điều khiển và bộ `isLoading` thích hợp để UI có thể hiển thị trạng thái mới.
- ❑ `_onSearchPressed ()` cấu hình và khởi tạo truy vấn tìm kiếm. Điều này sẽ khởi động khi nút Go được nhấn.
- ❑ Để hoàn thành việc này, chúng ta cần thay đổi `onPress={this._onSearchPressed}` với phương thức hiển thị và thay thế phương thức `onPress` cho nút Go như sau:

□ Cuối cùng thêm hàm sau vào SearchPage

```
1 function urlForQueryAndPage(key, value, pageNumber) {  
2   const data = {  
3     country: 'uk',  
4     pretty: '1',  
5     encoding: 'json',  
6     listing_type: 'buy',  
7     action: 'search_listings',  
8     page: pageNumber,  
9   };  
10  data[key] = value;  
11  
12  const querystring = Object.keys(data)  
13    .map(key => key + '=' + encodeURIComponent(data[key]))  
14    .join('&');  
15  
16  return 'https://api.nestoria.co.uk/api?' + querystring;  
17 }
```

□ Tạo file mới tên là ***SearchResults.js***

```
1  'use strict';
2
3  import React, { Component } from 'react'
4  import {
5    StyleSheet,
6    Image,
7    View,
8    TouchableHighlight,
9    FlatList,
10   Text,
11  } from 'react-native';
```

```

12 export default class SearchResults extends Component {
13   _keyExtractor = (item, index) => index;
14
15   _renderItem = ({item}) => {
16     return (
17       <TouchableHighlight
18         underlayColor='#dddddd'>
19         <View>
20           <Text>{item.title}</Text>
21         </View>
22       </TouchableHighlight>
23     );
24
25   };
26
27   render() {
28     return (
29       <FlatList
30         data={this.props.listings}
31         keyExtractor={this._keyExtractor}
32         renderItem={this._renderItem}
33       />
34     );
35   }
36 }

```

□ Vẫn trong *SearchResults.js* ta viết code định dạng

```

1  const styles = StyleSheet.create({
2    thumb: {
3      width: 80,
4      height: 80,
5      marginRight: 10
6    },
7    textContainer: {
8      flex: 1
9    },
10   separator: {
11     height: 1,
12     backgroundColor: '#dddddd'
13   },
14   price: {
15     fontSize: 25,
16     fontWeight: 'bold',
17     color: '#48BBEC'
18   },
19   title: {
20     fontSize: 20,
21     color: '#656565'
22   },
23   rowContainer: {
24     flexDirection: 'row',
25     padding: 10
26   },
27 });
  
```


□ Đăng ký component, ta dùng cú pháp sau:

```
AppRegistry.registerComponent('PropertyFinder', () => PropertyFinder);
```

□ Định nghĩa style, ta dùng cú pháp sau:

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  },
  welcome: {
    fontSize: 20,
    textAlign: 'center',
    margin: 10,
  },
  instructions: {
    textAlign: 'center',
    color: '#333333',
    marginBottom: 5,
  },
});
```








❑ Khai báo styling inline:

```
46 <Text style={styles.instructions}>  
47   To get started, edit index.android.js  
48 </Text>
```

```
❑ import {  
  AppRegistry,  
  StyleSheet,  
  Text,  
  View  
} from 'react-native';
```

- ❑ **Chú ý:** Các vấn đề sau sẽ xảy ra ở phía server khi người dùng chạy chương trình:
- ❑ Khi server ràng buộc thành công tới một port, nodejs sẽ tiến hành listening
- ❑ Sau khi nhận kết nối, một connection sẽ được mở ra
- ❑ Nếu tắt chương trình, nodejs sẽ gọi phương thức close
- ❑ Để xác định kết nối, ta cần lấy địa chỉ ip bằng phương thức `net.Server.GetIpAddress()`
- ❑ Nếu lỗi, `Net.Server` sinh ra sự kiện error và thông báo về client

KẾT QUẢ CUỐI CÙNG SAU TÌM KIẾM

| | | |
|---|-------------------|---------------------------------------|
| Carrier | 10:02 PM | |
| < Property Finder Results | | |
|  | £5,250,000 | Duncan Terrace, London N1 - En... |
|  | £925,000 | Little Britain, London EC1A - Balc... |
|  | £1,100,000 | Worcester Point, Central Street, L... |
|  | £595,000 | Shepherdess Walk, London N1 - ... |
|  | £695,000 | The Cloisters, 145 Commercial S... |
|  | £725,000 | Grosvenor Avenue, London N5 - ... |
|  | £750,000 | Southgate Road, London N1 - Flat |

- ❑ Điểm qua quy trình xây dựng ứng dụng trên React Native
- ❑ Áp dụng các kiến thức đã học nhằm xây dựng ứng dụng React Native hoàn chỉnh





Cảm ơn