SCROLLVIEW
- FLATLIST - SECTIONLIST

ScrollView

- ScrollView là một component có khả năng cuộn để thể hiển thị các component bên trong nó khi kích thước của component vượt quá kích thước của màn hình
- Thường sẽ ít sử dụng hơn so với FlatList vì lúc nào cũng sẽ render là toàn bộ dữ liệu, điều này sẽ gây ảnh hưởng đến hiệu năng
- ScrollView cũng là 1 thẻ bố cục
- Cách sử dụng: dùng ScrollView để bao bọc bên ngoài 1 component có nội dung dài
- Ví dụ:

```
<ScrollView>
    <Text>Nội dung rất dài</Text>
</ScrollView>
```

ScrollView

```
import { View, Text, ScrollView, StyleSheet } from 'react-native'
import React from 'react'
export default function DemoScrollView() {
  return (
    <View style={styles.container}>
    <ScrollView style={styles.scrollView}>
    <Text style={styles.text}>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
      eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
      minim veniam, quis nostrud exercitation ullamco laboris nisi ut
      aliquip ex ea commodo consequat. Duis aute irure dolor in
      reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
      pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
      culpa qui officia deserunt mollit anim id est laborum.
    </Text>
  </ScrollView>
  </View>
  )
}
```

ScrollView

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    paddingTop: 10,
  },
  scrollView: {
    backgroundColor: 'pink',
    marginHorizontal: 20,
  },
  text: {
    fontSize: 42,
  },
  });
```

- Muốn scroll theo chiều ngang: khai báo thuộc tính horizontal của ScrollView là true

FlatList

- Cũng giống như ScrollView thì FlastList một component có khả năng cuộn để thể hiển thị các component bên trong nó khi kích thước của component vượt quá kích thước của màn hình
- Nhưng thay vì chỉ render ra toàn bộ data bên trong thì FlatList hỗ trợ thêm một số tính năng khác nhằm tối ưu hiệu năng hơn so với ScrollView
- Các thuộc tính:
  - data: mảng dữ liệu nguồn cho FlastList
  - renderItem: return giao diện của từng phần tử
  - keyExtractor: return key cho từng phần tử (ít dùng)
  - numColumns: số cột trình bày (vd: numColumns=2) (ít dùng)

FlatList

- Hàm của tham số renderItem:
  - Có 1 tham số đại diện cho từng phần tử trong mảng (ví dụ: tham số tên data)
  - Truy xuất nội dung của tham số: tên_tham_số.item (vd: data.item)
  - Thân của hàm return các component giao diện của từng phần tử
  - Ví dụ khai báo hàm:

```
const renderItem = (data) => (
  <View style={styles.item}>
    <Text style={styles.title}>{data.item.title}</Text>
  </View>
);
```

  - Với: tittle là tên key dữ liệu trong mảng

=> FlatList ⇔ map + ScrollView

FlatList

Ví dụ:

- Mảng dữ liệu

```
const DATA = [
        {
          id: 'bd7acbea-c1b1-46c2-aed5-3ad53abb28ba',
          title: 'First Item',
        },
        {
          id: '3ac68afc-c605-48d3-a4f8-fbd91aa97f63',
          title: 'Second Item',
        },
]
```

- Bước 2: Khai báo hàm để return về giao diện mong muốn của từng phần tử

```
 const renderItem = (data) => (
        <View style={styles.item}>
          <Text style={styles.title}>{data.item.title}</Text>
        </View>
      )
```

- Bước 3: Khai báo FlatList:

```
    <View style={styles.container}>
      <FlatList
        data={DATA}
        renderItem={(item)=>(renderItem(item))}
      />
    </View>
```

SectionList

- Giống như FlatList, nhưng sẽ có sự phân cấp dạng: list -> sublist
- Sublist phải có tên là data
- Dữ liệu dạng:

```
const DATA = [
  {
    title: "Main dishes",
    data: ["Pizza", "Burger", "Risotto"]
  },
  {
    title: "Sides",
    data: ["French Fries", "Onion Rings", "Fried Shrimps"]
  },
.....
```

SectionList

- Giao diện thường gồm 2 phần:
  - Header: (Main dishes, Sides, ...
  - Item: Pizza, Burger, French Fries, ...

SectionList

- Các thuộc tính:
  - sections: mảng dữ liệu nguồn
  - renderSectionHeader: hàm return giao diện cho phần Header, có 1 tham số đại diện cho từng phần tử trong Header. Truy xuất nội dung của tham số: tên_tham_số.section
  - renderItem: hàm return giao diện cho phần Item, có 1 tham số đại diện cho từng phần tử trong sublist. Truy xuất nội dung của tham số: tên_tham_số.item

SectionList

- Ví dụ:

```
const listItem=[
  {
    title:'Header 1',
    data:[
      {id:'1.1', content:'Item 1.1'},
      {id:'1.2', content:'Item 1.2'},
      {id:'1.3', content:'Item 1.3'}
    ]
  },
  {
    title:'Header 2',
    data:[
      {id:'2.1', content:'Item 2.1'},
      {id:'2.2', content:'Item 2.2'},
      {id:'2.3', content:'Item 2.3'}
    ]
  }, ......
```

SectionList

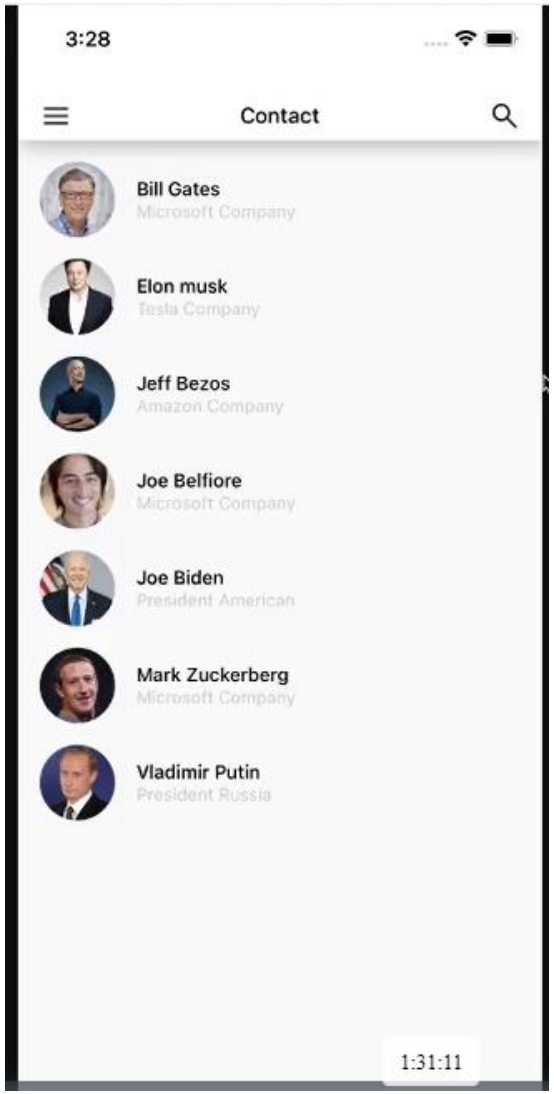
- Ví dụ:

```
const _renderHearder=(data)=>{
  return <Text>{data.section.title}</Text>
}
const _renderItem=(data)=>{
  return <Text>{data.item.content}</Text>
}
return (
  <View>
    <SectionList
      sections={listItem}
      renderSectionHeader={(data)=>_renderHearder(data)}
      renderItem={(data)=>_renderItem(data)}
      >
    </SectionList>
  </View>
)
```

Excercise 1

Khai báo mảng [1, 2, ..., 15]
Render như hình, màu nền ngẫu nhiên

Excercise 2

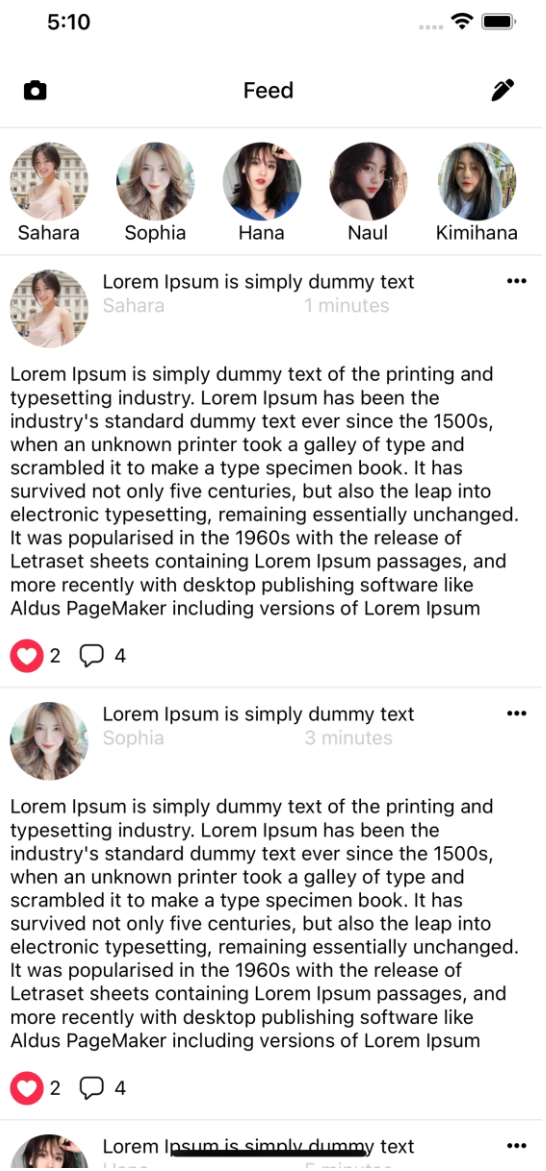



```
datas = [
{       title:"Bill Gates",       desc:"Microsoft Company",
image:require("../../assets/images/billgates.jpeg")    },
 {       title:"Elon musk",       desc:"Tesla Company",
image:require("../../assets/images/elon_musk.jpeg")    },
 {       title:"Jeff Bezos",       desc:"Amazon Company",
image:require("../../assets/images/jeff.jpeg")    },
 {       title:"Joe Belfiore",       desc:"Microsoft Company",
image:require("../../assets/images/joe_belfiore.webp")    },
{       title:"Joe Biden",       desc:"President American",
image:require("../../assets/images/joe_biden.jpeg")    },
{       title:"Mark Zuckerberg",       desc:"Microsoft Company",
image:require("../../assets/images/mark.jpeg")    },
{       title:"Vladimir Putin",       desc:"President Russia",
image:require("../../assets/images/putin.jpeg")    }
]
```

Các file hình được cho sẵn

Yêu cầu: Dùng ScrollView và dùng FlatList

Excercise 3



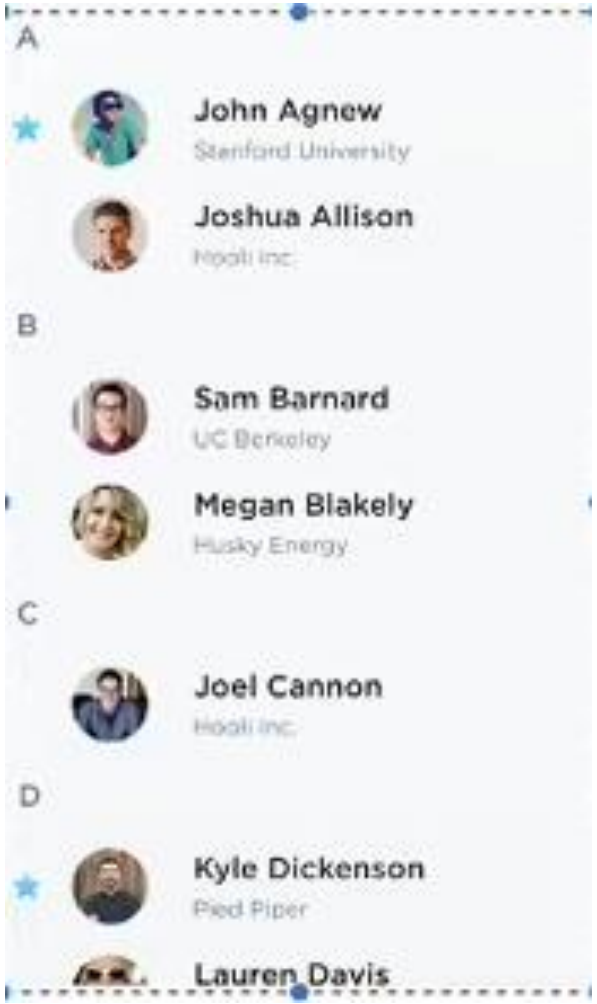
Hình ảnh được cho sẵn:

Excercise 3 (tt)

Dữ liệu cho sẵn:

```
const dataAvatar = [
   {
      name: 'Sahara',
      image: require('../../asset/images/image1.jpeg'),
   },
   {
      name: 'Sophia',
      image: require('../../asset/images/image2.jpeg'),
   },
.....
const dataFeeds = [
  {
     title: "Lorem Ipsum is simply dummy text",
     name: "Sahara",
     image: require('../../asset/images/image1.jpeg'),
     content: "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum",
     time: "1 minutes"
  },
.....
```

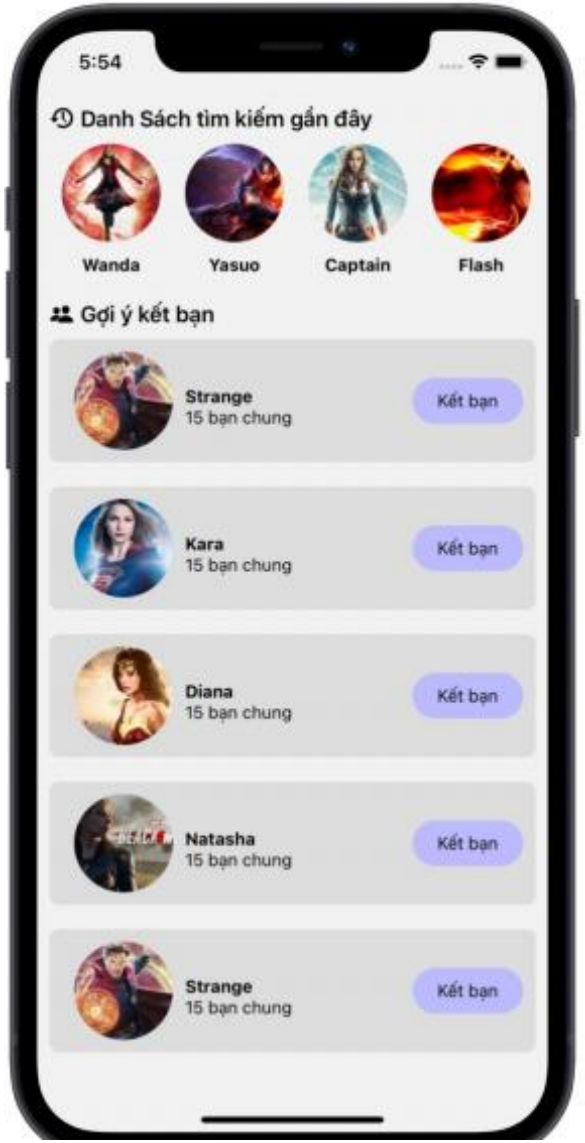
Excercise 4



Yêu cầu:

- Tạo dữ liệu, chuẩn bị hình ảnh
- Dùng SectionList

Excercise 5



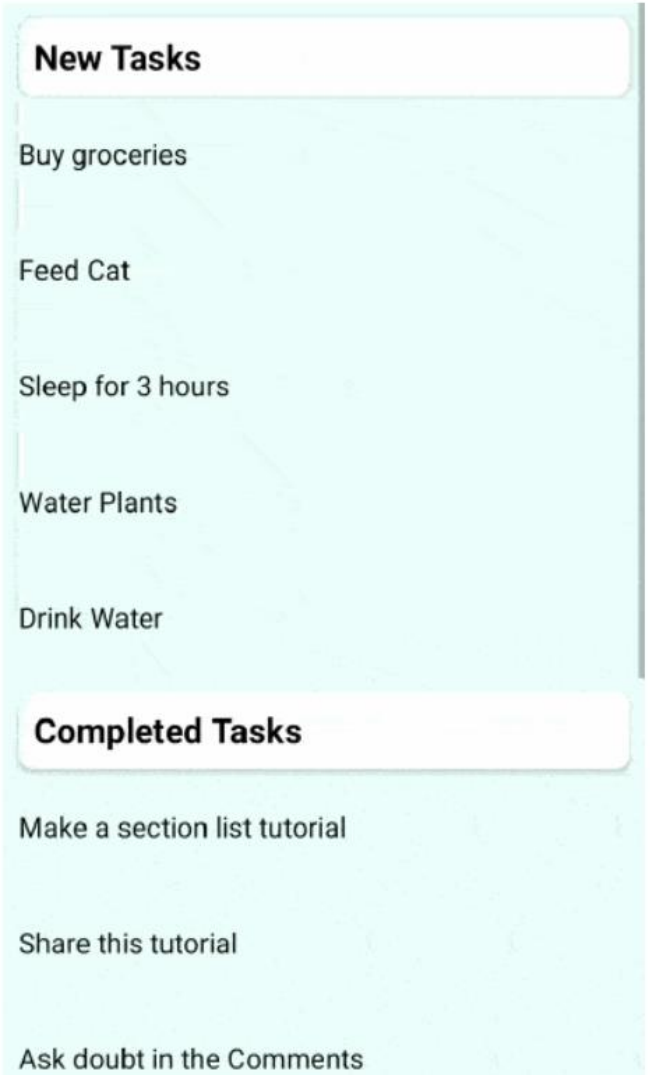
Hình ảnh cho tại:

https://drive.google.com/drive/folders/1ewYyjfO7OrjaLbgpYlpUNHdRgLSppEU9

Excercise 6


```
const DATA = [
  {
    title: "Main dishes",
    data: ["Pizza", "Burger", "Risotto"]
  },
  {
    title: "Sides",
    data: ["French Fries", "Onion Rings", "Fried Shrimps"]
  },
  {
    title: "Drinks",
    data: ["Water", "Coke", "Beer"]
  },
  {
    title: "Desserts",
    data: ["Cheese Cake", "Ice Cream"]
  }
];
```

Excercise 6



Dữ liệu tại file task.js đính kèm, dạng:

```
const newTaskData = [{
    title: "New Tasks",
    data: [
      {
        id: "1",
        task: "Buy groceries"
      },
      {
        id: "2",
        task: "Feed Cat"
      },
```

……