Lab: Flower Shop Landing Page using Basic Components

Estimated time needed: 30 minutes

What application will you build?

In this lab, you will build a landing page for a flower shop using React Native. The application will include various sections featuring different types of flowers with images and descriptions.

What you will learn

You will gain a solid understanding of how to structure a React Native app using functional components. You will learn how to use the ScrollView component for vertical and horizontal content scrolling. You will also learn how to handle user interactions using TouchableOpacity and trigger actions, such as displaying alerts.

Learning objectives

After completing this lab, you will be able to:

- Use a `ScrollView` to allow vertical content scrolling and include a horizontally scrollable section for showcasing featured flowers using the horizontal prop within the ScrollView.
- Render Dynamic `Image` Content component to dynamically load and display images from external URLs, allowing flexibility in showing different flower types.
- Use a `TouchableOpacity` component to create a clickable "Shop Now" button, which triggers an action (an alert) when the user selects it, demonstrating basic event handling.

Prerequisites

- Basic knowledge of HTML and JavaScript
- Basic knowledge of React

Understand folder structure

1. The "FlowerShop" folder contains the boilerplate for the React Native code. It includes one `App.js` component that uses a `SafeAreaView` to ensure the layout adapts to the device's safe areas. It renders a custom component **FlowerLandingPage** inside the main **App** component.
2. Click the `FlowerLandingPage.js` component inside the **FlowerShop** folder. It contains:
    - The `FlowerlandingPage` component which is a basic React Native screen that uses a `ScrollView` to allow scrolling through the content.
    - Inside the scrollable area, there's a header section `<View>` containing two `Text` elements:
        - One for the **Blooming Flowers**
        - Another subtitle describes the flower delivery service.
        - The styles are referenced from the `styles` object.

Create image and search bar section

1. Navigate to the `FlowerLandingPage.js` file. Here, you need to create an image section to display as a starting point. For this, use the `<View>` component to contain your image for layout purposes.

```
<View style={styles.imageContainer}></View>
```

Note: Include the above code within the `<ScrollView style={styles.container}>...</ScrollView>` after the `<View>` component with the `style={styles.header}` prop.

2. Inside this newly added `<View>` component, add an `<Image>` component. Set the image source using a URI link to the image you want to display.

```
<Image source={{ uri: 'https://images.unsplash.com/photo-1463043254199-
7a3efd782ad1?q=80&w=2069&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D' }}
style={styles.image} />
```

Note: The styles for both the `<View>` and `<Image>` components to control dimensions, alignment, and other visual properties are provided by default.

3. Now create another `<View>` to contain the search bar, placing it after the `<View>` of the image section.

```
<View style={styles.searchContainer}></View>
```

Note: Include this code after the `<View>` component with the `style={styles.imageContainer}` prop.

4. Now, inside this `<View>` component of the search bar, add a `TextInput` component to act as the search bar. Then, set a placeholder to display hint text, such as **Search for flowers...**.

```
<TextInput
          style={styles.textInput}
          placeholder="Search for flowers..."
        />
```

Note: Define styles for the `TextInput` to control its appearance, such as height, padding, and border.

Create featured flowers section

1. In this step, you will develop the featured flowers section, which will display a list of the application's famous flowers.

2. To begin, you need to create a container using the `<View>` component, which will act as the main container for the featured flowers section. You will also create a button for this section to proceed to the shop. Include this container after the `View` component of the search bar.

```
<View>


</View>
```

Note: Include the above code after the `<View>` component with the prop named `style={styles.searchContainer}`.

3. Inside the newly created `<View>` component, create another `<View>` component to feature the famous flower section.

```
<View>
    <View style={styles.featuredSection}>
    </View>
</View>
```

4. Now, inside the `<View>` container with the prop named `style={styles.featuredSection}`, add a `Text` component for the section title, such as **Featured Flowers**.

```
<View>
    <View style={styles.featuredSection}>
        <Text style={styles.sectionTitle}>Featured Flowers</Text>
    </View>
</View>
```

5. Next, add a `ScrollView` component after the `Text` component with the horizontal prop set to **true** to allow horizontal scrolling.
   Set **showsHorizontalScrollIndicator** to **false** to disable the horizontal scroll

indicators and prevent the scrollbar from appearing while the user scrolls horizontally through the content.

```
<ScrollView horizontal showsHorizontalScrollIndicator={false}>


</ScrollView>
```

6. Now, inside the `<ScrollView>`, add the `<View>` component to display the flower image and name using the `<Image>` and `<Text>` components.

```
<View style={styles.flowerCard}>
            <Image
              source={{ uri: 'https://images.unsplash.com/photo-
1686125616977-34f6d5979eb1?q=80&w=1856&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D' }}
              style={styles.flowerImage}
            />
            <Text style={styles.flowerTitle}>White Base</Text>
          </View>
```

   - The above code contains an `Image` component for the flower's picture, which uses the `source` prop to set the image URI.
   - It also includes a `<Text>` component for the flower's name or title.

7. Now, create several other `<View>` components to display other flower images and names using the `<Image>` and `<Text>` components within the `<ScrollView>` component.

```
          <View style={styles.featuredSection}>
        <Text style={styles.sectionTitle}>Featured Flowers</Text>
        <ScrollView horizontal showsHorizontalScrollIndicator={false}>
          <View style={styles.flowerCard}>
            <Image
              source={{ uri: 'https://images.unsplash.com/photo-
1686125616977-34f6d5979eb1?q=80&w=1856&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D' }}
```

```
              style={styles.flowerImage}
            />
            <Text style={styles.flowerTitle}>White Base</Text>
          </View>
          <View style={styles.flowerCard}>
            <Image
              source={{ uri: 'https://images.unsplash.com/photo-
1581264692636-3cf6f29655c2?q=80&w=1887&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D' }}
            />
            <Text style={styles.flowerTitle}>Red Rose</Text>
          </View>
          <View style={styles.flowerCard}>
            <Image
              source={{ uri: 'https://images.unsplash.com/photo-
1579847621515-b40fcc20831e?q=80&w=1887&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D' }}
              style={styles.flowerImage}
            />
            <Text style={styles.flowerTitle}>Lily</Text>
          </View>
          <View style={styles.flowerCard}>
            <Image
              source={{ uri: 'https://images.unsplash.com/photo-
1487435636644-3ad040f0195b?q=80&w=2070&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D' }}
              style={styles.flowerImage}
            />
            <Text style={styles.flowerTitle}>Tulip</Text>
          </View>
```

```
          <View style={styles.flowerCard}>
            <Image
              source={{ uri: 'https://images.unsplash.com/photo-1556216750-2108e1e54e9a?q=80&w=1887&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D' }}
              style={styles.flowerImage}
            />
            <Text style={styles.flowerTitle}>Orchid</Text>
          </View>
          <View style={styles.flowerCard}>
            <Image
              source={{ uri: 'https://images.unsplash.com/photo-1695112691738-5227cabb206f?q=80&w=1887&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D' }}
              style={styles.flowerImage}
            />
            <Text style={styles.flowerTitle}>Chrysanthemums</Text>
          </View>
        </ScrollView>
      </View>
```

Entire `View` component with `style={styles.featuredSection}` will look like above code snippet.

8. After the `<View>` component with the prop `style={styles.featuredSection}`, include one more component for the button section using `<view>` with the prop named `style={styles.buttonContainer}`.

```
<View style={styles.buttonContainer}>
</View>
```

9. Inside the newly created `<View>` component, add a button using the `<TouchableOpacity>` component.

```
<TouchableOpacity style={styles.button} onPress={() => alert('Shop Now')}>
```

```
<Text style={styles.buttonText}>Shop Now</Text>
</TouchableOpacity>
```

Let's break down the above code:

- `<TouchableOpacity>`:
  - This is a React Native component used to create a touchable area that provides visual feedback when pressed (by reducing opacity). It wraps other components like `Text` to make them touchable (like a button).
  - `style={styles.button}`: Applies the custom styling from the `styles.button` object to the `TouchableOpacity`. You'd define `styles.button` elsewhere in your code to control how the button looks (width, height, background color).
- `onPress={() => alert('Shop Now')}`:
  - This is a function triggered when you click the button. Here, it shows an alert with the message "Shop Now."
- `<Text style={styles.buttonText}>Shop Now</Text>`:
  - The Text component displays the "Shop Now" text inside the button. The `styles.buttonText` object contains styles that determine the appearance of the text (font size, color).

Note: You can also use Button instead of TouchableOpacity. The main difference is that a button comes with a default style, and if you want to style it manually, you can use touchable opacity.

Check the output

1. Make sure to save all the files where you have made changes.
2. If the output window is open, refresh the application. Otherwise, check the output by running your application in the terminal