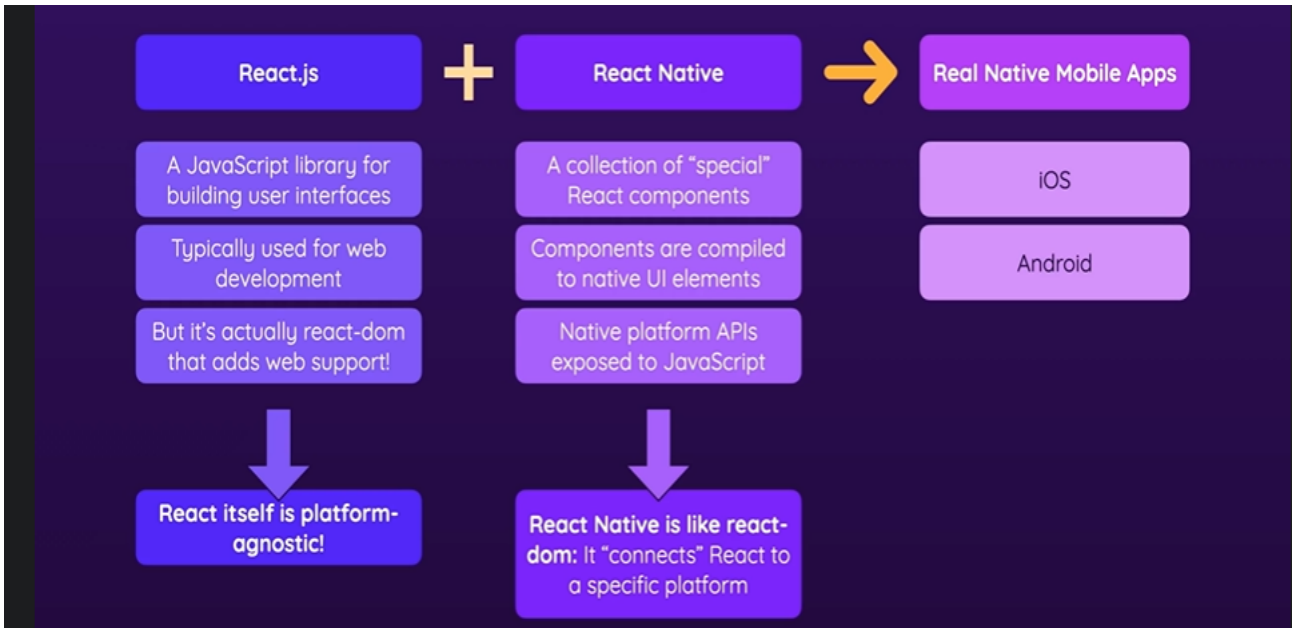
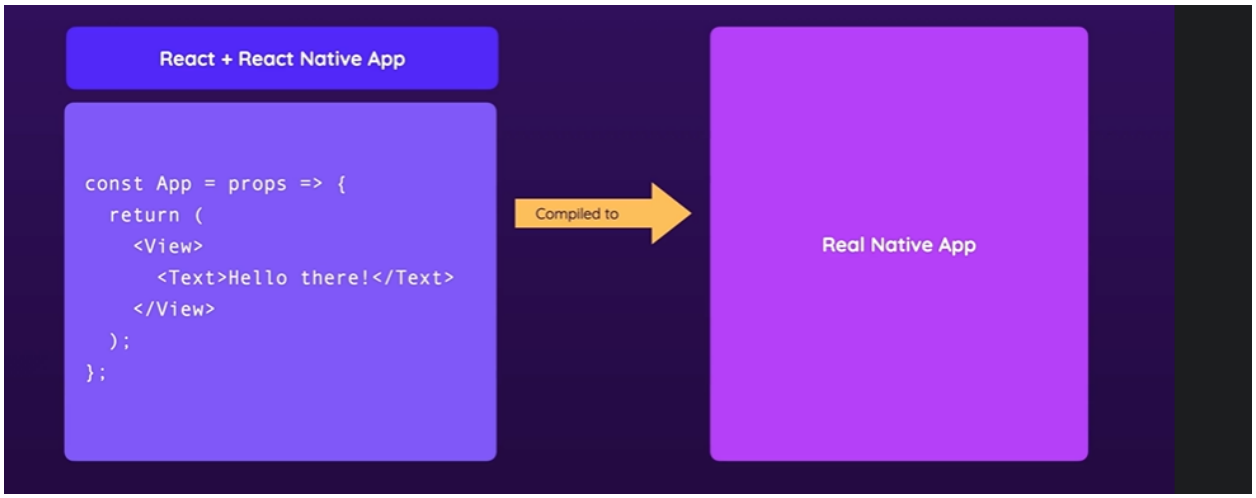
1.What is react native?

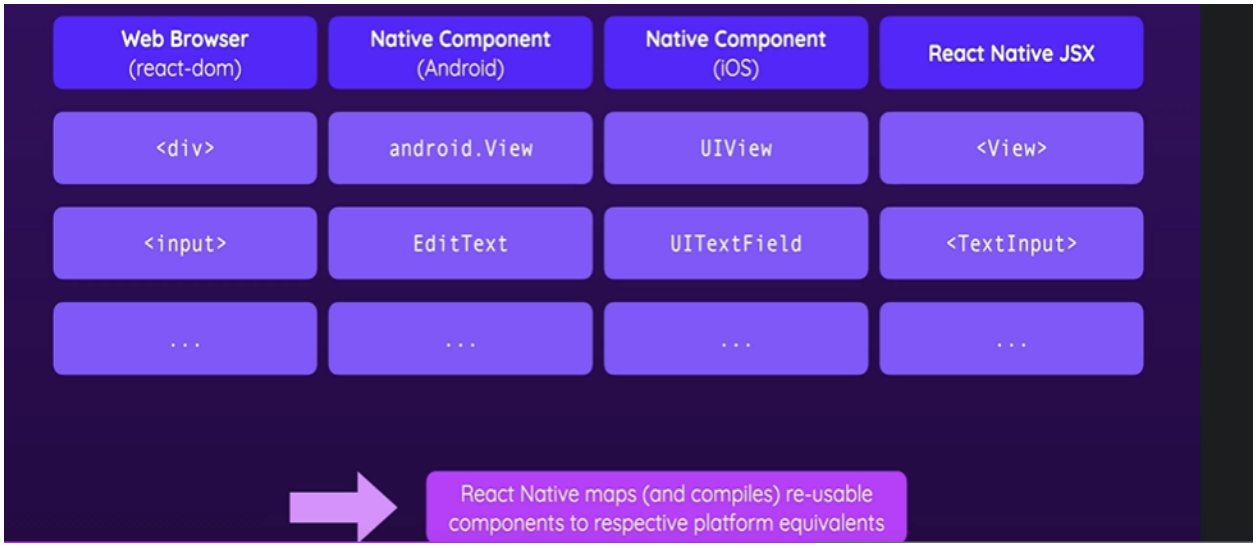


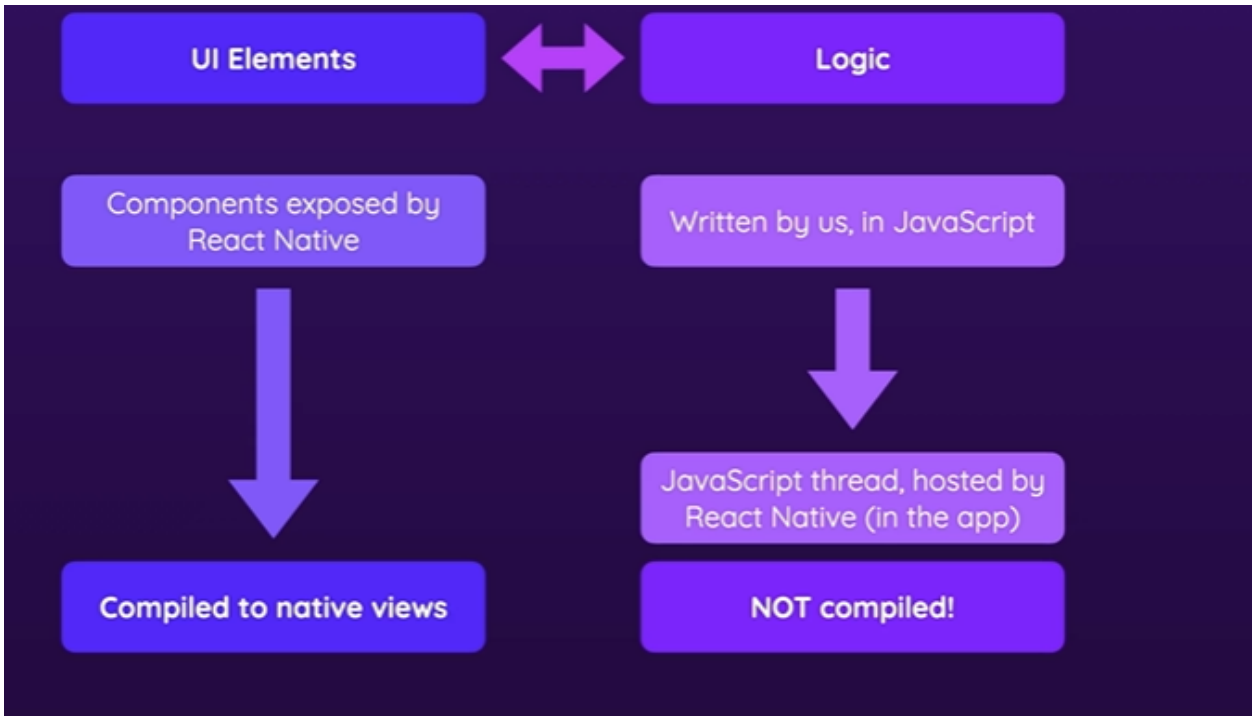
React native +React-js :for building mobile apps.
React-native+React.dom: for building web apps

2.How does React-native work?



Components are complied





The ui components are only complied .the javascript logic runs as athread hosted by react-native app

3.Creating a New React-native App

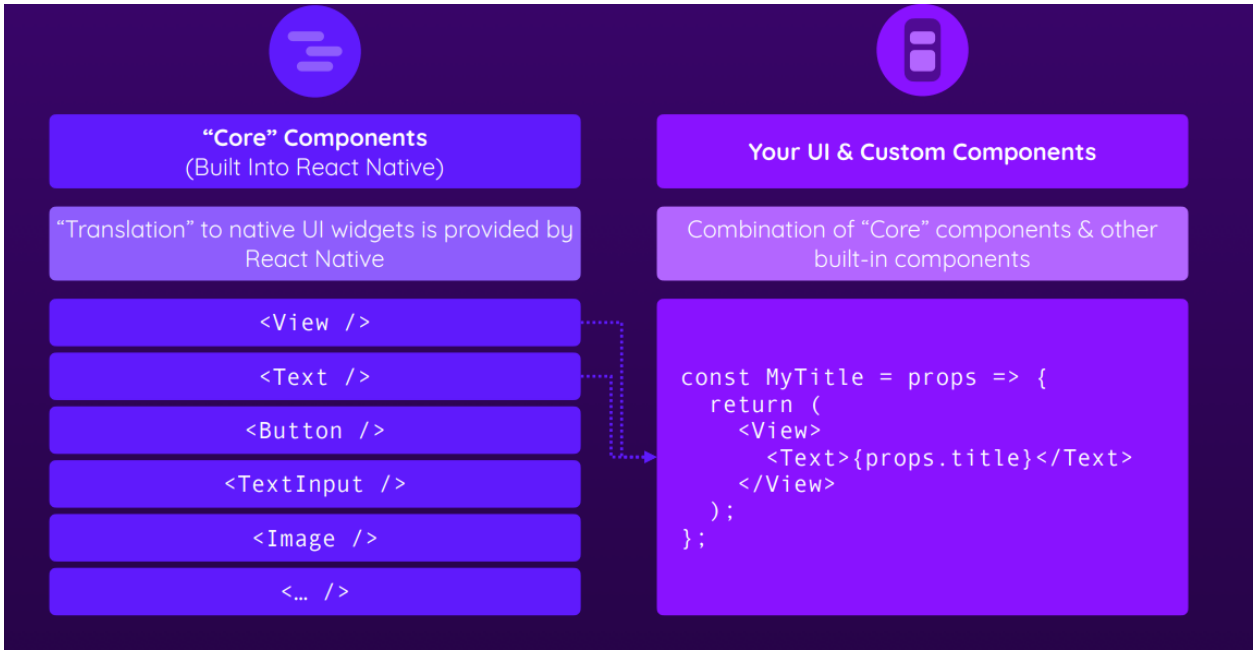
1.Install node.js
2.To create a project use the following commands

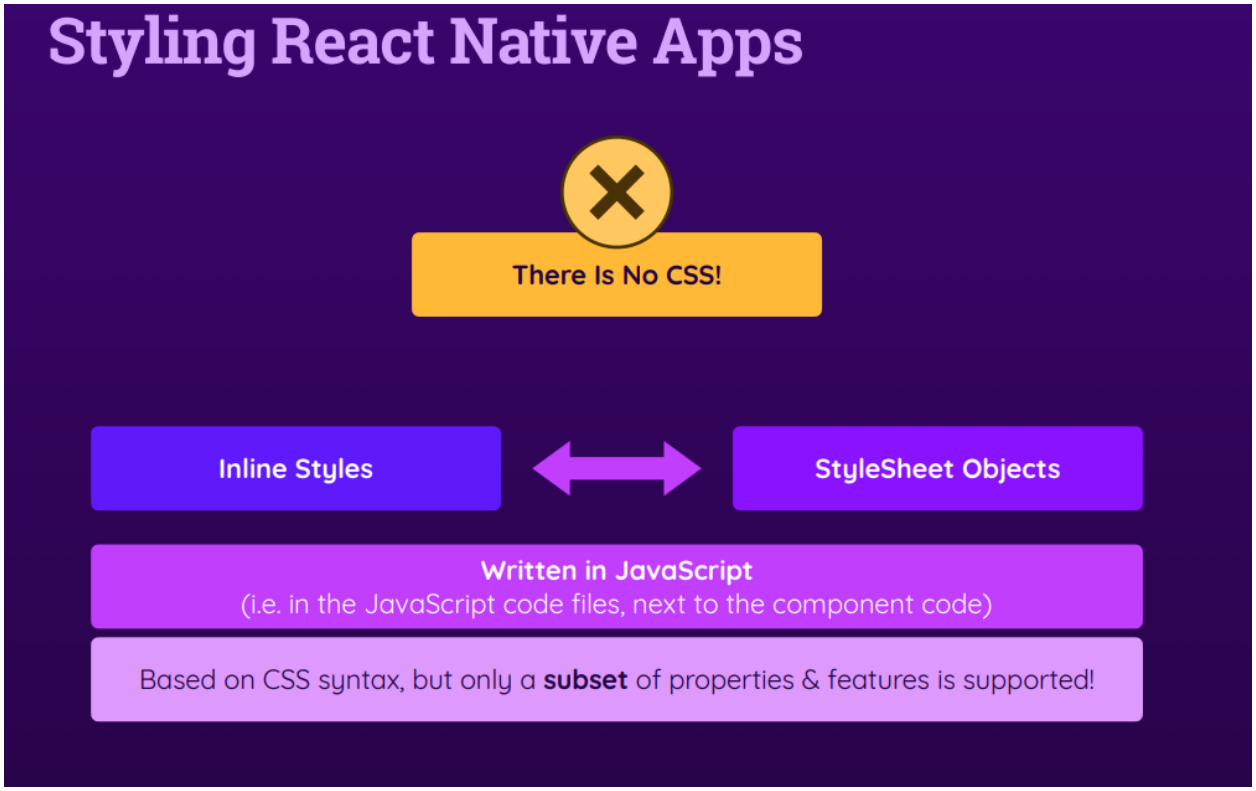
```
npx create-expo-app AwesomeProject // project name

cd AwesomeProject
npm start
```

4.Running our app in real device

1.go to play store ->install expo
2.run npm start
3.scan the qr code in terminal in the expo app of your phone
4.the app runs on the android

5.Working with core components


"Core" Components
(Built Into React Native)
"Translation" to native UI widgets is provided by React Native
<View />
<Text />
<Button />
<TextInput />
<Image />
<… />
Your UI & Custom Components
Combination of "Core" components & other built-in components
const MyTitle = props => {
return (
<View>
<Text>{props.title}</Text>
</View>
);
};

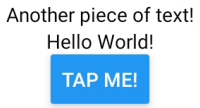

Styling React Native Apps
There Is No CSS!
Inline Styles
StyleSheet Objects
Written in JavaScript
(i.e. in the JavaScript code files, next to the component code)
Based on CSS syntax, but only a subset of properties & features is supported!

6.Basic React Project

code:

```
export default function App() {
  return (
    <View style={styles.container}>
     <View>
        <Text>Another piece of text!</Text>
      </View>
      <Text>Hello World!</Text>
      <Button title='Tap me!' />
    </View>
  );
}
```

Output:

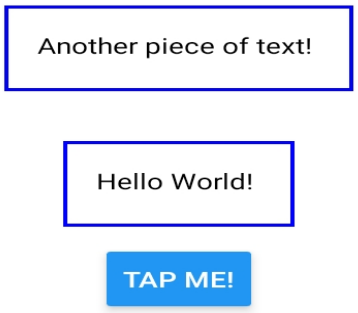


7.Styling React apps

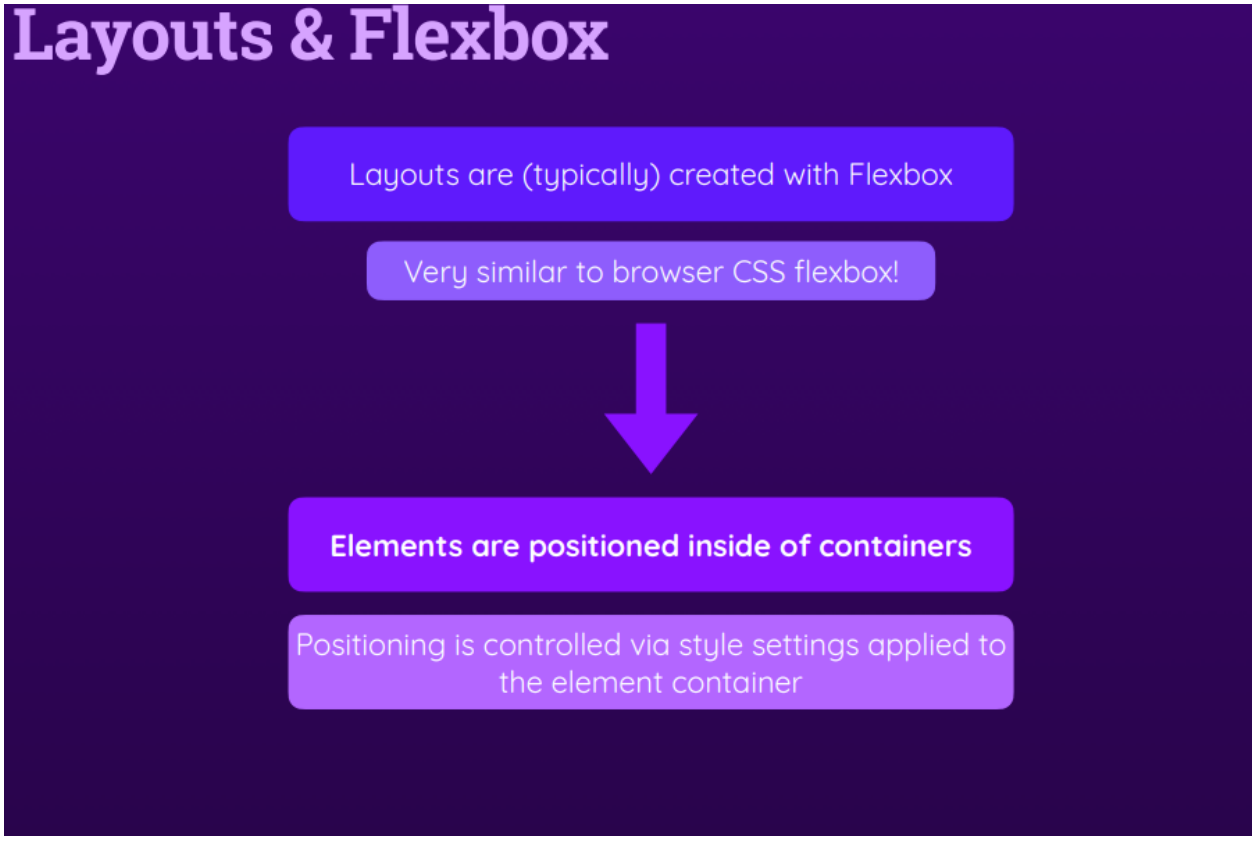
```
export default function App() {
  return (
    <View style={styles.container}>
     <View>
```

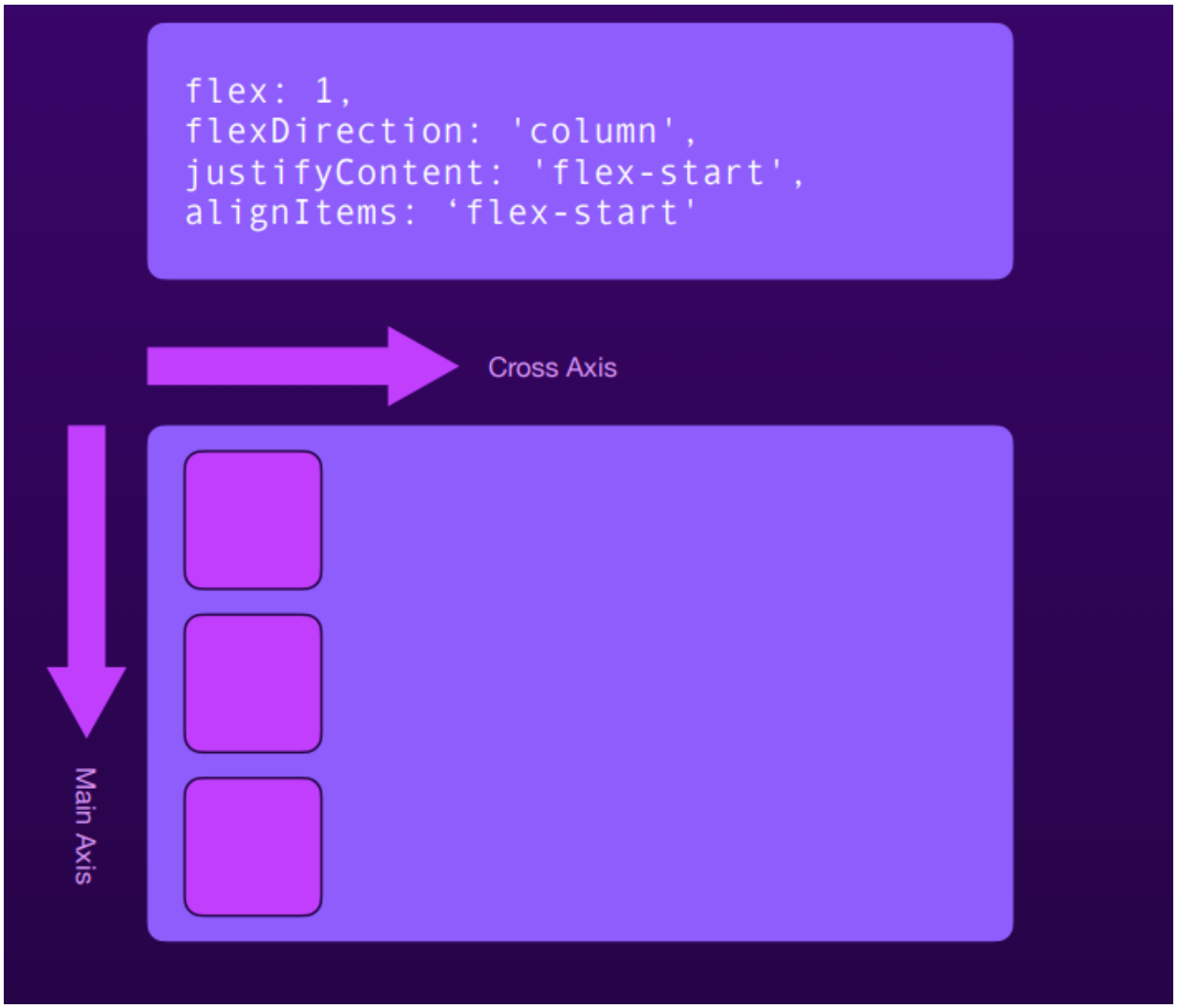
```
        <Text  style={styles.dummyText}>Another piece of text!</Text>
      </View>
      <Text  style={styles.dummyText}>Hello World!</Text>
      <Button title='Tap me!' />
    </View>
  );
}
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  dummyText: {
    margin: 16,
    padding: 16,
    borderWidth: 2,
    borderColor: 'blue',
  }
});
```

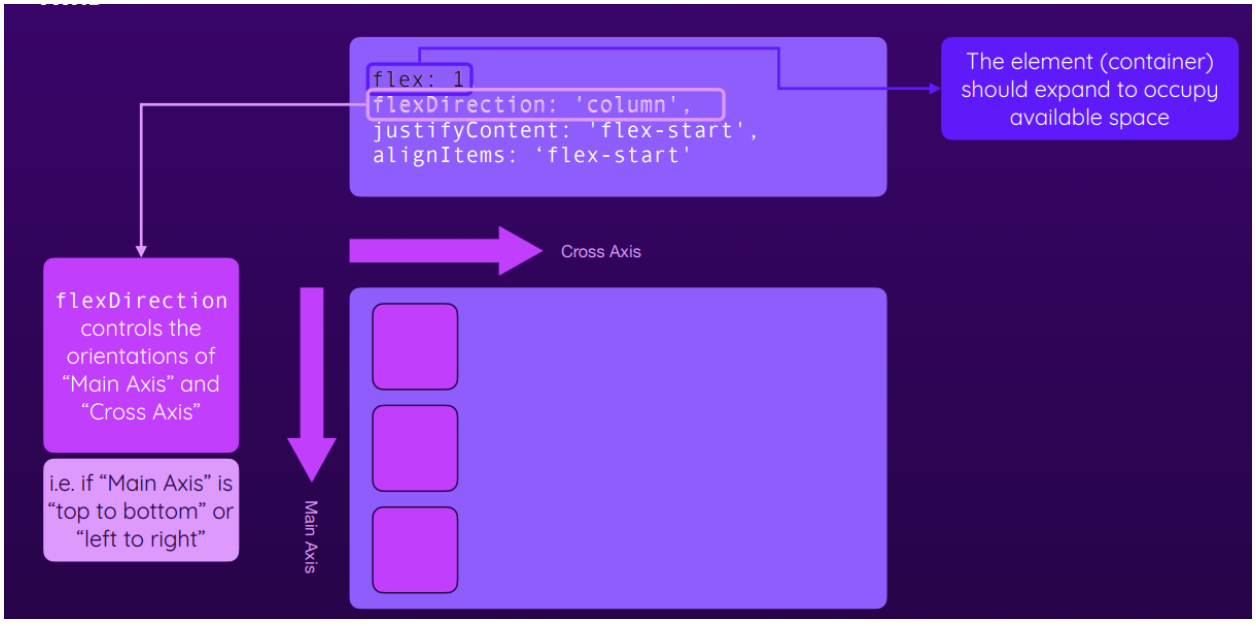
Output:

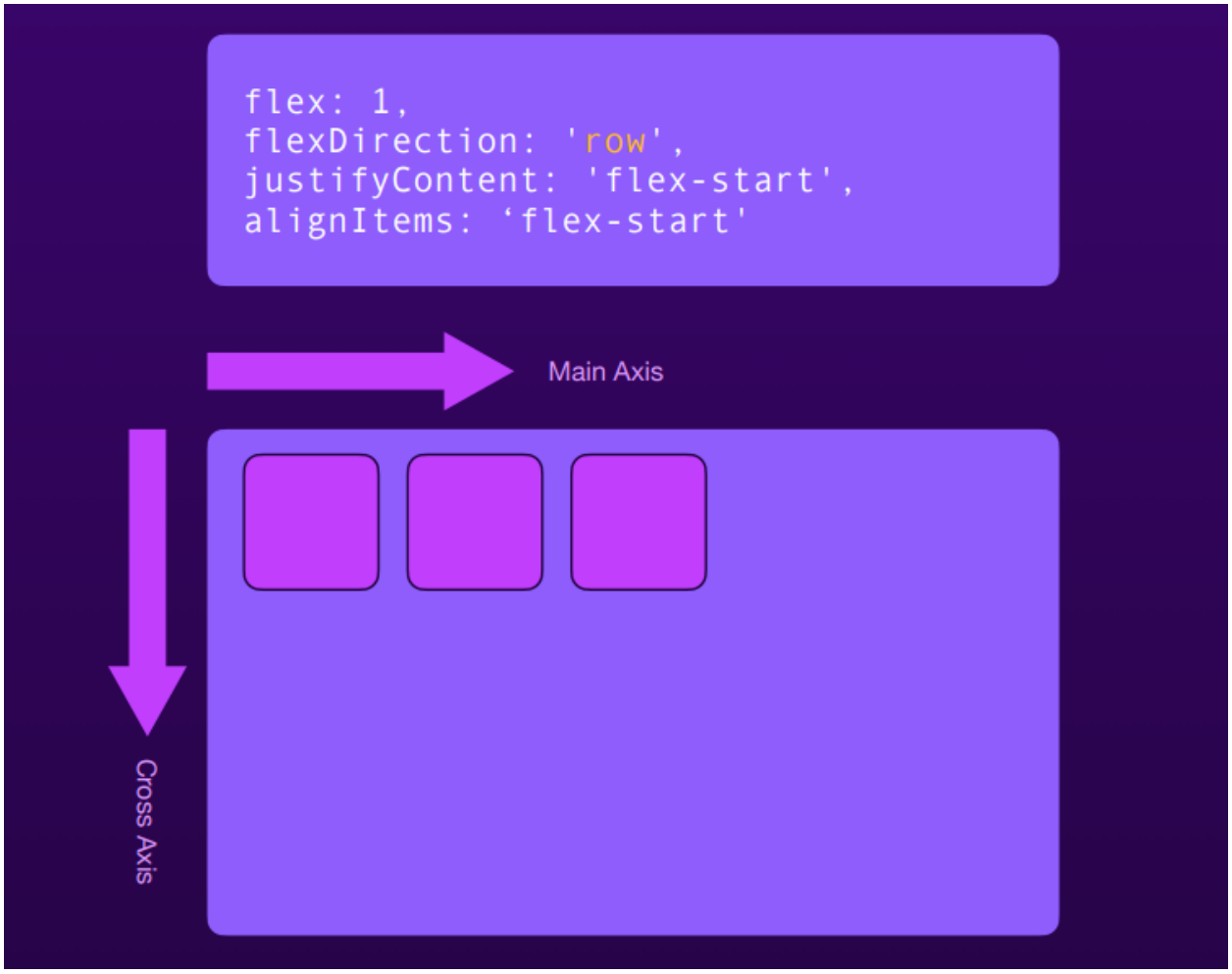


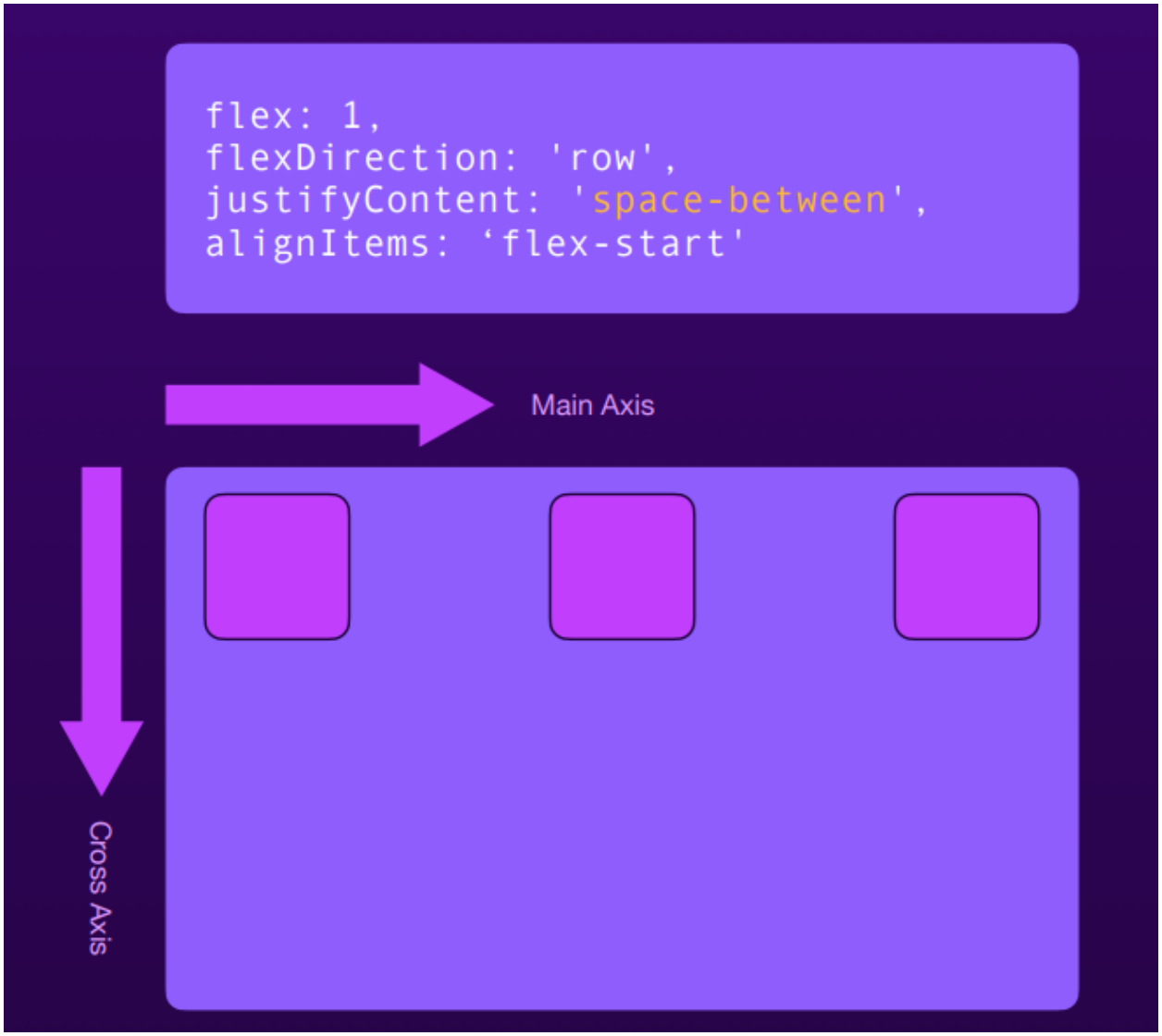
8.Exploring Layouts and Flexbox

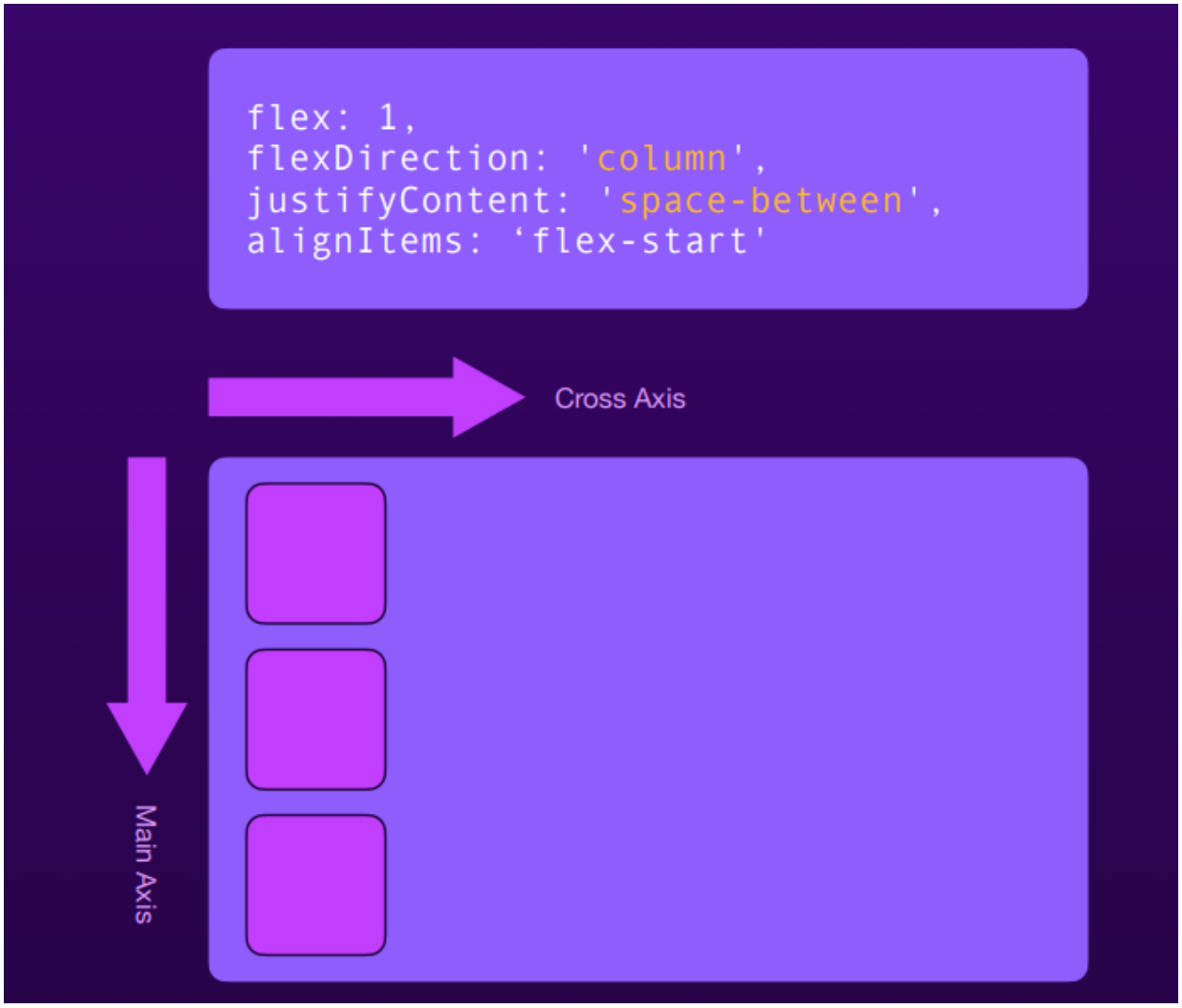

Layouts & Flexbox
Layouts are (typically) created with Flexbox
Very similar to browser CSS flexbox!
Elements are positioned inside of containers
Positioning is controlled via style settings applied to the element container

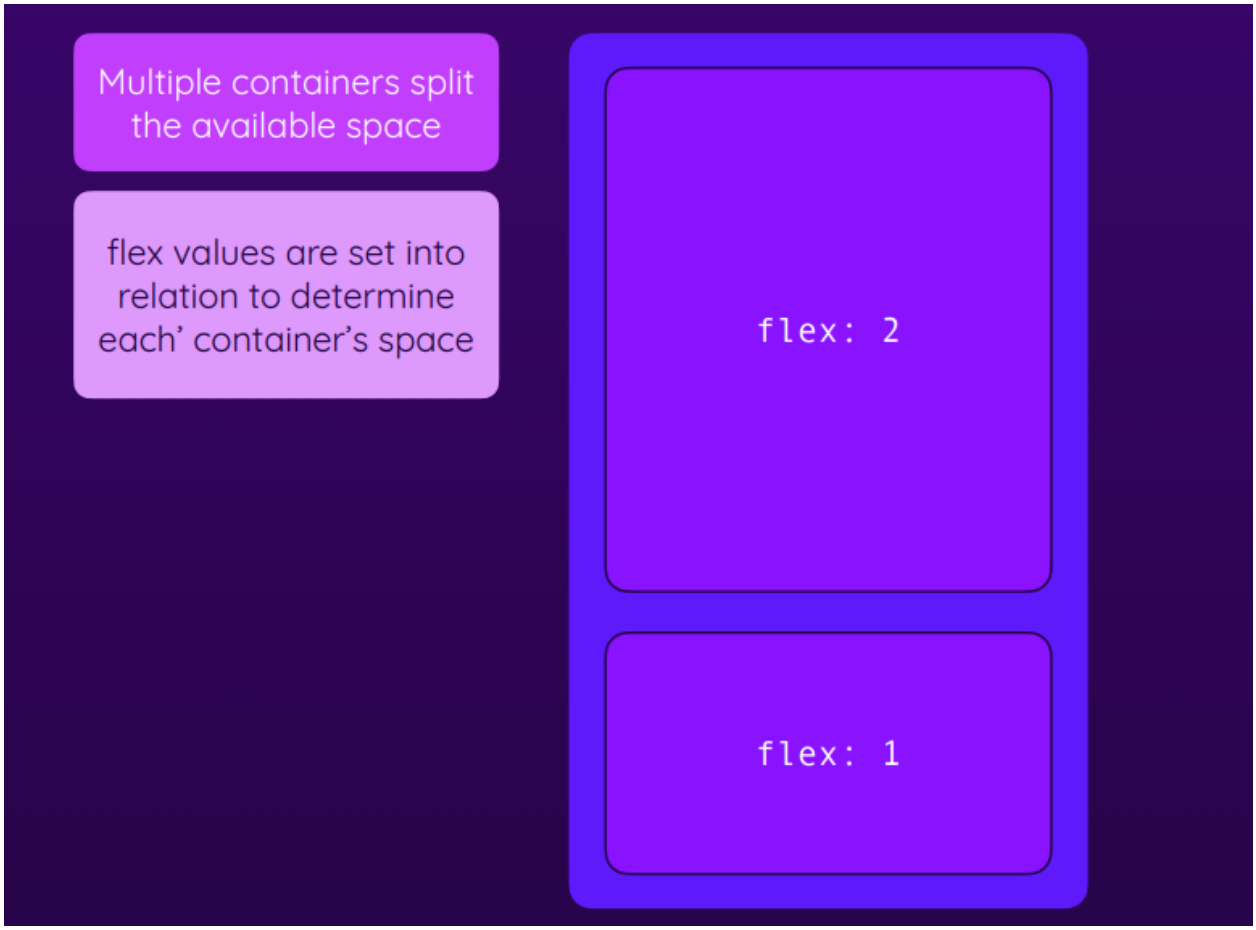

flex: 1,
flexDirection: 'column',
justifyContent: 'flex-start',
alignItems: 'flex-start'
Cross Axis
Main Axis


flex: 1
flexDirection: 'column',
justifyContent: 'flex-start',
alignItems: 'flex-start'
The element (container) should expand to occupy available space
Cross Axis
flexDirection controls the orientations of "Main Axis" and "Cross Axis"
i.e. if "Main Axis" is "top to bottom" or "left to right"
Main Axis


flex: 1,
flexDirection: 'row',
justifyContent: 'flex-start',
alignItems: ‘flex-start'
Main Axis
Cross Axis


flex: 1,
flexDirection: 'row',
justifyContent: 'space-between',
alignItems: ‘flex-start'
Main Axis
Cross Axis


flex: 1,
flexDirection: 'column',
justifyContent: 'space-between',
alignItems: ‘flex-start'
Cross Axis
Main Axis

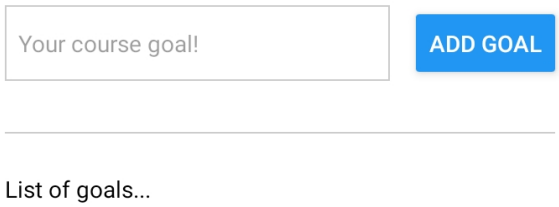


9.Improving the layout

```
import { StyleSheet, Text, View, Button, TextInput } from 'react-native';

export default function App() {
  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput style={styles.textInput} placeholder="Your course
goal!" />
        <Button title="Add Goal" />
      </View>
      <View style={styles.goalsContainer}>
        <Text>List of goals...</Text>
      </View>
    </View>
  );
}
```

```
const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16
  },
  inputContainer: {
    flex: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 24,
    borderBottomWidth: 1,
    borderBottomColor: '#cccccc'
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '70%',
    marginRight: 8,
    padding: 8
  },
  goalsContainer: {
    flex: 5
  }
});
```

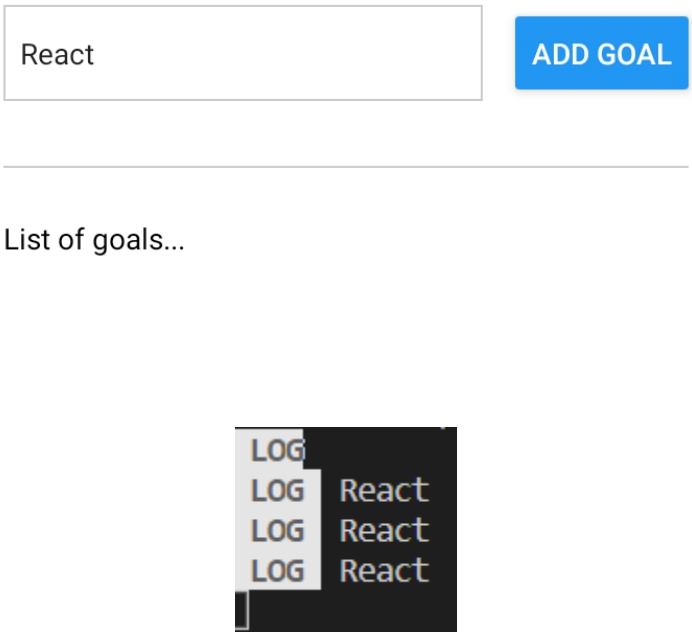
Output:



10.Handling events

```
export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    console.log(enteredGoalText);
  }
  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput style={styles.textInput} placeholder="Your course
goal!"  onChangeText={goalInputHandler}/>
        <Button title="Add Goal" onPress={addGoalHandler} />
      </View>
      <View style={styles.goalsContainer}>
        <Text>List of goals...</Text>
      </View>
    </View>
  );
}
```

Output:

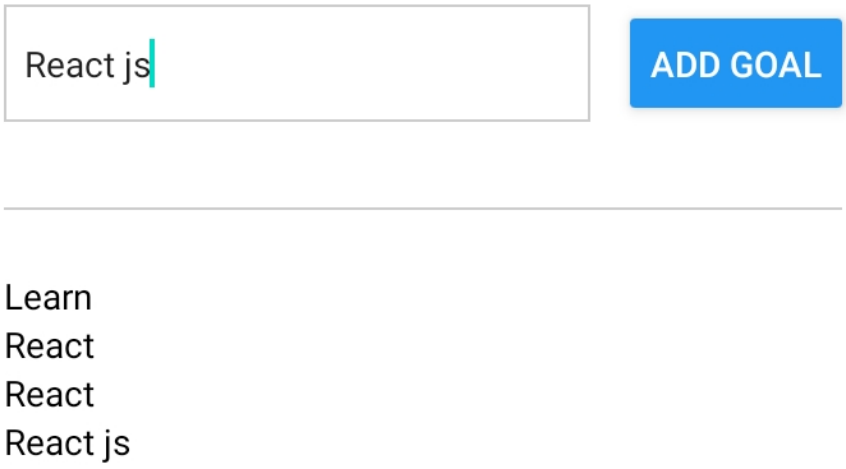


11.Managing a list of goals

```
import { StyleSheet, Text, View, Button, TextInput } from 'react-native';
import { useState } from 'react';

export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');
  const [courseGoals, setCourseGoals] = useState([]);

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
// 2.retrieves all the variables created in the courseGoals
      enteredGoalText,
//3.adds the new variable to the courseGoals state.
    ]);
```

```
  }
  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput style={styles.textInput} placeholder="Your course
goal!"  onChangeText={goalInputHandler}/>
        <Button title="Add Goal" onPress={addGoalHandler} />
      </View>
      <View style={styles.goalsContainer}>
      {courseGoals.map((goal) => <Text key={goal}>{goal}</Text>)}
//1.maps the list into the courseGoals variable in the state
      </View>
    </View>
  );
}
```

Output:



12.ios and android styling differences

```
export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');
  const [courseGoals, setCourseGoals] = useState([]);

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      enteredGoalText,
    ]);
  }

  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.textInput}
          placeholder="Your course goal!"
          onChangeText={goalInputHandler}
        />
        <Button title="Add Goals" onPress={addGoalHandler} />
      </View>
      <View style={styles.goalsContainer}>
        {courseGoals.map((goal) => (
          <View key={goal} style={styles.goalItem}>
            <Text style={styles.goalText}>{goal}</Text>
          </View>
        ))}
      </View>
    </View>
  );
}
```

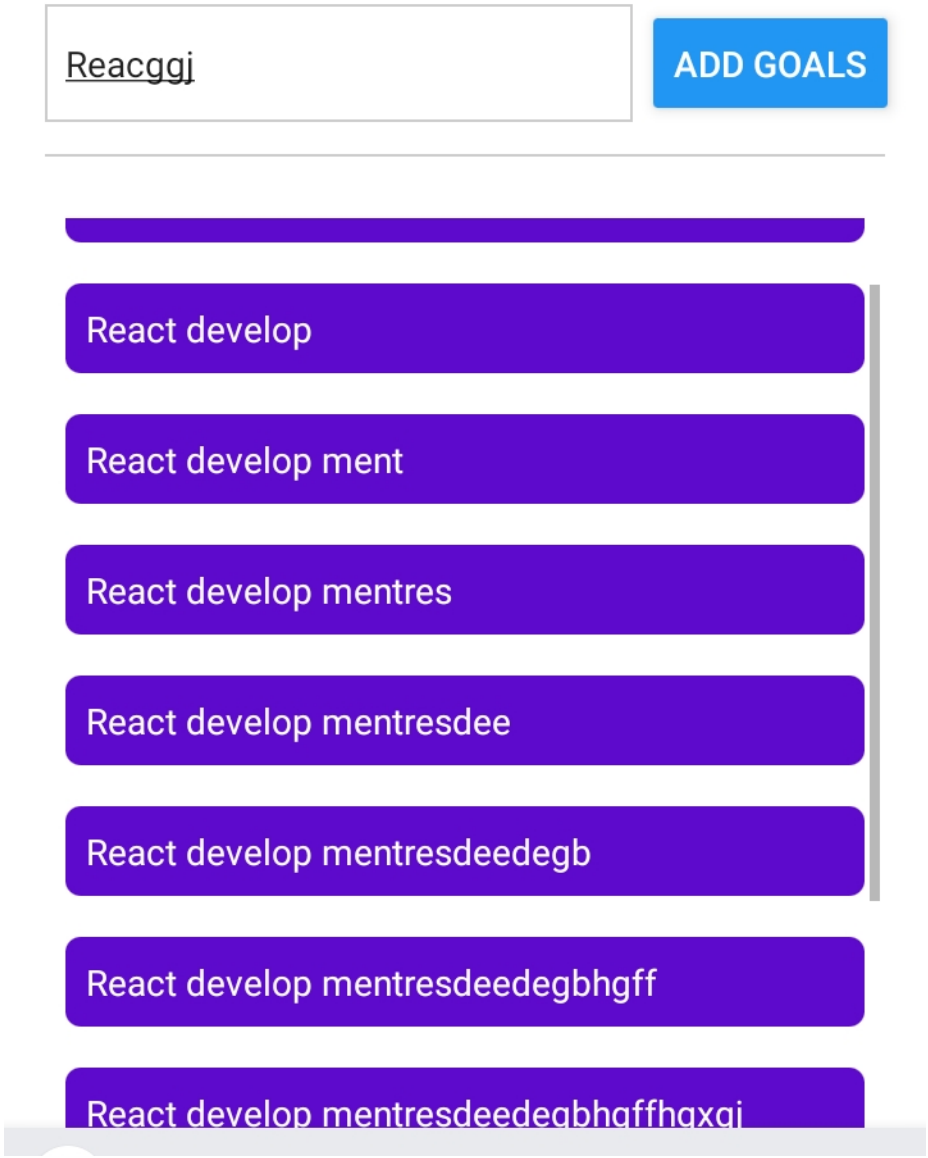
Output:

13.Making content scrollable with scrollview

```
import { useState } from 'react';
import { StyleSheet, Text, View, Button, TextInput,ScrollView } from
'react-native';

export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');
  const [courseGoals, setCourseGoals] = useState([]);

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      enteredGoalText,
    ]);
  }

  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.textInput}
```

```
          placeholder="Your course goal!"
          onChangeText={goalInputHandler}
        />
        <Button title="Add Goals" onPress={addGoalHandler} />
      </View>
      <View style={styles.goalsContainer}>
        <ScrollView>
        {courseGoals.map((goal) => (
          <View key={goal} style={styles.goalItem}>
            <Text style={styles.goalText}>{goal}</Text>
          </View>
        ))}

        </ScrollView>

      </View>
    </View>
  );
}
```

Output:



14.Optimizing lists with flatlist

Disadvantage of scrollview: scrollview renders all its child items.so ,for very dynamic long list it is not feasible.
Hence,we use flatlist.

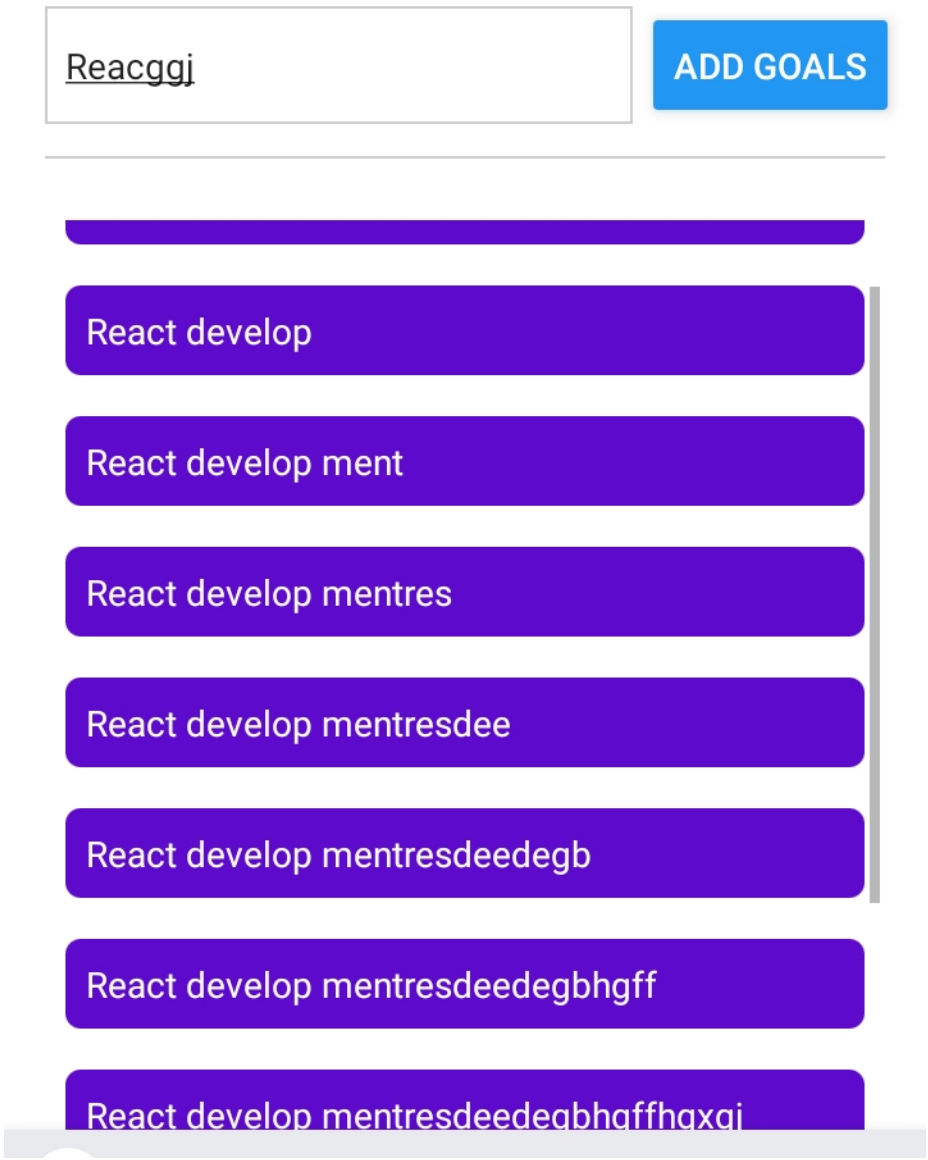
FlatList is used when we want to render only those elements that are currently being displayed on the screen (default: 10 items). For automatic memory management and to handle data changes in list we use a flat list.

Code:

```
import { useState } from 'react';
import { StyleSheet, Text, View, Button, TextInput,ScrollView,FlatList }
from 'react-native';

export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');
  const [courseGoals, setCourseGoals] = useState([]);

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() },
    ]);
  }

  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.textInput}
          placeholder="Your course goal!"
          onChangeText={goalInputHandler}
        />
        <Button title="Add Goals" onPress={addGoalHandler} />
      </View>
      <View style={styles.goalsContainer}>
      <FlatList
          data={courseGoals}
          renderItem={(itemData) => {
            return (
```

```
            <View style={styles.goalItem}>
              <Text style={styles.goalText}>{itemData.item.text}</Text>
            </View>
          );
        }}
        keyExtractor={(item, index) => {
          return item.id;
        }}
        alwaysBounceVertical={false}
      />

    </View>
  </View>
);
}
```

Output:



15.Spiltting components into smaller components

1)in GoalItem.js file (components)

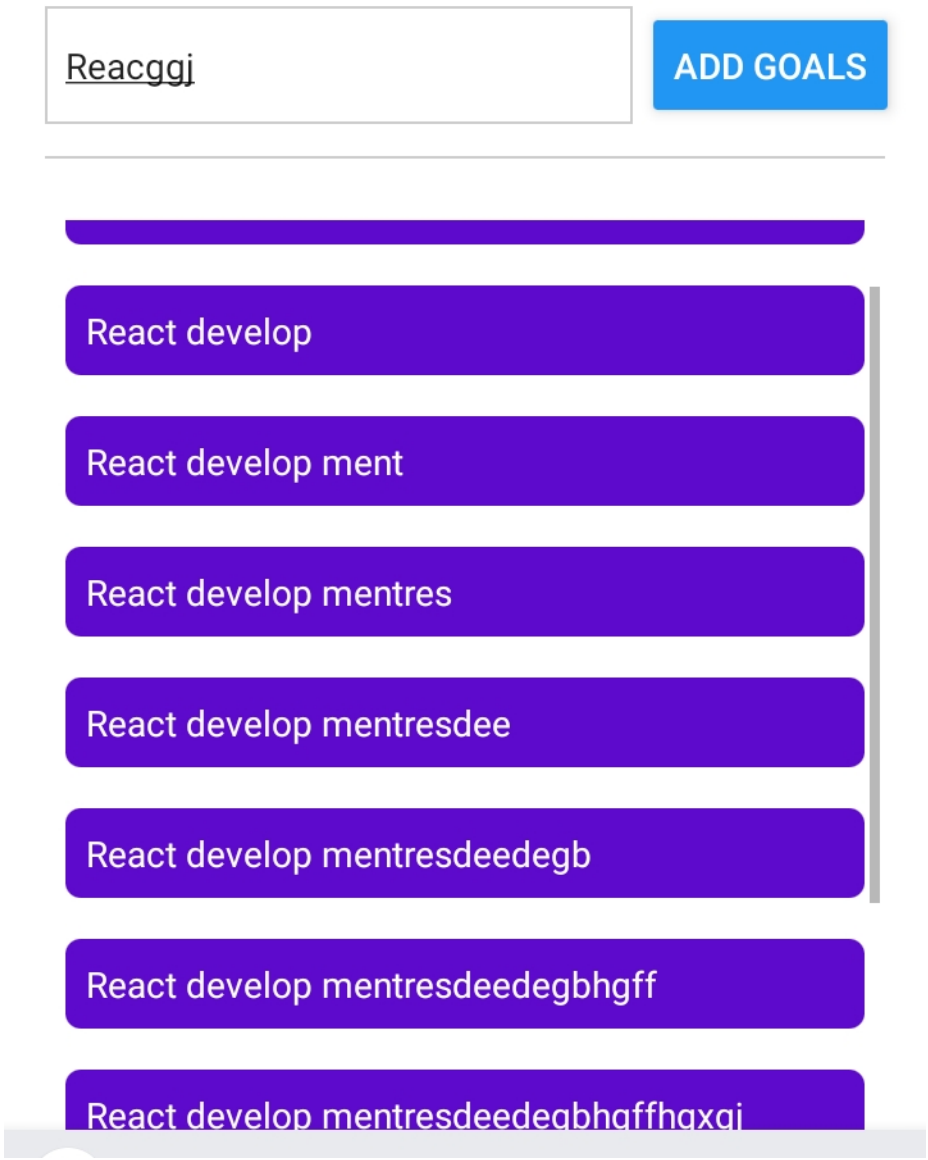
```
function GoalItem(props) {
  return (
    <View style={styles.goalItem}>
      <Text style={styles.goalText}>{props.text}</Text>
    </View>
```

```
  );
}
export default GoalItem;
```

2)in app.js

```
import { useState } from 'react';
import { StyleSheet, Text, View, Button, TextInput,ScrollView,FlatList }
from 'react-native';
import GoalItem from './components/GoalItem';
export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');
  const [courseGoals, setCourseGoals] = useState([]);

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() },
    ]);
  }

  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.textInput}
          placeholder="Your course goal!"
          onChangeText={goalInputHandler}
        />
        <Button title="Add Goals" onPress={addGoalHandler} />
      </View>
      <View style={styles.goalsContainer}>
      <FlatList
          data={courseGoals}
          renderItem={(itemData) => {
            return <GoalItem text={itemData.item.text} />;
```

```
          }}
          keyExtractor={(item, index) => {
            return item.id;
          }}
          alwaysBounceVertical={false}
        />

      </View>
    </View>
  );
}
```



16.Utilizing props

1)in GoalInput.js

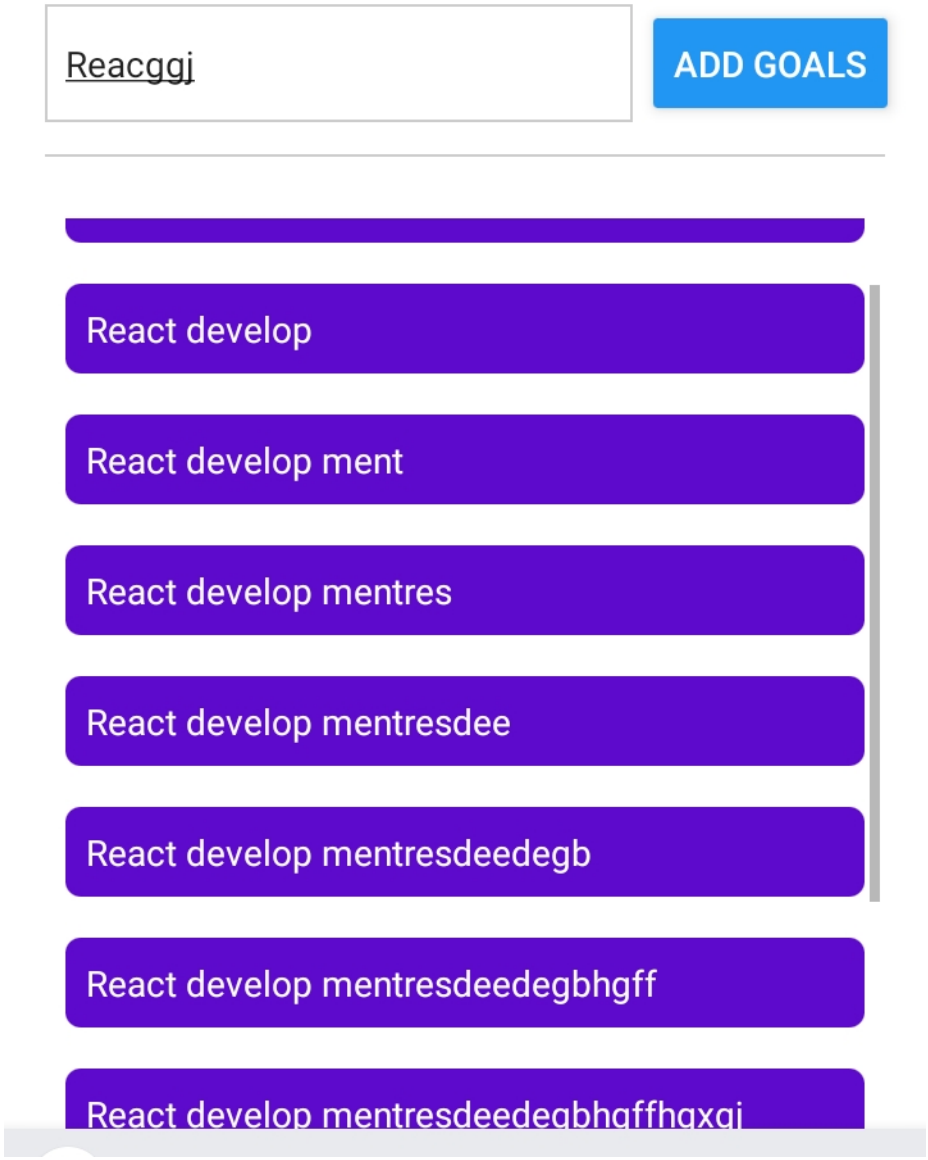
```
import { useState } from 'react';
import { View, TextInput, Button, StyleSheet } from 'react-native';

function GoalInput(props) {
  const [enteredGoalText, setEnteredGoalText] = useState('');

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
```

```
  }

  function addGoalHandler() {
    props.onAddGoal(enteredGoalText);
    setEnteredGoalText('');
  }

  return (
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.textInput}
        placeholder="Your course goal!"
        onChangeText={goalInputHandler}
        value={enteredGoalText}
      />
      <Button title="Add Goal" onPress={addGoalHandler} />
    </View>
  );
}

export default GoalInput;

const styles = StyleSheet.create({
  inputContainer: {
    flex: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 24,
    borderBottomWidth: 1,
    borderBottomColor: '#cccccc',
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '70%',
    marginRight: 8,
    padding: 8,
  },
});
```

In app.js

```
import { useState } from 'react';
import { StyleSheet, Text, View, Button, TextInput,ScrollView,FlatList }
from 'react-native';
import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';
export default function App() {

  const [courseGoals, setCourseGoals] = useState([]);



  function addGoalHandler(enteredGoalText) {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() },
    ]);
  }

  return (
    <View style={styles.appContainer}>
      <GoalInput onAddGoal={addGoalHandler} />
      <View style={styles.goalsContainer}>
      <FlatList
          data={courseGoals}
          renderItem={(itemData) => {
            return <GoalItem text={itemData.item.text} />;
          }}
          keyExtractor={(item, index) => {
            return item.id;
          }}
          alwaysBounceVertical={false}
        />

      </View>
    </View>
  );
}

const styles = StyleSheet.create({
```

```
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16,
  },

  goalsContainer: {
    flex: 5,
  }
});
```



17.Handling taps with pressale and deletable items

In GoalItem.js(components)

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

function GoalItem(props) {
  return (
    <Pressable onPress={props.onDeleteItem.bind(this, props.id)}>
        <View style={styles.goalItem}>
        <Text style={styles.goalText}>{props.text}</Text>
        </View>
    </Pressable>

  );
}
export default GoalItem;

const styles = StyleSheet.create({
  goalItem: {
    margin: 8,
    padding: 8,
    borderRadius: 6,
    backgroundColor: '#5e0acc',
  },
  goalText: {
    color: 'white',
  },
});
```

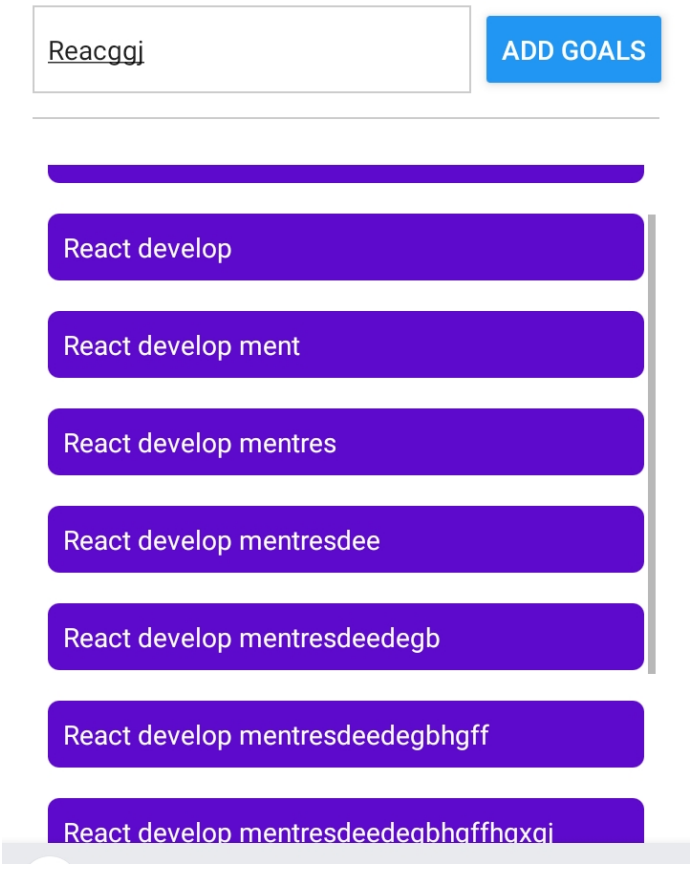
In app.js

```
import { useState } from 'react';
import { StyleSheet, View, FlatList } from 'react-native';
import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';
export default function App() {

  const [courseGoals, setCourseGoals] = useState([]);
```

```
  function addGoalHandler(enteredGoalText) {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() },
    ]);
  }
  function deleteGoalHandler(id) {
    setCourseGoals((currentCourseGoals) => {
      return currentCourseGoals.filter((goal) => goal.id !== id);
    });
  }

  return (
    <View style={styles.appContainer}>
      <GoalInput onAddGoal={addGoalHandler} />
      <View style={styles.goalsContainer}>
      <FlatList
          data={courseGoals}
          renderItem={(itemData) => {
            return (
            <GoalItem
            text={itemData.item.text}
            id={itemData.item.id}
            onDeleteItem={deleteGoalHandler} />
            );
          }}
          keyExtractor={(item, index) => {
            return item.id;
          }}
          alwaysBounceVertical={false}
        />

      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  appContainer: {
```

```
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16,
  },

  goalsContainer: {
    flex: 5,
  }
});
```

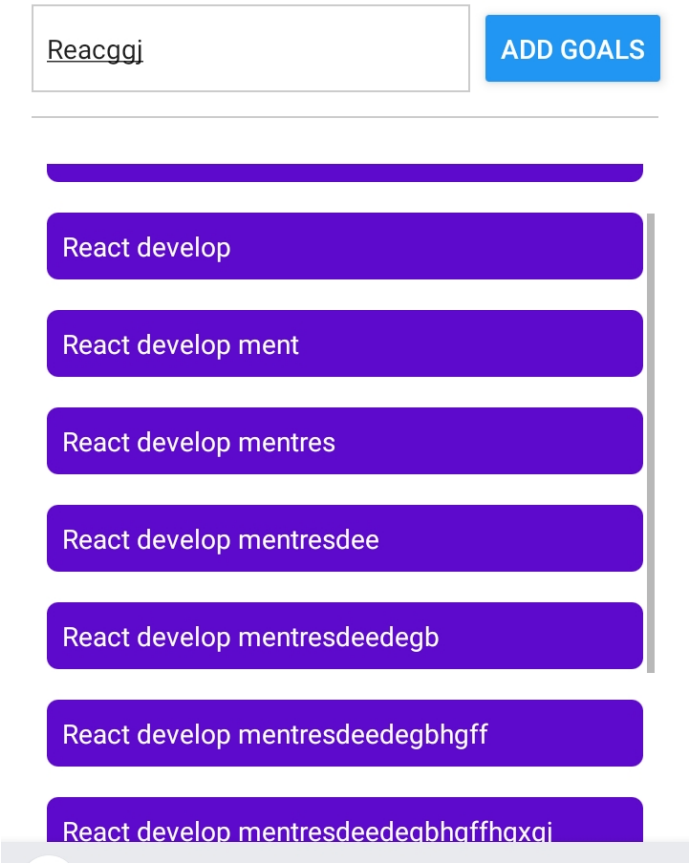
Output:



18.Adding an android ripple effect In GoalItem.js(components)

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

```
function GoalItem(props) {
  return (

        <View style={styles.goalItem}>
            <Pressable  android_ripple={{ color: '#210644' }}
            onPress={props.onDeleteItem.bind(this, props.id)}
            style={({ pressed }) => pressed && styles.pressedItem}
            >
            <Text style={styles.goalText}>{props.text}</Text>
            </Pressable>


        </View>


  );
}
export default GoalItem;

const styles = StyleSheet.create({
  goalItem: {
    margin: 8,
    padding: 8,
    borderRadius: 6,
    backgroundColor: '#5e0acc',
  },
  pressedItem: {
    opacity: 0.5,
  },
  goalText: {
    color: 'white',
  },
});
```



19.Opening and closing the modal

In app.js

```
import { useState } from 'react';
import { StyleSheet, View, FlatList, Button } from 'react-native';

import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';

export default function App() {
//creating the modal  to  display the files and functions to control the
model  closing  and  opening
  const [modalIsVisible, setModalIsVisible] = useState(false);
  const [courseGoals, setCourseGoals] = useState([]);

  function startAddGoalHandler() {
    setModalIsVisible(true);
  }
  function endAddGoalHandler() {
```

```
    setModalIsVisible(false);
  }

  function addGoalHandler(enteredGoalText) {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() },
    ]);
    endAddGoalHandler();
  }

  function deleteGoalHandler(id) {
    setCourseGoals((currentCourseGoals) => {
      return currentCourseGoals.filter((goal) => goal.id !== id);
    });
  }

  return (
    <View style={styles.appContainer}>
      <Button
        title="Add New Goal"
        color="#5e0acc"
        onPress={startAddGoalHandler}
      />
      <GoalInput visible={modalIsVisible} onAddGoal={addGoalHandler}
onCancel={endAddGoalHandler}
/>
      <View style={styles.goalsContainer}>
        <FlatList
          data={courseGoals}
          renderItem={(itemData) => {
            return (
              <GoalItem
                text={itemData.item.text}
                id={itemData.item.id}
                onDeleteItem={deleteGoalHandler}
              />
            );
          }}
          keyExtractor={(item, index) => {
```

```
            return item.id;
          }}
          alwaysBounceVertical={false}
        />
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16,
  },
  goalsContainer: {
    flex: 5,
  },
});
```

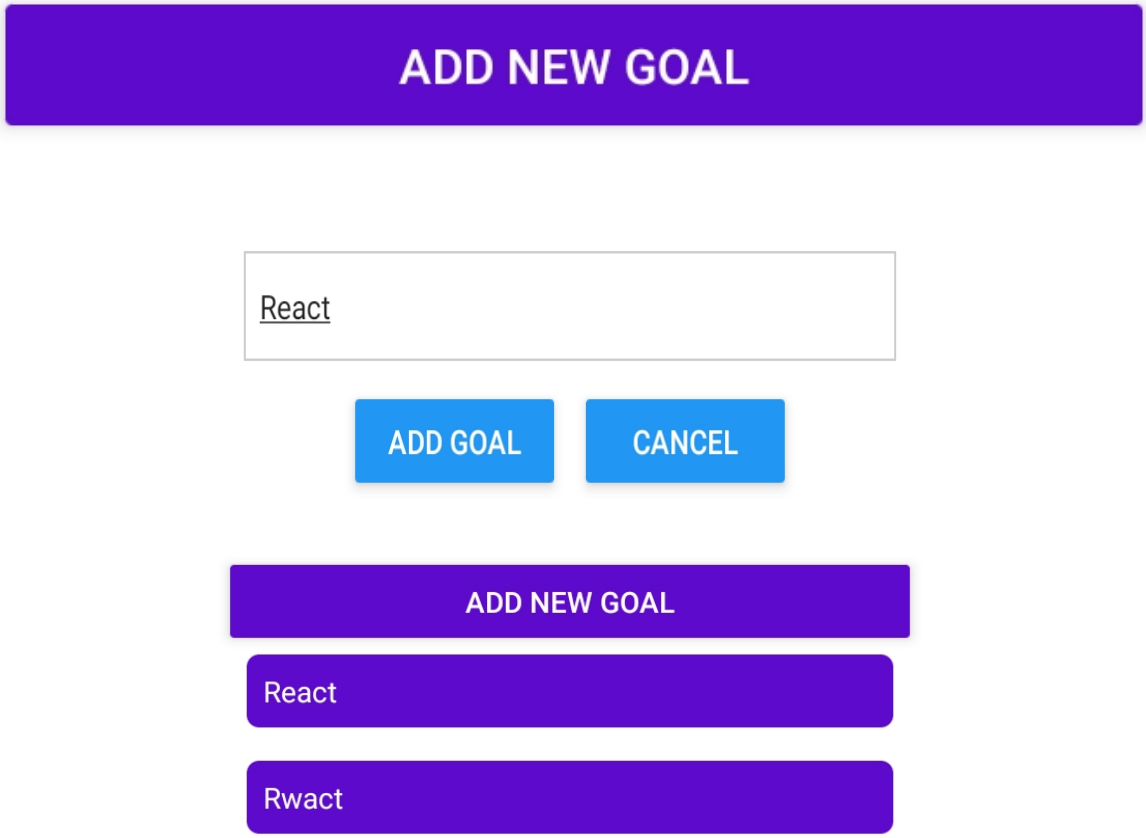
In GoalInput.js

```
import { useState } from 'react';
import { View, TextInput, Button, StyleSheet, Modal } from 'react-native';

function GoalInput(props) {
  const [enteredGoalText, setEnteredGoalText] = useState('');

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    props.onAddGoal(enteredGoalText);
    setEnteredGoalText('');
  }

  return (
    <Modal visible={props.visible} animationType="slide">
```

```
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.textInput}
          placeholder="Your course goal!"
          onChangeText={goalInputHandler}
          value={enteredGoalText}
        />
        <View style={styles.buttonContainer}>
          <View style={styles.button}>
            <Button title="Add Goal" onPress={addGoalHandler} />
          </View>
          <View style={styles.button} onPress={props.onCancel}
color="#f31282"
>
            <Button title="Cancel" />
          </View>
        </View>
      </View>
    </Modal>
  );
}

export default GoalInput;

const styles = StyleSheet.create({
  inputContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    marginBottom: 24,
    padding: 16,
    borderBottomWidth: 1,
    borderBottomColor: '#cccccc',
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '100%',
    padding: 8,
  },
```

```
  buttonContainer: {
    marginTop: 16,
    flexDirection: 'row',
  },
  button: {
    width: 100,
    marginHorizontal: 8
  }
});
```

Output:



20.Working with images and changing colors

In app.js

```
import { useState } from 'react';
import { StyleSheet, View, FlatList, Button } from 'react-native';

import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';
```

```
export default function App() {
  const [modalIsVisible, setModalIsVisible] = useState(false);
  const [courseGoals, setCourseGoals] = useState([]);

  function startAddGoalHandler() {
    setModalIsVisible(true);
  }

  function endAddGoalHandler() {
    setModalIsVisible(false);
  }

  function addGoalHandler(enteredGoalText) {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() },
    ]);
    endAddGoalHandler();
  }

  function deleteGoalHandler(id) {
    setCourseGoals((currentCourseGoals) => {
      return currentCourseGoals.filter((goal) => goal.id !== id);
    });
  }

  return (
    <View style={styles.appContainer}>
      <Button
        title="Add New Goal"
        color="#5e0acc"
        onPress={startAddGoalHandler}
      />
      <GoalInput
      visible={modalIsVisible}
      onAddGoal={addGoalHandler}
      onCancel={endAddGoalHandler}
      />
      <View style={styles.goalsContainer}>
```

```
        <FlatList
          data={courseGoals}
          renderItem={(itemData) => {
            return (
              <GoalItem
                text={itemData.item.text}
                id={itemData.item.id}
                onDeleteItem={deleteGoalHandler}
              />
            );
          }}
          keyExtractor={(item, index) => {
            return item.id;
          }}
          alwaysBounceVertical={false}
        />
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16,
  },
  goalsContainer: {
    flex: 5,
  },
});
```

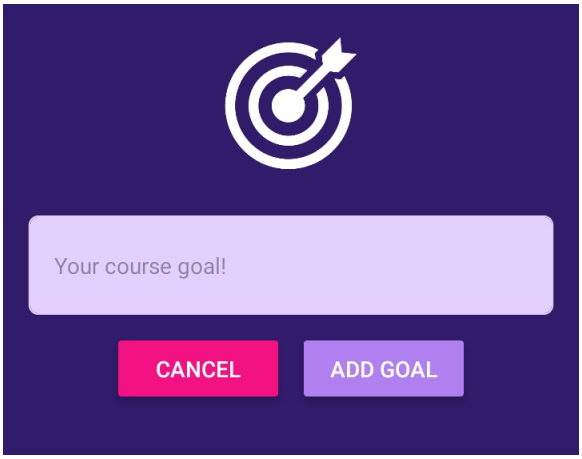
In GoalInput.js(components)

```
import { useState } from 'react';
import {
  View,
  TextInput,
  Button,
  StyleSheet,
```

```
  Modal,
  Image,
} from 'react-native';

function GoalInput(props) {
  const [enteredGoalText, setEnteredGoalText] = useState('');

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    props.onAddGoal(enteredGoalText);
    setEnteredGoalText('');
  }

  return (
    <Modal visible={props.visible} animationType="slide">
      <View style={styles.inputContainer}>
        <Image
          style={styles.image}
          source={require('../assets/images/goal.png')}
        />
        <TextInput
          style={styles.textInput}
          placeholder="Your course goal!"
          onChangeText={goalInputHandler}
          value={enteredGoalText}
        />
        <View style={styles.buttonContainer}>
          <View style={styles.button}>
            <Button title="Cancel" onPress={props.onCancel}
color="#f31282" />
          </View>
          <View style={styles.button}>
            <Button title="Add Goal" onPress={addGoalHandler}
color="#b180f0" />
          </View>
        </View>
      </View>
```

```
    </Modal>
  );
}

export default GoalInput;

const styles = StyleSheet.create({
  inputContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    marginBottom: 24,
    padding: 16,
    backgroundColor: '#311b6b',
  },
  image: {
    width: 100,
    height: 100,
    margin: 20,
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#e4d0ff',
    backgroundColor: '#e4d0ff',
    color: '#120438',
    borderRadius: 6,
    width: '100%',
    padding: 16,
  },
  buttonContainer: {
    marginTop: 16,
    flexDirection: 'row',
  },
  button: {
    width: 100,
    marginHorizontal: 8,
  },
});
```

Output:



20. Status bar component

Component to control the app's status bar.

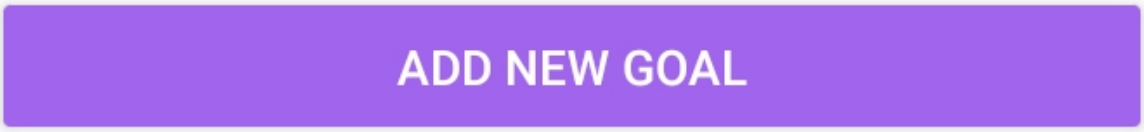
In app.js

```
import { useState } from 'react';
import { StyleSheet, View, FlatList, Button } from 'react-native';
import { StatusBar } from 'expo-status-bar';

import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';

export default function App() {
  const [modalIsVisible, setModalIsVisible] = useState(false);
  const [courseGoals, setCourseGoals] = useState([]);

  function startAddGoalHandler() {
    setModalIsVisible(true);
  }

  function endAddGoalHandler() {
    setModalIsVisible(false);
  }

  function addGoalHandler(enteredGoalText) {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() },
    ]);
```

```
  endAddGoalHandler();
}

function deleteGoalHandler(id) {
  setCourseGoals((currentCourseGoals) => {
    return currentCourseGoals.filter((goal) => goal.id !== id);
  });
}

return (
  <>
    <StatusBar style="light" />
  <View style={styles.appContainer}>
    <Button
      title="Add New Goal"
        color="#a065ec"
      onPress={startAddGoalHandler}
    />
    <GoalInput
    visible={modalIsVisible}
    onAddGoal={addGoalHandler}
    onCancel={endAddGoalHandler}
    />
    <View style={styles.goalsContainer}>
      <FlatList
        data={courseGoals}
        renderItem={(itemData) => {
          return (
            <GoalItem
              text={itemData.item.text}
              id={itemData.item.id}
              onDeleteItem={deleteGoalHandler}
            />
          );
        }}
        keyExtractor={(item, index) => {
          return item.id;
        }}
        alwaysBounceVertical={false}
      />
```

```
      </View>
    </View>
    </>
  );
}

const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16,
  },
  goalsContainer: {
    flex: 5,
  },
});
```

Output:



Project -2 :guessing game app

1.Setting up screen components

Create 2 folders

1)components(PrimaryButton.js)

2)screens(StartGameSCreen.js,GameOverScreen.js, GameScreen.js)

Code:

In app.js

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

```
import StartGameScreen from './screens/StartGameScreen';

export default function App() {
  return <StartGameScreen />;
}

const styles = StyleSheet.create({});
```

In PrimaryButton.js

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

function PrimaryButton({ children }) {
  return (
    <View>
      <Text>{children}</Text>
    </View>
  );
}

export default PrimaryButton;
```

In StartGameSCreen.js

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

import PrimaryButton from '../components/PrimaryButton';

function StartGameScreen() {
  return (
    <View>
      <TextInput />
      <PrimaryButton>Reset</PrimaryButton>
      <PrimaryButton>Confirm</PrimaryButton>
    </View>
  );
}

export default StartGameScreen;
```

Output:

2.Styling the Number Input element
In StartGameScreen.js

```
import { TextInput, View, StyleSheet } from 'react-native';

import PrimaryButton from '../components/PrimaryButton';

function StartGameScreen() {
  return (
    <View style={styles.inputContainer}>
      <TextInput style={styles.numberInput} maxLength={2} />
      <PrimaryButton>Reset</PrimaryButton>
      <PrimaryButton>Confirm</PrimaryButton>
    </View>
  );
}

export default StartGameScreen;

const styles = StyleSheet.create({
  inputContainer: {
    marginTop: 100,
    marginHorizontal: 24,
    padding: 16,
    backgroundColor: '#72063c',
    borderRadius: 8,
    elevation: 4,
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 6,
    shadowOpacity: 0.25
```

```
  },
  numberInput: {
    height: 50,
    width: 50,
    fontSize: 32,
    borderBottomColor: '#ddb52f',
    borderBottomWidth: 2,
    color: '#ddb52f',
    marginVertical: 8,
    fontWeight: 'bold',
    textAlign: 'center'
  }
});
```

Output:

3.configuring text input
In start game screen.js

```
import { TextInput, View, StyleSheet } from 'react-native';

import PrimaryButton from '../components/PrimaryButton';

function StartGameScreen() {
  return (
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.numberInput}
        maxLength={2}
        keyboardType="number-pad"
        autoCapitalize="none"
```

```
        autoCorrect={false}
      />
      <PrimaryButton>Reset</PrimaryButton>
      <PrimaryButton>Confirm</PrimaryButton>
    </View>
  );
}

export default StartGameScreen;

const styles = StyleSheet.create({
  inputContainer: {
    marginTop: 100,
    marginHorizontal: 24,
    padding: 16,
    backgroundColor: '#72063c',
    borderRadius: 8,
    elevation: 4,
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 6,
    shadowOpacity: 0.25,
  },
  numberInput: {
    height: 50,
    width: 50,
    fontSize: 32,
    borderBottomColor: '#ddb52f',
    borderBottomWidth: 2,
    color: '#ddb52f',
    marginVertical: 8,
    fontWeight: 'bold',
    textAlign: 'center',
  },
});
```

Output:

4.Improving the buttons
In primarybutton.js(components)

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

function PrimaryButton({ children }) {
  function pressHandler() {
    console.log('Pressed!');
  }

  return (
    <View style={styles.buttonOuterContainer}>
      <Pressable
        style={({ pressed }) =>
          pressed
            ? [styles.buttonInnerContainer, styles.pressed]
            : styles.buttonInnerContainer
        }
//pass the  call function  on press
        onPress={pressHandler}

        android_ripple={{ color: '#640233' }}
      >
        <Text style={styles.buttonText}>{children}</Text>
      </Pressable>
    </View>
  );
}

export default PrimaryButton;

const styles = StyleSheet.create({
  buttonOuterContainer: {
```

```
    borderRadius: 28,
    margin: 4,
    overflow: 'hidden',
  },
  buttonInnerContainer: {
    backgroundColor: '#72063c',
    paddingVertical: 8,
    paddingHorizontal: 16,
    elevation: 2,
  },
  buttonText: {
    color: 'white',
    textAlign: 'center',
  },
  pressed: {
    opacity: 0.75,
  },
});
```

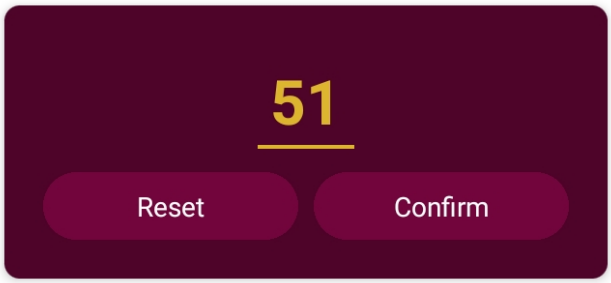
In StartGameScreen.js

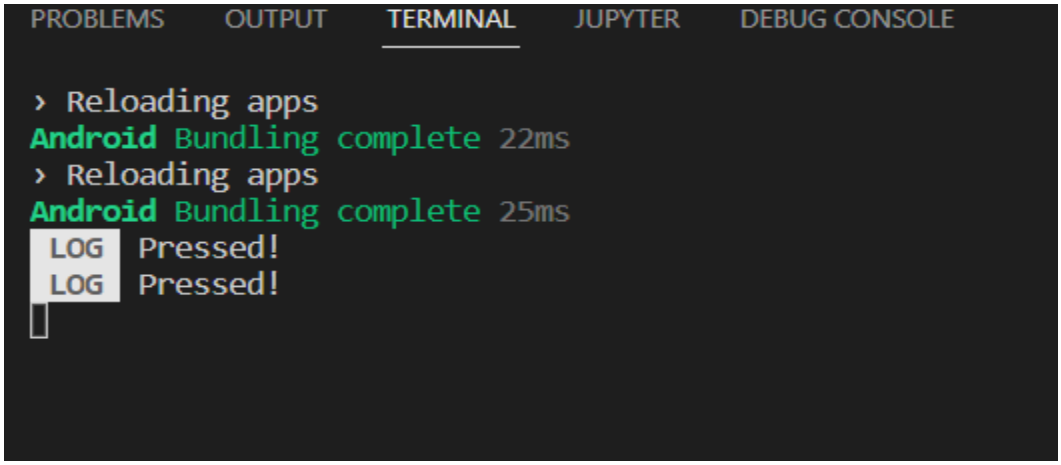
```
import { TextInput, View, StyleSheet } from 'react-native';

import PrimaryButton from '../components/PrimaryButton';

function StartGameScreen() {
  return (
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.numberInput}
        maxLength={2}
        keyboardType="number-pad"
        autoCapitalize="none"
        autoCorrect={false}
      />
      <View style={styles.buttonsContainer}>
        <View style={styles.buttonContainer}>
          <PrimaryButton>Reset</PrimaryButton>
        </View>
        <View style={styles.buttonContainer}>
```

```
          <PrimaryButton>Confirm</PrimaryButton>
        </View>
      </View>
    </View>
  );
}

export default StartGameScreen;

const styles = StyleSheet.create({
  inputContainer: {
    justifyContent: 'center',
    alignItems: 'center',
    marginTop: 100,
    marginHorizontal: 24,
    padding: 16,
    backgroundColor: '#4e0329',
    borderRadius: 8,
    elevation: 4,
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 6,
    shadowOpacity: 0.25,
  },
  numberInput: {
    height: 50,
    width: 50,
    fontSize: 32,
    borderBottomColor: '#ddb52f',
    borderBottomWidth: 2,
    color: '#ddb52f',
    marginVertical: 8,
    fontWeight: 'bold',
    textAlign: 'center',
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1
```

```
  }
});
```

Output:



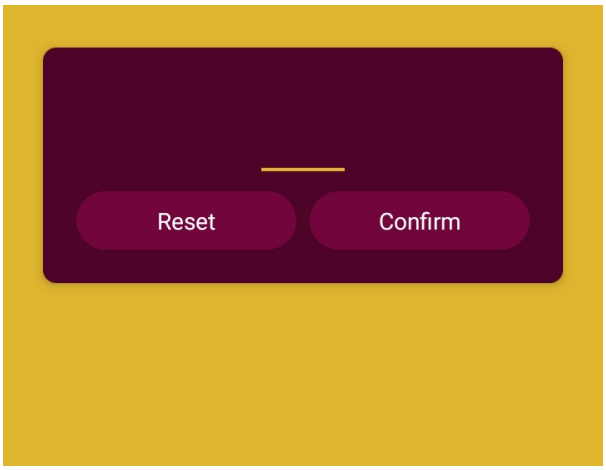


5.coloring components overall app

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

import StartGameScreen from './screens/StartGameScreen';

export default function App() {
  return (
    <View style={styles.rootScreen}>
      <StartGameScreen />
    </View>
  );
}

const styles = StyleSheet.create({
```

```
  rootScreen: {
    flex: 1,
    backgroundColor: '#ddb52f'
  }
});
```

Output:



6.Adding linear gradient

Linear gradient:sets multiple color background

Use:

Install :

```
npx expo install expo-linear-gradient
```

Code:

```
import { StyleSheet } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';

import StartGameScreen from './screens/StartGameScreen';

export default function App() {
  return (
    <LinearGradient colors={['#4e0329', '#ddb52f']}
style={styles.rootScreen}>
      <StartGameScreen />
```

```
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
  rootScreen: {
    flex: 1,
  },
});
```

7.adding background image

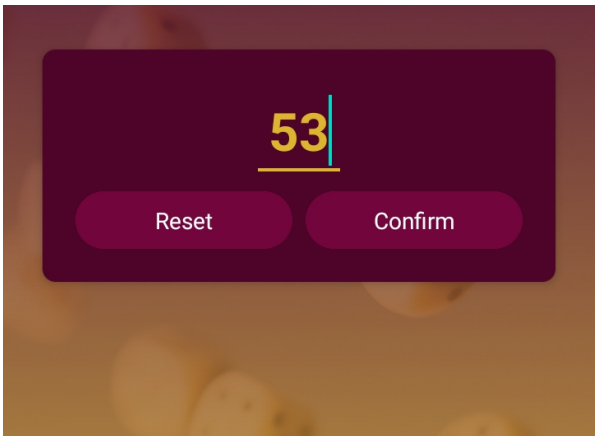
1.in assets/images folder save the image

Code:

```
import { StyleSheet, ImageBackground } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';

import StartGameScreen from './screens/StartGameScreen';

export default function App() {
  return (
    <LinearGradient colors={['#4e0329', '#ddb52f']}
style={styles.rootScreen}>
      <ImageBackground
        source={require('./assets/images/background.png')}
        resizeMode="cover"
        style={styles.rootScreen}
        imageStyle={styles.backgroundImage}
      >
        <StartGameScreen />
      </ImageBackground>
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
  rootScreen: {
    flex: 1,
  },
  backgroundImage: {
```

```
    opacity: 0.15
  }
});
```

Output:



8.game logic

In startgamescreen.js

```
import { useState } from 'react';
import { TextInput, View, StyleSheet, Alert } from 'react-native';

import PrimaryButton from '../components/PrimaryButton';

function StartGameScreen() {
  const [enteredNumber, setEnteredNumber] = useState('');

  function numberInputHandler(enteredText) {
    setEnteredNumber(enteredText);
  }

  function resetInputHandler() {
    setEnteredNumber('');
  }

  function confirmInputHandler() {
    const chosenNumber = parseInt(enteredNumber);
// checking the number between 0  and 99
```

```
    if (isNaN(chosenNumber) || chosenNumber <= 0 || chosenNumber > 99) {
     //alert message
      Alert.alert(
        'Invalid number!',
        'Number has to be a number between 1 and 99.',
        [{ text: 'Okay', style: 'destructive', onPress: resetInputHandler
}]
      );
      return;
    }

    console.log('Valid number!');
  }

  return (
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.numberInput}
        maxLength={2}
        keyboardType="number-pad"
        autoCapitalize="none"
        autoCorrect={false}
        onChangeText={numberInputHandler}
        value={enteredNumber}
      />
      <View style={styles.buttonsContainer}>
        <View style={styles.buttonContainer}>
          <PrimaryButton onPress={resetInputHandler}>Reset</PrimaryButton>
        </View>
        <View style={styles.buttonContainer}>
          <PrimaryButton
     onPress={confirmInputHandler}>Confirm</PrimaryButton>
        </View>
      </View>
    </View>
  );
}

export default StartGameScreen;
```

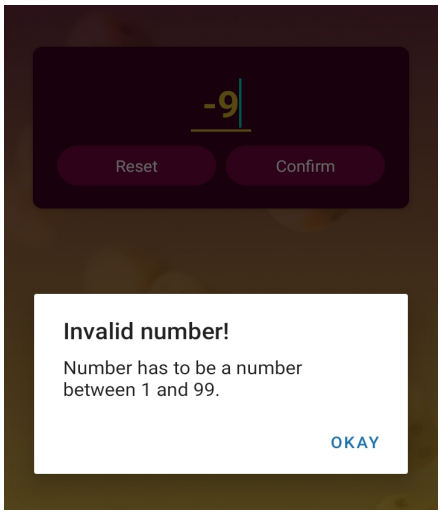
```
const styles = StyleSheet.create({
  inputContainer: {
    justifyContent: 'center',
    alignItems: 'center',
    marginTop: 100,
    marginHorizontal: 24,
    padding: 16,
    backgroundColor: '#3b021f',
    borderRadius: 8,
    elevation: 4,
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 6,
    shadowOpacity: 0.25,
  },
  numberInput: {
    height: 50,
    width: 50,
    fontSize: 32,
    borderBottomColor: '#ddb52f',
    borderBottomWidth: 2,
    color: '#ddb52f',
    marginVertical: 8,
    fontWeight: 'bold',
    textAlign: 'center',
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
});
```

In primary button.js(components)

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

function PrimaryButton({ children, onPress }) {
  return (
```

```
    <View style={styles.buttonOuterContainer}>
      <Pressable
        style={({ pressed }) =>
          pressed
            ? [styles.buttonInnerContainer, styles.pressed]
            : styles.buttonInnerContainer
        }
        onPress={onPress}
        android_ripple={{ color: '#640233' }}
      >
        <Text style={styles.buttonText}>{children}</Text>
      </Pressable>
    </View>
  );
}

export default PrimaryButton;

const styles = StyleSheet.create({
  buttonOuterContainer: {
    borderRadius: 28,
    margin: 4,
    overflow: 'hidden',
  },
  buttonInnerContainer: {
    backgroundColor: '#72063c',
    paddingVertical: 8,
    paddingHorizontal: 16,
    elevation: 2,
  },
  buttonText: {
    color: 'white',
    textAlign: 'center',
  },
  pressed: {
    opacity: 0.75,
  },
});
```

Output:



9.switching screens

In app.js

```
import { useState } from 'react';
import { StyleSheet, ImageBackground } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';

import StartGameScreen from './screens/StartGameScreen';
import GameScreen from './screens/GameScreen';

export default function App() {
  const [userNumber, setUserNumber] = useState();

  function pickedNumberHandler(pickedNumber) {
    setUserNumber(pickedNumber);
  }

//switching the screens if the user number is valid
  let screen = <StartGameScreen onPickNumber={pickedNumberHandler} />;

  if (userNumber) {
    screen = <GameScreen />;
  }

  return (
```

```
    <LinearGradient colors={['#4e0329', '#ddb52f']}
style={styles.rootScreen}>
      <ImageBackground
        source={require('./assets/images/background.png')}
        resizeMode="cover"
        style={styles.rootScreen}
        imageStyle={styles.backgroundImage}
      >
        {screen}
      </ImageBackground>
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
  rootScreen: {
    flex: 1,
  },
  backgroundImage: {
    opacity: 0.15,
  },
});
```

In gamescreen.js

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

function GameScreen() {
  return <Text>Game Screen!</Text>
}

export default GameScreen;
```

In startgamescreen.js

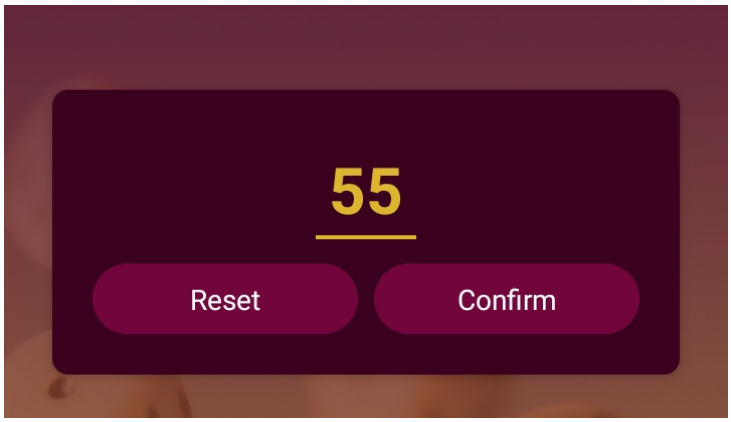
```
import { useState } from 'react';
import { TextInput, View, StyleSheet, Alert } from 'react-native';

import PrimaryButton from '../components/PrimaryButton';

function StartGameScreen({onPickNumber}) {
```

```
  const [enteredNumber, setEnteredNumber] = useState('');

  function numberInputHandler(enteredText) {
    setEnteredNumber(enteredText);
  }

  function resetInputHandler() {
    setEnteredNumber('');
  }

  function confirmInputHandler() {
    const chosenNumber = parseInt(enteredNumber);

    if (isNaN(chosenNumber) || chosenNumber <= 0 || chosenNumber > 99) {
      Alert.alert(
        'Invalid number!',
        'Number has to be a number between 1 and 99.',
        [{ text: 'Okay', style: 'destructive', onPress: resetInputHandler
}]
      );
      return;
    }

    onPickNumber(chosenNumber);
  }

  return (
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.numberInput}
        maxLength={2}
        keyboardType="number-pad"
        autoCapitalize="none"
        autoCorrect={false}
        onChangeText={numberInputHandler}
        value={enteredNumber}
      />
      <View style={styles.buttonsContainer}>
        <View style={styles.buttonContainer}>
          <PrimaryButton onPress={resetInputHandler}>Reset</PrimaryButton>
```

```
        </View>
        <View style={styles.buttonContainer}>
          <PrimaryButton
onPress={confirmInputHandler}>Confirm</PrimaryButton>
    </View>
      </View>
    </View>
  );
}

export default StartGameScreen;

const styles = StyleSheet.create({
  inputContainer: {
    justifyContent: 'center',
    alignItems: 'center',
    marginTop: 100,
    marginHorizontal: 24,
    padding: 16,
    backgroundColor: '#3b021f',
    borderRadius: 8,
    elevation: 4,
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 6,
    shadowOpacity: 0.25,
  },
  numberInput: {
    height: 50,
    width: 50,
    fontSize: 32,
    borderBottomColor: '#ddb52f',
    borderBottomWidth: 2,
    color: '#ddb52f',
    marginVertical: 8,
    fontWeight: 'bold',
    textAlign: 'center',
  },
  buttonsContainer: {
    flexDirection: 'row',
```

```
  },
  buttonContainer: {
    flex: 1,
  },
});
```

Output:



10.showing up the screen and safeareaview

Safeareaview:to make sure that our text do not go over the notch.

In game screen.js

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

function GameScreen() {
  return (
    <View style={styles.screen}>
      <Text>Opponent's Guess</Text>
      {/* GUESS */}
      <View>
        <Text>Higher or lower?</Text>
        {/* + - */}
      </View>
      {/* <View>LOG ROUNDS</View> */}
    </View>
  );
}

export default GameScreen;
```

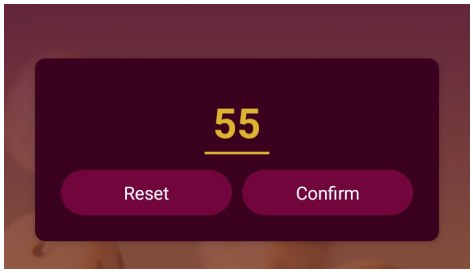
```
const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24
  }
});
```

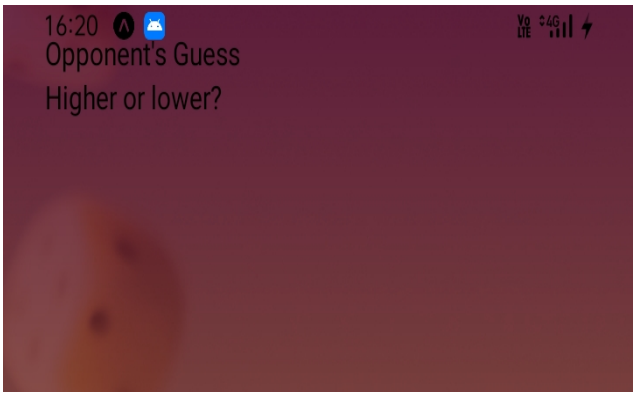
In app.js

```
import { useState } from 'react';
import { StyleSheet, ImageBackground,SafeAreaView } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';

import StartGameScreen from './screens/StartGameScreen';
import GameScreen from './screens/GameScreen';

export default function App() {
  const [userNumber, setUserNumber] = useState();

  function pickedNumberHandler(pickedNumber) {
    setUserNumber(pickedNumber);
  }

  let screen = <StartGameScreen onPickNumber={pickedNumberHandler} />;

  if (userNumber) {
    screen = <GameScreen />;
  }

  return (
    <LinearGradient colors={['#4e0329', '#ddb52f']}
style={styles.rootScreen}>
      <ImageBackground
        source={require('./assets/images/background.png')}
        resizeMode="cover"
        style={styles.rootScreen}
        imageStyle={styles.backgroundImage}
      >
         <SafeAreaView style={styles.rootScreen}>{screen}</SafeAreaView>
```

```
      </ImageBackground>
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
  rootScreen: {
    flex: 1,
  },
  backgroundImage: {
    opacity: 0.15,
  },
});
```

Output:





11.creating atitle component and managing colors globally In gamescreen.js

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

import Title from '../components/Title';

function GameScreen() {
  return (
    <View style={styles.screen}>
      <Title>Opponent's Guess</Title>
      {/* GUESS */}
      <View>
        <Text>Higher or lower?</Text>
```

```
        {/* + - */}
      </View>
      {/* <View>LOG ROUNDS</View> */}
    </View>
  );
}

export default GameScreen;

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24
  }
});
```

In title.js

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

import Colors from '../constants/colors';

function Title({ children }) {
  return <Text style={styles.title}>{children}</Text>;
}

export default Title;

const styles = StyleSheet.create({
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    color: Colors.accent500,
    textAlign: 'center',
    borderWidth: 2,
    borderColor: Colors.accent500,
    padding: 12,
  },
});
```

Output:

Creating colors globally
In startgamescreen.js

```
import { useState } from 'react';
import { TextInput, View, StyleSheet, Alert } from 'react-native';

import PrimaryButton from '../components/PrimaryButton';
import Colors from '../constants/colors';

function StartGameScreen({onPickNumber}) {
  const [enteredNumber, setEnteredNumber] = useState('');

  function numberInputHandler(enteredText) {
    setEnteredNumber(enteredText);
  }

  function resetInputHandler() {
    setEnteredNumber('');
  }

  function confirmInputHandler() {
    const chosenNumber = parseInt(enteredNumber);

    if (isNaN(chosenNumber) || chosenNumber <= 0 || chosenNumber > 99) {
      Alert.alert(
        'Invalid number!',
        'Number has to be a number between 1 and 99.',
        [{ text: 'Okay', style: 'destructive', onPress: resetInputHandler
}]
      );
      return;
    }
```

```
    onPickNumber(chosenNumber);
  }

  return (
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.numberInput}
        maxLength={2}
        keyboardType="number-pad"
        autoCapitalize="none"
        autoCorrect={false}
        onChangeText={numberInputHandler}
        value={enteredNumber}
      />
      <View style={styles.buttonsContainer}>
        <View style={styles.buttonContainer}>
          <PrimaryButton onPress={resetInputHandler}>Reset</PrimaryButton>
        </View>
        <View style={styles.buttonContainer}>
          <PrimaryButton
onPress={confirmInputHandler}>Confirm</PrimaryButton>
        </View>
      </View>
    </View>
  );
}

export default StartGameScreen;

const styles = StyleSheet.create({
  inputContainer: {
    justifyContent: 'center',
    alignItems: 'center',
    marginTop: 100,
    marginHorizontal: 24,
    padding: 16,
    backgroundColor: Colors.primary800,
    borderRadius: 8,
    elevation: 4,
    shadowColor: 'black',
```

```
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 6,
    shadowOpacity: 0.25,
  },
  numberInput: {
    height: 50,
    width: 50,
    fontSize: 32,
    borderBottomColor: Colors.accent500,
    borderBottomWidth: 2,
    color: Colors.accent500,
    marginVertical: 8,
    fontWeight: 'bold',
    textAlign: 'center',
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
});
```

In color.js

```
const Colors = {
    primary500: '#72063c',
    primary600: '#640233',
    primary700: '#4e0329',
    primary800: '#3b021f',
    accent500: '#ddb52f'
  };

  export default Colors;
```

In primarybutton.js

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

import Colors from '../constants/colors';

function PrimaryButton({ children, onPress }) {
```

```
  return (
    <View style={styles.buttonOuterContainer}>
      <Pressable
        style={({ pressed }) =>
          pressed
            ? [styles.buttonInnerContainer, styles.pressed]
            : styles.buttonInnerContainer
        }
        onPress={onPress}
        android_ripple={{ color: Colors.primary600 }}
      >
        <Text style={styles.buttonText}>{children}</Text>
      </Pressable>
    </View>
  );
}

export default PrimaryButton;

const styles = StyleSheet.create({
  buttonOuterContainer: {
    borderRadius: 28,
    margin: 4,
    overflow: 'hidden',
  },
  buttonInnerContainer: {
    backgroundColor: Colors.primary500,
    paddingVertical: 8,
    paddingHorizontal: 16,
    elevation: 2,
  },
  buttonText: {
    color: 'white',
    textAlign: 'center',
  },
  pressed: {
    opacity: 0.75,
  },
});
```

12.creating -using-displaying-random number
In game screen.js

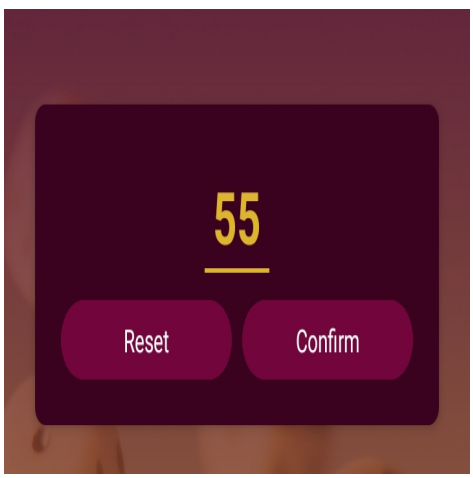
```
import { useState } from 'react';
import { View, Text, StyleSheet } from 'react-native';

import NumberContainer from '../components/game/NumberContainer';
import Title from '../components/ui/Title';

function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
  }
}

function GameScreen({ userNumber }) {
  const initialGuess = generateRandomBetween(1, 100, userNumber);
  const [currentGuess, setCurrentGuess] = useState(initialGuess);

  return (
    <View style={styles.screen}>
      <Title>Opponent's Guess</Title>
      <NumberContainer>{currentGuess}</NumberContainer>
      <View>
        <Text>Higher or lower?</Text>
        {/* + - */}
      </View>
      {/* <View>LOG ROUNDS</View> */}
    </View>
  );
}

export default GameScreen;

const styles = StyleSheet.create({
  screen: {
```

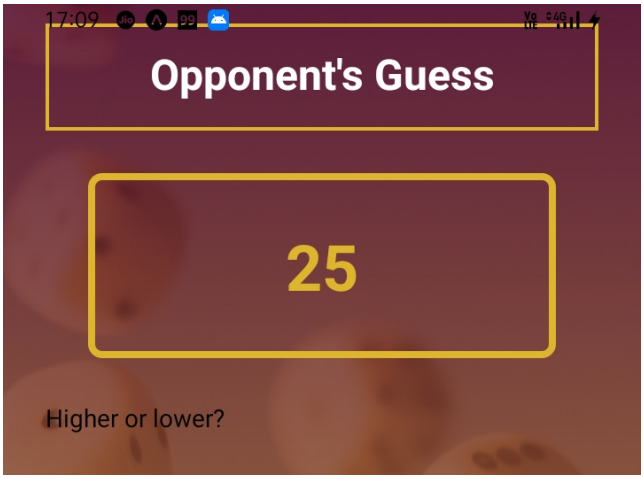
```
    flex: 1,
    padding: 24,
  },
});
```

In numberconatiner.js

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

import Colors from '../../constants/colors';

function NumberContainer({ children }) {
  return (
    <View style={styles.container}>
      <Text style={styles.numberText}>{children}</Text>
    </View>
  );
}

export default NumberContainer;

const styles = StyleSheet.create({
  container: {
    borderWidth: 4,
    borderColor: Colors.accent500,
    padding: 24,
    margin: 24,
    borderRadius: 8,
    alignItems: 'center',
    justifyContent: 'center',
  },
  numberText: {
    color: Colors.accent500,
    fontSize: 36,
    fontWeight: 'bold',
  },
});
```

Output:





13.adding game control buttons

In app.js

```
import { useState } from 'react';
import { StyleSheet, ImageBackground, SafeAreaView } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';

import StartGameScreen from './screens/StartGameScreen';
import GameScreen from './screens/GameScreen';
import Colors from './constants/colors';

export default function App() {
  const [userNumber, setUserNumber] = useState();

  function pickedNumberHandler(pickedNumber) {
    setUserNumber(pickedNumber);
  }

  let screen = <StartGameScreen onPickNumber={pickedNumberHandler} />;

  if (userNumber) {
    screen = <GameScreen userNumber={userNumber} />;
  }

  return (
    <LinearGradient
```

```
      colors={[Colors.primary700, Colors.accent500]}
      style={styles.rootScreen}
    >
      <ImageBackground
        source={require('./assets/images/background.png')}
        resizeMode="cover"
        style={styles.rootScreen}
        imageStyle={styles.backgroundImage}
      >
        <SafeAreaView style={styles.rootScreen}>{screen}</SafeAreaView>
      </ImageBackground>
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
  rootScreen: {
    flex: 1,
  },
  backgroundImage: {
    opacity: 0.15,
  },
});
```

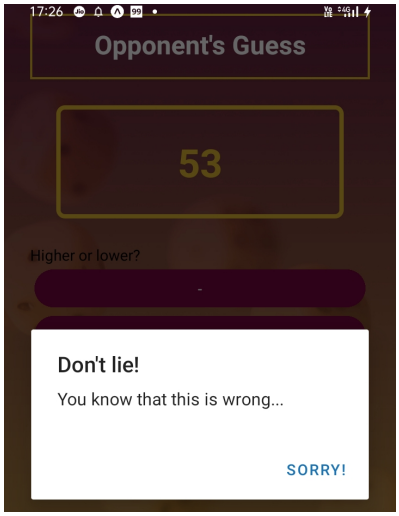
In game screen.js

```
import { useState } from 'react';
import { View, Text, StyleSheet, Alert } from 'react-native';

import NumberContainer from '../components/game/NumberContainer';
import PrimaryButton from '../components/ui/PrimaryButton';
import Title from '../components/ui/Title';

function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
```

```
  }
}

let minBoundary = 1;
let maxBoundary = 100;

function GameScreen({ userNumber }) {
  const initialGuess = generateRandomBetween(
    minBoundary,
    maxBoundary,
    userNumber
  );
  const [currentGuess, setCurrentGuess] = useState(initialGuess);

  function nextGuessHandler(direction) {
    // direction => 'lower', 'greater'
    if (
      (direction === 'lower' && currentGuess < userNumber) ||
      (direction === 'greater' && currentGuess > userNumber)
    ) {
      Alert.alert("Don't lie!", 'You know that this is wrong...', [
        { text: 'Sorry!', style: 'cancel' },
      ]);
      return;
    }

    if (direction === 'lower') {
      maxBoundary = currentGuess;
    } else {
      minBoundary = currentGuess + 1;
    }

    const newRndNumber = generateRandomBetween(
      minBoundary,
      maxBoundary,
      currentGuess
    );
    setCurrentGuess(newRndNumber);
  }
```

```
  return (
    <View style={styles.screen}>
      <Title>Opponent's Guess</Title>
      <NumberContainer>{currentGuess}</NumberContainer>
      <View>
        <Text>Higher or lower?</Text>
        <View>
          <PrimaryButton onPress={nextGuessHandler.bind(this, 'lower')}>
            -
          </PrimaryButton>
          <PrimaryButton onPress={nextGuessHandler.bind(this, 'greater')}>
            +
          </PrimaryButton>
        </View>
      </View>
      {/* <View>LOG ROUNDS</View> */}
    </View>
  );
}

export default GameScreen;

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24,
  },
});
```

Output:



14.checking for game over
In app.js

```
import { useState } from 'react';
import { StyleSheet, ImageBackground, SafeAreaView } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';

import StartGameScreen from './screens/StartGameScreen';
import GameScreen from './screens/GameScreen';
import GameOverScreen from './screens/GameOverScreen';
import Colors from './constants/colors';

export default function App() {
  const [userNumber, setUserNumber] = useState();
  const [gameIsOver, setGameIsOver] = useState(true);

  function pickedNumberHandler(pickedNumber) {
    setUserNumber(pickedNumber);
    setGameIsOver(false);
  }

  function gameOverHandler() {
    setGameIsOver(true);
  }

  let screen = <StartGameScreen onPickNumber={pickedNumberHandler} />;
```

```
  if (userNumber) {
    screen = (
      <GameScreen userNumber={userNumber} onGameOver={gameOverHandler} />
    );
  }

  if (gameIsOver && userNumber) {
    screen = <GameOverScreen />;
  }

  return (
    <LinearGradient
      colors={[Colors.primary700, Colors.accent500]}
      style={styles.rootScreen}
    >
      <ImageBackground
        source={require('./assets/images/background.png')}
        resizeMode="cover"
        style={styles.rootScreen}
        imageStyle={styles.backgroundImage}
      >
        <SafeAreaView style={styles.rootScreen}>{screen}</SafeAreaView>
      </ImageBackground>
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
  rootScreen: {
    flex: 1,
  },
  backgroundImage: {
    opacity: 0.15,
  },
});
```

In gamescreen.js

```
import { useState, useEffect } from 'react';
import { View, Text, StyleSheet, Alert } from 'react-native';
```

```
import NumberContainer from '../components/game/NumberContainer';
import PrimaryButton from '../components/ui/PrimaryButton';
import Title from '../components/ui/Title';

function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
  }
}

let minBoundary = 1;
let maxBoundary = 100;

function GameScreen({ userNumber, onGameOver }) {
  const initialGuess = generateRandomBetween(
    1,
    100,
    userNumber
  );
  const [currentGuess, setCurrentGuess] = useState(initialGuess);

  useEffect(() => {
    if (currentGuess === userNumber) {
      onGameOver();
    }
  }, [currentGuess, userNumber, onGameOver]);

  function nextGuessHandler(direction) {
    // direction => 'lower', 'greater'
    if (
      (direction === 'lower' && currentGuess < userNumber) ||
      (direction === 'greater' && currentGuess > userNumber)
    ) {
      Alert.alert("Don't lie!", 'You know that this is wrong...', [
        { text: 'Sorry!', style: 'cancel' },
```

```
      ]);
      return;
    }

    if (direction === 'lower') {
      maxBoundary = currentGuess;
    } else {
      minBoundary = currentGuess + 1;
    }

    const newRndNumber = generateRandomBetween(
      minBoundary,
      maxBoundary,
      currentGuess
    );
    setCurrentGuess(newRndNumber);
  }

  return (
    <View style={styles.screen}>
      <Title>Opponent's Guess</Title>
      <NumberContainer>{currentGuess}</NumberContainer>
      <View>
        <Text>Higher or lower?</Text>
        <View>
          <PrimaryButton onPress={nextGuessHandler.bind(this, 'lower')}>
            -
          </PrimaryButton>
          <PrimaryButton onPress={nextGuessHandler.bind(this, 'greater')}>
            +
          </PrimaryButton>
        </View>
      </View>
      {/* <View>LOG ROUNDS</View> */}
    </View>
  );
}

export default GameScreen;
```

```
const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24,
  },
});
```

In gameoverscreen.js

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

function GameOverScreen() {
  return <Text>Game is over!</Text>
}

export default GameOverScreen;
```

Output:

15. Improving game screen visuals

Code:

In card.js

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

import Colors from '../../constants/colors';

function Card({ children }) {
  return <View style={styles.card}>{children}</View>;
}

export default Card;

const styles = StyleSheet.create({
```

```
  card: {
    justifyContent: 'center',
    alignItems: 'center',
    marginTop: 36,
    marginHorizontal: 24,
    padding: 16,
    backgroundColor: Colors.primary800,
    borderRadius: 8,
    elevation: 4,
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 6,
    shadowOpacity: 0.25,
  },
});
```

In instructiontext.js

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

import Colors from '../../constants/colors';

function InstructionText({ children, style }) {
  return <Text style={[styles.instructionText, style]}>{children}</Text>;
}

export default InstructionText;

const styles = StyleSheet.create({
  instructionText: {
    color: Colors.accent500,
    fontSize: 24,
  },
});
```

In startgamescreen.js

```
import { useState } from 'react';
import { TextInput, View, StyleSheet, Alert } from 'react-native';

import PrimaryButton from '../components/ui/PrimaryButton';
```

```
import Title from '../components/ui/Title';
import Colors from '../constants/colors';
import Card from '../components/ui/Card';
import InstructionText from '../components/ui/InstructionText';

function StartGameScreen({ onPickNumber }) {
  const [enteredNumber, setEnteredNumber] = useState('');

  function numberInputHandler(enteredText) {
    setEnteredNumber(enteredText);
  }

  function resetInputHandler() {
    setEnteredNumber('');
  }

  function confirmInputHandler() {
    const chosenNumber = parseInt(enteredNumber);

    if (isNaN(chosenNumber) || chosenNumber <= 0 || chosenNumber > 99) {
      Alert.alert(
        'Invalid number!',
        'Number has to be a number between 1 and 99.',
        [{ text: 'Okay', style: 'destructive', onPress: resetInputHandler
}]
      );
      return;
    }

    onPickNumber(chosenNumber);
  }

  return (
    <View style={styles.rootContainer}>
      <Title>Guess My Number</Title>
      <Card>
        <InstructionText>
          Enter a Number
        </InstructionText>
        <TextInput
```

```
          style={styles.numberInput}
          maxLength={2}
          keyboardType="number-pad"
          autoCapitalize="none"
          autoCorrect={false}
          onChangeText={numberInputHandler}
          value={enteredNumber}
        />
        <View style={styles.buttonsContainer}>
          <View style={styles.buttonContainer}>
            <PrimaryButton
onPress={resetInputHandler}>Reset</PrimaryButton>
          </View>
          <View style={styles.buttonContainer}>
            <PrimaryButton
onPress={confirmInputHandler}>Confirm</PrimaryButton>
          </View>
        </View>
      </Card>
    </View>
  );
}

export default StartGameScreen;

const styles = StyleSheet.create({
  rootContainer: {
    flex: 1,
    marginTop: 100,
    alignItems: 'center',
  },
  numberInput: {
    height: 50,
    width: 50,
    fontSize: 32,
    borderBottomColor: Colors.accent500,
    borderBottomWidth: 2,
    color: Colors.accent500,
    marginVertical: 8,
    fontWeight: 'bold',
```

```
    textAlign: 'center',
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
});
```

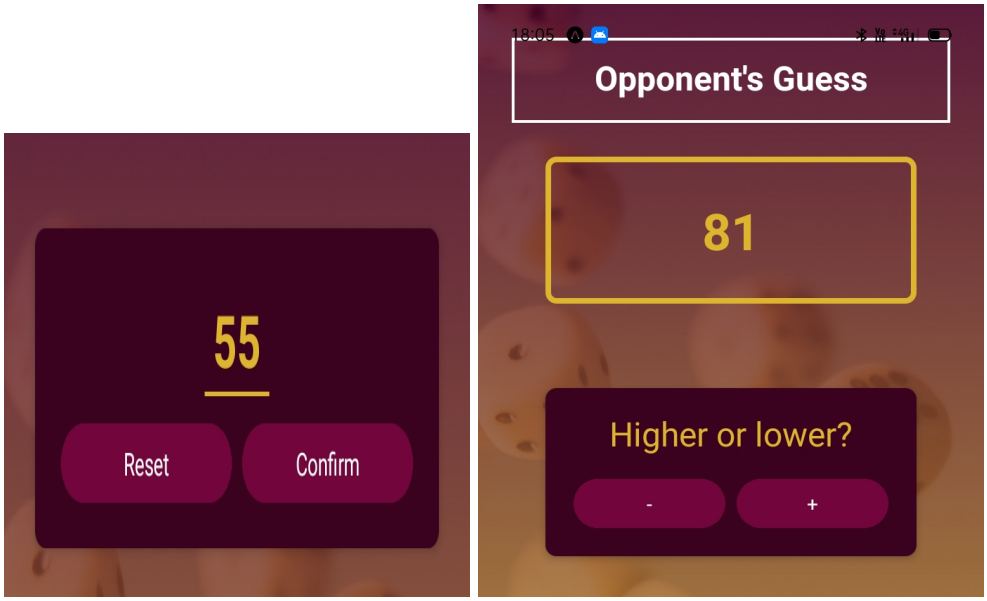
In game screen.js

```
import { useState, useEffect } from 'react';
import { View, StyleSheet, Alert } from 'react-native';

import NumberContainer from '../components/game/NumberContainer';
import Card from '../components/ui/Card';
import InstructionText from '../components/ui/InstructionText';
import PrimaryButton from '../components/ui/PrimaryButton';
import Title from '../components/ui/Title';

function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
  }
}

let minBoundary = 1;
let maxBoundary = 100;

function GameScreen({ userNumber, onGameOver }) {
  const initialGuess = generateRandomBetween(1, 100, userNumber);
  const [currentGuess, setCurrentGuess] = useState(initialGuess);

  useEffect(() => {
```

```
  if (currentGuess === userNumber) {
    onGameOver();
  }
}, [currentGuess, userNumber, onGameOver]);

function nextGuessHandler(direction) {
  // direction => 'lower', 'greater'
  if (
    (direction === 'lower' && currentGuess < userNumber) ||
    (direction === 'greater' && currentGuess > userNumber)
  ) {
    Alert.alert("Don't lie!", 'You know that this is wrong...', [
      { text: 'Sorry!', style: 'cancel' },
    ]);
    return;
  }

  if (direction === 'lower') {
    maxBoundary = currentGuess;
  } else {
    minBoundary = currentGuess + 1;
  }

  const newRndNumber = generateRandomBetween(
    minBoundary,
    maxBoundary,
    currentGuess
  );
  setCurrentGuess(newRndNumber);
}

return (
  <View style={styles.screen}>
    <Title>Opponent's Guess</Title>
    <NumberContainer>{currentGuess}</NumberContainer>
    <Card>
      <InstructionText style={styles.instructionText}>
        Higher or lower?
      </InstructionText>
      <View style={styles.buttonsContainer}>
```

```
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this, 'lower')}>
              -
            </PrimaryButton>
          </View>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this,
'greater')}>
              +
            </PrimaryButton>
          </View>
        </View>
      </Card>
      {/* <View>LOG ROUNDS</View> */}
    </View>
  );
}

export default GameScreen;

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24,
  },
  instructionText: {
    marginBottom: 12,
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
});
```

Output:



16.Working with icons(button icons)

In gamescreen.js

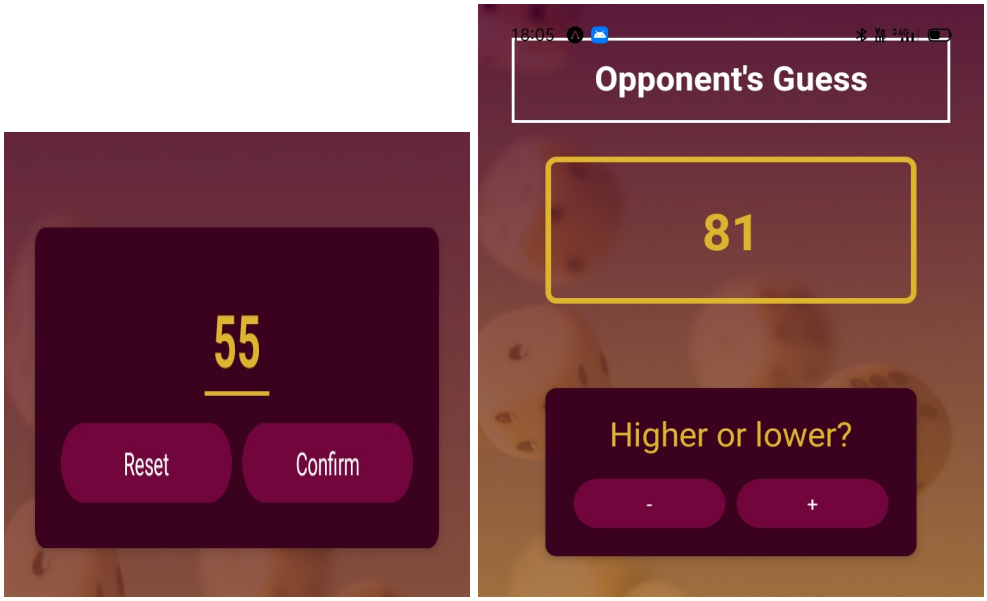
```
import { useState, useEffect } from 'react';
import { View, StyleSheet, Alert } from 'react-native';
import { Ionicons } from '@expo/vector-icons';

import NumberContainer from '../components/game/NumberContainer';
import Card from '../components/ui/Card';
import InstructionText from '../components/ui/InstructionText';
import PrimaryButton from '../components/ui/PrimaryButton';
import Title from '../components/ui/Title';

function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
  }
}
```

```
let minBoundary = 1;
let maxBoundary = 100;

function GameScreen({ userNumber, onGameOver }) {
  const initialGuess = generateRandomBetween(1, 100, userNumber);
  const [currentGuess, setCurrentGuess] = useState(initialGuess);

  useEffect(() => {
    if (currentGuess === userNumber) {
      onGameOver();
    }
  }, [currentGuess, userNumber, onGameOver]);

  function nextGuessHandler(direction) {
    // direction => 'lower', 'greater'
    if (
      (direction === 'lower' && currentGuess < userNumber) ||
      (direction === 'greater' && currentGuess > userNumber)
    ) {
      Alert.alert("Don't lie!", 'You know that this is wrong...', [
        { text: 'Sorry!', style: 'cancel' },
      ]);
      return;
    }

    if (direction === 'lower') {
      maxBoundary = currentGuess;
    } else {
      minBoundary = currentGuess + 1;
    }

    const newRndNumber = generateRandomBetween(
      minBoundary,
      maxBoundary,
      currentGuess
    );
    setCurrentGuess(newRndNumber);
  }
```

```
  return (
    <View style={styles.screen}>
      <Title>Opponent's Guess</Title>
      <NumberContainer>{currentGuess}</NumberContainer>
      <Card>
        <InstructionText style={styles.instructionText}>
          Higher or lower?
        </InstructionText>
        <View style={styles.buttonsContainer}>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this, 'lower')}>
              <Ionicons name="md-remove" size={24} color="white" />
            </PrimaryButton>
          </View>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this,
'greater')}>
              <Ionicons name="md-add" size={24} color="white" />
            </PrimaryButton>
          </View>
        </View>
      </Card>
      {/* <View>LOG ROUNDS</View> */}
    </View>
  );
}

export default GameScreen;

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24,
  },
  instructionText: {
    marginBottom: 12,
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
```

```
  buttonContainer: {
    flex: 1,
  },
});
```

Output:



17.adding-using-custom-font

1)Install the following packages.

expo install expo-font

expo install expo-app-loading

2)Create a folder named fonts where you store the fonts you want to use and download the fonts

3)In app.js

```
import { useState } from 'react';
import { StyleSheet, ImageBackground, SafeAreaView } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';
import { useFonts } from 'expo-font';
import AppLoading from 'expo-app-loading';
```

```
import StartGameScreen from './screens/StartGameScreen';
import GameScreen from './screens/GameScreen';
import GameOverScreen from './screens/GameOverScreen';
import Colors from './constants/colors';

export default function App() {
  const [userNumber, setUserNumber] = useState();
  const [gameIsOver, setGameIsOver] = useState(true);

  const [fontsLoaded] = useFonts({
    'open-sans': require('./assets/fonts/OpenSans-Regular.ttf'),
    'open-sans-bold': require('./assets/fonts/OpenSans-Bold.ttf'),
  });

  if (!fontsLoaded) {
    return <AppLoading />;
  }

  function pickedNumberHandler(pickedNumber) {
    setUserNumber(pickedNumber);
    setGameIsOver(false);
  }

  function gameOverHandler() {
    setGameIsOver(true);
  }

  let screen = <StartGameScreen onPickNumber={pickedNumberHandler} />;

  if (userNumber) {
    screen = (
      <GameScreen userNumber={userNumber} onGameOver={gameOverHandler} />
    );
  }

  if (gameIsOver && userNumber) {
    screen = <GameOverScreen />;
  }
```

```
  return (
    <LinearGradient
      colors={[Colors.primary700, Colors.accent500]}
      style={styles.rootScreen}
    >
      <ImageBackground
        source={require('./assets/images/background.png')}
        resizeMode="cover"
        style={styles.rootScreen}
        imageStyle={styles.backgroundImage}
      >
        <SafeAreaView style={styles.rootScreen}>{screen}</SafeAreaView>
      </ImageBackground>
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
  rootScreen: {
    flex: 1,
  },
  backgroundImage: {
    opacity: 0.15,
  },
});
```

In title.js

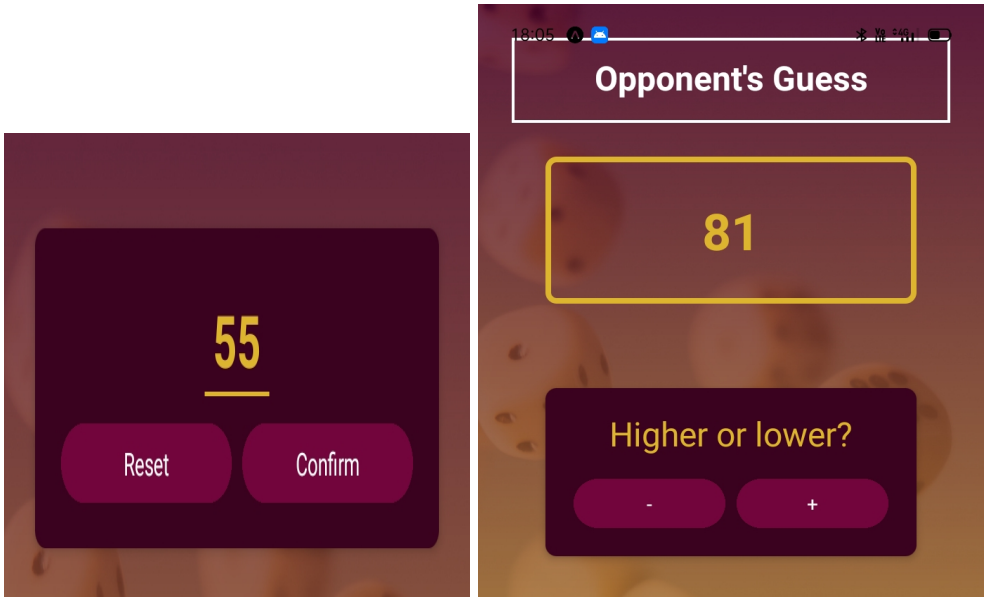
```
import { Text, StyleSheet } from 'react-native';

function Title({ children }) {
  return <Text style={styles.title}>{children}</Text>;
}

export default Title;

const styles = StyleSheet.create({
  title: {
    fontFamily: 'open-sans-bold',
    fontSize: 24,
```

```
    // fontWeight: 'bold',
    color: 'white',
    textAlign: 'center',
    borderWidth: 2,
    borderColor: 'white',
    padding: 12,
  },
});
```

In instruction.js

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

import Colors from '../../constants/colors';

function InstructionText({ children, style }) {
  return <Text style={[styles.instructionText, style]}>{children}</Text>;
}

export default InstructionText;

const styles = StyleSheet.create({
  instructionText: {
    fontFamily: 'open-sans',
    color: Colors.accent500,
    fontSize: 24,
  },
});
```

Output:



18.using styling-nested-text

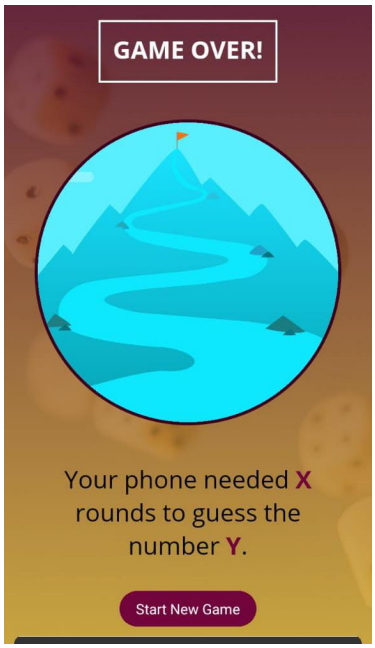
1)in images folder save the background image of gameoverscreen.js(success.png)

2)in gameoverscreen.js

```
import { View, Image, Text, StyleSheet } from 'react-native';
import Title from '../components/ui/Title';
import PrimaryButton from '../components/ui/PrimaryButton';
import Colors from '../constants/colors';

function GameOverScreen() {
  return (
    <View style={styles.rootContainer}>
      <Title>GAME OVER!</Title>
      <View style={styles.imageContainer}>
        <Image
          style={styles.image}
          source={require('../assets/images/success.png')}
        />
      </View>
      <Text style={styles.summaryText}>
        Your phone needed <Text style={styles.highlight}>X</Text> rounds
to
        guess the number <Text style={styles.highlight}>Y</Text>.
      </Text>
```

```
      <PrimaryButton>Start New Game</PrimaryButton>
    </View>
  );
}

export default GameOverScreen;

const styles = StyleSheet.create({
  rootContainer: {
    flex: 1,
    padding: 24,
    justifyContent: 'center',
    alignItems: 'center',
  },
  imageContainer: {
    width: 300,
    height: 300,
    borderRadius: 150,
    borderWidth: 3,
    borderColor: Colors.primary800,
    overflow: 'hidden',
    margin: 36,
  },
  image: {
    width: '100%',
    height: '100%',
  },
  summaryText: {
    fontFamily: 'open-sans',
    fontSize: 24,
    textAlign: 'center',
    marginBottom: 24
  },
  highlight: {
    fontFamily: 'open-sans-bold',
    color: Colors.primary500,
  },
});
```

Output:



19.adding-logic-to-restart-games

In gameoverscreen.js

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

import Title from '../components/ui/Title';
import PrimaryButton from '../components/ui/PrimaryButton';
import Colors from '../constants/colors';

function GameOverScreen({ roundsNumber, userNumber, onStartNewGame }) {
  return (
    <View style={styles.rootContainer}>
      <Title>GAME OVER!</Title>
      <View style={styles.imageContainer}>
        <Image
          style={styles.image}
          source={require('../assets/images/success.png')}
        />
      </View>
      <Text style={styles.summaryText}>
        Your phone needed <Text
style={styles.highlight}>{roundsNumber}</Text>{' '}
```

```
        rounds to guess the number{' '}
        <Text style={styles.highlight}>{userNumber}</Text>.
      </Text>
      <PrimaryButton onPress={onStartNewGame}>Start New
Game</PrimaryButton>
    </View>
  );
}

export default GameOverScreen;

const styles = StyleSheet.create({
  rootContainer: {
    flex: 1,
    padding: 24,
    justifyContent: 'center',
    alignItems: 'center',
  },
  imageContainer: {
    width: 300,
    height: 300,
    borderRadius: 150,
    borderWidth: 3,
    borderColor: Colors.primary800,
    overflow: 'hidden',
    margin: 36,
  },
  image: {
    width: '100%',
    height: '100%',
  },
  summaryText: {
    fontFamily: 'open-sans',
    fontSize: 24,
    textAlign: 'center',
    marginBottom: 24,
  },
  highlight: {
    fontFamily: 'open-sans-bold',
    color: Colors.primary500,
```

```
  },
});
```

In app.js

```
import { useState } from 'react';
import { StyleSheet, ImageBackground, SafeAreaView } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';
import { useFonts } from 'expo-font';
import AppLoading from 'expo-app-loading';

import StartGameScreen from './screens/StartGameScreen';
import GameScreen from './screens/GameScreen';
import GameOverScreen from './screens/GameOverScreen';
import Colors from './constants/colors';

export default function App() {
  const [userNumber, setUserNumber] = useState();
  const [gameIsOver, setGameIsOver] = useState(true);
  const [guessRounds, setGuessRounds] = useState(0);

  const [fontsLoaded] = useFonts({
    'open-sans': require('./assets/fonts/OpenSans-Regular.ttf'),
    'open-sans-bold': require('./assets/fonts/OpenSans-Bold.ttf'),
  });

  if (!fontsLoaded) {
    return <AppLoading />;
  }

  function pickedNumberHandler(pickedNumber) {
    setUserNumber(pickedNumber);
    setGameIsOver(false);
  }

  function gameOverHandler() {
    setGameIsOver(true);
  }

  function startNewGameHandler() {
```

```
    setUserNumber(null);
    setGuessRounds(0);
  }

  let screen = <StartGameScreen onPickNumber={pickedNumberHandler} />;

  if (userNumber) {
    screen = (
      <GameScreen userNumber={userNumber} onGameOver={gameOverHandler} />
    );
  }

  if (gameIsOver && userNumber) {
    screen = (
      <GameOverScreen
        userNumber={userNumber}
        roundsNumber={guessRounds}
        onStartNewGame={startNewGameHandler}
      />
    );
  }

  return (
    <LinearGradient
      colors={[Colors.primary700, Colors.accent500]}
      style={styles.rootScreen}
    >
      <ImageBackground
        source={require('./assets/images/background.png')}
        resizeMode="cover"
        style={styles.rootScreen}
        imageStyle={styles.backgroundImage}
      >
        <SafeAreaView style={styles.rootScreen}>{screen}</SafeAreaView>
      </ImageBackground>
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
```

```
  rootScreen: {
    flex: 1,
  },
  backgroundImage: {
    opacity: 0.15,
  },
});
```

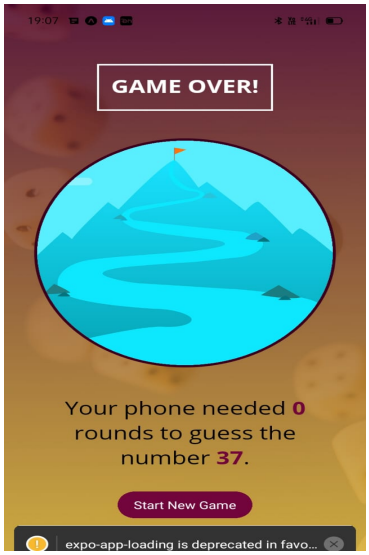
In gamescreen.js

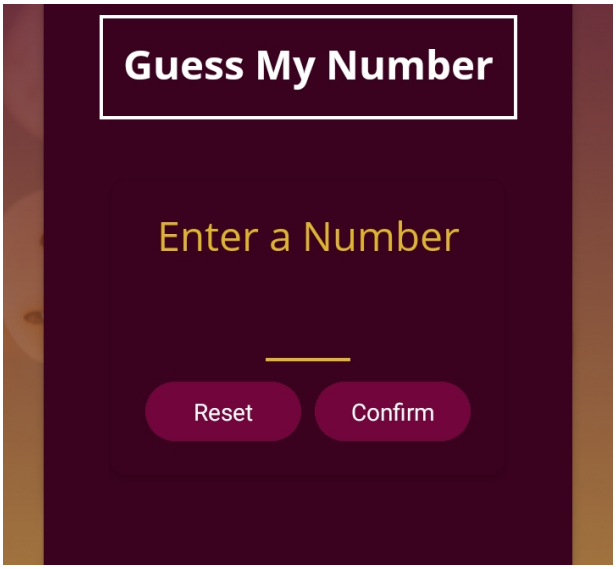
```
import { useState, useEffect } from 'react';
import { View, StyleSheet, Alert } from 'react-native';
import { Ionicons } from '@expo/vector-icons';

import NumberContainer from '../components/game/NumberContainer';
import Card from '../components/ui/Card';
import InstructionText from '../components/ui/InstructionText';
import PrimaryButton from '../components/ui/PrimaryButton';
import Title from '../components/ui/Title';

function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
  }
}

let minBoundary = 1;
let maxBoundary = 100;

function GameScreen({ userNumber, onGameOver }) {
  const initialGuess = generateRandomBetween(1, 100, userNumber);
  const [currentGuess, setCurrentGuess] = useState(initialGuess);

  useEffect(() => {
    if (currentGuess === userNumber) {
      onGameOver();
```

```
  }
}, [currentGuess, userNumber, onGameOver]);

useEffect(() => {
  minBoundary = 1;
  maxBoundary = 100;
}, []);

function nextGuessHandler(direction) {
  // direction => 'lower', 'greater'
  if (
    (direction === 'lower' && currentGuess < userNumber) ||
    (direction === 'greater' && currentGuess > userNumber)
  ) {
    Alert.alert("Don't lie!", 'You know that this is wrong...', [
      { text: 'Sorry!', style: 'cancel' },
    ]);
    return;
  }

  if (direction === 'lower') {
    maxBoundary = currentGuess;
  } else {
    minBoundary = currentGuess + 1;
  }

  const newRndNumber = generateRandomBetween(
    minBoundary,
    maxBoundary,
    currentGuess
  );
  setCurrentGuess(newRndNumber);
}

return (
  <View style={styles.screen}>
    <Title>Opponent's Guess</Title>
    <NumberContainer>{currentGuess}</NumberContainer>
    <Card>
      <InstructionText style={styles.instructionText}>
```

```
          Higher or lower?
        </InstructionText>
        <View style={styles.buttonsContainer}>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this, 'lower')}>
              <Ionicons name="md-remove" size={24} color="white" />
            </PrimaryButton>
          </View>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this,
'greater')}>
              <Ionicons name="md-add" size={24} color="white" />
              </PrimaryButton>
          </View>
        </View>
      </Card>
      {/* <View>LOG ROUNDS</View> */}
    </View>
  );
}

export default GameScreen;

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24,
  },
  instructionText: {
    marginBottom: 12,
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
});
```

Output:





20.styling-game-round-logs

1.create a file inside of

components/game…(components/game/guestlogitem.js)

2.In GuessLogitem.js

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

import Colors from '../../constants/colors';

function GuessLogItem({ roundNumber, guess }) {
  return (
    <View style={styles.listItem}>
      <Text style={styles.itemText}>#{roundNumber}</Text>
      <Text style={styles.itemText}>Opponent's Guess: {guess}</Text>
    </View>
  );
}

export default GuessLogItem;

const styles = StyleSheet.create({
  listItem: {
```

```
    borderColor: Colors.primary800,
    borderWidth: 1,
    borderRadius: 40,
    padding: 12,
    marginVertical: 8,
    backgroundColor: Colors.accent500,
    flexDirection: 'row',
    justifyContent: 'space-between',
    width: '100%',
    elevation: 4,
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 0 },
    shadowOpacity: 0.25,
    shadowRadius: 3,
  },
  itemText: {
    fontFamily: 'open-sans'
  }
});
```

GameScreen.js

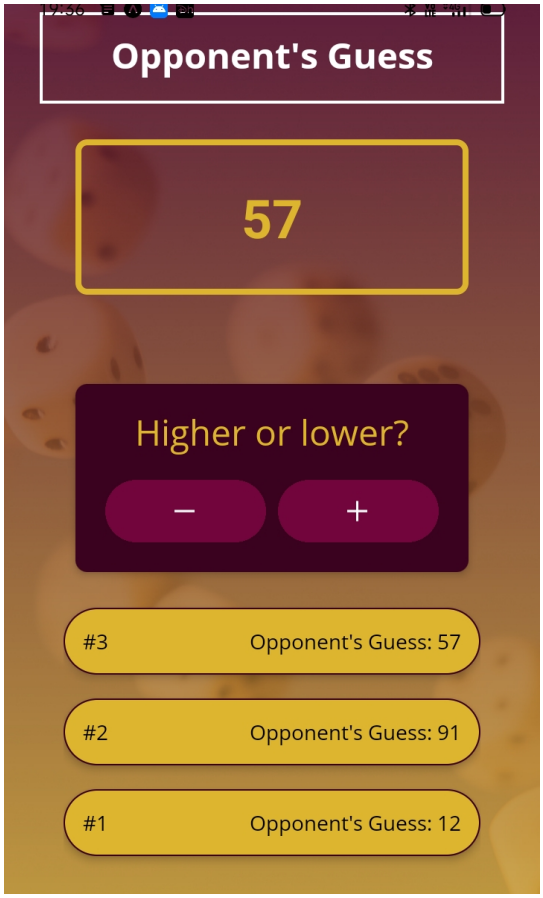
```
import { useState, useEffect } from 'react';
import { View, StyleSheet, Alert, Text, FlatList } from 'react-native';
import { Ionicons } from '@expo/vector-icons';

import NumberContainer from '../components/game/NumberContainer';
import Card from '../components/ui/Card';
import InstructionText from '../components/ui/InstructionText';
import PrimaryButton from '../components/ui/PrimaryButton';
import Title from '../components/ui/Title';
import GuessLogItem from '../components/game/GuessLogItem';

function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
```

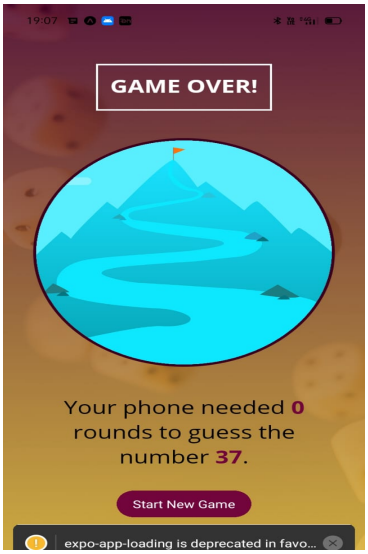
```
  }
}

let minBoundary = 1;
let maxBoundary = 100;

function GameScreen({ userNumber, onGameOver }) {
  const initialGuess = generateRandomBetween(1, 100, userNumber);
  const [currentGuess, setCurrentGuess] = useState(initialGuess);
  const [guessRounds, setGuessRounds] = useState([initialGuess]);

  useEffect(() => {
    if (currentGuess === userNumber) {
      onGameOver();
    }
  }, [currentGuess, userNumber, onGameOver]);

  useEffect(() => {
    minBoundary = 1;
    maxBoundary = 100;
  }, []);

  function nextGuessHandler(direction) {
    // direction => 'lower', 'greater'
    if (
      (direction === 'lower' && currentGuess < userNumber) ||
      (direction === 'greater' && currentGuess > userNumber)
    ) {
      Alert.alert("Don't lie!", 'You know that this is wrong...', [
        { text: 'Sorry!', style: 'cancel' },
      ]);
      return;
    }

    if (direction === 'lower') {
      maxBoundary = currentGuess;
    } else {
      minBoundary = currentGuess + 1;
    }
```

```
    const newRndNumber = generateRandomBetween(
      minBoundary,
      maxBoundary,
      currentGuess
    );
    setCurrentGuess(newRndNumber);
    setGuessRounds((prevGuessRounds) => [newRndNumber,
...prevGuessRounds]);
  }

  const guessRoundsListLength = guessRounds.length;

  return (
    <View style={styles.screen}>
      <Title>Opponent's Guess</Title>
      <NumberContainer>{currentGuess}</NumberContainer>
      <Card>
        <InstructionText style={styles.instructionText}>
          Higher or lower?
        </InstructionText>
        <View style={styles.buttonsContainer}>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this, 'lower')}>
              <Ionicons name="md-remove" size={24} color="white" />
            </PrimaryButton>
          </View>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this,
'greater')}>
              <Ionicons name="md-add" size={24} color="white" />
            </PrimaryButton>
          </View>
        </View>
      </Card>
      <View style={styles.listContainer}>
        {/* {guessRounds.map(guessRound => <Text
key={guessRound}>{guessRound}</Text>)} */}
        <FlatList
          data={guessRounds}
          renderItem={(itemData) => (
```

```
            <GuessLogItem
              roundNumber={guessRoundsListLength - itemData.index}
              guess={itemData.item}
            />
          )}
          keyExtractor={(item) => item}
        />
      </View>
    </View>
  );
}

export default GameScreen;

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24,
  },
  instructionText: {
    marginBottom: 12,
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
  listContainer: {
    flex: 1,
    padding: 16,
  },
});
```

Output:





20.showing the game rounds

In app.js

```
import { useState } from 'react';
import { StyleSheet, ImageBackground, SafeAreaView } from 'react-native';
import { LinearGradient } from 'expo-linear-gradient';
import { useFonts } from 'expo-font';
import AppLoading from 'expo-app-loading';

import StartGameScreen from './screens/StartGameScreen';
import GameScreen from './screens/GameScreen';
import GameOverScreen from './screens/GameOverScreen';
import Colors from './constants/colors';

export default function App() {
  const [userNumber, setUserNumber] = useState();
  const [gameIsOver, setGameIsOver] = useState(true);
  const [guessRounds, setGuessRounds] = useState(0);
```

```
const [fontsLoaded] = useFonts({
  'open-sans': require('./assets/fonts/OpenSans-Regular.ttf'),
  'open-sans-bold': require('./assets/fonts/OpenSans-Bold.ttf'),
});

if (!fontsLoaded) {
  return <AppLoading />;
}

function pickedNumberHandler(pickedNumber) {
  setUserNumber(pickedNumber);
  setGameIsOver(false);
}

function gameOverHandler(numberOfRounds) {
  setGameIsOver(true);
  setGuessRounds(numberOfRounds);
}

function startNewGameHandler() {
  setUserNumber(null);
  setGuessRounds(0);
}

let screen = <StartGameScreen onPickNumber={pickedNumberHandler} />;

if (userNumber) {
  screen = (
    <GameScreen userNumber={userNumber} onGameOver={gameOverHandler} />
  );
}

if (gameIsOver && userNumber) {
  screen = (
    <GameOverScreen
      userNumber={userNumber}
      roundsNumber={guessRounds}
      onStartNewGame={startNewGameHandler}
    />
```

```
    );
  }

  return (
    <LinearGradient
      colors={[Colors.primary700, Colors.accent500]}
      style={styles.rootScreen}
    >
      <ImageBackground
        source={require('./assets/images/background.png')}
        resizeMode="cover"
        style={styles.rootScreen}
        imageStyle={styles.backgroundImage}
      >
        <SafeAreaView style={styles.rootScreen}>{screen}</SafeAreaView>
      </ImageBackground>
    </LinearGradient>
  );
}

const styles = StyleSheet.create({
  rootScreen: {
    flex: 1,
  },
  backgroundImage: {
    opacity: 0.15,
  },
});
```

In gamescreen.js

```
import { useState, useEffect } from 'react';
import { View, StyleSheet, Alert, Text, FlatList } from 'react-native';
import { Ionicons } from '@expo/vector-icons';

import NumberContainer from '../components/game/NumberContainer';
import Card from '../components/ui/Card';
import InstructionText from '../components/ui/InstructionText';
import PrimaryButton from '../components/ui/PrimaryButton';
import Title from '../components/ui/Title';
```

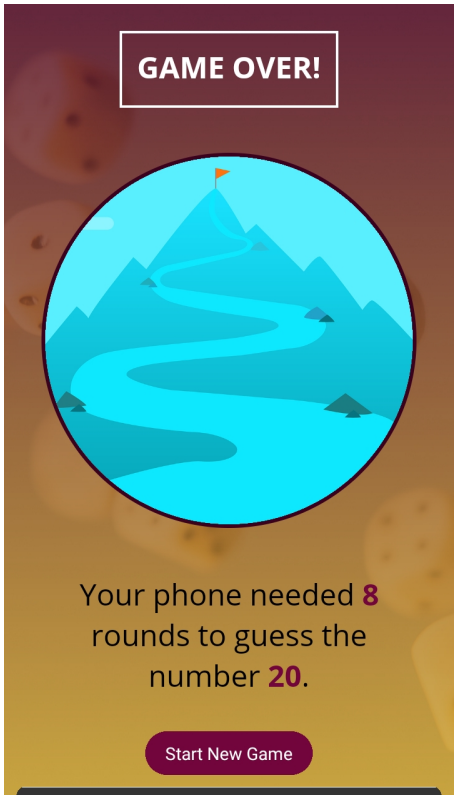
```
import GuessLogItem from '../components/game/GuessLogItem';

function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
  }
}

let minBoundary = 1;
let maxBoundary = 100;

function GameScreen({ userNumber, onGameOver }) {
  const initialGuess = generateRandomBetween(1, 100, userNumber);
  const [currentGuess, setCurrentGuess] = useState(initialGuess);
  const [guessRounds, setGuessRounds] = useState([initialGuess]);

  useEffect(() => {
    if (currentGuess === userNumber) {
      onGameOver(guessRounds.length);
    }
  }, [currentGuess, userNumber, onGameOver]);

  useEffect(() => {
    minBoundary = 1;
    maxBoundary = 100;
  }, []);

  function nextGuessHandler(direction) {
    // direction => 'lower', 'greater'
    if (
      (direction === 'lower' && currentGuess < userNumber) ||
      (direction === 'greater' && currentGuess > userNumber)
    ) {
      Alert.alert("Don't lie!", 'You know that this is wrong...', [
        { text: 'Sorry!', style: 'cancel' },
      ]);
```

```
      return;
    }

    if (direction === 'lower') {
      maxBoundary = currentGuess;
    } else {
      minBoundary = currentGuess + 1;
    }

    const newRndNumber = generateRandomBetween(
      minBoundary,
      maxBoundary,
      currentGuess
    );
    setCurrentGuess(newRndNumber);
    setGuessRounds((prevGuessRounds) => [newRndNumber,
...prevGuessRounds]);
  }

  const guessRoundsListLength = guessRounds.length;

  return (
    <View style={styles.screen}>
      <Title>Opponent's Guess</Title>
      <NumberContainer>{currentGuess}</NumberContainer>
      <Card>
        <InstructionText style={styles.instructionText}>
          Higher or lower?
        </InstructionText>
        <View style={styles.buttonsContainer}>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this, 'lower')}>
              <Ionicons name="md-remove" size={24} color="white" />
            </PrimaryButton>
          </View>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this,
'greater')}>
              <Ionicons name="md-add" size={24} color="white" />
            </PrimaryButton>
```

```
          </View>
        </View>
      </Card>
      <View style={styles.listContainer}>
        {/* {guessRounds.map(guessRound => <Text
key={guessRound}>{guessRound}</Text>)} */}
        <FlatList
          data={guessRounds}
          renderItem={(itemData) => (
            <GuessLogItem
              roundNumber={guessRoundsListLength - itemData.index}
              guess={itemData.item}
            />
          )}
          keyExtractor={(item) => item}
        />
      </View>
    </View>
  );
}

export default GameScreen;

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 24,
  },
  instructionText: {
    marginBottom: 12,
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
  listContainer: {
    flex: 1,
    padding: 16,
```

```
  },
});
```

Output:


GAME OVER!
Your phone needed 8 rounds to guess the number 20.
Start New Game