1.setting dynamic widths

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,
    maxWidth: '80%',
    width: 300
  },
});
```

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';
```

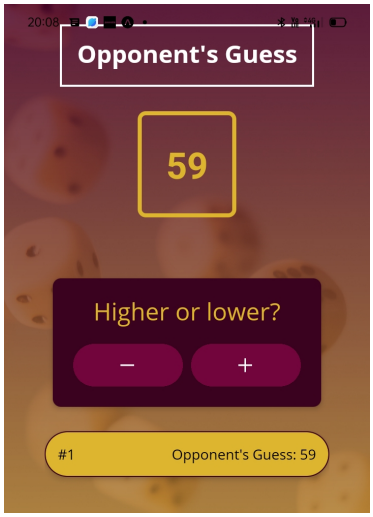
```
function generateRandomBetween(min, max, exclude) {
  const rndNum = Math.floor(Math.random() * (max - min)) + min;

  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,
    alignItems: 'center'
  },
  instructionText: {
    marginBottom: 12,
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
  listContainer: {
    flex: 1,
    padding: 16,
  },
```

```
});
```

Output:



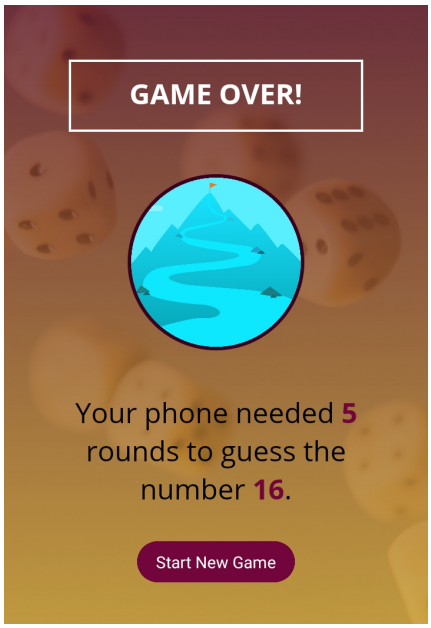
2.adjusting images sizes with dimensions api

```
import { View, Image, Text, StyleSheet, Dimensions } 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 deviceWidth = Dimensions.get('window').width;

const styles = StyleSheet.create({
  rootContainer: {
    flex: 1,
    padding: 24,
    justifyContent: 'center',
    alignItems: 'center',
  },
  imageContainer: {
    width: deviceWidth < 380 ? 150 : 300,
    height: deviceWidth < 380 ? 150 : 300,
    borderRadius: deviceWidth < 380 ? 75 : 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:



3.setting sizes dynamically(for rotations,)

1)for turning on auto rotations

Go to app.json

And set `"orientation": "default",`

2)setting sizes dynamically

In startgamescreen.js

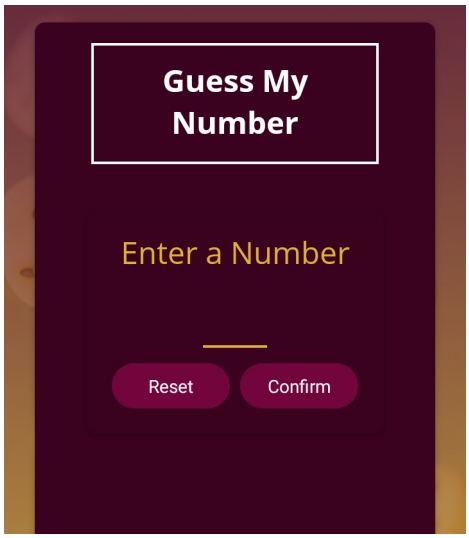
```
import { useState } from 'react';
import {
  TextInput,
  View,
  StyleSheet,
  Alert,
```

```
  useWindowDimensions,
} 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('');

  const { width, height } = useWindowDimensions();

  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);
  }

  const marginTopDistance = height < 380 ? 30 : 100;
```

```
  return (
    <View style={[styles.rootContainer, { marginTop: marginTopDistance
}]}>
      <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 deviceHeight = Dimensions.get('window').height;

const styles = StyleSheet.create({
  rootContainer: {
    flex: 1,
    // marginTop: deviceHeight < 380 ? 30 : 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,
  },
});
```

Output:



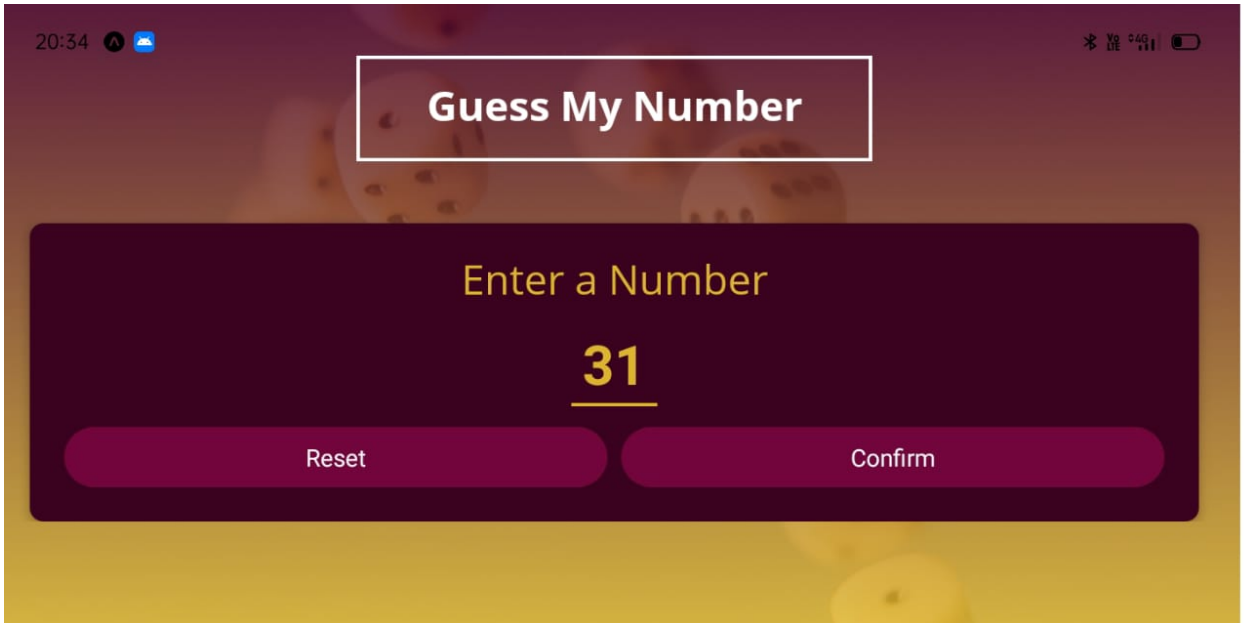
4.keyboardavoiding view problem solving
In startgamescreen.js

```
import { useState } from 'react';
import {
  TextInput,
  View,
  StyleSheet,
  Alert,
  useWindowDimensions,
  KeyboardAvoidingView,
  ScrollView,
} 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('');

  const { width, height } = useWindowDimensions();

  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);
  }

  const marginTopDistance = height < 380 ? 30 : 100;

  return (
    <ScrollView style={styles.screen}>
      <KeyboardAvoidingView style={styles.screen} behavior="position">
        <View style={[styles.rootContainer, { marginTop: marginTopDistance
}]}>
          <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>
      </KeyboardAvoidingView>
    </ScrollView>
  );
}

export default StartGameScreen;

// const deviceHeight = Dimensions.get('window').height;

const styles = StyleSheet.create({
  screen: {
    flex: 1,
  },
  rootContainer: {
    flex: 1,
    // marginTop: deviceHeight < 380 ? 30 : 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,
  },
});
```

Output:



5.improvement in landscapemode ui

```
import { useState, useEffect } from 'react';
import {
  View,
  StyleSheet,
  Alert,
  FlatList,
  useWindowDimensions,
} 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]);
  const { width, height } = useWindowDimensions();

  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;

  let content = (
    <>
      <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>
    </>
  );

  if (width > 500) {
    content = (
      <>
        <View style={styles.buttonsContainerWide}>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this, 'lower')}>
              <Ionicons name="md-remove" size={24} color="white" />
            </PrimaryButton>
          </View>
          <NumberContainer>{currentGuess}</NumberContainer>
          <View style={styles.buttonContainer}>
            <PrimaryButton onPress={nextGuessHandler.bind(this,
'greater')}>
              <Ionicons name="md-add" size={24} color="white" />
            </PrimaryButton>
          </View>
        </View>
      </>
    );
  }

  return (
    <View style={styles.screen}>
      <Title>Opponent's Guess</Title>
      {content}
      <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,
    alignItems: 'center',
  },
  instructionText: {
    marginBottom: 12,
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
  buttonContainer: {
    flex: 1,
  },
  buttonsContainerWide: {
    flexDirection: 'row',
    alignItems: 'center',
  },
  listContainer: {
    flex: 1,
    padding: 16,
  },
});
```

6.styling the status bar

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 { StatusBar } from 'expo-status-bar';

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 (
  <>
    <StatusBar style="light" />
    <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,
  },
});
```

Output: