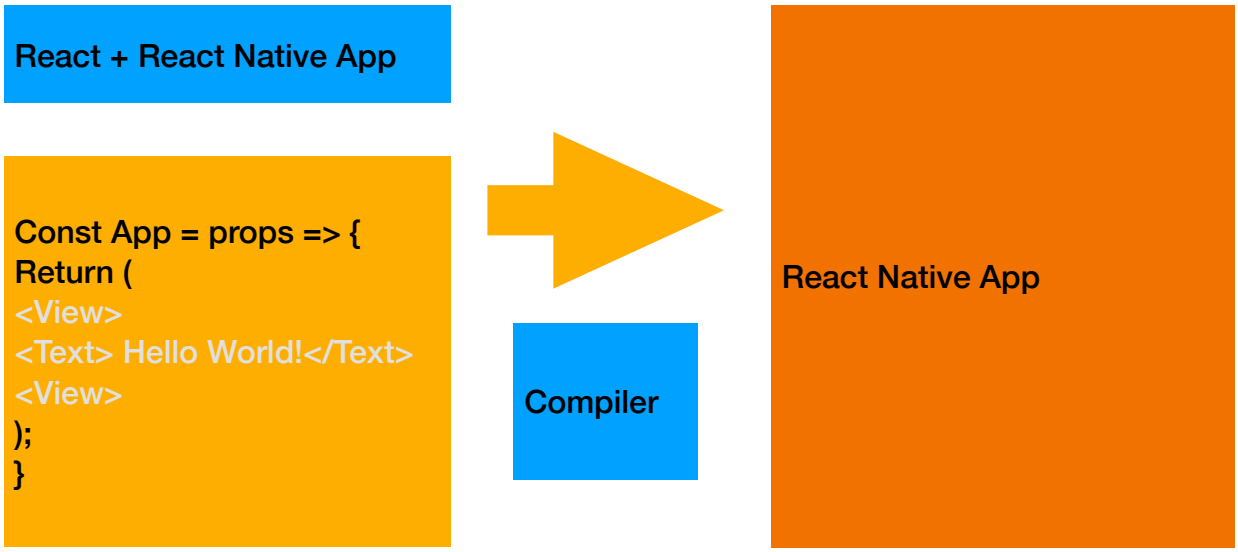
3 January 2022 Monday

React Native

React Native Nedir?

- Facebook tarafından oluşturulan IOS ve Android mobil uygulama geliştirmek için platformdur.
- React üzerinde kullanılan özelleştirilmiş compponentları kullanmaktadır.
- Geliştirilen componentları native widgetlara derlemektedir.
- Native platformda bulunan abilerin JavaScript içerisinde geliştirilebilmektedir.
- Javascript aracılığıyla native platforma erişebilmektedir.
- Temel olarak javascript ile native arasında bir bridge kurularak native code geliştirilebilmektedir.

Nasıl Çalışır ?



Web	Android	IOS	React Native
	android.view	UIView	<View>
<input>	EditText	UITextField	<TextInput>
...	...	...	...

Kurulumlar ?

https://reactnative.dev

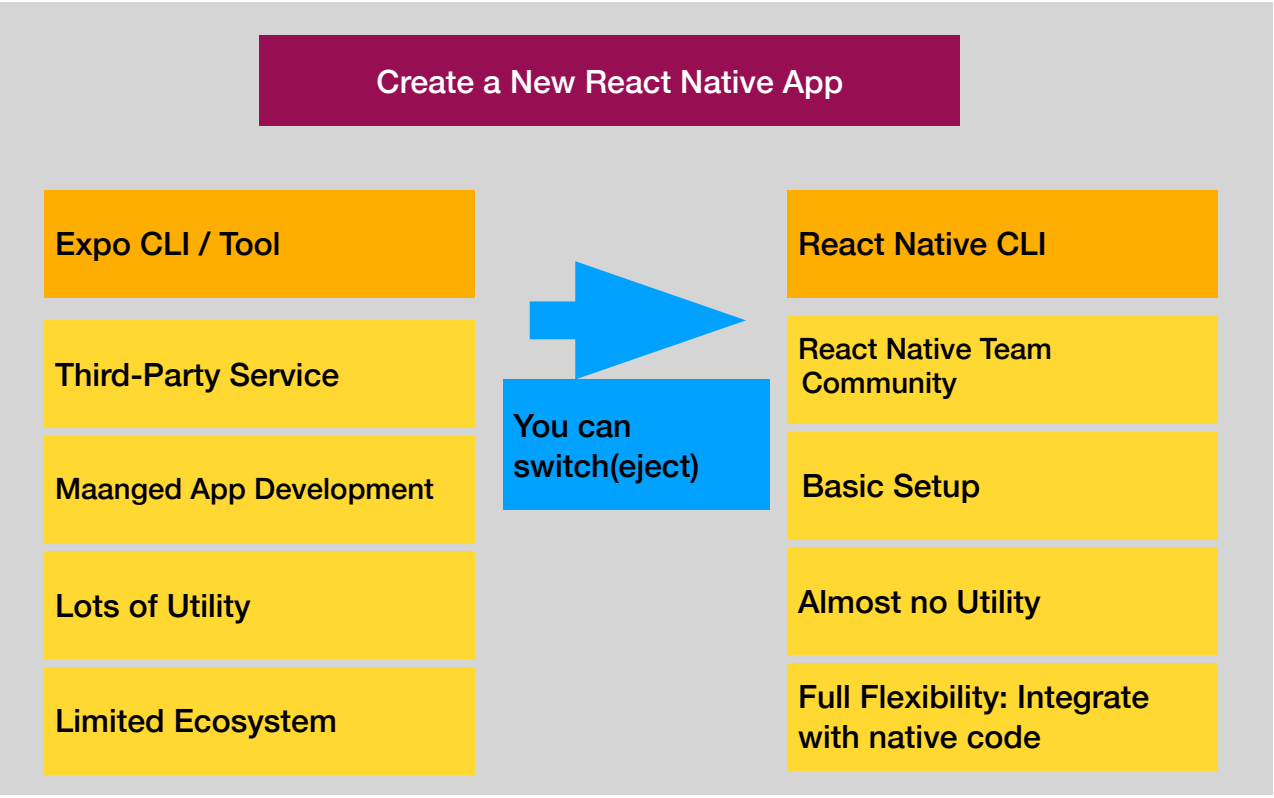
https://expo.dev

https://nodejs.org/en/

https://code.visualstudio.com

XCODE (Mac)

Android Studio



İlk Proje

İlk proje oluşturmak için ilk önce node js üzerinde Expo entegrasyonu yapılmalıdır. Bunun için konsolunuzdan aşağıdaki komutu çalıştırınız.

npm install --global expo-cli

Daha sonra bir klasör üzerinden proje oluşturmak için terminalimizden o klasör üzerinden aşağıdaki Expo scriptini çalıştırıyoruz.

expo init my-project

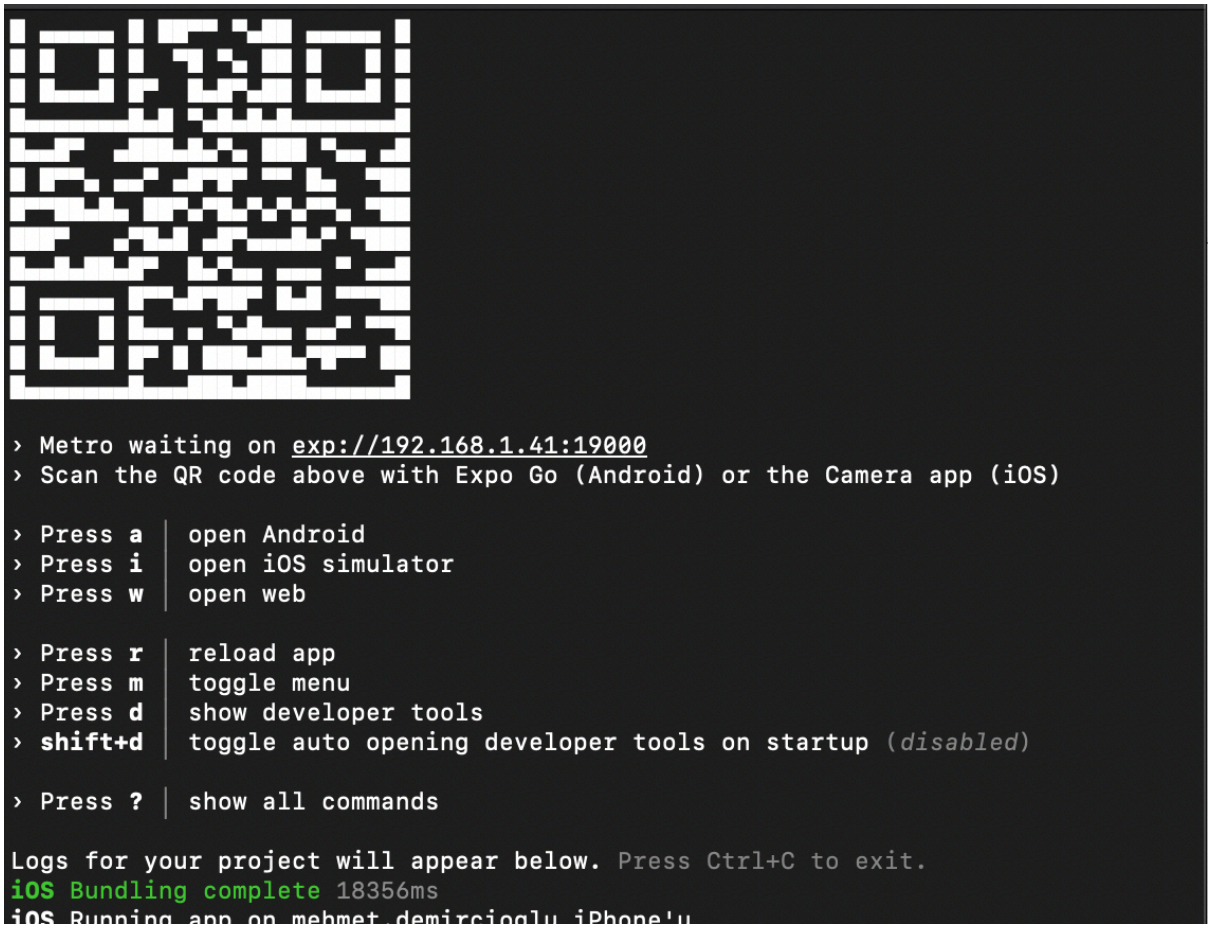
```
[mehmetdemircioglu@Mehmets-MacBook-Pro-2 ReactNativeCode % expo init my-project  ]
? Choose a template: › - Use arrow-keys. Return to submit.
    ----- Managed workflow -----
❯   blank                a minimal app as clean as an empty canvas
    blank (TypeScript)   same as blank but with TypeScript configuration
    tabs (TypeScript)    several example screens and tabs using react-navigation
and TypeScript
    ----- Bare workflow -----
    minimal              bare and minimal, just the essentials to get you started
```

Konsoldan ise blank olanı seçiyoruz ve projenin oluşturulmasını bekliyoruz.

```
[mehmetdemircioglu@Mehmets-MacBook-Pro-2 ReactNativeCode % expo init my-project  ]
✔ Choose a template: › blank                a minimal app as clean as an empty ca
nvas
✔ Downloaded and extracted project files.
📦 Using npm to install packages.
✔ Installed JavaScript dependencies.

✅ Your project is ready!

To run your project, navigate to the directory and run one of the following npm
commands.

- cd my-project
- npm start # you can open iOS, Android, or web from here, or run them directly
with the commands below.
- npm run android
- npm run ios
- npm run web
mehmetdemircioglu@Mehmets-MacBook-Pro-2 ReactNativeCode %
```

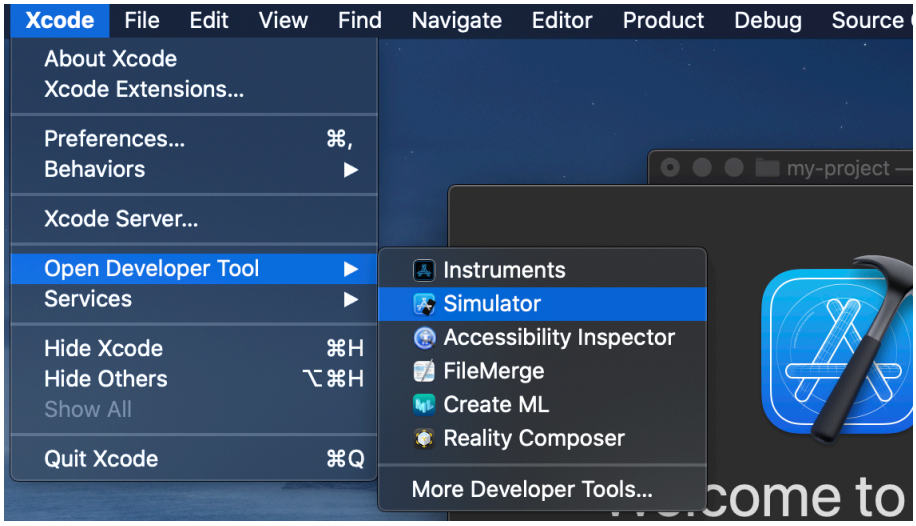
Daha sonra proje oluşturulduktan sonra o klasör içerisine konsol içerisinden komutla gidiyoruz.

Proje içerisine gittikten sonra ise artık projemizi çalıştırabiliriz. Çalıştırmak için birden fazla seçenek sunulmaktadır. Biz npm start diyerek projeyi derliyor ve Expo üzerinden(lokalinizde) yayınlamış oluyoruz.



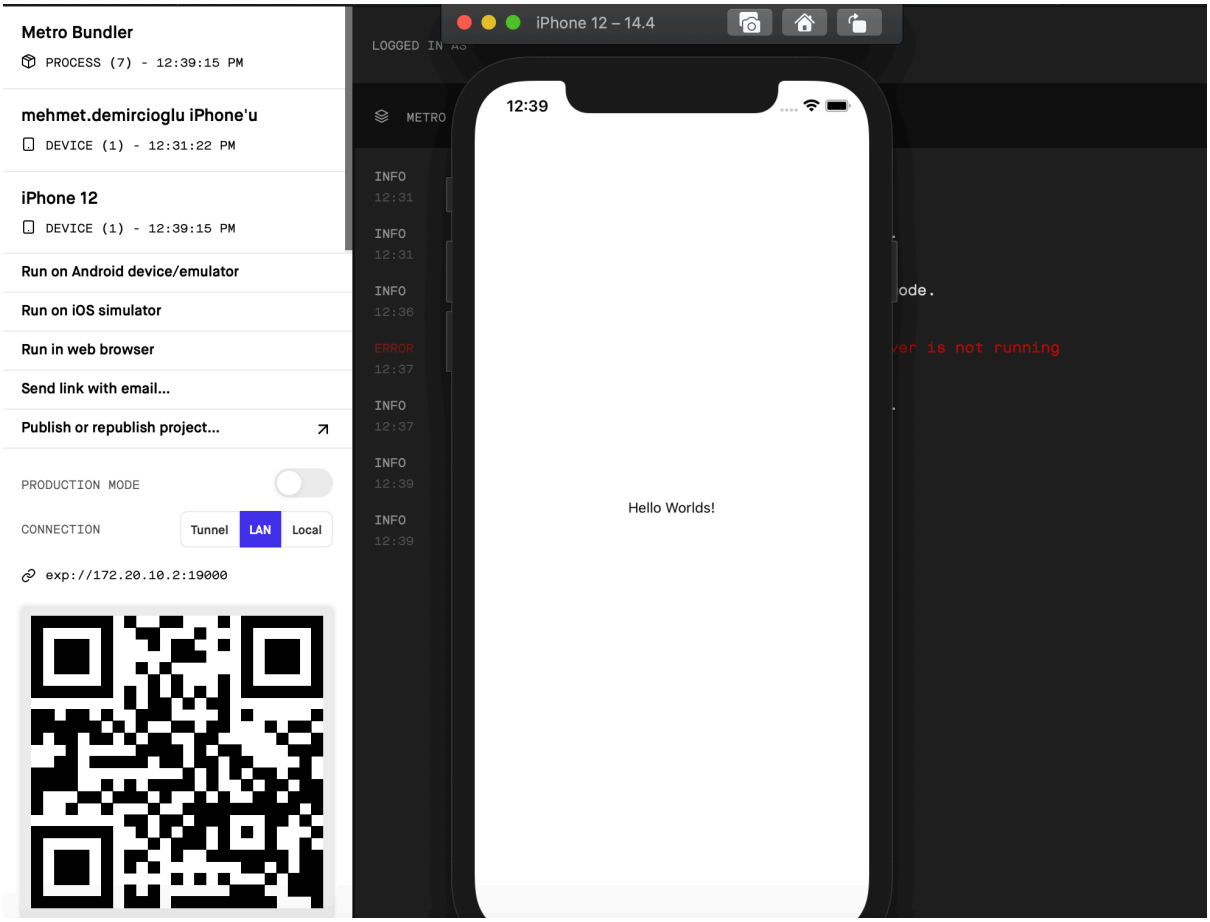
Telefonuzun kamerası ile qr kodu okutup telefonunuzda çalıştırınız.
Dikkat ! Aynı ağ üzerinde olmanız gerekmektedir.
Dikkat ! Telefonunuza Expo Go uygulamasını indirmeniz gerekmektedir.

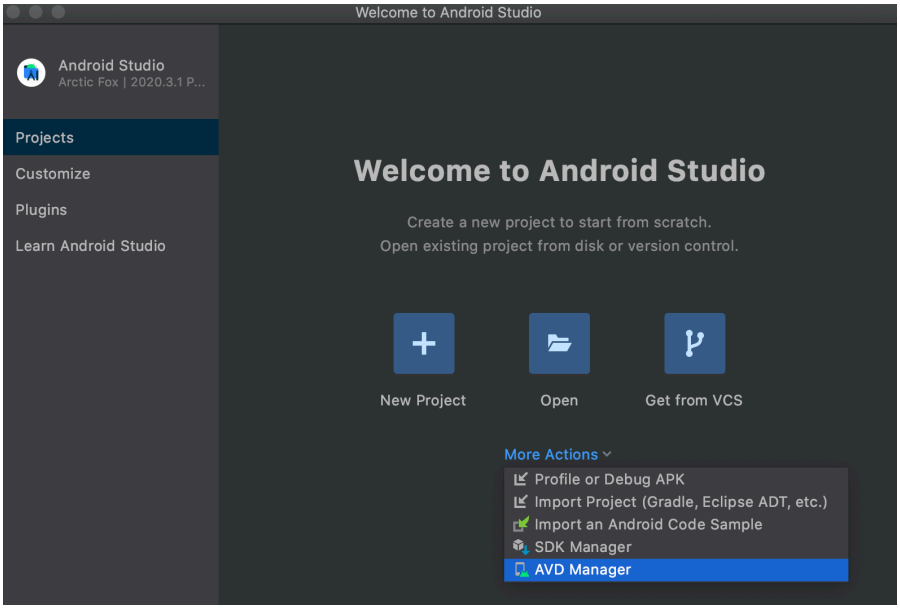
Kodun düzenlemesi için VSCODE kullanıyoruz. Oluşturduğumuz dosyayı şimdi vscode ile acınız ve düzenleme yapınız.



Uygulamanın simülatörler üzerinde çalıştırılması için;
IOS : MacOs üzerine XCODE kurularak bir tane simulator oluşturulur.

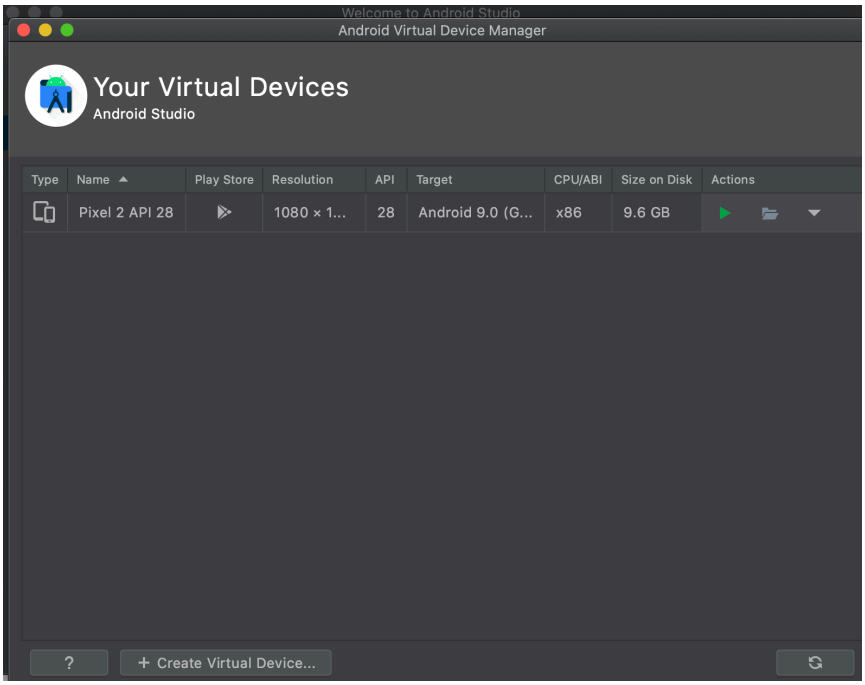
Konsoldan calıstırdıgımız projeye d tusuna basılarak developer tools sayfası acılır, ve Run on IOS simulator tıklanılarak simülatör üzerinde çalıştırılması sağlanır.

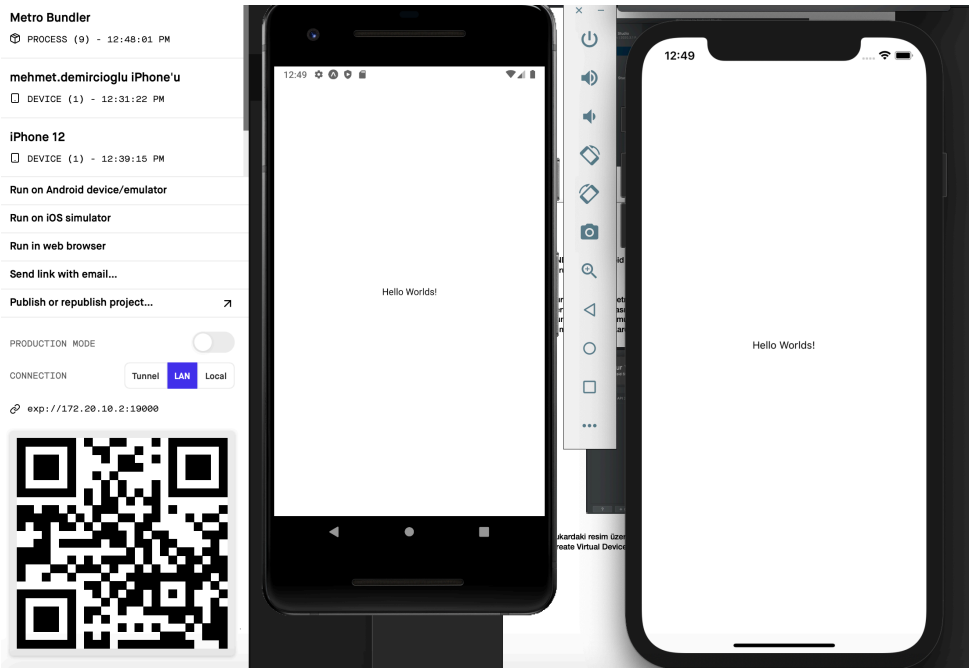




ANDROID : Android için ilk olarak Android Studio official sitesinde indirilerek kurulur.

Kurulurken dikkat etmeniz gereken nokta simülatörler üzerinde Google Play Servicelerinin olması gerekmektedir. Daha sonra IOS da yapmış oldugumuz gibi Run on Android Simulator tıklayarak calıstırıyrouz. Android üzerinde simulartörü acmak için ise yukardaki resim üzerinde gösterilemiştir.





Yukardaki resim üzerinde ise kurulu olan bir tane simülatör başlatılır eğer yok ise Create Virtual Device tıklayarak yeni bir Android Simülatör oluşturulabilir.

Component, FlexBox,Layouts,State,Events

1- İlk olarak Expo init rn-chapter-1 komutunu kullanılarak yeni bir proje oluşturulur. VSCODE üzerinden proje dosyası acılır. Terminal üzerinden npm start komutu verilerek proje çalıştırılır.

2. Proje içerisinde bulunan App.js dosyası içerisine aşağıdaki kod parçacığı ile yer değiştirilir. Branch : step1-components

```
<View style={{ padding: 40 }}>
      <View>
        <TextInput
          placeholder='Enter Word'
          style={{ borderColor: 'black', borderWidth: 1,
padding: 10 }} />
        <Button title="ADD" />
      </View>
      <View>
      </View>
    </View>
```

3. Vİew üzerinde flexDirection komutunun incelemesi yapılır. Ayrıca justifyContent tag kullanımı aşağıdaki kod parcası üzerinde oluşturulmuştur. Branch: step2

```
<View
style={{flexDirection:’row',
justifyContent:’space-between',
alignItems:'center'}}>
        <TextInput
          placeholder='Enter Word'
          style={{ width:'80%', borderColor: 'black',
borderWidth: 1, padding: 10 }} />
        <Button title="ADD" />
      </View>
```

4. App.js dosyası üzerine style eklemek için StyleSheet fonksiyonun Create methodu kullanılır. Aşağıdaki kod parcasını app.js ekleyiniz ve View üzerindeki styleları değiştiriniz. Branch: step2

```
const styles = StyleSheet.create({
  screen: {
    padding: 40
  },
  container: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center'
  },
  inputStyle: {
    width: '80%',
    borderColor: 'black',
    borderWidth: 1,
    padding: 10
  }
})
```

4. App.js dosyası üzerine textınput ve button için eventları ekleyerek konsola input içerisine girilen değerleri yazdırmak için aşağıdaki kod parçacığı seklinde revize ediniz. Branch: step3

```
export default function App() {

  const [word, setWord] = useState('');

  const wordInputHandler = (text) => {
    setWord(text);
  }

  const addWordHandler = () => {
    console.log(word);
  }

  return (
    <View style={styles.screen}>
      <View style={styles.container}>
        <TextInput
          placeholder='Enter Word'
          onChangeText={wordInputHandler}
```

```
          style={styles.inputStyle} />
        <Button title="ADD" onPress={addWordHandler} />
      </View>
      <View style={styles.container}>

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

5. App.js dosyası üzerine input içerisinde girilen değerleri bir liste içerisine almak için const liste tanımı yapılarak addInputHandler fonksiyonu içerisinde input değerlerini liste içerisine atınız. Daha sonra item style tanımlamak için styles tanımı içerisinde item adında bir tanım yapınız. Branch step3

```
  const [wordList, setWordList] = useState([]);
```

```
 const addWordHandler = () => {
    console.log(word);
    setWordList(currentList => [...currentList, word]);
  }
```

```
 {wordList.map((word) => (
          <View style={styles.item}>
            <Text key={word}>{word}</Text>
          </View>
        ))}
```

```
 item: {
    padding: 10,
    marginVertical:5,
    backgroundColor: '#ccc',
    borderColor: 'black',
    borderWidth: 1
  }
```

6. Listeleme içerisinde elementleri scroll aracılığıyla erişmek için ScrollView komponent ekleyiniz.

```
    <ScrollView>
        {wordList.map((word) => (
```

```
          <View key={word} style={styles.item}>
            <Text >{word}</Text>
          </View>
        ))}
      </ScrollView>
```

7. ScrollView yerine daha kullanışlı ve otomatik key ataması yapan FlatList kullanabilirsiniz. Kullanımı için ScrolView olarak tanımladığımız yeri FlatİIst olarak değiştirelim. Branch: step4

```
  const addWordHandler = () => {
    console.log(word);
    setWordList(currentList => [...currentList, {key:
Math.random().toString(),value: word}]);
  }
```

```
<FlatList data={wordList}
        renderItem={itemData =>
          <View style={styles.item}>
          <Text >{itemData.item.value}</Text>
        </View>
         }
         />
```

7. Proje içerisinde bulunan elementleri komponent seklinde parcalama işlemi için components adında bir klasör oluşturularak input alanı için InputCustom.js componenti ve liste içerisindeki gösterim için ise Item.js adında dosya oluşturunuz. Oluşturulan dosya içine ise aşağıdaki kod bloklarını yazınız. Burada oluşturulan kod ise App.js içerisinde bulunan elementleri parçalama yani komponent olarak kullanma işlemi yapılmaktadır.

```
import React from "react";
import { View, Text, StyleSheet, TouchableOpacity } from
"react-native";
const Item = props => {
    return (
        <TouchableOpacity activeOpacity={0.8}
onPress={props.onDelete.bind(this, props.id)}>
            <View style={styles.item}>
                <Text >{props.value}</Text>
            </View>
        </TouchableOpacity>
```

```
    )
}

const styles = StyleSheet.create({
    item: {
        padding: 10,
        marginVertical: 5,
        backgroundColor: '#ccc',
        borderColor: 'black',
        borderWidth: 1
    }
})

export default Item;
```

PItem dosyası liste içerisindeki her bir elementi gösteren componenttir. Burada dikkat edilmesi gereken 2 nokta vardır.

1. Etkileşimli olduğu componentlarda veri aktarımını propslar üzerinde keylerle yapmaktadır
2. Touchable componentları, bu komponent ise element üzerine tıklama kontrolu yapan ve gerekli eventları ortaya çıkaran componenttır.

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


const InputCustom = props => {

    const [word, setWord] = useState('');

    const wordInputHandler = (text) => {
        setWord(text);
    }
    return (
        <View style={styles.container}>
            <TextInput
```

```
                placeholder='Enter Word'
                onChangeText={wordInputHandler}
                style={styles.inputStyle} />
            <Button title="ADD"
onPress={props.addWordHandler.bind(this, word)} />
        </View>
    )
}

const styles = StyleSheet.create({
    container: {
        flexDirection: 'row',
        justifyContent: 'space-between',
        alignItems: 'center'
    },
    inputStyle: {
        width: '80%',
        borderColor: 'black',
        borderWidth: 1,
        padding: 10
    }
})

export default InputCustom;

import { StatusBar } from 'expo-status-bar';
import { useState } from 'react';
import { Button, StyleSheet, Text, TextInput, View,
ScrollView, FlatList } from 'react-native';
import InputCustom from './components/InputCustom';
import Item from './components/Item';

export default function App() {

  const [wordList, setWordList] = useState([]);

  const addWordHandler = word => {
    console.log(word);
```

```
    setWordList(currentList => [...currentList, { key:
Math.random().toString(), value: word }]);
  }

  const removeElement = id => {
    setWordList(currentList => {
      return currentList.filter((item) => item.key !== id)
    })
  }

  return (
    <View style={styles.screen}>
      <InputCustom addWordHandler={addWordHandler} />
      <FlatList data={wordList}
        renderItem={itemData =>
          <Item
            id={itemData.item.key}
            onDelete={removeElement}
            value={itemData.item.value} />
        }
      />
    </View>
  );
}

const styles = StyleSheet.create({
  screen: {
    padding: 40
  },
  container: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center'
  },
  inputStyle: {
    width: '80%',
    borderColor: 'black',
    borderWidth: 1,
```

```
    padding: 10
  },

})
```

Uygulama 1 Sayı Tahmin Oyunu

Bu uygulama ile component, layout, style ve React Native üzerinde gercek bir uygulama nasıl yapılır onu inceleyeceğiz.

Oyun acıklaması ise rastgele bir sayı tutup bu sayı tahmin edebilmek için aşağı ve yukarı butonları ile sayıya yaklaşmaya çalışacağız.

Uygulama için Expo init komutunu kullanarak yeni bir proje açınız ve VSCode üzerinde açınız.

1. İlk olarak proje içerisine component klasörü acıyoruz.
2. Daha sonra ilk olarak projenin Header kısmını bir komponent olarak tanımladığımız Header.js dosyasını tanımlıyoruz. Branch: master

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

const Header = props => {

    return (
        <View style={styles.header}>
            <Text style={styles.title}>{props.title}</Text>
        </View>
    )

}
const styles = StyleSheet.create({
    header :{
        width: '100%',
        height: 90,
        paddingTop:35,
        backgroundColor: '#7B0CF2',
        alignItems:'center'
    },
```

```
    title: {
        color: 'white',
        fontSize: 18,
        paddingTop:10
    }
})
export default Header;
```

3. Tanımlamış oldugumuz Header.js App.js içerisine import ederek aşağıdaki sekilde yazıyoruz.
4. Proje içerisindeki diğer sayfaların dosyalarını tutmak içn projeye screens adında bir klasör ekliyoruz.
5. Screen klasörünün içerisine uygulamamızın ana container olan StartScreen adında bir js dosya oluşturuyoruz. Bu dosya uygulamanın ana layoutunun tanımının yapılacağı komponent olacaktır. Kod blogu ise aşağıdaki gibidir. Branch : step1

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

const StartScreen = props => {
    return (
        <View style={styles.screen}>
            <Text style={styles.title}>Yeni Oyna Başla!</Text>
            <View style={styles.inputContainer}>
                <Text>Bir sayı giriniz!</Text>
                <TextInput />
                <View style={styles.buttonContainer}>
                    <Button title="Sıfırla" onPress={() =>
{ }} />
                    <Button title="OK" onPress={() => { }} />
                </View>
            </View>
        </View>
    )
}
const styles = StyleSheet.create({
    screen: {
        flex: 1,
        padding: 10,
        alignItems: 'center'
```

```
    },
    title: {
        fontSize: 20,
        marginVertical: 10
    },
    buttonContainer: {
        flexDirection: 'row',
        width: '100%',
        justifyContent: 'space-between',
        paddingHorizontal: 15
    },
    inputContainer: {
        width: 300,
        maxWidth: '80%',
        alignItems: 'center',
        shadowColor: 'black',
        shadowOffset: { width: 0, height: 2 },
        shadowRadius: 6,
        shadowOpacity: 0.30,
        backgroundColor: 'white',
        padding: 20,
        borderRadius: 10
    }
})

export default StartScreen;
```

6. Proje içerisinde oluşturmuş oldugumuz StartScreen içerindeki input alanını bir komponent olarak oluşturmak için Card.js adında bir dosyayı components içerisinde tanımlayınız.

Kod içerisinde burada dikkat edilmesi gereken nokta ..styles.card ve …props.style komutlarıdır. Burada aslında Javascript te olan özellik kullanılarak ekstra gelen propslarıda style içerisine eklemektedir. Ayrıca props.children ise Card komponent içerisinde bulunan componentları buradaki View içerisine doğrudan alacaktır ve styleları etki edecektir.

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

const Card = props => {
    return (
        <View style={{ ...styles.card, ...props.style }}>
```

```
            {props.children}
        </View>
    )
}
const styles = StyleSheet.create({
    card: {
        shadowColor: 'black',
        shadowOffset: { width: 0, height: 2 },
        shadowRadius: 6,
        shadowOpacity: 0.30,
        backgroundColor: 'white',
        padding: 20,
        borderRadius: 10
    }

})

export default Card;
```

7. App.js dosyası içerisine Card.js componenti inputContainer view yerine Tanımlayınız ve style içerisindeki card içerisindeki tanımlanan style değerlerini cıkartınız.
8. StartScreen içerisinde tanımlanan butonlara style eklenerek Constantlar içerisinde tanımlanan renk kodları almak için Constant klasörü oluşturularak Colors.js dosyası tanımlayınız ve oradan renkleri alınız. Branch: step2

```
export default {
    mainColor : '#F20CD6',
    secondColor: '#6283F1'
}
```

9. StartScreen içerisinde bulunan TextInput komponent için Component klasörü altına Input.js diye bir komponent oluşturularak gerekli style eklenilerek yeni bir compnent oluşturunuz. Branch: step2

```
import React from "react";
import { StyleSheet, TextInput } from "react-native";

const Input = props => {
```

```
    return <TextInput {...props}
style={{ ...styles.input, ...props.style }} />
}
const styles = StyleSheet.create({
    input: {
        height: 30,
        borderBottomColor: 'grey',
        borderBottomWidth: 1,
        marginVertical: 10
    }

})

export default Input;
```

10. Oluşturmuş oldugumuz input değeri için bir state tanımı yapılarak değeri setleyen fonksiyonu yazınız.Branch: step2

```
 const {number,setNumber} = useState('');
```

```
    const numberInputHandler = inputText => {
        setNumber(inputText.replace(/[^0-9]/g,''));
    }
```

11. StartScreen içerisinde çıkan pad kapatmak için TouchableWithoutFeedback komponent en üst noktaya ekleyiniz ve Keyboard dismiss komutu yazınız. Branch: step2

```
  <TouchableWithoutFeedback
        onPress={() => {
            Keyboard.dismiss();
        }}>
```

12. StartScreen dosyasının son hali ise; Branch: step2

```
import React, { useState } from "react";
import { View, StyleSheet, Text, TextInput, Button, Keyboard }
from "react-native";
import { TouchableWithoutFeedback } from "react-native-web";
import Card from "../components/Card";
```

```
import Input from "../components/Input";
import Colors from '../constants/colors';

const StartScreen = props => {

    const [numberValue, setNumberValue] = useState('');

    const numberInputHandler = inputText => {
        setNumberValue(inputText.replace(/[^0-9]/g, ''));
    }

    return (
        <TouchableWithoutFeedback
            onPress={() => {
                Keyboard.dismiss();
            }}>
            <View style={styles.screen}>
                <Text style={styles.title}>Yeni Oyna Başla!</
Text>
                <Card style={styles.inputContainer}>
                    <Text>Bir sayı giriniz!</Text>
                    <Input
                        style={styles.input}
                        blurOnSubmit
                        autoCapitalize="none"
                        autoCorrect={false}
                        keyboardType="numeric"
                        maxLength={2}
                        onChangeText={numberInputHandler}
                        value={numberValue}
                    />
                    <View style={styles.buttonContainer}>
                        <View style={styles.button}>
                            <Button title="Sıfırla"
onPress={() => { }} color={Colors.mainColor} />
                        </View>
                        <View style={styles.button}>
```

```
                            <Button title="OK" onPress={() =>
{ }} color={Colors.secondColor} />
                        </View>
                    </View>
                </Card>
            </View>
        </TouchableWithoutFeedback>
    )
}
const styles = StyleSheet.create({
    screen: {
        flex: 1,
        padding: 10,
        alignItems: 'center'
    },
    title: {
        fontSize: 20,
        marginVertical: 10
    },
    buttonContainer: {
        flexDirection: 'row',
        width: '100%',
        justifyContent: 'space-between',
        paddingHorizontal: 15
    },
    inputContainer: {
        width: 300,
        maxWidth: '80%',
        alignItems: 'center'
    },
    button: {
        width: 100
    },
    input: {
        width: 50,
        textAlign: 'center'
    }
})
```

```
export default StartScreen;
```

13. StartScreen içerisinde bulunan sıfırla ve Ok butonun eventlerını yazmak için 2 tane fonksiyonu tanımı yapınız. Sıfırla için yapılacak işlem input içerisindeki değeri temizlemek olacaktır. Diğeri ise değeri onaylamak ve karşılaştırma yapacaktır. Branch: step 3

```
const resetInputHandler = () => {
        setNumberValue('');
    }
    const confirmInputHandler = () => {
        const chosenNumber = parseInt(numberValue);
        if (chosenNumber === NaN || chosenNumber <= 0 ||
chosenNumber > 99) {
            Alert.alert(
                'Invalid Number!',
                'Sayı 1 ila 99 arasında olmalıdır!',
                [{ text: 'OK', style: 'destructive', onPress:
resetInputHandler }]);
            return;
        }
        setConfirmed(true);
        setSelectedNumber(chosenNumber);
        setNumberValue('');
    }

    let confirmedOutput;

    if (confirmed) {
        confirmedOutput = <Text>Seçilen Numara :
{selectedNumber}</Text>
    }
```

14. StartScreen içerisinde bulunan confirmedOutput component daha düzenli hale getirmek için daha önce tanımladığımız Card component içerisine alıyoruz. Daha sonra burada seçmiş oldugumuz numarayı gösterim için bir tane NumberContainer componenti component klasörünün içerisinde oluşturuyoruz. Branch:step4

```
   if (confirmed) {
        confirmedOutput = (
```

```
        <Card style={styles.confirmContainer}>
          <Text>Seçilen Numara</Text>
          <NumberContainer>{selectedNumber}</NumberContainer>
          <Button title="Başla" onPress={() =>
props.onStartGame(selectedNumber)}/>
        </Card>
        )
    }
```

NumberContainer.js

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

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

const NumberContainer = props => {
  return (
    <View style={styles.container}>
      <Text style={styles.number}>{props.children}</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    borderWidth: 2,
    borderColor: Colors.mainColor,
    padding: 10,
    borderRadius: 10,
    marginVertical: 10,
    alignItems: 'center',
    justifyContent: 'center'
  },
  number: {
    color: Colors.secondColor,
    fontSize: 22
```

```
  }
});

export default NumberContainer;
```

15. Oyunun tahmin ekranı için GameScreen adında bir component oluşturulup seçmiş olduğumuz sayıya göre tahmin yapılmasını sağlayacak ekranı aşağıdaki gibi kodluyoruz. Burada kullanıcı ile aynı sayı tutulmuş ise tekrar tutmanısı isteyen generateRandomBetween methodu random fonksiyonu kullanılarak yazılır. Branch:step4

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

import NumberContainer from '../components/NumberContainer';
import Card from '../components/Card';

const generateRandomBetween = (min, max, exclude) => {
  min = Math.ceil(min);
  max = Math.floor(max);
  const rndNum = Math.floor(Math.random() * (max - min)) +
min;
  if (rndNum === exclude) {
    return generateRandomBetween(min, max, exclude);
  } else {
    return rndNum;
  }
};

const GameScreen = props => {
  const [currentGuess, setCurrentGuess] = useState(
    generateRandomBetween(1, 100, props.userChoice)
  );

  return (
    <View style={styles.screen}>
      <Text>Karşı Tahmin</Text>
      <NumberContainer>{currentGuess}</NumberContainer>
      <Card style={styles.buttonContainer}>
        <Button title="AŞAĞI" onPress={() => {}} />
```

```
        <Button title="YUKARI" onPress={() => {}} />
      </Card>
    </View>
  );
};

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    padding: 10,
    alignItems: 'center'
  },
  buttonContainer: {
    flexDirection: 'row',
    justifyContent: 'space-around',
    marginTop: 20,
    width: 300,
    maxWidth: '80%'
  }
});

export default GameScreen;
```

16. İki adet ekranımız olduğu için oyunu içerisinde ekranlar arası gecişleri saplayabilmek için App.js içerisini aşağıdaki gibi revize etmemiz gerekmektedir. Eğer kullanıcı bir sayı seçmiş ise GameScreen yani oyun ekranına geçiş yapması istiyoruz. Branch:step4

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

import Header from './components/Header';
import StartScreen from './screen/StartScreen';
import GameScreen from './screen/GameScreen';

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

  const startGameHandler = selectedNumber => {
    setUserNumber(selectedNumber);
```

```
  };

  let content = <StartScreen onStartGame={startGameHandler} /
>;

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

  return (
    <View style={styles.screen}>
      <Header title="Tahmin Et!" />
      {content}
    </View>
  );
}

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

17. Oyunun bitmesi durumunda başka bir ekrana yönlendirmek için screen klasörünün altına GameOverScreen.js tanımlanarak oynunun bitmesi ile birlikte bu ekrana yönlendirilmelidir. Burada yönlendirme işlemi için App.js içerisindeki content değişkenini oyunun bittiğine göre ayarlanmaldır. Oyunun bitişi ise tahmin edilen sayı kullanıcının girmiş olduğu sayıya denk geldiğinde yönlendirme yapmaldır. Ayrıca her aksiyonda kontrol için useEffect fonksiyonu kullanılmış olup useEffect sayfa üzerindeki her değişiliğe göre tekrardan render olmaktadır.

```
import React from "react";
import { Button, StyleSheet, Text, View } from "react-native";

const GameOverScreen = props => {
    return(
        <View style={styles.screen}>
            <Text>
                Oyun Bitti!
            </Text>
```

```
            <Text>Tahmin Sayısı : {props.roundsNumber}</Text>
            <Button title="Yeni Oyun!"
onPress={props.onNewGame} />
        </View>
    )
}

const styles = StyleSheet.create({
    screen: {
        flex: 1,
        justifyContent: 'center',
        alignItems: 'center'
    }
})

export default GameOverScreen;
```

GameScreen son hali.

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

import NumberContainer from '../components/NumberContainer';
import Card from '../components/Card';

const generateRandomBetween = (min, max, exclude) => {
    min = Math.ceil(min);
    max = Math.floor(max);
    const rndNum = Math.floor(Math.random() * (max - min)) +
min;
    if (rndNum === exclude) {
        return generateRandomBetween(min, max, exclude);
    } else {
        return rndNum;
    }
};
```

```
const GameScreen = props => {
    const [currentGuess, setCurrentGuess] = useState(
        generateRandomBetween(1, 100, props.userChoice)
    );

    const [rounds, setRounds] = useState(0);

    const currentLow = useRef(1);
    const currentHigh = useRef(100);

    const { userChoice, onGameOver } = props;

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

    const nextGuessHandler = direction => {

        if (
            (direction === 'lower' && currentGuess <
props.userChoice) ||
            (direction === 'greater' && currentGuess >
props.userChoice)
        ) {
            Alert.alert("Ohhhhh !", "Hatalı bir tahmin....",
                [{ text: "Üzgünüm!", style: 'cancel' }]);
            return;
        }

        if (direction === 'lower') {
            currentHigh.current = currentGuess;
        }
        else {
            currentLow.current = currentGuess;
        }
```

```
        const nextNumber =
generateRandomBetween(currentLow.current, currentHigh.current,
currentGuess);
        setCurrentGuess(nextNumber);
        setRounds(curRounds => curRounds + 1);
    }

    return (
        <View style={styles.screen}>
            <Text>Karşı Tahmin</Text>
            <NumberContainer>{currentGuess}</NumberContainer>
            <Card style={styles.buttonContainer}>
                <Button title="AŞAĞI"
onPress={nextGuessHandler.bind(this, 'lower')} />
                <Button title="YUKARI"
onPress={nextGuessHandler.bind(this, 'greater')} />
            </Card>
        </View>
    );
};

const styles = StyleSheet.create({
    screen: {
        flex: 1,
        padding: 10,
        alignItems: 'center'
    },
    buttonContainer: {
        flexDirection: 'row',
        justifyContent: 'space-around',
        marginTop: 20,
        width: 300,
        maxWidth: '80%'
    }
});

export default GameScreen;
```

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

import Header from './components/Header';
import StartScreen from './screen/StartScreen';
import GameScreen from './screen/GameScreen';
import GameOverScreen from './screen/GameOverScreen';

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

  const startGameHandler = selectedNumber => {
    setUserNumber(selectedNumber);
    setGuessRounds(0);
  };

  const gameOverHandler = number => {
    setGuessRounds(number);
  }

  const newGameHandler = () => {
    setUserNumber(null);
    setGuessRounds(0);
  }

  let content = <StartScreen onStartGame={startGameHandler} /
>;

  if (userNumber && guessRounds <= 0) {
    content = <GameScreen userChoice={userNumber}
onGameOver={gameOverHandler} />;
  }
  else if (guessRounds > 0) {
    content = <GameOverScreen roundsNumber={guessRounds}
onNewGame={newGameHandler} />
  }
```

```
  return (
    <View style={styles.screen}>
      <Header title="Tahmin Et!" />
      {content}
    </View>
  );
}

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

Uygulama 2 Shop App
Font,Navigation,Redux,Web Service

1. Uygulama için yeni bir tane rn-chapter-3 adında uygulama acınız.
2. Uygulama bir alışveriş uygulaması olup ürünlerin listelendiği, ürün detayının görüntülendiği, alışveriş sepetinin oluşturulduğu gibi temel özelliklere sahip bir uygulamadır. Uygulama içerisinde react native içerisinde kullanılan bir çok özellikler kullanılacaktır.
3. Proje içerisindeki oluşturacağımız js dosyaları düzenli bir folder yapısında tutmak için aşağıdaki klasörleri oluşturunuz.

    -components : Componentları tutan klasör
    -constants : Constant olarak tanımlanacak variable tutacak klasör
    -navigation: Uygulamanın menüler arası gecişleri için tanımlanacak klasör
    -screens : Uygulama içerisinde kullanılacak ekranları tutan klasör
        shop: Alışveriş ile ilgili detay ekranları için
        user : kullanıcıların detay ekranları için
    -store: Redux ile sayfaların datalarını yöneteceğimiz js dosyaları

4. Proje içerisindeki oluşturacağımız js dosyaları düzenli bir folder yapısında tutmak için aşağıdaki klasörleri oluşturunuz.
5. Proje içerisinde kullanılacak kütüphaneleri npm aracılığıyla indiriniz.

npm install —save redux react-redux react-navigation

expo install react-native-gesture-handler react-native-reanimated

6. Proje bulunan tanımladığımız productOverview içerisinde productları tanımlamak için models klasörü tanımlayıp içerisine product.js adında bir class oluşturunuz. .Branch: step2

```
class Product {

    constructor(id,ownerId,title,imageUrl,description,price){
        this.id = id;
        this.ownerId = ownerId;
        this.title = title;
        this.imageUrl = imageUrl;
        this.description = description;
        this.price =price;
    }


}
```

7. Projenin product için redux tanımı yapmak için reducers içerisine products.js dosyası oluşturunuz ve tanımlanan reducer App.js içerisinde proje içerisine injecte edecek kodu yazınız.Branch: step2

```
import PRODUCTS from "../../data/dummy-data";

const initialState = {
    avaiableProducts: PRODUCTS,
    userProducts: PRODUCTS.filter(prod => prod.ownerId === 
'u1')
};

export default (state = initialState,action) => {
    return state;
}
```

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';
import { createStore, combineReducers } from 'redux';
import { Provider } from 'react-redux';
import productReducer from './store/reducers/products';
```

```
const rootReducer = combineReducers({
  products : productReducer
});

const store = createStore(rootReducer);

export default function App() {
  return (
    <Provider store={store}>
      <View>

      </View>
      </Provider>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

8. ProductOverviewScreen ekranını oluşturduğunuz dummy dataya göre useSelector kullanılarak cekecek olan kodu yazınız. Kodu son hali aşağıdaki gibidir. Branch:step2

```
import React from "react";
import { FlatList, Text } from "react-native";
import { useSelector } from "react-redux";

const ProductOverviewScreen = props => {

    const products = useSelector(state =>
state.products.avaiableProducts)

    return <FlatList
        data={products}
```

```
        keyExtractor={item => item.id}
        renderItem={
            itemData => <Text>{itemData.item.title
            }</Text>
        } />
}

export default ProductOverviewScreen;
```

9. Proje içerisine Navigator eklemek için navigator klasörü içerisine ShopNavigator.js dosyası eklenerek stack navigator tanımı yapınız. React native navigator kütüphanesi 4 ile 5 sürümü arasında fark vardır. Bunun için resim sitesi üzerinden kontrole ederek eklenmesi doğru sonucu verecektir. Shop navigator oluşturulduktan sonra App.js dosyasının içerisine Provider ana tagı arasına navigator tanımını yapınız. Branch: step3

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import { Platform } from 'react-native';
import ProductOverviewScreen from '../screens/shop/
ProductOverviewScreen';
import Colors from '../constants/Colors';

const ProductsNavigator = createStackNavigator({
    ProductsOverview: ProductOverviewScreen
},
    {
        defaultNavigationOptions: {
            headerStyle: {
                backgroundColor: Platform.OS === 'android' ?
Colors.color1 : ''
            },
            headerTintColor: Platform.OS === 'android' ?
'white' : Colors.color1
        }
    }
);

export default createAppContainer(ProductsNavigator);
```

10. Proje içerisine bulunan bütün ürünleri gösterdiğimiz ProductOverview içindeki bulunan flatlist itemlarını bir component klasörünün altına shop adında alt klasör oluşturulup productıtem adında bir component oluşturunuz ve gerekli style bilgisini de ekleyerek lsiteyi düzenleyiniz. Branch :step4

```
import React from "react";
import { View,Text,Image,StyleSheet, Button } from "react-
native";

const ProductItem = props => {
    return (
    <View style={styles.product}>
        <View style={styles.imageContainer}>
        <Image style={styles.image} source={{uri:props.image}}
/>
        </View>
        <View style={styles.details}>
        <Text style={styles.title}>{props.title}</Text>
        <Text style={styles.price}>{props.price.toFixed(2)}
TL</Text>
        </View>
        <View style={styles.actions}>
            <Button title="Detay"
onPress={props.onViewDetail}/>
            <Button title="Sepete At"
onPress={props.onAddToCart} />
        </View>
    </View>
    )
}

const styles = StyleSheet.create({

    product: {
        shadowColor: 'black',
        shadowOpacity:0.26,
        shadowOffset:{width:0,height:2},
        shadowRadius:8,
        elevation:5,
        borderRadius:10,
```

```
        backgroundColor: 'white',
        height:300,
        margin:20
    },
    imageContainer:{
        width:'100%',
        height:'60%',
        borderTopLeftRadius:10,
        borderTopRightRadius:10,
        overflow:'hidden'
    },
    image: {
        width:'100%',
        height:'100%'
    },
    details:{
        alignItems: 'center',
        height:'15%',
        padding:10
    },
    title:{
        fontSize:18,
        marginVertical:4
    },
    price: {
        fontSize: 14,
        color: '#888'
    },
    actions :{
        flexDirection: 'row',
        justifyContent: 'space-between',
        alignItems: 'center',
        height:'25%',
        paddingHorizontal:15
    }

})
```

```
export default ProductItem;
```

PRODUCT OVERVIEW IÇERISINDEKI FLATLIST SON HALI

```
<FlatList
        data={products}
        keyExtractor={item => item.id}
        renderItem={
            itemData => <ProductItem
            image={itemData.item.imageUrl}
            title={itemData.item.title}
            price={itemData.item.price}
            onViewDetail={()=> {} }
            onAddToCart={() => {} } />
        } />
```