ANKARA

ÜNİVERSİTESİ

MÜHENDİSLİK

FAKÜLTESİ

BİLGİSAYAR

MÜHENDİSLİĞİ

BLM4538- IOS İLE MOBİL UYGULAMA-II

GELİŞTİRME I PROJESİ

MELİKE ŞAHİN

20290290

HAZİRAN 2023

https://github.com/melike-sahin/Cocktails-React-Native

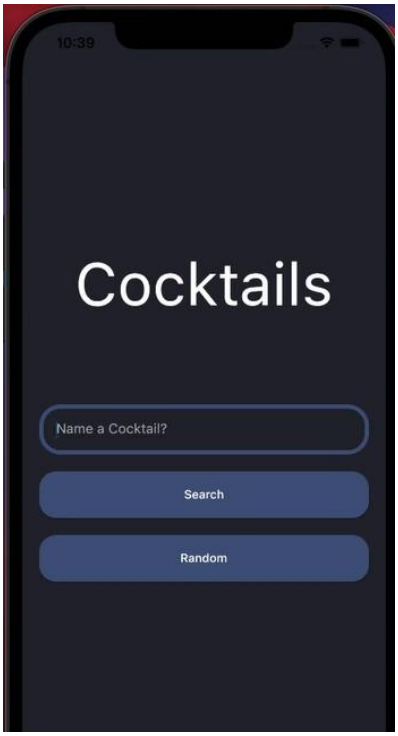
1. UYGULAMANIN TASARIMI

Cocktails uygulamasını IOS ile Mobil Uygulama Geliştirme-II Dersi için geliştirdim. Projemde 2 panel bulunmaktadır. Bu paneller aşağıdakilerden oluşmaktadır:

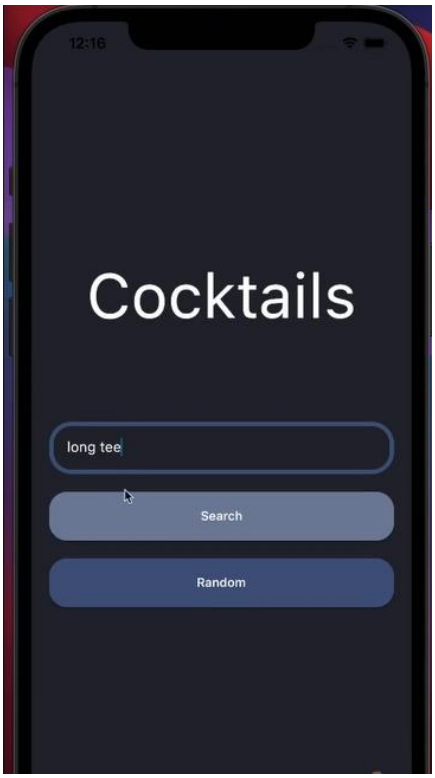
1- Kokteyl arama ekranı
2- Kokteyl tariflerinin bulunduğu ekran

1.1 Kokteyl Arama Ekranı

Kokteyl arama ekranında kokteyl ismi girilerek arama yapılabilir ve kokteylin tarifine, görseline ve malzemelerine ulaşılabilmektedir. Aynı zamanda kokteyl arama ekranından random seçeneği ile herhangi bir kokteyle ulaşılabilmektedir.



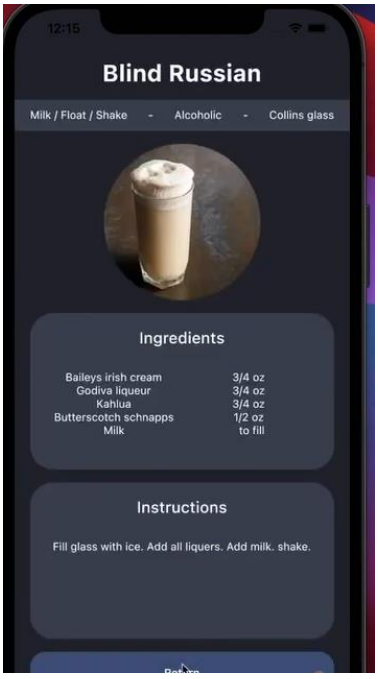
Şekil 1.1 Arama Ekranı



Şekil 1.1 Arama Ekranı

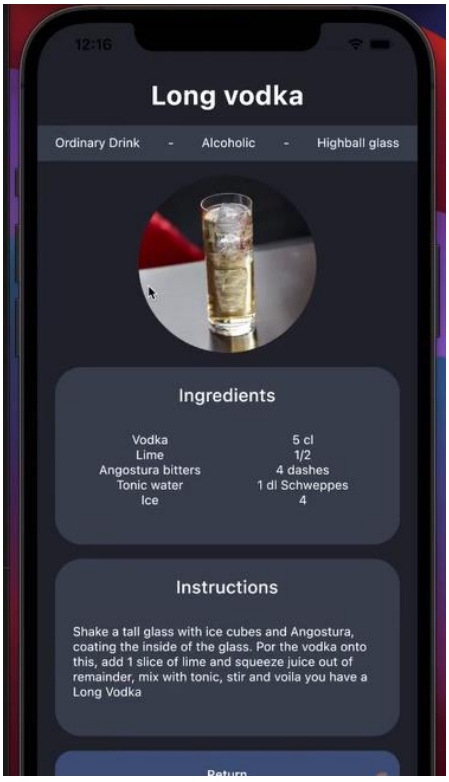
1.2 Kokteyl Tariflerinin Bulunduğu Ekran

Kokteyl tariflerinin bulunduğu ekranda arama yapılan ya da random seçeneği sonucunda bulunan kokteyle ait görsel, malzemeler ve kokteylin tarifi karşımıza çıkmaktadır.



Şekil 1.2.1 Tarif Ekranı

Şekil 1.2.2 Tarif Ekranı

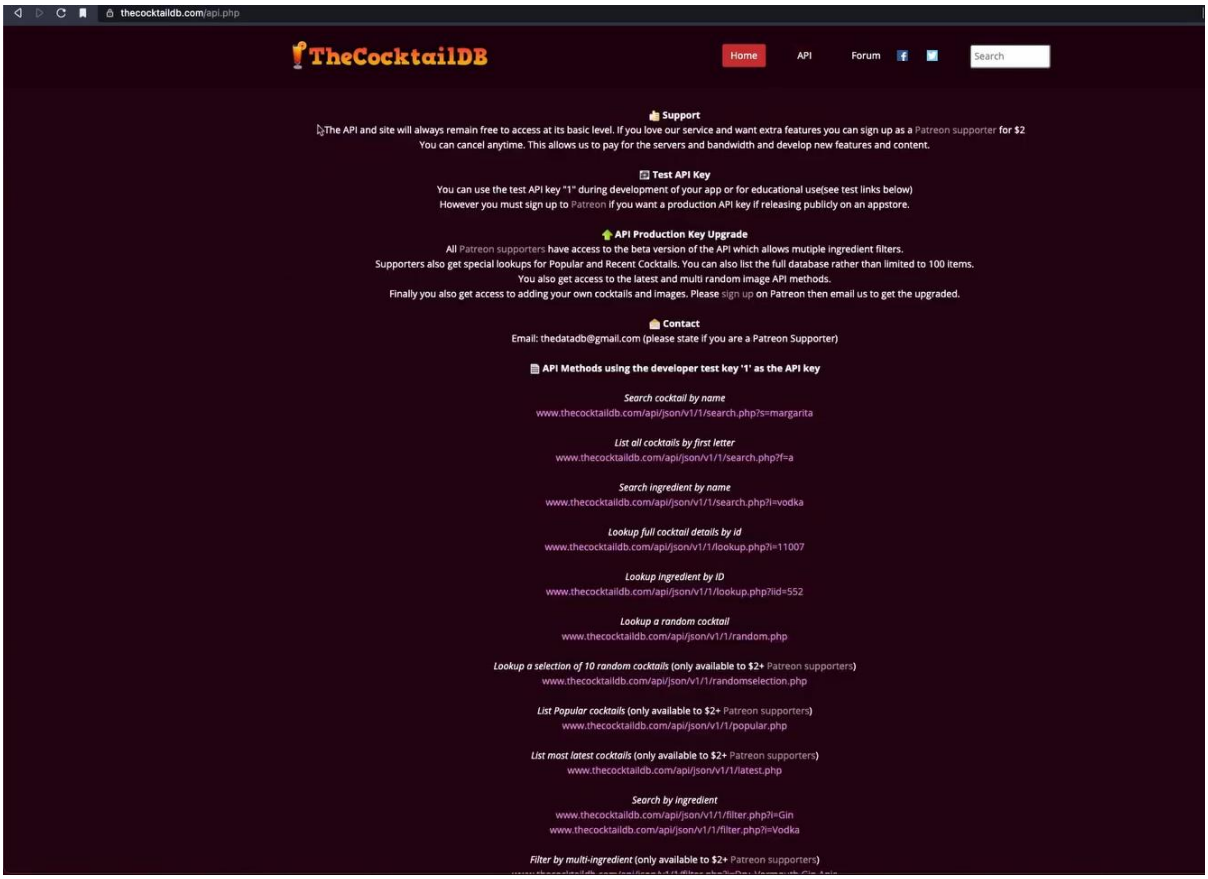


Şekil 1.2.3 Tarif Ekranı

2. API

Uygulamamızdaki kokteyl türlerine, malzemelerine ve tariflerine ulaşabilmek için çevrimiçi bir web sitesi olan TheCocktailDB üzerinden bir veritabanı ile uygulamamı bağladım.

Şekil 2.1 API Adresi



Şekil 2.1 API Adresi

3. BACKEND

Proje kapsamlı olduğu için kodların belli bir kısmı paylaşılacaktır. Kodlar hakkında daha detaylı bilgi için https://github.com/melike-sahin/Cocktails-React-Native adresinden Github hesabımı ziyaret edebilirsiniz.

1. Aşağıdaki görselde Cocktail interface'inin tanımlaması yapılmıştır.

```
export interface Cocktail {
    cocktailId: string;
    cocktailName: string;
    cocktailImageLink?: string;
    cocktailIngredients?: string[];
}
```

2. Aşağıdaki görselde ise API kaynağından verileri çeken async fonksiyon tanımı yapılmıştır.

```
export const getCocktailsFromApiAsync = async (url: Urls, parameter?: string) => {
    try {
        const response = await fetch(
            parameter ? url + parameter : url
        );
        const json = await response.json();
        if (url === Urls.RandomCocktailUrl || url === Urls.IdCocktailUrl) return randomOrIdResponseToCocktail(json);
        if (url === Urls.SearchCocktailUrl) return searchResponseToCocktail(json);
        return json;
    } catch (error) {
        return 'No data available';
    }
};
```

3. Bu görselde kokteyl aramak için kullanılacak olan komponent tanımlanmıştır.

```
const SearchCocktail = ({ navigation }: { navigation: any }): JSX.Element => {

    const searchText: string = 'Search';
    const [text, setText] = useState<string>('');
    const [data, setData] = useState<null | string | Cocktail[]>(null);

    const isString = (data: any): data is String => {
        return typeof data === 'string' || data instanceof String
    }

    const getData = async () => {
        Keyboard.dismiss();
        const fetchedData = await getCocktailsFromApiAsync(Urls.SearchCocktailUrl, text);
        setData(fetchedData);
    }

    const renderItem = ({ item }: { item: string }) => (
        <Pressable onPress={() => {
            navigation.navigate('Id Cocktail',
                { cocktailId: data && !isString(data) && data.find(el => el.cocktailName === item)?.cocktailId })
        }}>
            <View style={styles.item}>
                <Text style={styles.textItem}>{item}</Text>
            </View>
        </Pressable>
    )

    return (
        <View style={styles.container}>
            <TextInput
                style={styles.input}
                onChangeText={setText}
                value={text}
                autoFocus={true}
                onSubmitEditing={getData}
            />
            <Pressable style={styles.button} onPress={getData}>
                <Text style={styles.text}>{searchText}</Text>
            </Pressable>
            {data && !isString(data) &&
                <View style={styles.list}>
                    <FlatList data={data.map(el => el.cocktailName)}
                        renderItem={renderItem}
                        keyExtractor={(item: string, index: number) => index.toString()}
                    />
                </View>
            }
            {data && isString(data) &&
                <Text style={styles.noData}>{data}</Text>
            }
        </View>
    )
}
```

4. Bu görselde ise kokteyl arama komponenti için stil kuralları yazılmıştır.

```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        backgroundColor: '#fff',
        alignItems: 'center',
        justifyContent: 'center',
    },
    input: {
        backgroundColor: '#99423d',
        color: 'white',
        paddingVertical: 5,
        paddingHorizontal: 10,
        width: '50%',
        borderRadius: 5,
        fontSize: 16,
    },
    button: {
        paddingVertical: 5,
        paddingHorizontal: 10,
        backgroundColor: 'black',
        marginTop: 15,
        borderRadius: 5,
    },
    text: {
        fontSize: 16,
        color: 'white',
    },
    list: {
        marginTop: 20,
        maxHeight: 160,
    },
    item: {
        margin: 5,
        padding: 10,
        backgroundColor: '#3d9970',
        borderRadius: 4,
        borderWidth: 3,
        borderColor: '#99423d',
    },
    textItem: {
        textAlign: 'center',
        color: 'white',
    },
    noData: {
        fontSize: 18,
        fontWeight: 'bold',
        color: '#993d94',
        marginTop: 15,
    }
});
```

5. Son olarak ilgili komponentler ve servisler toplanarak çatı bir

elementte uygulamanın çalışması sağlanmıştır.

```
const App = (): JSX.Element => {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName='Landing'>
        <Stack.Screen name="Landing" component={Landing} options={{ headerShown: false }} />
        <Stack.Screen name="Home" component={Home} />
        <Stack.Screen name="Id Cocktail" component={IdCocktail} />
      </Stack.Navigator>
      <StatusBar style="auto" />
    </NavigationContainer>
  );
}
```