



# İSTANBUL OKAN ÜNİVERSİTESİ

OKAN ÜNİVERSİTESİ

BULUT BİLİŞİM FİNAL ÖDEVİ  
GERÇEK ZAMANLI SOHBET UYGULAMASI

19MY93019  
ABDULBAKİ ZİRİH

React Native ve Firebase kullanılarak geliştirilen bu uygulamanın amacı kullanıcıların hesap oluşturup isterlerse kendileri sohbet odası kurup veya daha önce açılmış olan bir sohbet odasına bağlanıp kullanıcılarla mesajlaşmaları hedeflenmiştir.

# İçindekiler

1 Giriş.....	3
1.1 Teorik Algoritma.....	3
1.2 Projede Kullanılan Teknolojiler.....	3
1.3 npm ve Yarn.....	3
1.4 Projede Kullanılan Paket ve Kütüphaneler.....	3
1.4.1 Projedeki Tüm Npm Paketleri.....	4
2 Uygulamanın Geliştirilmesi.....	5
2.1 Klasör Yapısı.....	5
2.2 Dosyaların İçerik Aktarılması.....	6
2.2.1 Proje İçerisinden Örnek Kullanım.....	7
2.3 Özel Font Kullanımı.....	8
2.4 Firebase.....	10
2.4.1 Firebase Entegrasyonu.....	11
2.4.2 Firebase Projede Kullanımı.....	11
2.5 Stillerin Kullanımı.....	11
2.6 Redux.....	13
2.7 Oturum Ekranı.....	15
2.8 Ana Ekran.....	17
2.9 Ayarlar Ekranı.....	18
2.10 Sohbet Odası Ekleme Ekranı.....	19
2.11 Sohbet Ekranı.....	20
3 Uygulamanın Test Edilmesi.....	21
4 Sonuç.....	22

# 1 Giriş

## 1.1 Teorik Algoritma

1. Uygulama ilk açıldığında oturum sayfası açılır.
  1. İsterse var olan bir hesap ile oturum açar.
  2. İsterse yeni bir hesap oluşturur.
2. Kullanıcı girişi yapıldığında sohbet odaları listelenir.
3. Kullanıcı isterse bir sohbet odasına bağlanabilir.
4. Kullanıcı yeni bir sohbet odası açabilir.
5. Sohbet odasında odadaki diğer kullanıcılar ile mesajlaşabilir.
6. Ayarlar bölümünden profil bilgilerini güncelleyebilir.
7. Ayarlar bölümünden hesabını silebilir.

## 1.2 Projede Kullanılan Teknolojiler

- JavaScript
- React Native
- JSX
- AsyncStorage
- Firebase

## 1.3 npm ve Yarn

npm Node.js ile birlikte gelen paket yönetim sistemidir. Projeye bir paket kurulacağı zaman bu görevi üstlenen npm'dir. Yarn ise Facebook tarafından geliştirilen npm alternatifidir. Yarn'ın görevi ise npm'in yaptığı işi üstlenmektir. Yani ikiside aynı amaca hizmet ediyor. Bu proje geliştirilirken Yarn kullanımı tercih edilmiştir.

## 1.4 Projede Kullanılan Paket ve Kütüphaneler

Projede bir çok npm paketi kullanıldı. Bunlardan en öne çıkanları "Redux" ve "React Navigation" bu iki pakettir tabi burada firebase ve benzeri sistemler için kullanılan paketlerde mevcuttur. Fakat bu paketler ilgili teknolojilerin kullanımı için gerekli olan paketlerdir. Redux ve React Navigation ise uygulama geliştirilebilirliğini kolaylaştıran, kullanımı isteğe bağlı olup gelişmiş npm paketi ve sistemlerdir.

Bir Node.js projesinde kurulan paketler "package.json" dosyasında eğer geliştirici modu için eklemişsek "devDependencies" altına, sadece projeye dahil etmişsek "dependencies" altına gelir.

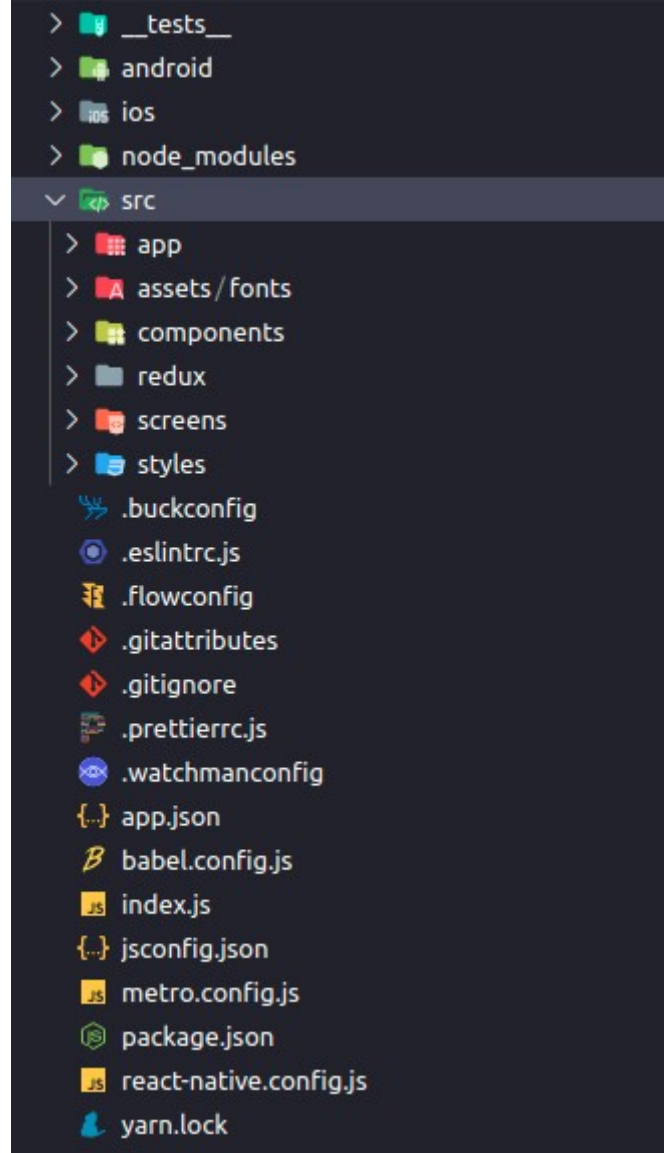
### 1.4.1 Projedeki Tüm Npm Paketleri

```
{
  "dependencies": {
    "@react-native-async-storage/async-storage": "^1.13.2",
    "@react-native-community/masked-view": "^0.1.10",
    "@react-native-firebase/app": "^10.4.0",
    "@react-native-firebase/auth": "^10.4.0",
    "@react-native-firebase/database": "^10.4.0",
    "@react-navigation/native": "^5.8.10",
    "@react-navigation/stack": "^5.12.8",
    "querystring": "^0.2.0",
    "react": "16.13.1",
    "react-native": "0.63.4",
    "react-native-gesture-handler": "^1.9.0",
    "react-native-reanimated": "^1.13.2",
    "react-native-safe-area-context": "^3.1.9",
    "react-native-screens": "^2.16.1",
    "react-native-vector-icons": "^7.1.0",
    "react-redux": "^7.2.2",
    "redux": "^4.0.5",
    "redux-thunk": "^2.3.0"
  },
  "devDependencies": {
    "@babel/core": "^7.12.10",
    "@babel/runtime": "^7.12.5",
    "@react-native-community/eslint-config": "^2.0.0",
    "babel-jest": "^26.6.3",
    "babel-plugin-module-resolver": "^4.1.0",
    "eslint": "^7.17.0",
    "jest": "^26.6.3",
    "metro-react-native-babel-preset": "^0.64.0",
    "react-test-renderer": "16.13.1"
  }
}
```

## 2 Uygulamanın Geliştirilmesi

### 2.1 Klasör Yapısı

Klasör yapısı projenin geliştirilebilirliği açısından çok önemli bir konu bu proje için bu konuya dikkat edilmiş ve düzenli bir yapı kurulmuştur. Bir çok proje ve framework de src adında bir klasör ve bu klasörün içerisinde uygulamanın geliştirilmesinde kullanılan çeşitli dosya türleri bulunur. Bu projede de global bir düzen benimsenmiştir.



## 2.2 Dosyaların İçe Aktarılması

Proje ilerledikçe dosya sayısı da artamay başlar bazen bir dosyayı başka bir yere taşımak istediğimizde ilgili dosya başka dosyalar tarafından içe aktarılmış veya ilgili dosyada başka dosyalar içe aktarılmış olabilir. Her ikisinde aynı anda olabilir. Bu sebeple içe aktarılan dosyaların yollarını değiştirmek gerekebilir. Kullanılan kod editörleri bu işlemi otomatik yapabilmekte fakat hem kodun okunabilirliği hemde olası editör hatalarından kaçınmak için takma adlar kullanarak temiz bir kod elde etmiş oluruz. JavaScript için babel eklentisi olan “babel-plugin-module-resolver” eklentisini bu projede kullanılmıştır. Öncelikle ilgili paketin yarn ile kurulumu yapılmıştır.

```
abdulbaki@Abdulbaki:~/Desktop/final_rtca$ yarn add --dev babel-plugin-module-resolver
```

Ardından babel.js'in yapılandırma dosyası olan babel.config.js içerisinde alias tanımlamaları yapıldı.

```
const { join } = require('path');

const src = uri => join(__dirname, 'src', uri);

module.exports = {
  presets: ['module:metro-react-native-babel-preset'],
  plugins: [
    [
      "module-resolver",
      {
        root: [".src"],
        alias: {
          "@app": src('app'),
          "@assets": src('assets'),
          "@components": src('components'),
          "@screens": src('screens'),
          "@redux": src('redux'),
          "@styles": src('styles')
        }
      }
    ]
  ]
};
```

Bu aşamadan sonra uygulamanın geliştirildiği kod editörü "Visual Studio Code" editörü için ilgili path tanımlamalarını birde jsconfig.js adında bir dosya oluşturulup ilgili dosyanın içerisinde

yapılmıştır. Bu sayede bir dosyayı içe aktarırken otomatik path tamamlaması ve ipucu gibi çeşitli özellikleri kullanabildik.

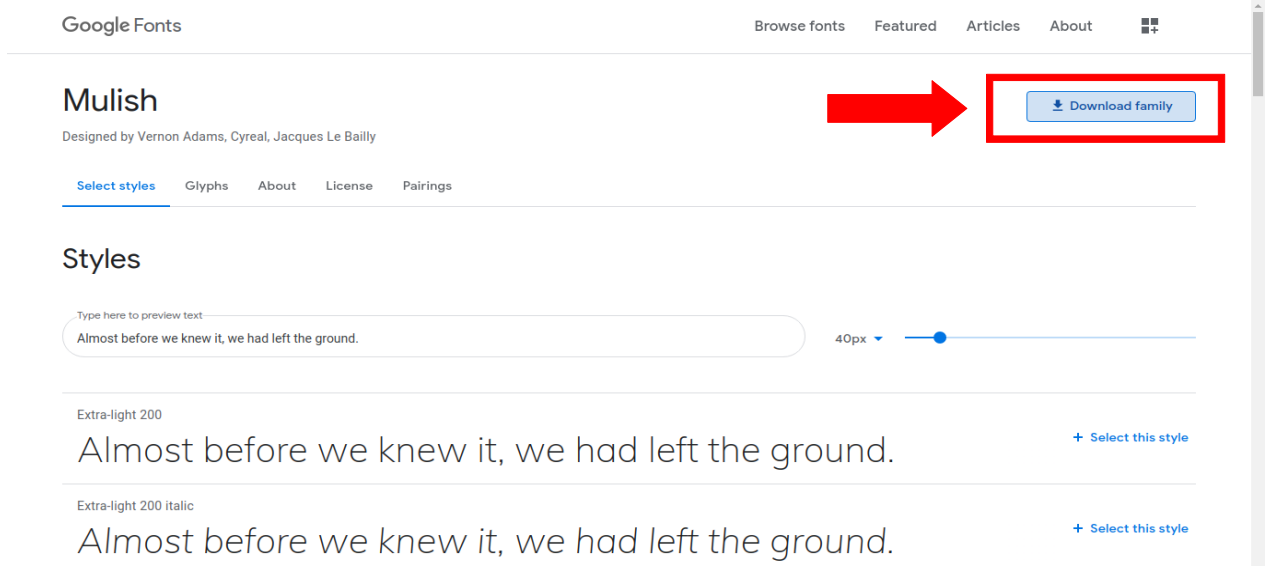
```
{
  "compilerOptions": {
    "baseUrl": "./src",
    "paths": {
      "@app/*": ["app/*"],
      "@assets/*": ["assets/*"],
      "@components/*": ["components/*"],
      "@screens/*": ["screens/*"],
      "@redux/*": ["redux/*"],
      "@styles/*": ["styles/*"]
    }
  }
}
```

### 2.2.1 Proje İçerisinden Örnek Kullanım

```
import { updateUser, deleteUser } from '@redux/actions/auth';
import $ from '@styles/settings';
import Button from '@components/form/Button';
```

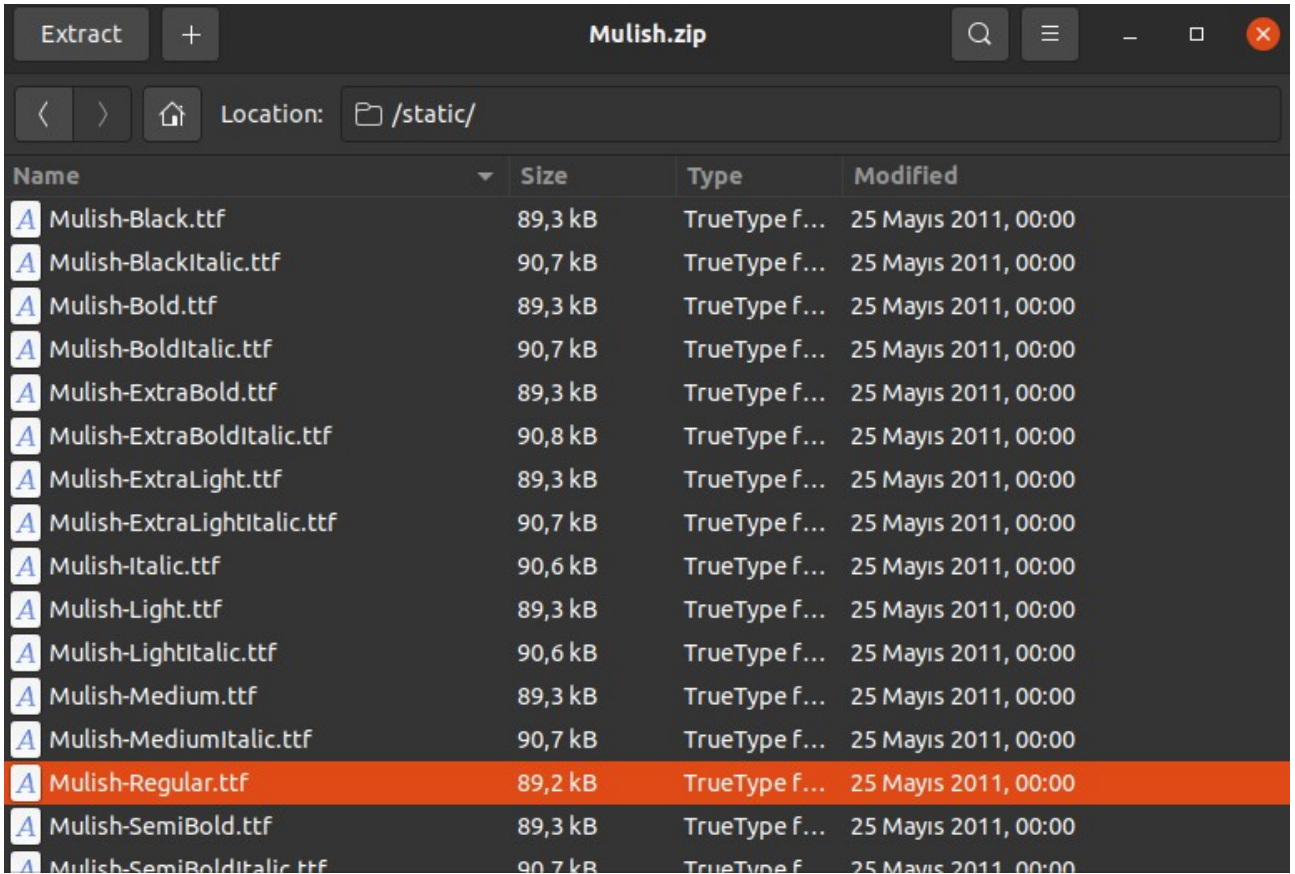
## 2.3 Özel Font Kullanımı

Geliştirilen uygulamanın tasarım için önemli olan bir diğer konu ise kullanılan yazı tipidir. Bu proje için bazı alanlarda kullanılmak üzere eski adı "Muli" olan ve adı "Mulish" olarak değiştirilen font kullanılmıştır. React Native'de font kullanabilmek için ilgili fontu projeye dahil etmek gerekir. Bu işlem için fonts.google.com üzerinden ilgili font indirilmiştir.

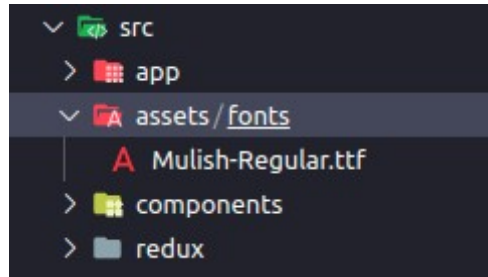


İndirilen dosya bir ".zip" dosyası olarak geliyor. "zip" dosyasından projede kullanılan "Mulish-Regular.ttf" dosyası dışa aktarılmıştır.





İlgili dosya "src/assets/fonts" klasörüne atılmıştır.



Bu aşamadan sonra özel font kullanmak istediğimizi React Native'e bildirmemiz gerekiyor bunun için projenin kök dizinine "react-native.config.js" adında bir dosya oluşturduk. İçerisine "fonts" klasörümüzün yolunu belirttik.



Bu işlemden sonra yapılan değişikliklerin geçerli olabilmesi için React Native'in link özelliğini kullandık.

```
abdulbaki@Abdulbaki:~/Desktop/final_rtca$ npx react-native link
```

Ve uygulamayı tekrar ayağa kaldırıp fontumuzu kullanmaya başladık.



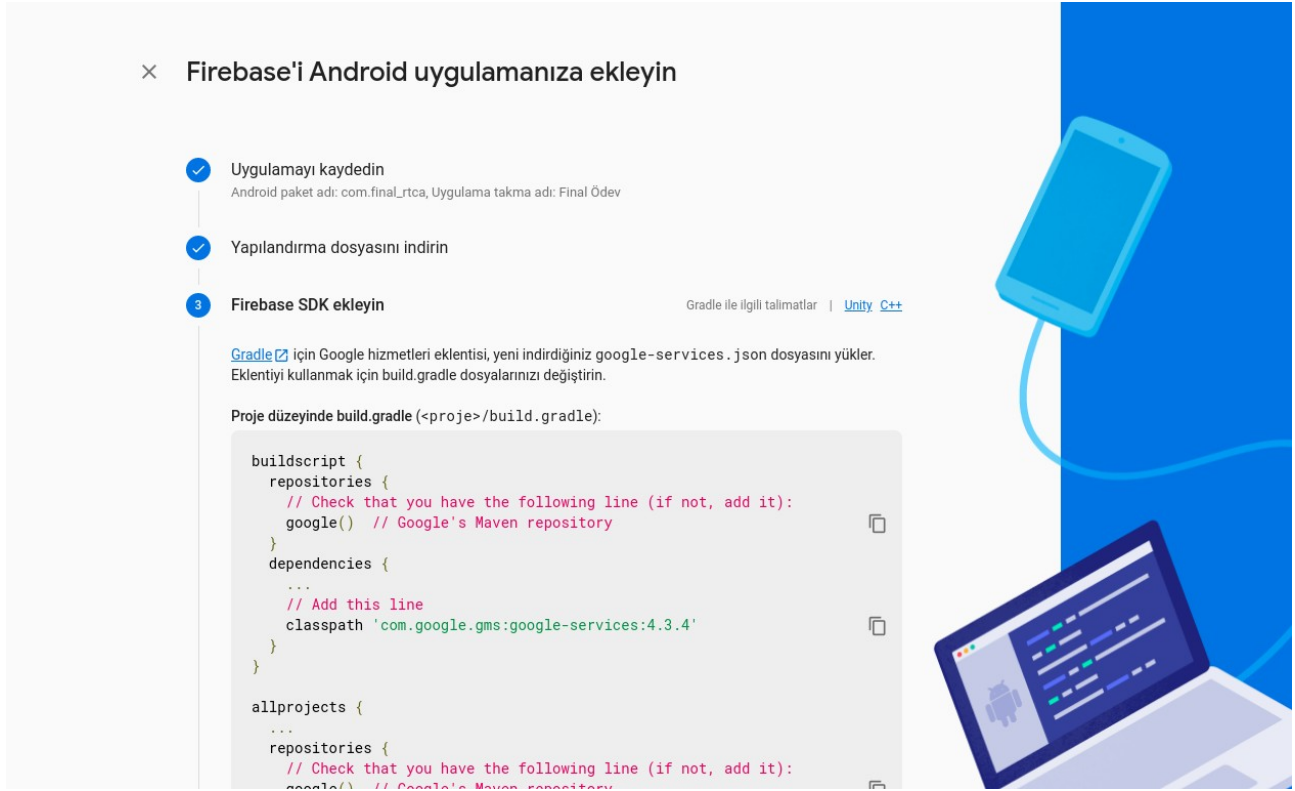
Kullanıcı Adı

## 2.4 Firebase

Projede kullanılan veritabanı google'ın geliştirdiği "Firebase" veritabanıdır. Firebase bizlere bir çok avantaj sağlamakta bunlardan en öne çıkan bazı özellikler bu projede kullanılmıştır. Firebase Auth ve Firebase Real Time özellikleri ile kullanıcıları sisteme kayıt edip gerçek zamanlı olarak veri akışını ve veri kayıt işlemi gibi özellikler projenin geliştirilmesine çok büyük kolaylıklar sağlamakta.

### 2.4.1 Firebase Entegrasyonu

Firebase'in projede kullanmak için firebase'in yönergeleri takip edilmiştir. Firebase'in yapılandırma dosyasını indirilip projeye eklenmiştir. Ardından Firebase SDK "build.gradle" dosyasına bağımlılık olarak eklenmiştir.



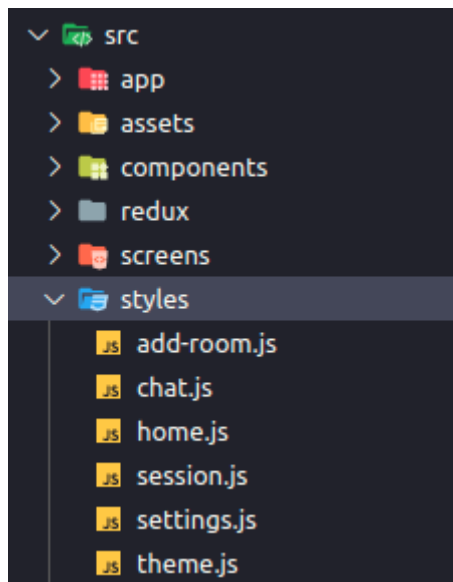
### 2.4.2 Firebase Projede Kullanımı

Firebase'i projede kullanılırken redux actions işlemlerinde kullanılmıştır. Redux konusuna birazdan değinilecektir. Firebase'i React projesinde kullanmak için ilgili npm paketleri kurulmuştur. Bu paketleri kurmak için kullanılan komutlar aşağıda yer almaktadır.

```
abdulbaki@Abdulbaki:~$ yarn add @react-native-firebase/app @react-native-firebase/auth @react-native-firebase/database
```

## 2.5 Stillerin Kullanımı

Tasarım konusuna giren stiller React için css yazılarak geliştirilir. React Native için biraz farklılıklar olsada genel olarak css kodları yazılıyor diyebiliriz. Söz Dizimi(Syntax) olarak yine css'den biraz farklı bir yapı kurulmaktadır. React Native'de stiller ilgili component'e "style" property tanımlanarak kullanılır. Bir çok stil tanımlamasında bu kullanım karışıklığı yol açabilmektedir. Bu sebeple stiller ayrı bir klasör altında ilgili bölümler için stil dosyaları yani JavaScript dosyalarında stil tanımlamaları yapılmıştır.

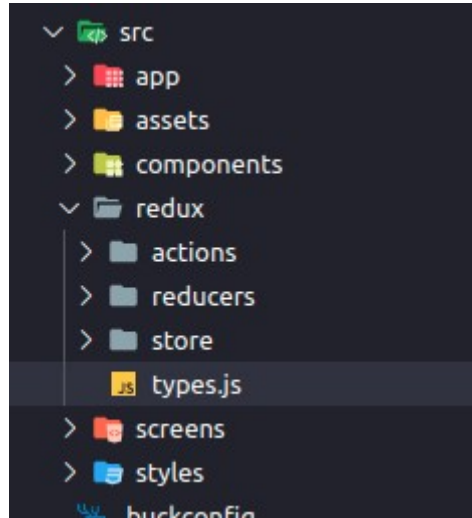


```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  headerToggleWrapper: {
    position: 'absolute',
    top: 0,
    backgroundColor: '#ffffff',
    minWidth: 150,
    borderWidth: 1,
    borderColor: '#d8d8d8',
    borderBottomLeftRadius: 5,
    borderBottomRightRadius: 5,
    right: 2,
    elevation: 1000,
    flex: 1,
    zIndex: 1
  },
  headerToggleItem: {
    padding: 5,
    height: 50,
    flex: 1,
    flexDirection: 'row',
    alignItems: 'center'
  },
  headerToggleItemText: {
    marginLeft: 5,
    fontSize: 17
  }
});
```

## 2.6 Redux

Redux state yönetimini kolaylaştıran bir kütüphanedir. React Context API adında state yönetimi için kendi geliştirmiş olduğu sistem yerine bu projede "Redux" kullanımı tercih edilmiştir.



```
// Settings
export const SETTINGS_TOGGLE_HEADER_MENU = 'SETTINGS_TOGGLE_HEADER_MENU';
export const SETTINGS_GET_WEATHER = 'SETTINGS_GET_WEATHER';

// Auth
export const AUTH_SIGN_IN = 'AUTH_SIGN_IN';
export const AUTH_SIGN_OUT = 'AUTH_SIGN_OUT';
export const AUTH_ERROR = 'AUTH_ERROR';
export const AUTH_SIGN_UP = 'AUTH_SIGN_UP';
export const AUTH_UPDATE = 'AUTH_UPDATE';
export const AUTH_DELETE_USER = 'AUTH_DELETE_USER';

// Chat
export const CHAT_ADD_ROOM = 'CHAT_ADD_ROOM';
export const CHAT_GET_ROOMS = 'CHAT_GET_ROOMS';
export const CHAT_GET_ROOM = 'CHAT_GET_ROOM';
export const CHAT_SET_SELECTED_ROOM = 'CHAT_SET_SELECTED_ROOM';
export const CHAT_SEND_MESSAGE = 'CHAT_SEND_MESSAGE';
export const CHAT_DISCONNECT = 'CHAT_DISCONNECT';
```

actions bölümünde ilgili yapılandırmalar yapıp isteğe bağlı olup reducers bölümüne işlenmek üzere veriler yollanır ve güncellenir. Örneğin yukarıda anlatılan Firebase işlemleri actions bölümünde yapılmaktadır. Aşağıda "chat" için geliştirilen actions ve reducers dosyalarının içeriği bulunmaktadır.

```
import {
  CHAT_GET_ROOMS,
  CHAT_GET_ROOM,
  CHAT_SET_SELECTED_ROOM,
  CHAT_DISCONNECT
} from '@redux/types';
import database from '@react-native-firebase/database';
import auth from '@react-native-firebase/auth';

const ref = database().ref('rooms');

export const addRoom = (name, capacity) => () => {
  ref.push().set({
    name,
    capacity,
    userCount: 0
  });
}

export const getRooms = () => dispatch => {
  ref.on('value', rooms => {
    if (rooms.exists()) {
      dispatch({
        type: CHAT_GET_ROOMS,
        payload: rooms.val()
      });
    }
  });
}

export const getRoom = id => dispatch => {
  database()
    .ref(`rooms/${id}`)
    .on('value', room => {
      dispatch({
        type: CHAT_GET_ROOM,
        payload: room.val()
      });
    });
}

export const setSelectedRoom = (id, userCount) => dispatch => {
  database()
    .ref(`rooms/${id}`)
    .update({
      userCount: userCount + 1
    })
    .then(() => dispatch({
      type: CHAT_SET_SELECTED_ROOM,
      payload: id
    }));
}

export const sendMessage = (roomId, message) => {
  database()
    .ref(`rooms/${roomId}/messages`)
    .push()
    .set({
      username: auth().currentUser.displayName,
      message,
      date: new Date().toString()
    });
}

export const disconnect = (roomId, userCount) => dispatch => {
  database()
    .ref(`rooms/${roomId}`)
    .update({
      userCount: userCount - 1
    })
    .then(() => setTimeout(() => {
      dispatch({ type: CHAT_DISCONNECT });
    }, 500))
    .catch(e => console.log('Error!', e.message));
}
```

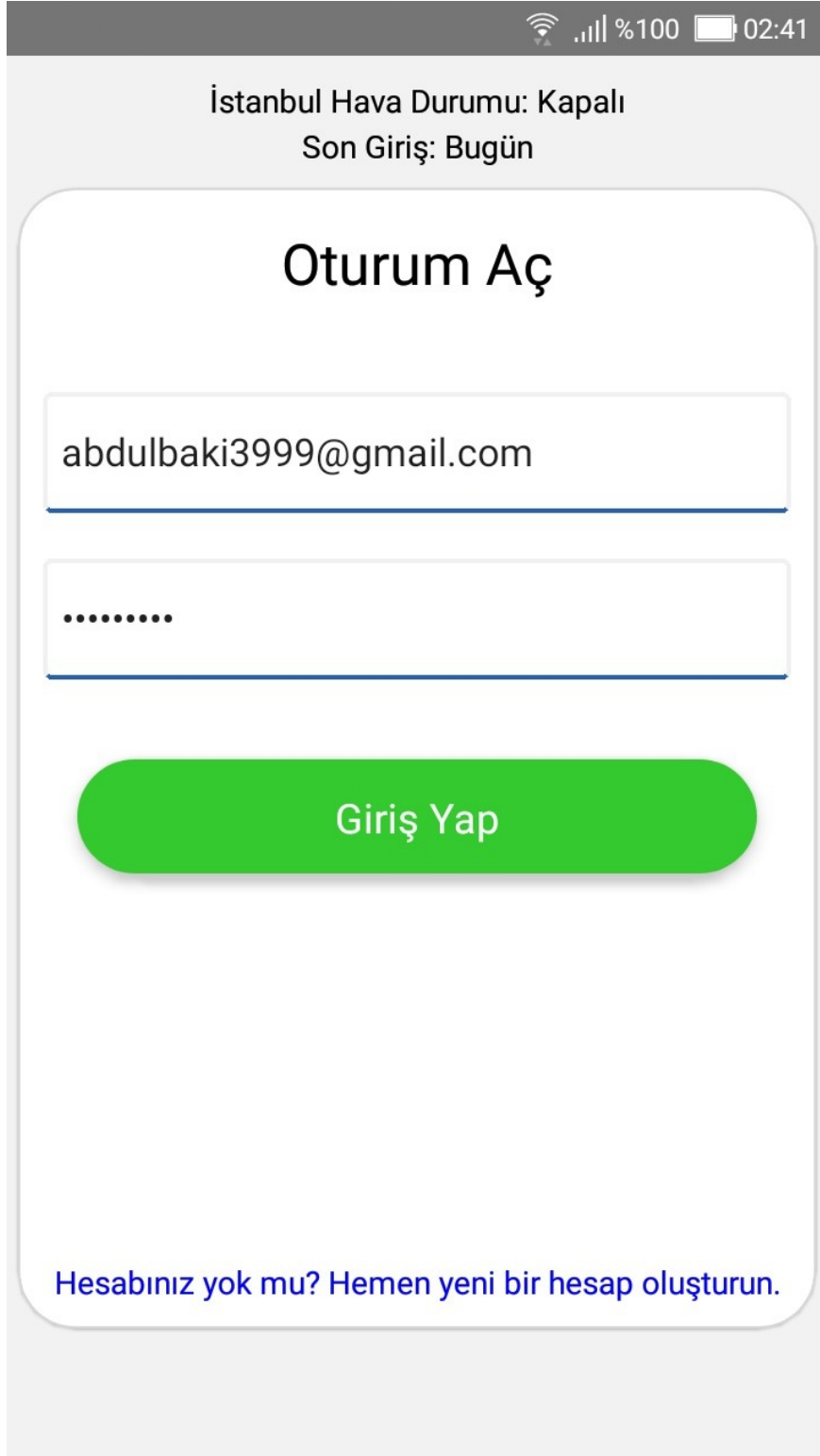
```
import {
  CHAT_GET_ROOMS,
  CHAT_GET_ROOM,
  CHAT_SET_SELECTED_ROOM,
  CHAT_DISCONNECT
} from '@redux/types';

const initialState = {
  rooms: {},
  room: {},
  selectedRoom: null
};






export default function chat(state = initialState, action) {
  switch (action.type) {
    case CHAT_GET_ROOMS:
      return {
        ...state,
        rooms: action.payload
      }
    case CHAT_SET_SELECTED_ROOM:
      return {
        ...state,
        selectedRoom: action.payload
      }
    case CHAT_GET_ROOM:
      return {
        ...state,
        room: { ...action.payload }
      }
    case CHAT_DISCONNECT:
      return {
        ...state,
        room: {},
        selectedRoom: null
      }
    default:
      return state;
  }
}
```

## 2.7 Oturum Ekranı

Uygulama ilk açıldığında oturum sayfası gelir. Oturum Sayfası (Session.js) içerisinde state kullanılarak istenilirse yeni bir hesap oluşturulur. İstenilirse de var olan bir hesaba bağlanılarak oturum açılma işlemi yapılır. Bu işlemler Firebase Auth sistemi kullanılarak yapılmaktadır.



The screenshot displays a mobile application's login screen. At the top, a status bar shows a Wi-Fi icon, signal strength, 100% battery, and the time 02:41. Below this, a header section displays 'İstanbul Hava Durumu: Kapalı' (Istanbul Weather: Closed) and 'Son Giriş: Bugün' (Last Login: Today). The main content area is titled 'Oturum Aç' (Login). It features two input fields: the first contains the email 'abdulbaki3999@gmail.com', and the second is a password field represented by dots. A prominent green button labeled 'Giriş Yap' (Login) is positioned below the password field. At the bottom, a link in blue text reads 'Hesabınız yok mu? Hemen yeni bir hesap oluşturun.' (Don't you have an account? Create a new account now.).

    %100  02:41

İstanbul Hava Durumu: Kapalı  
Son Giriş: Bugün

## Hesap Oluştur

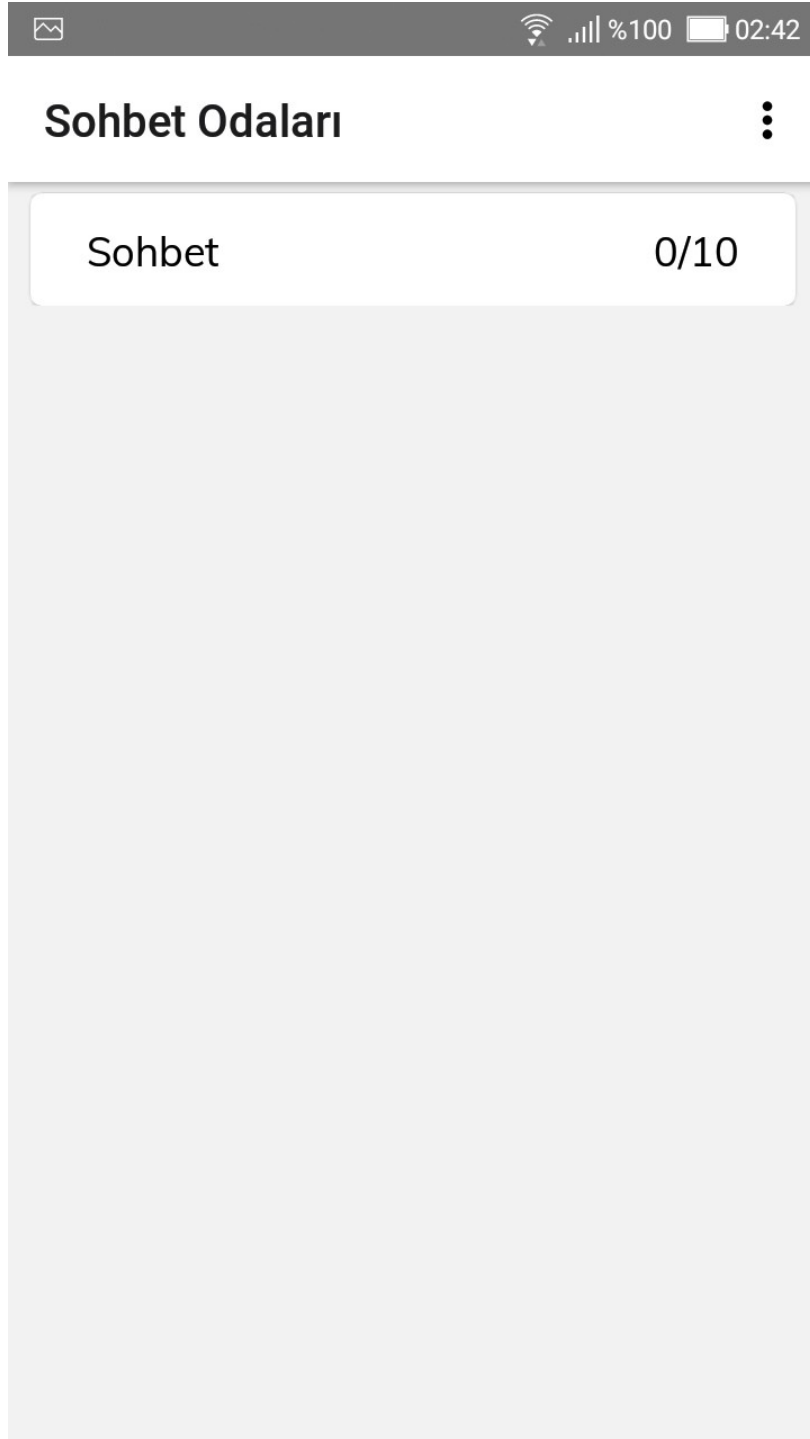
Hesap Oluştur

[Zaten bir hesabınız var mı? Oturum Açın.](#)




## 2.8 Ana Ekran


Oturum açıldıktan sonra gelecek olan ekrandır. Bu ekranda sohbet odaları listelenmektedir. Bir odaya bağlanılmak istenildiğinde odanın üzerine basılır. Eğer oda kapasitesi müsait ise o zaman odaya bağlanılır. Müsait değil ise alert ile mesaj verilir.



## 2.9 Ayarlar Ekranı

Bu bölümde kullanıcı profil bilgilerini düzenleyebilir ve hesabını silebilir.

📶 100% 02:42

 **Ayarlar**

Kullanıcı Adı

abdulbaki

Kullanıcı Adı


abdulbaki3999@gmail.com


Kaydet

Hesabımı Sil

## 2.10 Sohbet Odası Ekleme Ekranı

Kullanıcılar isteğe bağlı olarak yeni sohbet odaları oluşturabilirler. Bu yeni oda ekleme ekranında oda adı ve oda kapasitesi belirleyip her kullanıcı yeni bir oda ekleyebilir.



 **Yeni Oda Oluştur**

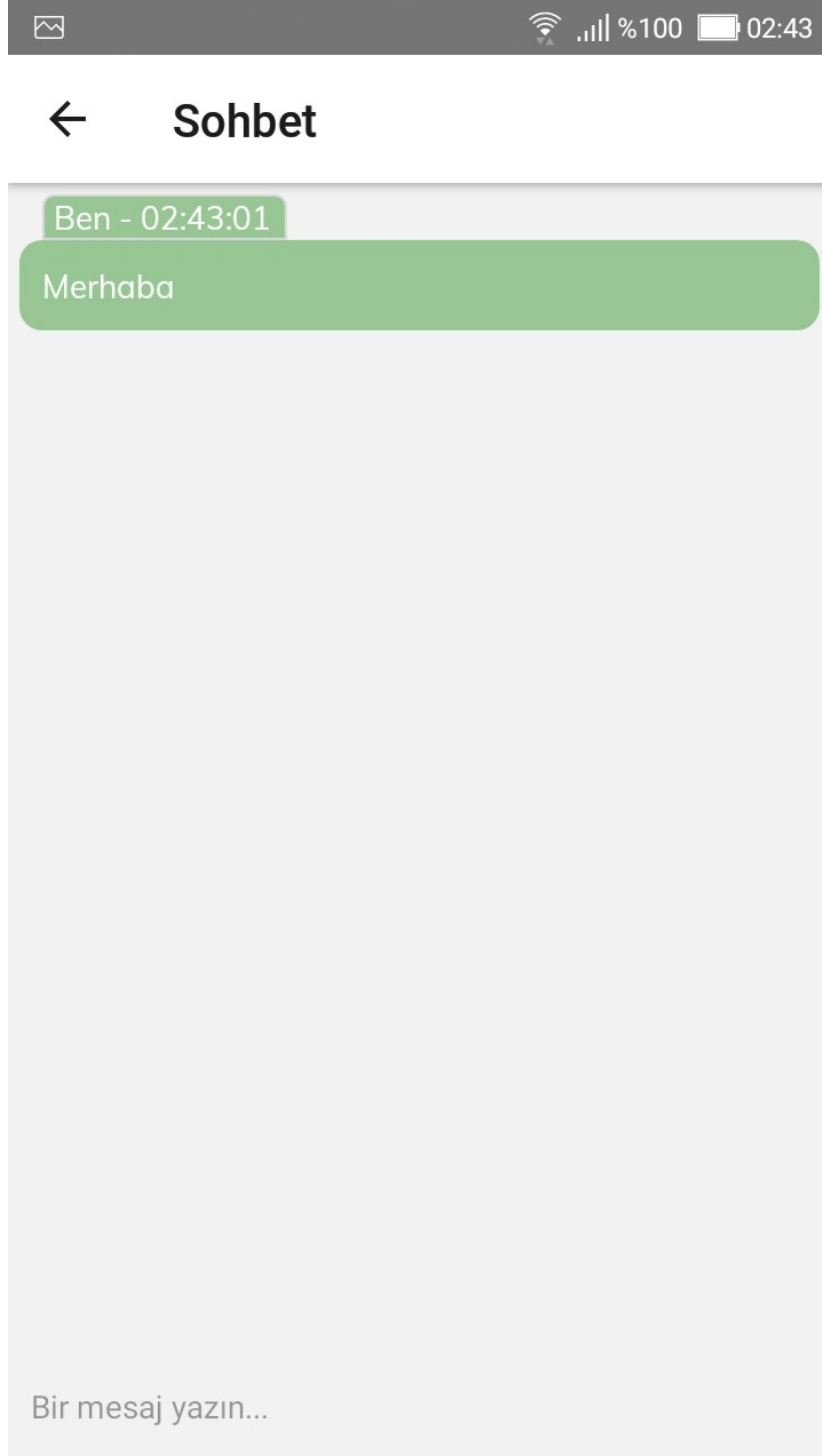
Oda Adı

Kapasite

Oluştur

## 2.11 Sohbet Ekranı

Bu alanda mesajların listelendiği ve mesaj göndermek için iki alan bulunmaktadır. Kullanıcılardan herhangi biri odaya eğer kapasite dolmamışsa bağlanabilir ve mesaj gönderebilirler. Gönderilen mesajlar anlık görüntülenirler. Bu özellik Firebase Real Time ile sağlanmıştır. Eğer Firebase kullanmayıp bu yapıyı kendimiz kurmak isteseydik socket yapısı geliştirerek yine aynı şekilde bir yapı elde edebilirdik. Firebase ile bu şekilde uğraşmak yerine kolay ve hızlı bir şekilde bu yapıyı kurabildik.



### 3 Uygulamanın Test Edilmesi

Uygulamanın geliştirilmesi bittikten sonra test etmek için iki fiziksel cihaz kullanılmıştır. Bu iki cihazda uygulama ayağa kaldırılarak test edilmiştir. Test etme işlemi için bir video hazırlanmıştır. Hem projeye .mp4 olarak eklenmiştir. Hemde Google Driver üzerinden yayımlanmıştır. İlgili bağlantı aşağıdadır.

Test Video Bağlantısı

<https://drive.google.com/file/d/1-KYbNArd81U2QaxuIUUcxPoPv0cs0A3/view?usp=sharing>

**Not: Uygulama yalnızca Android cihazlarında test edilmiştir.**

## 4 Sonu

Uygulamanın geliřtirilmesi tamamlanmıřtır. "Gerek Zamanlı Sohbet Uygulaması" temel anlamda istenileni karřılamaktadır. Gerek hayatta kullanılabilir. Kullanıma sunulmak istenirse yani dağıtım yapılmak istenirse iyileřtirmeler yapılmalıdır. Örneğın güvenlik için Firebase kuralları tanımlanmalıdır.

Bu rapor dosyasında temel anlamda uygulamanın geliřtirilme süreci ve kullanılan teknolojilere değinilmiřtir.