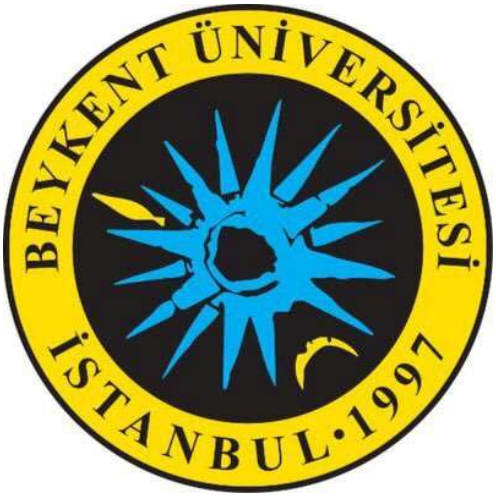

Beykent Üniversitesi

Yazılım Mühendisliği

Yazılım Mühendisliği Bitirme Projesi

2020 – BAHAR

İkinci Raporu

Benimki Nerede? (BN)

Abdurrahman Aydın - 160313020

Ahmet Burak Hapaz - 160313067

Muhammed Burak Sıvık - 160313003

Uğurcan Çakar - 160313024

Utku Erdemir - 160313037

İçindekiler

Yazılım Mühendisliği Bitirme Projesi Grup10 2. Raporu

1 Benimki Nerede?

1 a. Projenin Açıklaması

Projenin temel amacı kişiler için önemli olan varlıkların kaybolmasını önlemek, kaybolan varlıkların bulunmasına yardımcı olmak ve günümüz koşulları nedeniyle depremde enkaz altında kalan insanların bulunmasına yardım etmektedir. Projenin asıl amaçlarının haricinde bir artısı kullanıcı dostu olmaktır. Çünkü proje içerisindeki ara yüzler, teknolojilerin etkileşime geçmesi ve alış veriş sistemi mümkün olduğunca en basit düzeye indirgenmiştir. Bu işlemler yapılırken Beacon cihazlardan yararlanılmaktadır, mobil uygulamanın geliştirilmesinde tek kod platformu olarak React-Native kullanılmaktadır, web ön uç birimi için ReactJS kullanılmaktadır. Arka uç birimde son dönemlerde yüksek performansından ötürü GoLang ve veri yoğunluğunu kaldırabilmesi için MongoDB kullanılmaktadır. Tablo – 1 üzerinde ekip ve görev dağılımları belirtilmiştir.

1 b. Proje Ekibi

Tablo – 1: Proje Ekibi

	Numarası	Adı Soyadı	Projedeki Görev Tanımı
1	160313020	Abdurrahman Aydın	• Bütün arka uç biriminin geliştirmektir. • Web ön uç biriminin geliştirilmesine yardım etmektir. • Veri tabanının oluşturulmasına yardım etmektir.
2	160313067	Ahmet Burak Hapaz	• Veri tabanında ki ilişkileri belirlemek ve yapıyı oluşturmaktır. • Web ön uç biriminin geliştirilmesine yardım etmektir.
3	160313003	Muhammed Burak Sıvık	• Web ön uç birimini geliştirmektir. • Bütün kullanıcı ara yüzlerini ve grafiksel gereksinimleri tasarlamaktır.
4	160313024	Uğurcan Çakar	• Mobil ön uç birimini geliştirmektir. • Ön uç birimler için durum yönetimini geliştirmektir.
5	160313037	Utku Erdemir	• Proje ekibinin koordinasyonunu ve proje için gerekli kaynakların bulunmasını sağlamaktır. • Web ön uç biriminin geliştirilmesinde karşılaşılan problemlere çözüm bulmaktır. • Arka uç birimin geliştirilmesinde karşılaşılan problemlere çözüm bulmaktır. • Mobil ön uç biriminin genel yapısını oluşturmak ve gerekli platformların kurmaktır. • Veri tabanının oluşturulmasına yardım etmektir.

1 c. Projenin Aşamaları

Proje içerisinde yaklaşık on beş tane aşama ön görülmüştür ve bu aşamalardan yaklaşık dört tanesi geçtiğimiz dönem, dört tanesini birinci rapor sırasında, bu rapor içerisinde ise üç tanesi tamamlanmıştır. Toplam on birinci iş paketine kadar gelinmiştir. Tablo - 2 üzerinde iş paketleri belirtilmiştir.

Tablo – 2: İş Paketleri

No	İş Paketi Adı	Dönem	İçeriği
1	Planlama	12.10.2019 - 08.11.2019	Projenin hatlarının belirlendiği aşamadır.
2	Analiz	08.11.2019 - 27.11.2019	Kullanım, durum, gereksinimler ve kısıtların belirlendiği aşamadır.
3	Tasarım	27.11.2019 - 27.12.2019	Projenin veri tabanının, ara yüzlerinin ve akış senaryolarının belirlendiği aşamadır.
4	Test Planları Oluşturulması	27.11.2019 - 22.01.2020	Projenin bitimindeki test senaryolarının oluşturulduğu aşamadır.
5	Seçilmiş Platform Testlerinin Yapılması	11.02.2020 - 25.02.2020	Projenin gereksinimlerine uygun belirlenmiş platformların kullanım testlerinin gerçekleştirildiği aşamadır.
6	Veri Tabanının Oluşturulması	18.02.2020 - 29.02.2020	Proje için verilerin barındırılacak olduğu veri tabanının temel yapısının oluşturulmasını içeren aşamadır.
7	Arka Uç Birimin Geliştirilmesi	19.02.2020 - 30.03.2020	Kullanıcının ön uçta yaptığı işlemleri veri tabanına belirli kontroller ile kaydedecek olan birimin geliştirildiği aşamadır.
8	Mobil Ön Uç Biriminin Tasarlanması	20.02.2020 - 15.03.2020	Projenin asıl kısmı olan Beacon tarama kısmını oluşturan mobil uygulamanın daha önceden belirlenmiş ara yüzler referans alınarak tasarlandığı kısımdır.
9	Web Ön Uç Biriminin Tasarlanması	25.02.2020 - 17.03.2020	Kullanıcıların ve yöneticilerin kullanacağı web panelinin oluşturulduğu sayfadır.
10	Yayına Alma (BE)	25.03.2020 – 04.04.2020	Arka uç birimin yayına alınma aşamasıdır
11	Arka Uç Birimi İle Ön Uç Birimlerinin Bağlanması	30.03.2020 - 04.05.2020	Tasarımlar bittikten sonra daha önceden hazırlanmış olan arka uç birimiyle ön uç birimin bağlanmasını içeren aşamadır.
12	Test Planlarının Gerçekleştirilmesi	04.05.2020 - 12.05.2020	Belirlenmiş olan test planlarının uygulandığı aşamalardır.
13	Karşılaşılan Sorunların Çözülmesi	12.05.2020 - 20.05.2020	Test planları gerçekleştirildikten sonra hata ile karşılaşılan yerlerin çözüme kavuşturulması aşamasıdır.
14	İyileştirme Çalışmaları	20.05.2020 - 27.05.2020	Proje içerisinde kullanıcı deneyimi, performans/kaynak kullanımı açısından iyileştirme çalışmalarını içeren aşamadır.
15	Yayına Alınma (Mobil)	27.05.2020 - 30.05.2020	Projenin mobil marketlerden yayınlanma aşamasıdır.

2 Dönem İçinde Yapılan İşler

2 a. Sunucu Kurulumu ve Arka Uç Birimin Yayına Alınması (Utku Erdemir)

Bu kısım normal planlanma sürecinde en son aşama olarak yer almakta olan aşamadır. Lakin ekip üyelerinin bilgisayarlarının yetersizliğinden ve bunun yanında ekip üyelerinin yerel olarak kurma, sunucuyu açma ve kapama işlemleriyle uğraşması istenmemesinden ötürü yayına alma aşamasının arka uç birim ile alakalı olan kısmı öne alınmıştır. Bu süreç içerisinde GoLang'in gereksinimlerinden ötürü geleneksel konaklar kullanılamamıştır. Özel olarak bir sunucu kurulumu gerçekleştirilmiştir ve bu kurulumda Ubuntu işletim sistemi kullanılmıştır. Ardından bir sanallaştırma teknolojisi olan Docker kurulmuştur. Çünkü GoLang'in sunucuda çalışabilmesi için buna ihtiyacı vardır. Sonrasında ise GoLang'in gerekli paketlerinin ve MongoDB'nin kurulumu gerçekleştirilmiştir. Sunucuda gerekli ortam sağlanmasının ardından yazılmış olan arka uç birimi ve oluşturulmuş olan veri tabanı sunucu içine aktarılmıştır. Ardından sunucu başlatılmış ve gerekli olan testler yapılmıştır. Son olarak sunucunun işlemlerinin sürekli çalışabilmesi adına sunucuya Screen kurulmuştur. Screen bir sunucu üzerine SSH ile bağlandığınızda arkada işlem yapabilmenize olanak veren eklentidir. Screen kullanılarak sunucu sürekli açık tutulması sağlanmıştır.

2 b. Veri Tabanının Düzenlenmesi (Ahmet Burak Hapaz, Utku Erdemir, Abdurrahman Aydın)

Bu bölümde veritabanı düzenlenmesi yapılmıştır. Önceki dokümandan da görülebileceği üzere belirli başlı Id değerleri dizgi şeklinde bırakılmıştır. Bunun temel sebebi ekibin MongoDB'ye başladığında olan bilgisizliğinden kaynaklanmaktadır. Ekip, ilk raporu göndermesinin ardından MongoDB ile alakalı daha çok araştırma yapmıştır ve ObjectId'nin de el ile atanabileceğini fark etmiştir. ObjectId el ile atanabilince, olağan dışı durumlar için tutulan ilişki Id'lerinin dizgiden ObjectId'ye çevrilmiştir. Sadece bir kısımda sayısal bir Id değeri tutulmaktadır. Onun temel sebebi ise, o sayısal değerin yeri geldiğinde el ile girilecek olmasıdır. Bunun yanı sıra Beacons koleksiyonunda yapısal olarak hatalar bulunmuştur. Sonrasında ise bu hatalar giderilmiştir. Önceki mimaride Beacons koleksiyonu dizi olarak tanımlanmıştır. Lakin büyük bir hata olduğu fark edildiğinden dolayı bu hata giderilmiştir. Yeni yapısı, Kod Örneği-1'de yeni veri tabanı tasarımı gösterilmiştir.

```
{
  "infos": {
    "beacon_name": "asdasd",
    "uuid": "sadassasd",
    "id": {
      "$numberInt": "12"
    },
    "major": {
      "$numberInt": "31456"
    },
    "minor": {
      "$numberInt": "34564561"
    },
    "last_seen": {
      "$date": {
        "$numberLong": "1001883600000"
      }
    },
    "beacon_type": {
      "$numberLong": "0"
    }
  },
  "user": {
    "username": "",
    "user_id": {
      "$oid": "5e5180347854a42f3841ecb6"
    },
    "user_mail": "",
    "user_phone": ""
  }
}
```


Kod Örneği – 1: Veri Tabanının Oluşturulması / beacons.json

2 c. Arka Uç Geliştirme (Abdurrahman Aydın, Utku Erdemir)

Bu süreç içerisinde eski yazılmış kodların üstüne bir çok servis eklenmiştir. Bunun yanı sıra bir SOLID ilkesi olan Liskov prensibinin tamamen uyarlanması gerçekleştirilmiştir. Bu uyarlama gerçekleştirilir iken, sürekli kod tekrarından kaçınılması için, bir yapı oluşturulmuştur. Eski kodlar üzerinde yapısal değişiklikler yapılmıştır. Buna ek olarak Kullanıcı Kayıt, Kayıp İlanı Oluşturma, Ürün Listesi , Cihaz Ekleme ve Regex işlemleri gerçekleştirilmiştir. Bu bağlamda kullanıcıdan alınan kredi kartı bilgilerini kontrol etmek için, harici bir kütüphane kullanılmıştır. Daha sonra projede oluşan veri tabanı ve struct uyumsuzluklarını gidermek için structlar tekrardan tanımlanmıştır. Yapılan işlemlerle alakalı örnek kod, Kod Örneği-2'de belirtilmiştir. Bu örnektede görüldüğü gibi Liskov prensiblerini tamamen uyulmuş ve anlamsız hataların önüne geçilmiştir.

```
package main
import (
	"encoding/json"
	"net/http"
	"github.com/globalsign/mgo/bson"
)
func loginPage(w http.ResponseWriter, r *http.Request) {
	if r.Method == "POST" {
		if r.FormValue("email") == "" {
			writeResponse(w, requiredInputError("E-posta"))
		} else if r.FormValue("password") == "" {
			writeResponse(w, requiredInputError("Şifre"))
		} else {
			var user, control = findUser(r.FormValue("email"),
r.FormValue("password"))
			if control == "Login" {
				writeResponse(w, string(user))
			} else if control == "Notfound" {
				writeResponse(w, notFindRecordError())
			} else if control == "Lvl" {
				writeResponse(w, invalidLoginRequest())
			} else if control == "Parse" {
				writeResponse(w, incorrectInput("Parse"))
			} else if control == "Mail" {
				writeResponse(w, incorrectInput("Mail"))
			} else {
				writeResponse(w, someThingWentWrong())
			}
		}
	} else {
		writeResponse(w, notValidRequestError(r.Method))
	}
}
func findUser(userMail string, userPassword string) ([]byte, string) {
	person := &Person{}
```

```
	var data []byte
	controlMail := checkEmailValidity(userMail)
	if controlMail != true {
		return data, "Mail"
	}
	err := connection.Collection("users").FindOne(bson.M{"user_info.user_mail": userMail,
"user_info.user_password": userPassword}, person)
	if err != nil {
		return data, "Notfound"
	}
	lvl := person.UserInfos.RoleLvl
	if lvl == 0 {
		return data, "Lvl"
	}
	person.UserInfos.UserToken = tokenGenerator()
	connection.Collection("users").Save(person)
	user := &Userjon{person.UserInfos.UserToken}
	data, err = json.Marshal(user)
	if err != nil {
		return data, "Parse"
	}
	return addError(data), "Login"
}
```

Kod Örneği – 2: Arka Uç / Login.go

2 d. Mobil Uygulamanın Geliştirilmesi(Uğurcan Çakar, Utku Erdemir)

Bu bölümde mobil uygulamanın geliştirilmesine devam edilmiştir. Önceki dokümanda mobil uygulamanın başlanan ön yüzü bitirilmiştir, gerekli revizeler yapılmıştır ve sayfaların özelleştirilemediği noktalarda, sayfalarda kullanılan kütüphanelerin kodlarına müdahale edilmiştir. Sonra da gerekli sayfalarda doğrulama işlemleri yapılmıştır. Ardından arka uç birimi ile bağlanılması, durum yönetimi mekanizmasının geliştirilmesi ve yönlendirme yapısının oluşturulması işlemleri gerçekleştirilmiştir. Önceden geliştirilmiş olan sol menü yönlendirme mekanizması ile uyumsuz olmasından dolayı revize edilmiştir ve uyumlu hale getirilmiştir. Ardından bu kısımda React ailesi için saf Javascript kodlarından oluşan bir bağlantı kütüphanesi yazılmıştır bu kütüphane sayesinde arka uç ile bağlantı oluşturulurken kullanılan yönteme bağlılık kaldırılmış. Bu sayede her yönteme aynı girdilerin verilmesi sağlanmıştır. Kullanıcı anahtarını tutmak için Redux-Persist yapısından faydalanılmıştır. Redux-Persist yapısı temelinde “async-storage” kullanıyor olmasına rağmen Redux ile olan bağlılığı maaliyeti önemli ölçüde azaltmıştır. Yapılan işlemlerle alakalı örnek kod, Kod Örneği-3’de belirtilmiştir.

```
import React, {Component} from 'react';
import {StyleSheet, View, Image} from 'react-native';
import {Button, List, Text, Layout, Spinner} from '@ui-kitten/components';
import {ArrowRightIcon} from './extra/icons';
import { connect } from "react-redux";
import { bindActionCreators } from "redux";
import * as BeaconListActions from "../../redux/actions/beaconListActions";
import * as ProfileActions from "../../redux/actions/profileActions";
import { Actions } from 'react-native-router-flux';
class Device extends Component {
 constructor(props) {
  super(props);
  this.state={
   spinner: false
  }
 }
 onItemPress = (ID) => {
  Actions.DeviceDetail({ ID: ID })
 };
 renderLoading = () => (
  <View style={styles.loading}>
   <Spinner/>
  </View>
 );
 componentDidMount = () =>{
  this.props.actions.getProfile("profile",["token"],[this.props.login])
 }
 componentDidUpdate = () => {
  if(this.props.profile !="" && this.state.spinner==false){
   this.getBeaconList()
  }
 }
 getBeaconList()
 {
  this.props.actions.getBeacons("devices",["userId"],[this.props.profile.user_id]);
  this.setState({
   spinner:true
  })
 }
 renderItem = info => (
  <View style={styles.item}>
   <Layout style={styles.itemImage}>
    <View style={styles.layout}>
     <View style={styles.left}>
```

```
          <Text style={styles.itemTitle} category="h4" status="control">
            {info.item.beacon_name}
          </Text>
          <Text style={styles.itemDescription} category="s1" status="control">
            Tür: {info.item.type}
          </Text>
        </View>
      </View>
      <View style={styles.itemFooter}>
        <Image
          style={styles.headerImage}
          source={{
            uri:url
          }}
        />
        <View style={styles.space}></View>
        <Button
          style={styles.iconButton}
          appearance="outline"
          status="control"
          icon={ArrowRightIcon}
          onPress={() => this.onItemPress(info.item.beacon_id)}>
          {'Details'}
        </Button>
      </View>
    </Layout>
  </View>
);
render() {
  return (
    <Layout style={styles.layout}>
      {
        this.state.spinner == false ?
        this.renderLoading()
      :
        <List
        style={styles.list}
        contentContainerStyle={styles.listContent}
        data={this.props.beacons}
        renderItem={this.renderItem.bind()}
        />
      }
    </Layout>
  );
```

```
  }
}
function mapStateToProps(state) {
  return {
    beacons: state.beaconListReducer,
    profile: state.profileReducer,
    login:state.loginReducer
  };
}//reducer'dan çekilen veri props'lara işlendi
function mapDispatchToProps(dispatch) {
  return {
    actions: {
      getBeacons: bindActionCreators(BeaconListActions.getBeacons, dispatch),
      getProfile: bindActionCreators(ProfileActions.getProfile, dispatch)
    }
  };
}
export default connect(mapStateToProps, mapDispatchToProps)(Device);
```

Kod Örneği –3: Mobil Ön Uç Biriminin Tasarlanması / Device.js,

2 e. Web Ön Uç Geliştirme (Muhammed Burak Sıvık, Ahmet Burak Hapaz)

Bu süreç içerisinde Web Ön uç kısmının geliştirilmesine devam edilmiştir. Önceki dokümandan bu yana eksik olan bileşenler bitirilmiştir. Biten bileşenler baştan aşağıya düzenlenmiştir ve ayrı ayrı hepsi yapısına uygun olarak yapısallaştırılmıştır. Bileşenlerin birkaçının Action ve Reducer bağlantıları sağlanmıştır. Kullanıcının anahtarını tutmak amacıyla "Redux-Persist" yapıları kurulmuştur. Bu kısımda Next.JS'in sunucu tarafı işleme özelliğinden dolayı özel bir yapılandırılmaya gidilerek kalıcı depolama sağlanmıştır. Asenkron işlemlerden dolayı bazı kısımlarda Redux üzerine veriler geç eklenmektedir. Bu yüzden bileşenin yaşam döngüsü içerisinde yer alan "componentDidUpdate" fonksiyonu kullanılmıştır. Böylece, geç gelen verinin kullanımı sağlanmıştır. Belirli bir kısımda React'ın bileşenden bağımsız olan Hooks yapısı kullanılmıştır. Bu sayede yapıyla özel olarak Redux bütünleşmiş hale getirilmiştir. Yapılan işlemlerle alakalı örnek kod, Kod Örneği-4'de belirtilmiştir.

```
import React, { Component } from "react";
import { Button, Col, Card, Row } from "antd";
import { getConnectionLink } from "../../lib/connector";
import { connect } from "react-redux";
import * as authActions from "../../redux/actions/authActions";
import { bindActionCreators } from "redux";
import * as productActions from "../../redux/actions/productActions";
const { Meta } = Card;
class ProductOne extends Component {
  constructor(props) {
    super(props);
    this.state = {
      products: [],
      loaded: false
```

```
    };
  }
  componentDidMount() {
    if (this.props.product_data == "") {
      var paramsNames = [];
      var paramsValues = [];
      var obj = getConnectionLink("products",paramsNames,paramsValues, "POST");
      this.props.actions.ProductPage(obj);
      console.log(this.props.product_data);
      this.props.product_data;
    } else {
      this.setState({ products: this.props.product_data, loaded: true },function()
{console.log(this.state.products);});
    }
  }
  componentDidUpdate() {
    if (this.props.product_data != "" && !this.state.loaded) {
      this.setState({ products: this.props.product_data, loaded: true }, function() {
console.log(this.state.products); } );
    }
  }
  render() {
    var productlist = [];
    if (this.state.products != []) {
      this.state.products.forEach(product => {
        productlist.push(
          <div key={product.product_id}>
            <Col lg={6} md={12} style={{ margin: 5 }}>
              <Card bodyStyle={{ padding: 5 }} style={{ marginBottom: "20px" }}>
                <div float="center">
                  <Card
                    cover={
                      <img
                        alt="example"
                        src="https://www.patidogclub.com/wp-content/uploads/2017/12/yavru-kopekler-icin-
tasma-egitimi.jpg"
                      />
                    }
                    actions={[
                      <Button type="primary" block>
                        Sepete Ekle
                      </Button>
                    ]}
                  >
                    <Meta
```

```
                  style={{ textAlign: "center" }}
                  title={product.product_description}
                  description={product.product_name}
                />
                <br></br>
                <div className="price-container">
                  <h2>${product.product_price}</h2>
                </div>
              </Card>
            </div>
          </Card>
        </Col>
      </div>
    );
  });
 } else {
  productlist = "Yükleniyor.";
 }
 return (
  <div>
   <Row>{productlist}</Row>
  </div>
 );
 }
}
function mapStateToProps(state) {
 return { product_data: state.productlistReducer, currentToken: state.authReducer};
}
function mapDispatchToProps(dispatch) {
 return {
  actions: {
   productPage: bindActionCreators(productActions.productPage, dispatch),
   loginUser: bindActionCreators(authActions.loginUser, dispatch)
  }
 };
}
export default connect(mapStateToProps, mapDispatchToProps)(ProductOne);
```

Kod Örneği – 4: Web Ön Uç Geliştirme / ProductOne.js

3 Sonuç

3 a. Başarı İle Tamamlanan İşler

Bugüne kadar on birinci numaralı iş paketine kadar olan kısımların hepsi başarıyla tamamlanmıştır. On bir numaralı iş paketinde ise büyük bir yol alınmıştır. Muhtemelen tahmin süresinden daha önce bitirilecektir.

3 b. Başarı İle Tamamlanamayan İşler

Planlanan bütün işler başarıyla gerçekleşmiştir. Mobil uygulama üzerinde yönlendirme de kullanılması gereken bileşen arıza vermesinden dolayı başka bir bileşen kullanılmıştır.

3 c. Kişilerin Görevlerini Yerine Getirme Durumu

Kişilerin görevlerini yerine getirme durumu Tablo – 3 üzerinde belirtilmiştir.

Tablo – 3: Proje Ekibinin Görevlerini Yerine Getirme Durumu

	Numarası	**Adı Soyadı**	**Projedeki Görevini Yerine Getirme Durumu**
1	160313020	Abdurrahman Aydın	• Veri tabanı düzenlenmesine yardım etmiştir. (%100) • Web servislerin istenilen kısmını geliştirmiş ve eksiklikleri düzenlemiştir. (%100)
2	160313067	Ahmet Burak Hapaz	• Veri tabanının düzenlenmesine yardım etmiştir. (%100) • Web uygulaması kısmında kendine verilen görevi yerine getirmiştir. (%100)
3	160313003	Muhammed Burak Sıvık	• Kendinden istenen ikonu tasarlamıştır. (%100) • Web uygulamasının geliştirilmesinde beklenen yere kadar gelmiştir. (%100)
4	160313024	Uğurcan Çakar	• Mobil uygulamanın geliştirilmesinde kendine verilen görevi yerine getirmiştir. (%100) • Durum yönetimini düzenlemiştir. (%100)
5	160313037	Utku Erdemir	• Web ön uç biriminin geliştirilmesinde karşılaşılan problemlere çözüm bulmuştur. (%100) • Arka uç birimde düzenlemeler ve iyileştirmelere yardım etmiştir. (%100) • Mobil ön uç birimde iyileştirmeler yapmıştır. (%100) • Veri tabanının düzenlenmesine yardım etmiştir. (%100) • Sunucu kurulumunu gerçekleştirmiştir. (%100)

Not: Görevleri yerine getirme oranı günümüze kadar olan kendine verilmiş görevleri yerine getirme oranı olarak varsayılmıştır.