Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Yazılım Laboratuvarı II

ROTA PLANLAMA PROBLEMİ

Taha Uz/Mahmut Alper Yılmaz

190202013@kocaeli.edu.tr[Taha]/180202004@kocaeli.edu.tr[Alper]

ÖZET

Yazılım Laboratuvarı 2 projesi olarak bizden "Rota Planlama Problemi" adlı uygulama geliştirmemiz beklenmektedir.

Proje için React Native dilini kullanmayı tercih ettik. Visual Studio Code geliştirme ortamını kullandık.

Proje dökümanından bizden, otobüs duraklarındaki kişi sayılarını ve maaliyetlerini hesaba katarak bir rota çizdiren bir uygulama yapmamız beklenmektedir. Rotaya planlamak için kullanıcıdan bindiği durak bilgisini almak için arayüz oluşturuldu. Durağın koordinatları ve duraklarda kaç kişinin olduğu bilgisini toplamak için admin arayüzü oluşturuldu. Admin arayüzüden kullanıcı bilgileri, durak bilgileri, tur bilgileri ve harita bilgilerini ayarlamak için gerekli arayüz hazırlandı. Böylelikle kullanıcının admin tarafından eklenen duraklardan hangisine biniceğini seçerek otobüs firmasının buna göre en az maaliyetli rotayı planlayarak sefer planması yapan bir uygulama geliştirdik.

1.GİRİŞ

Proje için React Native dilini kullanmayı tercih ettik. Visual Studio Code geliştirme ortamını kullandık. Senaryo için gereken verileri depolamak için SQLite veri tabanı kullanıldı.

React Native programlama dili, Meta Platforms, Inc tarafından oluşturulan açık kaynaklı bir UI yazılım çerçevesidir. Geliştiricilerin React çerçevesini birlikte kullanmalarını sağlayarak Android, Android TV, iOS, macOS, tvOS, Web, Windows ve UWP için uygulamalar geliştirmek için kullanılır.

Visual Studio Code platformu , Microsoft tarafından Windows, Linux ve MacOs için geliştirilen bir kaynak kodu düzenleyicisidir. Hata ayıklama, gömülü Git kontrolü,

sözdizimi vurgulama, akıllı kod tamamlama, snippet'ler ve kod yeniden yapılandırma desteği içerir.

SQLite, dünyada en çok dağıtılan ve tavsiye edilen kaynak kodları halka açık, tamamen C/C++ programlama dilleriyle geliştirilmiş sunucu yazılımı ve yapılandırma gereksinimi olmayan, işlemsel ve ilişkisel bir SQL veritabanı motorudur.

2.TEMEL BİLGİLER

2.1.Genel Sistem Yapısı

Bu kısımda react native dilinde classların birbiri ile nasıl iletişim kurduğumuz ve senaryo için gereken veri tabanı bağlantısı, ".js" uzantılı bir dosyadan diğerine "navigatior" kütüphanesi sayesinde veri aktarımı, google harita sistemini detaylı bir biçimde anlatıcaz.

Öncelikle fonksiyonlar arası geçişi sağlamak için bir ana kısım oluşturuyoruz. "App.js" kısmında diğer işlemlerle aradaki bağlantıyı saylayabilmek için köprü diye tanımlayabiliceğimiz kısmı oluşturuyoruz. Bu kısımda "@react-navigation" kütüphanesi içerisinden "NavigationContainer", "useNavigation", "createStackNavigator" fonksiyonları yardımıyla fonksiyonların arasındaki haberleşme yapısını kuruyoruz.

```
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
```

Kurduğumuz bu yapı sayesinde menular arasında geçiş sağlarken veri tabanından çektiğimiz kullanıcı, durak ve bilet bilgisi gibi verileri fonksiyondan fonksiyona aktarabildiğimiz bir sistem elde ediyoruz.

Daha sonra SQLite veri tabanı ile local veri tabanı yapısını geliştirmek için ilk olarak "react-native-sqlite-storage" kütüphanesi içerisinden "openDatabase" fonksiyonu yardımı ile veri tabanı oluşturuyoruz.

```
import { openDatabase } from "react-native-sqlite-storage";
```

Oluşturduğumuz veri tabanı ile kullanıcı bilgilerini güncelle, sil ve kaydet fonksiyonlarını yazıyoruz. Bunun yanı sıra Admin kişisinin durak bilgileri, tur bilgileri, geziye kişi kaydetme ve kaydetilen kişilerin haritada göstermek için genel tablosu gibi verilerin güncelle, sil ve kaydet fonksiyonları olan sistemi kuruyoruz.

Google Harita yapısında ise "react-native-maps" kütüphanesi içerisinden "MapView" yardımı ile arayüzde göstericeğimiz harita için ihtiyaç duyduğumuz fonksiyonları projeye dahil ediyoruz.

```
import MapView from 'react-native-maps';
```

Arayüz kısmında "view" yapısının içerisinde haritamızın bilgilerini girerek haritanın genel görünümünü oluşturuyoruz.

```
<MapView
    style={styles.map}
    initialRegion={{
        latitude: 40.766666,
        longitude: 29.916668,
        latitudeDelta: 0.0922,
        longitudeDelta: 0.0421,
    }}
```

2.2.Kullanıcı Girişi

Kullanıcı giriş ekranına bilgilerini yazarak sisteme girişini gerçekleştirir. Karşısına çıkış yap ve durak seç seçenekleri çıkar. Durak seç adlı seçenek kullanıcıya durak bilgisi verilerek kullanıcının hangi duraktan bineceğini seçtiği bilet alma sistemi ile karşılaşıyor. Bu sistemde aldığı bileti güncelleyebilir, silebilir veya kayıt edebilir. Böylelikle kayıt ettiği veriler adminin ekranına gelir ve admin, kullanıcıyı tura kayıt ederek işlem sonlandırılır.

2.3.Admin Girişi

Admin girişi yapıldıktan sonra durak düzenle, durak harita, kullanıcı düzenle, tur düzenle, geziye kullanıcı ekleme ve tur rota belirleme seçenekler sunulur.

Durak düzenleme seçeneği kullanıcının durakları ekleme ve silme seçenekleri sunar.

Durak harita seçeneğinde kullanıcının bineceği durakları harita üzerinde konumlarının gösterildiği google haritası admine arayüzde gösterilir.

Kullanıcı düzenleme seçeneğinde kullanıcının username, password ve üyelik türünü güncellyip, silip ve yenisini ekleyebilir.

Tur düzenle seçeneğinde admin, kullanıcılar için veri tabanında yeni turlar ekler.

Geziye kullanıcı ekle seçeneğinde admine bilet alan kullanıcıların ve sistemde açılan turların bilgileri düşer. Böylelikle admin kullanıcıları turlara ekleyerek sistemi düzenler.

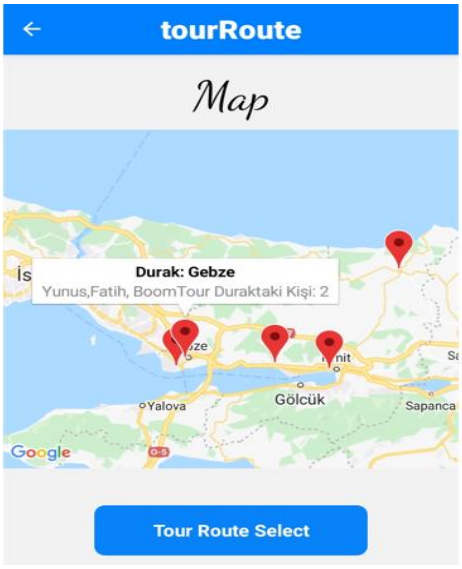
Tur rota seçeneğinde adminin düzenlediği tur bilgileri ile durakta kaç kişinin beklediği ve bekleyen kişilerin isimlerinin yazdığı google harita arayüzü gösterilir.

2.4.Harita Arayüzü

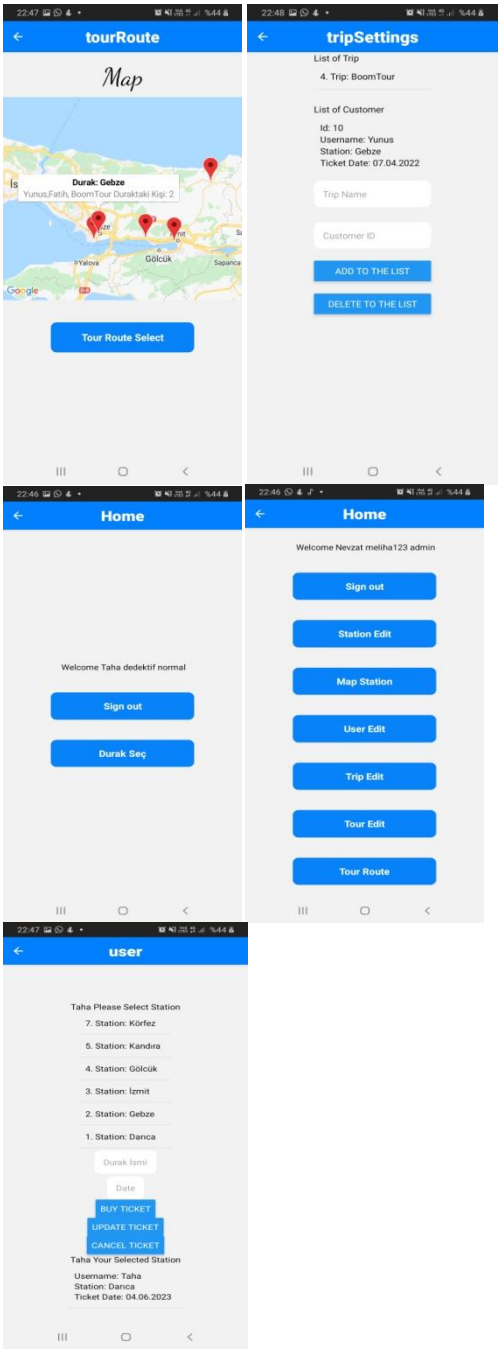
Google harita arayüzünde adminin oluşturduğu tur ile dinamik tur sistemi sayesinde oluşturulan turların tabloları admin oluşturduğunda oluşur. Böylelikle kullanıcının bilet bilgileri ile durak "lat" ve "long" bilgileri birleştirilir.

```
const getTripCategories = () => {
    db.transaction(txn => {
        txn.executeSql(
            `SELECT * FROM busTripName ORDER BY id DESC`,
            [],
            (sqlTxn, res) => {
                console.log("categories retrieved successfully");
                let len = res.rows.length;

                if (len > 0) {
                    let results = [];
                    for (let i = 0; i < len; i++) {
                        let item = res.rows.item(i);
                        results.push({ id: item.id, tripname: item.tripname });
                    }

                    setBusTrip(results);
                }
            },
            error => {
                console.log("error on getting categories " + error.message);
            },
        );
    });
};
```

Birleştirdiğimiz bu bilgileri arayüzde gösterilen durak işaretlerinin üzerinde bu bilgiler admine verilerek genel bir çerçevede gösterilir.



3.DENEYSEL SONUÇLAR



4.SONUC

Mobil bir uygulamanın kütüphane sürümleri ve emülatör gibi zorluklar gölgesinde nasıl geliştirildiğini tecrübe ettik.

5.REFERANSLAR

[1]https://reactnavigation.org/docs/params/

[2]https://www.youtube.com/watch?v=GkuPPJ7AOSQ&t=205s

[3]https://medium.com/infinitbility/react-native-sqlite-storage-422503634dd2

[4]https://stackoverflow.com/questions/58936356/dynamically-rendering-mapview-marker-react-native

[5]https://stackoverflow.com/questions/37500205/react-native-appinstalldebug-failed

[6]https://stackoverflow.com/questions/60310873/execution-failed-for-task-appmergedexdebug-firestore-flutter

[7]https://stackoverflow.com/questions/64254163/android-studio-warning-on-my-app-appstripdebugdebugsymbols-up-to-date

[8]https://stackoverflow.com/questions/63570597/typeerror-func-apply-is-not-a-function

6.AKIŞ DİAGRAMI