Yazılım Geliştirme Laboratuvarı
Swift Voice App

1. Selin Avcı
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye

2.Helen Hacer Uzunçayır
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye

***Özetçe*—Bu rapor, Swift programlama dili ve MapKit framework'ü kullanılarak geliştirilen bir sesli girişle rota belirleme uygulamasını tanıtmaktadır. Uygulama, kullanıcıların sesli komutlar ile başlangıç ve varış noktalarını belirleyerek, önerilen rotaları görüntülemelerini sağlamaktadır. Bu çözüm, kullanıcı deneyimini geliştirmeyi ve rota planlama süreçlerini daha hızlı ve etkili hale getirmeyi amaçlamaktadır. Raporda, uygulamanın geliştirme süreci, kullanılan teknolojiler, karşılaşılan zorluklar ve çözüm yolları detaylı bir şekilde ele alınmaktadır. Ayrıca, gelecekteki geliştirme önerileri ve uygulamanın potansiyel kullanım alanlarına da değinilmektedir.**

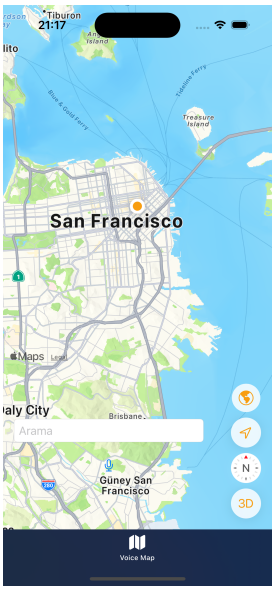
Keywords —Swift; mobile; Map; voice.

I. UYGULAMA YÖNETİM İŞLEMLERİ

Sistem, kullanıcıların otobüs biletlerini çevrimiçi olarak satın almalarını sağlar. Kullanıcılar, seyahat tarihlerini seçebilir, koltuk tercihlerini belirleyebilir ve kartla ödeme işlemlerini gerçekleştirebilirler.

SpeechRecognizer sınıfı **ObservableObject** protokolünü benimseyen bu sınıf, ses tanıma işlevselliğini yönetir ve tanınan metni bir yayınlanan (`@Published`) özellik olarak saklar.

- **startRecording()**: Ses kaydını başlatır ve ses tanıma isteğini yapılandırır. AVAudioSession ve AVAudioEngine'i kullanarak ses kaydını başlatır ve tanınan metni `recognizedText` değişkenine günceller.
- **stopRecording()**: Ses kaydını durdurur ve tanıma görevlerini sonlandırır.



TripMapView Yapısı Şekil 1.1

- **@Environment** ve **@State** değişkenleri**: Uygulamanın çeşitli durum ve bağlam bilgilerini yönetir, örneğin harita kamerasının pozisyonu, kullanıcı konumu, rotanın gösterilip gösterilmeyeceği vb.
- **Map Görünümü**: Haritayı görüntüler ve kullanıcı konumunu, yer işaretlerini ve rotayı gösterir. Kullanıcı etkileşimlerine ve harita durum değişikliklerine tepki verir.
- **TextField ve Mikrofon Butonu**: Kullanıcıların metinle veya sesle arama yapmasına olanak tanır. Ses tanıma butonu, ses tanımanın durumuna göre simge değiştirir.
- **Rota ve Adımlar**: Kullanıcı seçilen bir yer işaretine rota oluşturabilir ve adım adım yönlendirmeleri görebilir. Rota oluşturulduğunda haritada gösterilir ve detayları bir listede sunulur.

Ek İşlevler

- **updateCameraPosition()**: Harita kamerasının kullanıcı konumuna göre güncellenmesini sağlar.
- **fetchRoute()**: Kullanıcının konumundan seçilen yer işaretine kadar bir rota oluşturur.
- **removeRoute()**: Mevcut rotayı temizler ve harita kamerasını yeniden kullanıcı konumuna ayarlar.

Kullanıcı Arayüzü Elemanları

- **Map Controls**: Kullanıcı konumunu gösterme, harita stilini değiştirme ve diğer harita kontrolleri için düğmeler içerir.
- **Sheet ve Detay Görünümleri**: Seçilen yer işaretlerinin detaylarını ve adım adım yönlendirmeleri gösterir.

Ayrıca, yöneticiler, otobüs seferleri ve biletlerle ilgili bilgileri yönetmek için özel bir yönetici paneline erişebilirler.

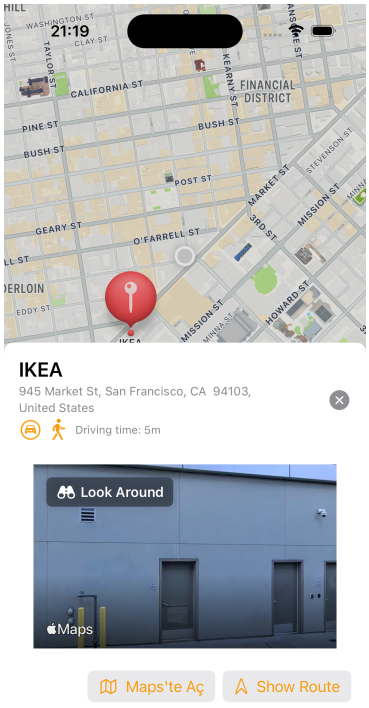
II. KONUM DETAYLARI

LocationDetailView Yapısı

- **@Environment**: `dismiss` özelliği, mevcut görünümleri kapatmak için kullanılır.
- **Bağlı Değişkenler (`@Binding`)**: `showRoute`, `travelInterval`, ve `transportType` gibi bağlanmış değişkenler, üst bileşenden gelen verileri temsil eder ve bu bileşen ile üst bileşen arasında veri akışını sağlar.
- **Durum Değişkenleri (`@State`)**: `name`, `address`, ve `lookaroundScene` gibi durum değişkenleri, bu görünümün içindeki dinamik durumu yönetir.
- **`travelTime` Hesaplanmış Özelliği**: `travelInterval`'ı insan tarafından okunabilir bir seyahat süresine dönüştürür.

Kullanıcı Arayüzü

- **VStack ve HStack**: Dikey ve yatay düzenlerde bileşenleri organize eder.
- **TextField ve Text**
- **Görünümleri**: `name` ve `address` için metin alanları, yer işaretinin adı ve adresini gösterir veya düzenlemeye izin verir.
- **Butonlar**: Kullanıcıların çeşitli eylemler yapmasına olanak tanır:
  - **Güncelle**: Değişiklikleri kaydetmek için.
  - **İşaretler (Icon Buttons)**: Taşıma türünü değiştirmek (otomobil veya yürüme).
  - **Sil/Ad**: Yer işaretini listeye eklemek veya çıkarmak.
  - **Maps'te Aç**: Haritalar uygulamasında yer işaretini açmak.
  - **Rota Göster**: Rotayı göstermek veya gizlemek için.



Şekil 2.1

- **LookAroundPreview**: Apple Maps'in Look Around özelliği ile yer işaretinin görsel önizlemesini sağlar.
- **ContentUnavailableView**: Look Around önizlemesi yoksa gösterilir.

Yardımcı Fonksiyonlar

- **isChanged**: `name` veya `address` değişmişse doğru döner.
- **fetchLookaroundPreview()**: Seçili yer işareti için Look Around önizlemesini getirir.

LocationDetailView İçin Önizleme

- **#Preview("Destination Tab")**: Bir hedef ve seçili yer işareti ile bu görünümün bir önizlemesini sağlar.
- **#Preview("TripMap Tab")**: Bir yer işareti ile bu görünümün bir önizlemesini sağlar.

Kullanım Durumları

- **Yer İşaretlerini Yönetme**: Kullanıcılar bir yer işaretinin adını ve adresini düzenleyebilir.
- **Rotayı Gösterme**: Kullanıcılar yer işaretine bir rota oluşturabilir ve seyahat süresini görebilir.
- **Look Around Görünümü**: Kullanıcılar yer işaretinin Look Around önizlemesini görebilir.
- **Haritada Açma**: Kullanıcılar yer işaretini Apple Maps'te açabilir.

Bu yapı, kullanıcıların bir yer işaretiyle etkileşimde bulunmasını kolaylaştırır ve çeşitli harita tabanlı işlevler sunar.

III. ROTA İŞLEMLERİ

İşlevsel olarak, `Destination` sınıfı bir varış noktasını temsil eder. Bu varış noktasının adı (`name`) ve konumu (`latitude`ve `longitude`) bulunur. Ayrıca, harita görünümünde bu varış noktasını nasıl göstereceğimizi belirlemek için bir bölge (`region`) hesaplar. Bölge, merkezi ve yakınlaştırma derecesi (`latitudeDelta` ve `longitudeDelta`) ile birlikte bir `MKCoordinateRegion` nesnesi olarak temsil edilir.

`Destination` sınıfı ayrıca `MTPlacemark` türünden bir diziye (`placemarks`) sahiptir. Bu, varış noktası ile ilişkilendirilmiş yer işaretlerini tutar.

`Destination` sınıfının `preview` adlı bir özellik uzantısı da vardır. Bu, uygulamanın önizlemesini oluşturur. Önizleme, `Destination` örnekleri içeren

bir `ModelContainer` oluşturur ve bu örneklerin bellekte saklanmasını sağlar.

1. `searchPlaces`: Bu fonksiyon, belirli bir metin sorgusuna (`searchText`) göre yerleri arar. `MKLocalSearch.Request` kullanarak yerel bir arama isteği oluşturur ve bu isteği başlatır. Eğer bir görünür bölge (`visibleRegion`) belirtilmişse, arama isteğine bu bölgeyi de ekler. Sonuçları alır ve bunları `MTPlacemark` olarak oluşturup model bağlamına ekler.

2. removeSearchResults`: Bu fonksiyon, önceki arama sonuçlarını temizler. Burada, `MTPlacemark` nesneleri arasında `destination` özelliği nil olanları siler.

3. `distance`: Bu fonksiyon, bir mesafeyi alır ve bunu kullanıcının tercih ettiği birimde biçimlendirir. Ölçüm biçimini ve birim seçeneklerini belirleyen bir `MeasurementFormatter` kullanır. Örneğin, mesafeyi metre cinsinden alır ve bunu kullanıcının yerel ayarlarına bağlı olarak metre veya yarda olarak biçimlendirir.

KAYNAKLAR

1. https://www.youtube.com/@StewartLynch
2. https://developer.apple.com/documentation/mapkit/
3. https://developer.apple.com/tutorials/app-dev-training/transcribing-speech-to-text