YAPAY ZEKÂ ARAMA ALGORİTMALARI KULLANILARAK MOBİL REHBER UYGULAMASI GELİŞTİRİLMESİ

Ahmet AYDIN

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Temmuz, 2015

Bu tez çalışması Anadolu Üniversitesi Bilimsel Araştırma Projeleri Komisyonu Başkanlığı tarafından desteklenmiştir. Proje No: 1402F047

JÜRİ VE ENSTİTÜ ONAYI

Ahmet AYDIN'ın **"Yapay Zekâ Arama Algoritmaları Kullanılarak Mobil Rehber Uygulaması Geliştirilmesi"** başlıklı **Bilgisayar Mühendisliği** Anabilim Dalındaki, Yüksek Lisans Tezi 31.07.2015 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

		Adı Soyadı	**İmza**
Üye (Tez Danışmanı)	**:**	**Yard. Doç. Dr. Sedat TELÇEKEN**	**……..**
Üye	**:**	**Yard. Doç. Dr. Muzaffer DOĞAN**	**……..**
Üye	**:**	**Yard. Doç. Dr. Mustafa Müjdat ATANAK**	**……..**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …….. tarih ve …….. sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

YAPAY ZEKÂ ARAMA ALGORİTMALARI KULLANILARAK MOBİL REHBER UYGULAMASI GELİŞTİRİLMESİ

Ahmet AYDIN

Anadolu Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yard. Doç. Dr. Sedat TELÇEKEN

2015, 73 sayfa

Mobil cihazların kullanımının artmasına bağlı olarak mobil uygulamalar birçok alanda kullanıcılara hizmet vermektedir. Bu alanlardan biri de turizmdir. Seyahat esnasında, bilinmeyen bir şehirde, seyahate yardımcı en önemli kaynaklardan biri de mobil rehber uygulamalarıdır. Ancak mobil platformlardaki uygulamalar incelendiğinde gezi planını bütünüyle oluşturan ve kullanıcı takibi ile öneriler sunarak seyahati daha verimli kılacak kullanıcı dostu bir mobil uygulamaya rastlanmamıştır. Bu çalışmanın amacı, mobil platformda bir gezi planı uygulaması geliştirilerek kullanıcıların seyahat tecrübelerini daha verimli hale getirebilmek ve turizme katkı sağlamaktır. Uygulamanın hizmetleri arasında bulunan rota hesaplama özelliği için literatürde bulunan arama algoritmaları araştırılmıştır. Benzer problemlerde kullanılan algoritmalar uygulamaya uygun hale getirilerek en kısa rotanın hesaplanması için tutarlılık ve çalışma süresi başlıklarında karşılaştırılmıştır. Pilot şehir olan Eskişehir iline ait veri seti ile geliştirilen uygulama uygun veri setleriyle diğer şehirler için de genişletilebilmektedir.

Anahtar Kelimeler: Yapay Zekâ, Gezi Rehberi Uygulaması, Arama Algoritmaları, A* Algoritması, Karınca Kolonisi Optimizasyonu, Genetik Algoritma

ABSTRACT

Master of Science Thesis

A MOBILE TRIP GUIDE APPLICATION DEVELOPMENT USING ARTIFICIAL INTELLIGENCE SEARCHING ALGORITHMS

Ahmet AYDIN

Anadolu University

Graduate School of Sciences

Computer Engineering Program

Supervisor: Assistant Prof. Dr. Sedat TELÇEKEN

2015, 73 pages

Mobile applications due to the increased use of mobile devices provide services to users in many areas such as tourism. During a trip in an unknown city, one of the most important travel assistants is a mobile guide application. However, when examining applications on mobile platforms, no mobile application exists that is user-friendly, makes up the trip plan, tracks users and offers suggestions to make the trip more efficient. The aim of this project is to develop a trip planner application on the mobile platform to make users' travel experiences more efficient and make contribution to tourism. A research has been done over studies about search algorithms for route calculation that is one of the services of the guide application. Algorithms that are used on similar problems have been implemented for the application and have been compared for shortest route calculation problem over consistency and running time issues. The application that have been developed with the data set which is constructed for the province of Eskişehir is able to be extended for other cities with a proper data set.

Keywords: Artificial Intelligence, Trip Guide Application, Search Algorithms, A* Algorithm, Ant Colony Optimization, Genetic Algorithm

TEŞEKKÜR

Yüksek lisans öğrenimim boyunca ve tez hazırlama sürecinde desteğini esirgemeyen, ilgisi ve tecrübesiyle her zaman yardımcı olan danışmanım Yard. Doç. Dr. Sedat TELÇEKEN'e teşekkürlerimi sunarım.

Anadolu Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği bölümünde görev yapmakta olan tüm hocalarıma ve çalışma arkadaşlarıma ayrıca teşekkürlerimi sunarım.

Tüm hayatım boyunca ilgi ve alâkalarını esirgemeyen ve yardımlarını her zaman hissettiğim aileme, özellikle anneme, teşekkürü bir borç bilirim.

Ahmet AYDIN

Temmuz, 2015

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Günümüzde, Türkiye'de ve dünyada gelişmiş mobil işletim sistemlerini çalıştıran cihazların kullanımında yıldan yıla büyük oranlarda gelişme izlenmektedir [1]. Çizelge 1.1'de bu cihazlar kullandıkları işletim sistemlerine göre sınıflandırılmış ve son 2 yıldaki kullanım verileri gösterilmiştir [2]. Mobil cihaz kullanımının artması, bu cihazlarda çalışan uygulamaların geliştirilmesini de hızlandırmıştır. Mobil cihazlarda en çok kullanılan işletim sistemlerinden biri olan ve Google tarafından geliştirilen Android'in uygulama marketi olan Google Play'de 2009 yılının Mart ayında 2.300 uygulama bulunmaktayken, 2014 yılının Aralık ayında bu sayı 1.430.000'e ulaşmıştır [3]. Android İşletim Sistemi'ni takip eden ve Apple firması tarafından geliştirilen IOS'un uygulama marketi olan App Store'da ise 2009 yılının Mart ayında 25.000 uygulama bulunmaktayken, 2015 yılının Ocak ayında bu sayı 1.400.000 olmuştur [4]. Bu uygulamaların toplam indirilme sayısı ise Haziran-Temmuz 2013'te bu iki işletim sistemi için, her birinde yaklaşık 50 milyar kadardır. Bu rakamlar, akıllı mobil cihazların günlük yaşamdaki yaygınlığını ve kullanıcılarının aktif olarak cihazın olanaklarından faydalandığını göstermektedir.

Çizelge 1.1. Akıllı Telefon İşletim Sistemleri ve Market Payları

İşletim Sistemi	2014 Satış Miktarı (Milyon)	2014 Market Payı	2013 Satış Miktarı (Milyon)	2013 Market Payı	Yıllık Değişim Oranı
Android	1.059,3	%81,5	802,2	%78,7	%32,0
IOS	192,7	%14,8	153,4	%15,1	%25,6
Windows Phone	34,9	%2,7	33,5	%3,3	%4,2
BlackBerry	5,8	%0,4	19,2	%1,9	%-69,8
Diğer	7,7	%0,6	2,3	%0,2	%234,8
Toplam	1.300,4	%100,0	1.018,7	%100,0	%27,7

Akıllı mobil cihazların çoğu alandaki yaygın kullanımı, seyahat alanında da kendini göstermektedir. Hareket halinde kullanımı, internet erişimi ve konum bilgilerini kullanabilmesi gibi özellikleriyle mobil cihazlar seyahat esnasında yardımcı olarak kullanılabilecek önemli araçlardır. Gezginler genellikle kısıtlı

zamanlarda en çok yeri gezmek-görmek isterler. Ancak gezilecek yerler hakkında yeterli bilgi sahibi olmamaktan kaynaklanan yer seçme kararsızlığı, toplam seyahatin süresini kestirememe ve gezilecek yerler arasında bağlantı kuramama gibi sorunlar sürenin yeterince verimli kullanılamamasına yol açmaktadır. Bu gibi nedenlerden dolayı, seyahat esnasında bilinmeyen bir şehirde gezerken mobil cihazlar, seyahatte kullanılabilecek en önemli yardımcı araçlar olarak kullanılabilir. Mobil cihazlardan erişilebilen web siteleri ve kullanılan mobil uygulamalar, seyahat esnasında şehrin turistik yerleri, yemek ve eğlence yerleri hakkında kullanıcıya açıklama, görsel veri ve kullanıcı tecrübeleri gibi içerik bilgileri sunabilmektedir. Böylece gezgin seyahati için ilgisini çeken yerler hakkında detaylı bilgi alabilir, diğer gezginlerin tavsiye ve değerlendirmelerini inceleyebilir ve böylelikle seyahatini verimli bir şekilde gerçekleştirebilir.

Web ve mobil platformlardaki seyahat uygulamaları için en önemli sorunlardan biri şehirler ve şehirlerdeki gezilecek yerler hakkındaki görsel ve açıklayıcı içerik bilgilerinin basılı kent rehberi broşürlerinde bulunması, ancak bunların dijital ortamlara sağlıklı aktarılmamasından kaynaklanmaktadır. Daha çok ziyaretçi çeken popüler şehirlerde bu konularda sorun yaşanmamakta fakat daha az ziyaret edilen ve gelişmekte olan şehirlerde bu gibi içerik sorunları gözlenmektedir. Bu sorunun bir çözümü olarak görülen kullanıcı tabanlı içerik sağlayan uygulamalar da maalesef yeterli güvenilirliği sağlayamamaktadır. Bu şartları sağlayan şehirler için güvenilir ve içerik açısından zengin bir uygulamanın eksikliği bu yönde bir ihtiyaç oluşturmuştur. Bu şartlar göz önüne alındığında turistik içerik zenginliği ile Eskişehir ili böyle bir uygulama için pilot şehir olarak seçilmesi uygun görülmüştür.

Mobil cihazlardan erişilebilecek bir uygulama geliştirilmek istendiğinde bir web sitesi oluşturulabilir ve kullanıcı bu uygulamaya mobil cihazındaki bir tarayıcıdan erişebilir. Bunun dışında ise aynı ihtiyaçları sağlayan mobil bir uygulama geliştirilebilir. Mobil uygulamalar, mobil cihazlarda hem görsel hem de çalışma performansı olarak daha iyi sonuçlar verdiğinden daha fazla tercih edilirler.

Bir mobil uygulama geliştirilirken göz önünde bulundurulması gereken başlıca etkenlerden birisi de kullanıcı istekleridir. Burada uygulamanın daha fazla kullanıcıya ulaşması için, dünya genelinde mobil işletim sistemleri kullanım

oranları incelendiğinde, Android İşletim Sistemi'nin tercih edilmesinin uygun olacağı öngörülmüştür.

Konuyla ilgili yapılan araştırmalarda, kullanıcının keyifli ve planlı bir gezi yapmasını sağlayan çeşitli uygulamalara ve web sitelerine rastlanmıştır. Bunlardan bazılarına örnek vermek gerekirse:

Tripomatic: Hem web platformunda hem de mobil cihazlarda hizmet veren, ara yüzü ve kullanılabilirliği ile kullanıcı dostu bir uygulama olarak değerlendirilmektedir. Gezinin tarihleri seçilip gezilecek yerler seçilen tarih ve belirtilen saatlere atanabilmektedir. Böylece seçilen yerler bir gezi planı haline getirilebilmektedir. Ayrıca şehirde bulunan kayıtlı tur ve aktiviteleri de listeleyebilmekte ve gezi seçeneklerini artırmaktadır. Bunun yanında kalınabilecek otel ve araba kiralama hizmetleri tavsiyeleri ile geziyi tam olarak kapsayabilmektedir. Gezilecek yerler isteğe bağlı olarak haritada gösterilebilmekte ve rota çizilebilmektedir hatta Google Maps uygulaması aracılığıyla bir varış noktasına gidiş yolu tarif edilebilmektedir. Ancak yol tarifleri tek bir varış noktasına göre düzenlendiğinden gezinin genelindeki izlenilen rotanın en kısa rota olma ihtimali düşüktür. Uygulama, genel özellikleriyle bu şekildedir ve dünya genelinde popüler şehirler için detaylı içeriklere ve seçeneklere sahiptir. Ancak diğer şehirler için sunabileceği seçenekler oldukça kısıtlıdır.

Tripadvisor: En çok kullanılan seyahat uygulamalarından biri olan Tripadvisor, geniş içeriğiyle ve kullanıcı dostu arayüzüyle kolay bir kullanım sunmaktadır. Kullanıcı sayısının çokluğu sayesinde gezilecek yerler hakkındaki tavsiyeler ve incelemeler daha bilgilendirici olmaktadır. Gezideki beğenilen yerlerin kaydedilebilmesi, haritada konumlandırılabilmesi ve Google Maps aracılığı ile yol tarifi hizmeti sunabilmesi uygulamayı daha kapsamlı kılmaktadır. Popüler şehirler için olan çevrimdışı seçenekleri de yurtdışı seyahatler için kullanıcıya kolaylık sunmaktadır. Ancak gezilecek yerlerin birbiriyle ilişkilendirilememesi ve plan oluşturulmaması gezinin bir bütün olarak incelenebilmesini zorlaştırmaktadır.

Gezi Rehberi: Türkiye genelinde hizmet veren bu mobil uygulama, şehirler ve önemli yerleri hakkında detaylı içeriğe ve "Nasıl gidilir?", "Tarihçe", "Ne yenir?" vb. seçeneklere sahiptir. Ancak bu içerikleri aynı hizmeti sunan başka

uygulamalardan temin ettiklerinden her zaman sağlıklı çalışmamakta ve doğru sonuç vermemektedir. Ayrıca bulunan harita seçeneği konum belirtmemekte ve başlangıç noktasını belirttikten sonra Google Maps uygulaması aracılığı ile yol tarifi hizmeti vermektedir.

Yerel Kent Rehberi Uygulamaları: Bu uygulamalar şehir bazlı olup şehrin önemli yerleri hakkında detaylı bilgiler içermektedir. İzmir, İstanbul, Konya, Bursa vb. şehirler için geliştirilen bu uygulamaların bazılarında konum bilgisi bulunmasına rağmen, kullanıcıya herhangi bir tavsiye veya rota bilgisi sunulmamaktadır.

1.1.Problemin Tanımı ve Amaç

Uygulama marketlerinde ve web platformundaki seyahat uygulamalarına bakıldığında, tam olarak bir seyahat planlamaya yardımcı uygulamaya rastlanmamıştır. Mevcut uygulamalar çoğunlukla şehirdeki gezi, yemek, eğlence vb. yerleri tanıtmakta ve bilgilendirici içerikler sunmakla yetinmektedir. Kullanıcının gezmek istediği yerleri bütünleştirip bir plan hazırlayan, kullanıcının seyahati sırasında bu planı takip ederek kullanıcıyı yönlendiren ve seyahat esnasında planda interaktif güncelleme yapabilecek bir uygulama bulunmamaktadır. Bu durum kullanıcının seyahat planını başka bir ortamda, kendi imkânlarıyla oluşturmasını gerektirmekte ve hem daha fazla uğraş hem de seyahatten alınacak verimin düşme ihtimalini doğurmaktadır.

Son yıllardaki öneri sistemlerinin gelişimiyle, özellikle kullanıcı profiline dayalı içerik tabanlı öneri sistemleri, çoğu bilgisayar bilimleri konularıyla özellikle bilgi erişim sistemleri ve yapay zekâ ile etkileşimli olarak çalışmaktadır [5]. Öneri sistemlerinde, kullanıcı tabanlı, içerik tabanlı ve bu ikisinin karışımından oluşan hibrit filtreleme sistemleri kullanılır. E-ticaret alanında kullanılmaya başlanan bu sistemler zamanla çoğu alanda kendini göstermektedir. Seyahat yardımcısı sistemlerde kullanılması da kullanıcının seyahat deneyimini daha zevkli hale getirmektedir. En basit haliyle örnek vermek gerekirse, kullanıcının belli kategorideki gezi yerlerini daha çok ziyaret etmesi kullanıcının ilgi alanları hakkında fikir verir ve önerilecek yeni gezi yerleri bu kategoriden seçilebilir.

Literatürde bulunan seyahat uygulamalarının, bahsedilen problemlerden ötürü kullanıcıya yeterli faydayı sağlayamadığı görülmüştür. Hem bu problemleri giderecek hem de öneri sistemleri ve yapay zekâ uygulamalarını bünyesinde barındırarak kullanıcıya daha keyifli bir seyahat deneyimi yaşatmayı amaçlayan bir mobil uygulama geliştirilmesi uygun görülmüştür. Bu uygulamanın kullanıcı ihtiyaçları doğrultusunda Android mobil platformunda ve pilot şehir olarak seçilen Eskişehir ili içerikleriyle geliştirilmesine karar verilmiştir.

"Akıllı Gezi Rehberi" ismi verilen mobil uygulama, arka planda çalışan bir sunucu uygulaması ve verilerin kayıtlı tutularak, veriler üzerinde değişiklikler yapılabildiği bir veri tabanı ile etkileşimli olarak çalışmaktadır.

Mobil uygulama, kullanım esnasında başlangıçta bir kullanıcı girişi sistemi barındırmaktadır. Kullanıcı hakkında temel bilgilerin kaydedildiği bu sistemin uygulamada bulunmasının amacı kullanıcının uygulamayı kullanma sırasındaki seçimlerinin ve hareketlerinin kullanıcı ile ilişkilendirilerek kullanıcının tanınması ve daha verimli öneriler sunulabilmesidir.

Kullanıcı, Eskişehir için kategorilere göre ayrılmış olan gezi yerleri üzerinden tercihi ve ziyaret süresine göre bir gezi planı tercih etmektedir. Burada gezi yerleri hakkında yazımsal ve görsel bilgiler kullanıcıya verilerek, tercih aşamasında kullanıcıya yardımcı olunmaktadır. Kullanıcı uygulamada yeni gezi planı oluşturup, mevcut oluşturduğu ve henüz tamamlamadığı planları güncelleyebilmektedir. Kullanıcı, oluşturduğu planlar için gezinin uygulanması hakkında planı başlatmadan bilgi sahibi olmakta ve bu bilgiler doğrultusunda planlarını değiştirebilmektedir.

Kullanıcı gezisi esnasında uygulama tarafından GPS sistemi vasıtasıyla takip edilmektedir. Kullanıcının uygulama üzerindeki geçmiş tercihleri göz önünde bulundurularak, gezi esnasında kullanıcıya yeni öneriler sunulmakta ve önerilerin kabul edilmesi durumunda alternatif rota ve plan bilgileri oluşturulmaktadır. Her kabul edilen önerinin sonucunda güncellenen plan, kullanıcı tarafından tüm gezi yerleri ziyaret edildiğinde tamamlanmaktadır. Bununla birlikte, ziyaret edilen yerlerin ortalama ziyaret süreleri ve kullanıcının ziyaret süresi dikkate alınarak, toplam plan süresinin önünde ya da gerisinde olunmasına göre, alternatif seneryalolar önerilmekte ve buna göre anlık gezi planı hesaplanabilmektedir.

1.2.Tezin Ana Hatları

Gezilecek yerler hakkındaki açıklayıcı bilgiler, yüksek çözünürlüklü görsel veriler ve konum bilgileri, Eskişehir Büyükşehir Belediyesi ve Eskişehir İl Kültür ve Turizm Müdürlüğü tarafından temin edilmiştir [6,7]. Bu sayede içeriğin hem zengin hem de güvenilir olması sağlanmıştır. Elde edilen veriler hakkındaki bilgiler ayrıntılarıyla Bölüm 2'de verilmiştir.

Yerel resmi kurumlardan temin edilen veriler işlenerek bir veri tabanı yapısında sınıflandırılarak saklanmaktadır. Geliştirilen bir sunucu programı sistem için gerekli hesaplama ve işlemleri gerçekleştirmektedir. Mobil uygulama ile kullanıcı etkileşimli ve kullanıcı dostu bir arayüz geliştirilmiştir. Sistemi oluşturan bu bileşenler, tasarımları ve birbirleriyle etkileşimleri hakkında Bölüm 3'te detaylı bilgiler verilmiştir.

Sistemin ve bileşenlerinin geliştirilmesi esnasında faydalanılan teknolojiler hakkında kısa ve tanıtıcı bilgiler Bölüm 4'te verilmiştir.

Tez süresince geliştirilen mobil uygulamanın çalışması sırasında kullanıcı seçimlerine göre bir gezi esnasında birden fazla rota oluşturulabilmektedir. Uygulama, kullanılış amacı gereği kullanıcıya hızlı cevap vermesi gerekmektedir. Bunun için rota oluşturma işleminde gerekli olan arama ve kısa yol bulma algoritmalarından yapılan literatür taramalarında öne çıkanlar ve bunların temelini oluşturan algoritmalar hakkında detaylı bilgiler Bölüm 5'te verilmiştir.

Rota oluşturma için gerekli olan ve Bölüm 5'te tanıtılan algoritmaların uygulamaya uygun olacak şekilde kodlaması yapılmıştır. Algoritmaların kodlama detayları, en uygun şekilde çalışmaları için yapılan test sonuçları ve belirli senaryolarda yapılan karşılaştırma sonuçları Bölüm 6'te gösterilmektedir.

Tezin içeriğinde yer alan mobil uygulamanın işleyişi ve yaşam döngüsü hakkında bilgiler detaylı bir şekilde Bölüm 7'da yer almaktadır. Tezde yapılan çalışmaların sonuçları ve gelecek için sunulan öneriler Bölüm 8'de yer almaktadır.

2. VERİ KÜMESİ TANITIMI

Eskişehir ili kültürel, tarihi, turistik zenginlikleriyle ve eğlence yerleriyle Türkiye'de öne çıkan şehirler arasındadır. Bu zenginliğin oluşmasında eskiye sahip çıkma ve sürekli yeni şeyler katma isteği bulunmaktadır. Belediyelerin ve İl Turizm Müdürlüğü'nün bu konudaki çalışmaları şehrin zenginliğini günden güne artırmaktadır. Tarihi dokusu bozulmadan restore edilen eserler, modern yapıtlar, sanatı destekleyen ve daha ileri taşımayı amaçlayan çeşitli müzeler ve sergiler, artan genç nüfusun da etkisiyle gelişen eğlence sektöründeki çeşitlilik, şehri seyahat için tercih edilebilecek şehirler arasına taşımıştır.

Tarih boyunca birçok uygarlığa ev sahipliği yapan Eskişehir, bu uygarlıkların medeniyete kazandırdığı çok çeşitli eserleri de bünyesinde barındırmaktadır. Hitit İmparatorluğu, Frig Krallığı, Bizans, Selçuklu ve Osmanlı Devletleri gibi uygarlıklardan kalan eserler şehrin çeşitli yerlerinde korunmakta ve sergilenmektedir. Bunlardan günümüzde en çok ziyaret edilenler arasında Frig Krallığı'nın Midas Yazılıkaya Anıtı ve çevresindeki tarihi eserler, Selçuklu Devleti'nin Alaeddin Camii, Osmanlı Devleti'nin Kurşunlu Cami Külliyesi ve tarihi Odunpazarı Evleri gösterilebilir [8].

Eskişehir'in Han İlçesi'ndeki Yazılıkaya Köyü'nde bulunan Midas Yazılıkaya Anıtı, Frigler tarafından M.Ö. 600'lerde yapılmıştır. Yüksekliği 17 mt. olan anıt, Antik Frigya'nın merkezi olarak kabul edilir [9].

1267 yılında Anadolu Selçuklu Devleti hükümdarı III. Gıyaseddin Keyhüsrev tarafından yapılan Alaeddin Camii, şehrin en eski camilerindendir. Çevresinde aynı adı taşıyan bir de park bulunmaktadır.

Kurşunlu Külliyesi, 16. yüzyıl Osmanlı dönemine ait bir eserdir. Osmanlı Devleti vezirlerinden Çoban Mustafa Paşa tarafından yaptırılmıştır. Külliye, camii, şadırvan, zaviye, talimhane, harem, imaret, Mevlevi şeyhlerine ait türbe ve iki kervansaraydan oluşmaktadır.

Eskişehir'in ilk yerleşim yeri olan Odunpazarı Semti'nde bulunan Odunpazarı Evleri, 19. yüzyıl Osmanlı mimarisinin en güzel örneklerindendir. Kıvrımlı yolları, ahşap süslemeli bitişik düzeni, ve cumbalı evleriyle günümüze

kadar taşınan bu bölge ve bölgedeki evlerin bir kısmı, tarihi dokusu bozulmadan restore edilmiştir.

(a) (b) (c)

Şekil 2.1. Eskişehir Tarihi Eser ve Anıtları a) Yazılıkaya Anıtı, b) Kurşunlu Külliyesi ve c) Odunpazarı Evleri

Kültür turizminin en önemli yapıtaşlarından biri de müzelerdir. Eskişehir'de bulunan çok sayıda ve çeşitli müzeler ziyaretçiler tarafından yoğun ilgi görmektedir. Bu müzeler arasında özellikle ETİ Arkeoloji Müzesi, Balmumu Müzesi, Hava Müzesi ve Çağdaş Cam Sanatları Müzesi çok sayıda ziyaretçi ağırlamaktadır [10].

2007 yılında Eti firması tarafından hayata geçirilen ETİ Arkeoloji Müzesi'ndeki eserler çevredeki antik kentlerden çıkartılan ve rastlantı sonucu ortaya çıkan mimari eserler ve heykellerdir. Öncesinde farklı yerlerde sergilenen eserler yenilerinin eklenmesiyle ve yer yetersizliği nedeniyle birkaç kez taşındıktan sonra ve ETİ Arkeoloji Müzesi binası inşa edildikten sonra burada sergilenmeye başlanmıştır.

Türkiye'de başka bir örneği olmayan Yılmaz Büyükerşen Balmumu Müzesi'nde ise başta Ulu Önder Mustafa Kemal Atatürk'ün sivil ve askeri dönemlerine ait olmak üzere çok sayıda balmumu heykeli bulunmaktadır. Bunların yanısıra Atatürk'ün aile fertlerinin, Kurtuluş Savaşı komutanlarının, Osmanlı Devleti padişahlarının, yerli ve yabancı çok sayıda siyasetçinin, gazeteci, aktör ve sanatçıların heykelleri bulunmaktadır.

1998 yılında açık teşhir olarak ziyarete açılan Hava Müzesi'nde çeşitli tip ve modellerde sivil ve savaş uçakları sergilenmektedir. Kapalı bir mekânda ise pilot giysileri, rozetler, maket uçaklar ve uçak motorları yer almaktadır.

2007 yılında Eskişehir Büyükşehir Belediyesi, Anadolu Üniversitesi ve Cam Dostları Grubu'nun işbirliği ile kurulan Türkiye'nin ilk cam sanatları müzesinde, 58 yerli ve 10 yabancı sanatçının eserleri sergilenmektedir.

Bunların yanında Kent Belleği Müzesi, Pessinus Açık Hava Müzesi ve Lületaşı Müzesi gibi birçok müze de ziyaretçilere hizmet vermektedir.

(a) (b)

(c) (d)

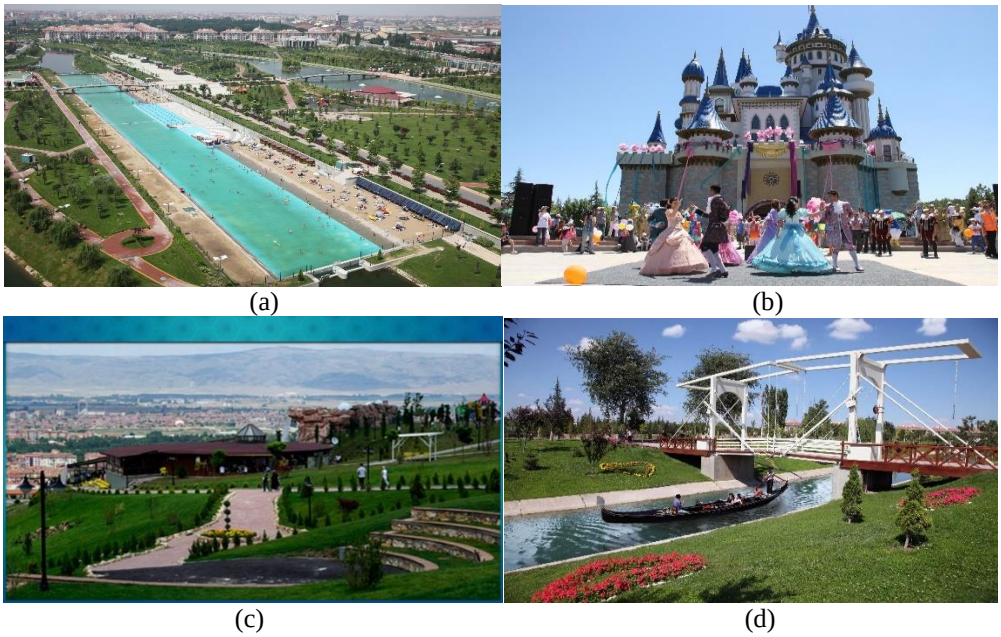
Şekil 2.2. Eskişehir Müzeler a) ETİ Arkeoloji Müzesi, b) Balmumu Müzesi, c) Uçak Müzesi ve d) Çağdaş Cam Sanatları Müzesi

Tarihi ve kültürel gezi yerlerinin yanısıra bir seyahati eğlenceli kılan ve görsel olarak ziyaretçiye hitap eden yerler vardır. Geniş ve görsel olarak tatmin edici öğelerle dolu parklar, şehrin güzelliklerinin yanında kafelerin olduğu caddeler ve eğlence yerleri seyahatin daha keyifli hale gelmesine katkıda bulunmaktadır. Eskişehir bu açıdan da ziyaretçilerini memnun bırakacak yerlere sahiptir [11].

Kentpark, 270 bin m^2'lik alanda yer alan geniş bir parktır. İçerisinde açık ve kapalı havuzlar, restoranlar, kafeler, at binme alanları ve yapay plajı bulunmaktadır. Türkiye'de bir ilk olan yapay plaj, 350 mt. uzunluğunda, gerçek deniz kumu ile donatılmış ve yaz aylarında hizmet vermektedir.

Sazova Bilim Sanat ve Kültür Parkı, yaklaşık 400 bin m^2'lik bir alanda bulunan Eskişehir'in en büyük parkıdır. İçerisinde çeşitli su sporlarına da imkân sunan büyük bir gölet, restoranlar, kafeler, açık hava konser alanı, anfi tiyatro, müze gemi, masal şatosu, bilim deney merkezi, uzay evi, kapalı akvaryum binası, hayvanat bahçesi ve japon bahçesi bulunan hem geniş hem de içerik bakımından oldukça zengin bir parktır. Park içinde hizmet veren özel gezi trenleriyle keyifli bir şekilde gezilebilmekte ve ulaşım kolaylığı da sağlanmaktadır.

Şelale Park, şehre hakim bir noktada bulunan, şehir manzarasıyla ön plana çıkan bir parktır. İçerisinde yapay şelale, yel değirmeni, Don Kişot ve Sanço Panço heykelleri bulunmaktadır. Bunların yanı sıra parkta mini anfi tiyatro, restoran-kafe ve seyir terası bulunmaktadır [12].



(a) (b)

(c) (d)

Şekil 2.3. Eskişehir Parklar a) Kentpark, b) Sazova Bilim ve Kültür Parkı, c) Şelale Park, d) Şehr-i Aşk Adası

Haller Gençlik Merkezi, tarihî "Yaş Sebze ve Meyve Hali" binasının restore edilmesi ile oluşmuştur. İçerisinde hediyelik eşya dükkânları, büfeler, kafeler ve barları barındırır. Bu nedenle hem şehri ziyaret edenlerin hem de şehir sakinlerinin ilgisini çekmektedir. Bunların yanı sıra Şehir Tiyatrosu sahnesi, sergi salonu ve kitabevi ile kültür-sanat etkinliklerine de ev sahipliği yapmaktadır.

Bu yerlerin haricinde ortasından Porsuk Çayı geçen ve etrafında çok sayıda kafe ve restoranın bulunduğu Adalar olarak da bilinen Porsuk Bulvarı ve etrafında

çeşitli bar ve eğlence yeri bulunan Barlar Sokağı (Vural Sk.) da Eskişehir ziyareti esnasında dinlenilebilecek ve eğlenilebilecek yerler arasındadır.

(a) (b) (c)

(d) (e)

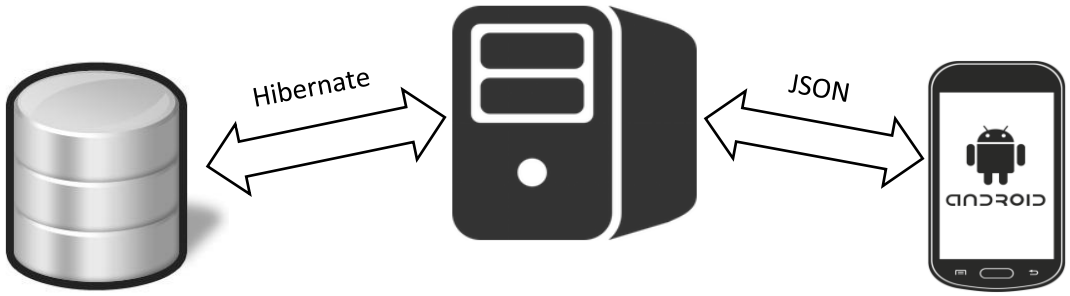
Şekil 2.4. Eskişehir Genel a) Haller Gençlik Merkezi, b) Tepebaşı Belediyesi Binası, c) Porsuk Çayı, d) Adalar, e) Reşadiye Camii

3. SİSTEM TASARIMI

Tez kapsamında bünyesinde yapay zekâ ve öneri sistemleri barındıran bir mobil gezi rehberi uygulaması geliştirilmiştir. Android mobil platformu için geliştirilen uygulama kullanıcı tercihlerine göre gezi planı oluşturabilmekte ve gezi esnasında kullanıcı takibi gerçekleştirerek farklı durumlarda farklı öneriler sunabilmektedir. Kullanıcının önerilere olan tepkisi ile uygulama, planı dinamik olarak güncelleyebilmektedir.

Uygulamanın kullandığı veri kümesindeki gezi yerlerine ait bilgiler pilot şehir olarak seçilen Eskişehir ili içerisindeki gezi yerleridir ve gerekli bilgilerin büyük bir çoğunluğu Eskişehir iline ait yerel resmi kurumlardan sağlanmıştır. Veri kümesindeki gezi yerlerine ait diğer bilgiler web ortamındaki resmi olmayan çeşitli tanıtım sitelerinin içeriklerinden elde edilmiştir.

Uygulamanın çalışması için oluşturulan sistemin genel yapısı 3 ana kısımdan oluşmaktadır. Bunlar uygulamaya ait verilerin saklandığı veri tabanı, kullanıcı ile etkileşimde kullanılan mobil uygulama ve ikisi arasında iletişimi sağlayan ve hesaplama işlemleri yapan sunucu programıdır.



Şekil 3.1. Sistemin Genel Yapısı Grafik Gösterimi

3.1.Veri Tabanı Yapısı

Eskişehir Büyükşehir Belediyesi ve Eskişehir İl Kültür ve Turizm Müdürlüğü'nden edinilen gezi yerlerine ait görsel ve yazılı içerikler ve koordinat verilerinin saklanması amacıyla Microsoft SQL Sunucusu'nda bir veritabanı

tasarlanmıştır. Verilerin kolay ulaşımı ve yönetimi için veritabanı, her biri ayrı bir parçanın betimleyicisi olacak şekilde tablolara ayrılarak ve bu tablolardan birbirleriyle bağlantılı olanlar arasında ilişki modelleri kurularak oluşturulmuştur.

Microsoft SQL Server veri tabanı sunucusunda oluşturulan veri tabanında verilerin tablolara ve sütunlara ayrılması işlemi ve tablolar arası ilişkiler, verilerin bütünlüğü ve işlevselliği de göz önünde bulundurularak gerçekleştirilmiştir. Ayrıca veri tabanının genel tasarımı esnasında gerekli verimin alınabilmesi için veri tabanı normlarına uygun olmasına dikkat edilmiştir. Veri tabanındaki her tabloda veri tabanı tarafından otomatik olarak üretilen ve tablolar için birincil anahtar olan "ID" alanları bulunmaktadır. Bu alanlar bulundukları tablodaki kayıtlar için belirleyici özellikte olup sistem içerisinde bir kayıt kullanılmak istendiğinde bu alanların değerlerinden faydalanılmaktadır.

Kullanıcı tablosunda kullanıcıya ait bilgiler bulunmaktadır. Bunlar veri tabanı tarafından otomatik olarak üretilen ve tablo için birincil anahtar olan "ID" alanı değeri, kullanıcı kaydı esnasında edinilen ad, soyad, e-posta adresi ve kullanıcının uygulama için belirlediği şifresidir. Kullanıcı şifresi sistem tarafından MD5 şifreleme algoritması ile şifrelenerek, oluşan sonuç değeri veri tabanında saklanmaktadır. MD5 tek yönlü bir şifreleme algoritmasıdır ve sonuç değerinden giriş değerini elde etmek mümkün değildir. Bu sebeple kullanıcı şifresi için güvenliği sağlamak amacıyla tercih edilmiştir.

Tourplan tablosunda kullanıcının oluşturduğu plana ait genel bilgiler bulunmaktadır. Bunlar, plana kullanıcı tarafından verilen isim, planı oluşturan kullanıcı, planın gerçekleşeceği şehir ve gün bilgileri, planın başlangıç ve bitiş saatleri, planın başlangıç noktası enlem ve boylam koordinatları ve planın tamamlanma bilgileridir.

Şehir, ülke ve kategori tabloları tanım tabloları olarak geçmektedir ve sadece isim bilgileri bulunmaktadır. Aynı şekilde fotoğraf bilgilerinin tutulduğu "PHOTO" tablosunda da veri tabanı tarafından otomatik olarak üretilen bir "ID" değeri ve byte formatında saklanan fotoğraf nesnesi bulunmaktadır.

Fotoğraf verilerinin boyutları çok çeşitli olabilmektedir. Kullanıcının daha iyi bir biçimde inceleyebilmesi için gezi yerlerine ait yüksek çözünürlüklü fotoğraflar veri tabanında saklanmaktadır ve ihtiyaç anında kullanıcıya

sunulmaktadır. Ancak yüksek çözünürlüklü görsel verilerin boyutları da yüksek olduğundan veri transferi sırasında yavaşlama olmaktadır. Özellikle birden fazla görsel veriye sahip gezi yerleri incelendiğinde bu durum gözle görülür yavaşlamalara sebep olmaktadır. Bunun önüne geçebilmek için gezi yerlerine ait fotoğrafların orijinallerinden daha düşük çözünürlüklü ve daha az alan kaplayan kopyaları oluşturulmuştur. Oluşturulan bu kopyalar orijinalleriyle ilişkilendirilerek "PHOTOLOW" adlı ayrı bir tabloda saklanmaktadır. Kullanıcı gezi yeri hakkında bilgi edinmek istediğinde bu tablodaki veriler kullanıcıya daha hızlı olarak sunulmaktadır. Orijinal fotoğraflar ise sadece kullanıcı tarafından özellikle görüntülenmek istendiğinde kullanılmaktadır. Bu aşamada da kullanıcı bir kerede sadece bir orijinal fotoğraf görüntüleyebildiği için başlangıçtaki verilerin yavaş transfer edilmesi sorunu büyük ölçüde engellenmiş olmaktadır.

Mekan tablosunda bir gezi yerine ait tüm bilgiler saklanmaktadır. Mekan tablosu veri tabanındaki tablolar arasında diğer tablolarla en çok ilişki barındıran tablodur ve bu ilişkilere ait alanlar barındırmaktadır. Bunların haricinde gezi yerinin isminin ve gezi yeri hakkında kullanıcıyı bilgilendirmeye yönelik detaylı açıklama bilgilerinin saklandığı metin alanları bulunmaktadır. Gezi yerlerine ait konum bilgileri de bu tabloda enlem ve boylam değerlerinde saklanmaktadır. Konum bilgileri gezi yerlerinin bir plan dahilinde sıralanmasında, uzaklık ve süre hesaplamalarında, kullanıcıya öneriler sunulmasında ve gezi yerlerinin harita üzerinde konumlandırılması gibi çeşitli alanlarda kullanılmaktadır. Tablo da gezi yerlerine ait açılış ve kapanış saatleri, ideal gezi saatleri ve gezi yerinin ziyarete kapalı olduğu günler de saklanmaktadır. Kullanıcı bu bilgiler doğrultusunda daha verimli planlar oluşturabilmektedir. Mevcut sistemde bu bilgiler kullanıcıyı bilgilendirme amacıyla kullanılmaktadır. Tabloda gezi yerlerine ait tavsiyelerin ve puanlamanın saklandığı bir alan bulunmaktadır. Sistemde bu alanlara sistem yöneticisi tarafından veri girişi sağlanmaktadır. Ancak sistem olabildiğince esnek olarak tasarlanmıştır ve ticari amaçlı kullanıma sunulmak istendiğinde en az değişiklikle bu amacın sağlanması düşünülmüştür. Bu şekilde sistemin sosyal becerilerinin artırılması için gerekli altyapı da hazırlamıştır. Bu alanlar ticari kullanım amaçlandığında veri girişini sağlayan kullanıcılarla ilişkilendirilecekleri ayrı tablolarda bulunmalıdır.

Veri tabanında sistem tarafından otomatik olarak üretilen ve her tabloda bulunan “ID” alanlarına otomatik olarak atanan kimlik değerleri bulunduğu tablodaki kayıtlar için belirleyici özellik taşımaktadır. Tablolar arasındaki ilişkiler bu kimlik değerleri sayesinde sağlanmaktadır.

Veri tabanındaki tablolar arasında ilişkiler oluşturulmasının temel amacı veri tutarsızlıklarının önüne geçmektir. Yazılım kısmında ise birbiriyle mantıksal olarak ilişkili verilerin veri tabanında ilişkili olması verilerin kullanımını kolaylaştırmaktadır. Veri tabanı ilişkileri bire bir ilişki, bire çok ilişki ve çoka çok ilişki olmak üzere 3 çeşitten oluşmaktadır.

Bir tablonun birincil anahtarında bağlı olarak diğer bir tabloda sadece bir kayıt bulunması şeklindeki ilişkiye bire bir ilişki denmektedir. Tablolardan birinde, diğer tablonun birincil anahtarı saklanarak ilişki oluşturulmaktadır.

Bir tablonun anahtar alanları ile ilişkili diğer bir tabloda birden fazla kayıt bulunmakta ise bu iki tablo arasındaki ilişkiye bire çok ilişki denmektedir. Çoğunlukla “çok” özelliğini sağlayan tabloda “bir” özelliğini sağlayan tabloya ait anahtar alanlarının bilgisi saklanarak ilişki sağlanmaktadır.

Bir tablonun anahtar alanı ile ilişkili diğer bir tabloda birden fazla kayıt bulunmakta ve bu tabloya ait anahtar alan ile ilişkili ilk tabloda da birden fazla kayıt bulunmakta ise bu iki tablo arasındaki ilişkiye çoka çok ilişki denmektedir. Çoka çok ilişkilerde veri tabanına bir geçiş tablosu eklenmesi gerekmektedir. Geçiş tablosunda ilişkili iki tablonun ilişkiyi sağlayan anahtar alanları bilgileri bulunmaktadır [13].

Sistemdeki her plan bir kullanıcı tarafından oluşturulduğundan, planların kullanıcılarla ilişkilendirilmesi gerekmektedir. Bir plan sadece bir kullanıcı tarafından oluşturulabilirken, bir kullanıcı birden çok plan oluşturabilmektedir. Bu sebeple kullanıcı tablosu ve plan tablosu arasında bire çok bir ilişki bulunmaktadır. Bu ilişki plan tablosundaki bir alanda kullanıcı tablosundaki kayıtların kimlik bilgisinin saklanmasıyla sağlanmaktadır. Bu sayede sistem tarafından ihtiyaç duyulduğunda bir kullanıcının oluşturduğu planlar rahatlıkla elde edilebilmektedir.

Tanımlanan bir plan sadece bir şehre ait olabilmekte ancak bir şehir için birden fazla plan oluşturulabilmektedir. Bu sebeple şehir ve plan tabloları arasında

da bire çok bir ilişki bulunmaktadır. Bu ilişki plan tablosundaki bir alanda şehir tablosundaki kayıtların kimlik bilgilerinin saklanmasıyla sağlanmaktadır.

Sistemde oluşturulan bir plan içerisinde birden fazla gezi yeri bulunabilmekte ve aynı şekilde bir gezi yeri birden fazla plana dahil olabilmektedir. Bu sebeple plan ve mekan tabloları arasında çoka çok bir ilişki bulunmaktadır. Çoka çok ilişkileri tanımlamakta mevcut tablolar yeterli olmadığından "PLANDETAY" adı verilen bir ilişki tablosu oluşturulmuştur. Bu tabloda plan ve mekan tablolarının ilgili kayıtlarının kimlik bilgileri saklanmaktadır. Bu sayede sistem bir planın bilgilerine ihtiyaç duyduğunda o plana ait gezi yerlerini de elde edebilmektedir.

Şehir tablosu ve ülke tablosu arasında bire çok ilişki bulunmaktadır. Bunun sebebi bir şehir sadece bir ülkeye ait olabilirken, bir ülkede birden çok şehir bulunabilmektedir. Bu ilişki şehir tablosunda ülke tablosuna ait kayıtların kimlik bilgisi değerlerinin saklanmasıyla sağlanmaktadır. Bu sayede sistem tarafından bir ülkenin bilgilerine ihtiyaç duyulduğunda o ülkeye ait şehirler de elde edilebilmektedir. Plan tablosu, şehir tablosu ile ilişkili olduğundan, şehir tablosunun ülke tablosu ile ilişkisi, plan tablosu ve ülke tablosu arasında dolaylı bir ilişki oluşturmaktadır. Böylelikle bir ülke için oluşturulmuş planlar ve bir planın oluşturulduğu ülke bilgileri rahatlıkla elde edilebilmektedir.

Veri tabanında en çok ilişki barındıran tablo mekan tablosudur. Gezi yerlerine ait bilgilerin saklandığı mekan tablosundaki ilk ilişki kategori tablosu ile olan ilişkidir. Kategori ve mekan tabloları arasında çoka çok bir ilişki bulunmaktadır. Yani bir kategoriye ait birden çok gezi yeri olabileceği gibi bir gezi yeri de birden çok kategoriye dahil olabilmektedir. Bu ilişki mekan tablosunda gezi yerinin ait olduğu kategorinin kimlik bilgisinin saklanmasıyla sağlanmaktadır. Bu sayede kullanıcının kategori seçimleri sonucunda seçilen kategoriye ait gezi yerleri listelenebilmektedir. Ayrıca kullanıcı tablosunun plan tablosu ile olan ilişkisi, plan tablosunun mekan tablosu ile ilişkisi ve mekan tablosunun da kategori tablosu ile ilişkisi sayesinde dolaylı olarak kullanıcı ve kategori tabloları ilişkili olmaktadır. Bu durum sistemin kullanıcının geçmiş hareketlerini değerlendirerek tercih ettiği kategoriler aracılığıyla verimli öneriler sunabilmesini sağlamaktadır.

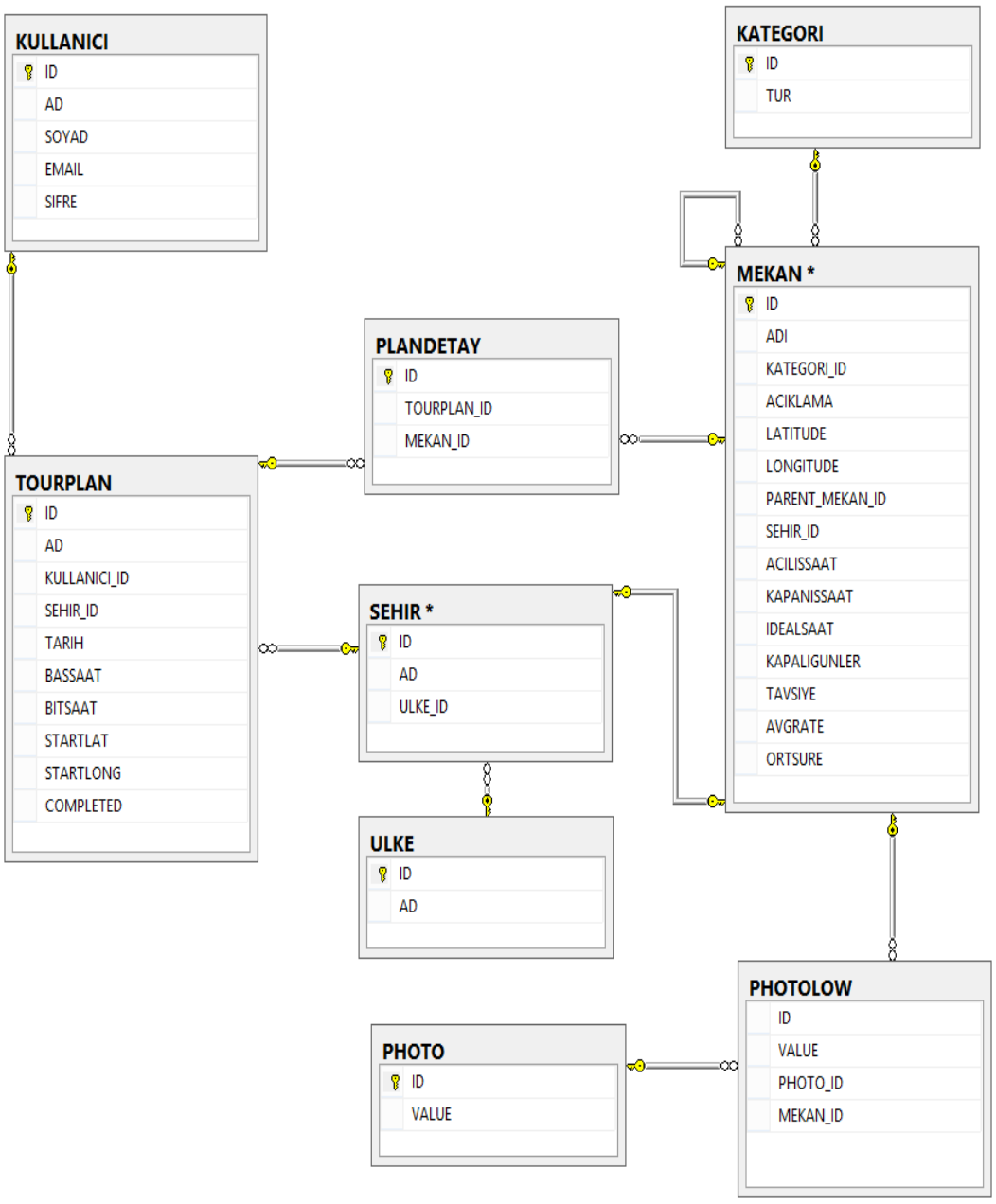
Mekan tablosunun ilişkili olduğu tablolardan birisi de kendisidir. Sistemin altyapısı geliştirilebilir birçok özelliğe uyarlanabilir şekilde tasarlanmıştır. Bu

özelliklerden biri de alt gezi yeri özelliğidir. Bu özelliğe göre bir gezi yeri birden fazla gezi yerini bünyesinde barındırabilmektedir. Örneğin Eskişehir ili için oluşturulan veri kümesinde bulunan Masal Şatosu adlı gezi yeri, Sazova Bilim Sanat ve Kültür Parkı adlı gezi yeri bünyesinde bulunmaktadır. Sistemin bu bilgiye sahip olması uygulamaya eklenebilecek ve bu özellikten türetilebilecek birçok özellik için zemin hazırlamaktadır. Bu ilişki mekan tablosunda alt gezi yeri olan kayıtta bünyesinde olduğu üst gezi yerinin kimlik bilgilerinin saklanması ile sağlanmaktadır.

Mekan tablosunda bulunan bir gezi yeri sadece bir şehrin bünyesinde olabildiğinden ve bir şehre ait birden fazla gezi yeri olabildiğinden şehir tablosu ve mekan tablosu arasında bire çok ilişki bulunmaktadır. Böylelikle kullanıcının plan oluşturma aşamasında seçtiği ülke ve şehre ait gezi yerleri kullanıcıya listelenebilmektedir. Bu ilişki mekan tablosundaki şehre ait kaydın kimlik bilgisi ile sağlanmaktadır.

Mekan tablosu ile düşük çözünürlüklü fotoğrafların bulunduğu "PHOTOLOW" tablosu arasında bire çok ilişki bulunmaktadır. Böylelikle bir gezi yerine ait düşük çözünürlüklü fotoğraflar kullanıcıya sunulabilmektedir. Bu ilişki "PHOTOLOW" tablosunda bulunan ve mekan tablosundaki gezi yerine ait kaydın kimlik bilgisi ile sağlanmaktadır.

Orijinal fotoğraf verilerinin bulunduğu "PHOTO" tablosu ile düşük çözünürlüklü kopyalarının bulunduğu "PHOTOLOW" tablosu arasında bire bir ilişki bulunmaktadır. Bu ilişki "PHOTOLOW" tablosundaki orijinal fotoğrafa ait kaydın kimlik bilgisi ile sağlanmaktadır. Bu sayede bir gezi yerine ait düşük çözünürlüklü fotoğraflar listelendiğinde, kullanıcının aynı fotoğrafın yüksek çözünürlüklü halini görmek istemesi durumunda bu bilgi kolaylıkla kullanıcıya sağlanabilmektedir.



Şekil 3.2. Veri Tabanı Diyagramı

Veritabanı tabloları, tablolardaki alanlar ve tablolar arasındaki ilişkiler Şekil 3.2'de belirtildiği gibidir. Bunlardan sol tarafında anahtar işareti bulunan alanlar bulundukları tablolar için birincil anahtar olup tablodaki satırların tekilliğini sağlamaktadır. Tablolar arasındaki ilişkiler birincil anahtarlar sayesinde sağlanmaktadır.

3.2.Sunucu Programı Tasarımı

Sistemin sunucu programı, Java programlama dili ile RESTful bir servis programı olarak geliştirilmiştir ve Jetty Web Sunucusu ile çalışmaktadır. Sistemin gerektirdiği tüm işlemler ve hesaplamalar sunucu programında gerçekleştirilmektedir. Mobil uygulama aracılığıyla elde edilen kullanıcı istekleri sunucu programında gerekli kısımlarda veri tabanı ile iletişime geçilerek işlenmekte ve kullanıcıya sonuç sunulmaktadır.

Sunucu programı ve mobil uygulama arasındaki veri transferi JSON formatıyla gerçekleşmektedir. JSON, veri transferinde nesne kullanılmasına izin vermektedir. Ancak istemci ve sunucu uygulamalarında aynı nesne tanımlarının olması gerekmektedir. Sistemde aynı sınıfların tekrar tanımlanmaması için ayrı bir proje oluşturulmuş ve sınıf tanımlamaları bu projede yapılmıştır. Sınıf tanımlamalarının yapıldığı bu proje bir kütüphaneye dönüştürülerek sunucu ve mobil uygulamalarda aynı nesneler kullanılabilmektedir.

Sunucu programı 3 katmandan oluşmaktadır. Katmanlar, projede farklı işlevi olan sınıfların ayrılmasıyla ve mobil uygulama ile kullanıcıdan gelen isteklerin işlenme aşamalarına göre birleşmeleriyle oluşmaktadır.

Sunucu programının üst katmanında mobil uygulamadan gelen istekleri elde eden ve mobil uygulama için cevap döndüren metotları içeren sınıflar bulunmaktadır. Program, mobil uygulama ile HTTP protokolleri vasıtasıyla haberleşmektedir. Sunucu programı, gelen istekleri gerekli hesaplamaların yapılması için bir alt katmandaki görevli alanlara iletmektedir.

Sunucu programının orta katmanında sisteme ait tüm hesaplama işlemleri yapılmaktadır. Üst katmanda kullanıcıdan elde edilen istekler bu katmanda aynı katman içerisinde görevli alanlara yönlendirilmektedir. Eğer kullanıcıdan gelen isteğin cevaplanabilmesi için veri tabanı bağlantısı gerekmiyorsa, gerekli işlemlerden sonra isteğin cevabı oluşturulmakta ve üst katmana iletilmektedir. Eğer isteğin cevaplanabilmesi için veri tabanı bağlantısı gerekmekteyse alt katmanla iletişime geçilmekte ve ilgili metotlar aracılığıyla gerekli bilgiler elde edilmekte veya veri tabanında yapılması gereken değişiklikler iletilmektedir. Veri tabanı

değişiklikleri sonucunda mobil uygulamaya işlemin başarı durumu hakkında bilgi veren cevaplar dönmektedir.

Sunucu programının alt katmanında veri tabanındaki varlıkların eşleniği olan varlık sınıfları ve veri tabanı işlemlerini gerçekleştiren sınıflar bulunmaktadır. Sunucu programı veri tabanı ile Hibernate aracı ile iletişim kurmaktadır. Bunun için sunucu programında veri tabanındaki tabloların eşleşebildiği varlık sınıflarının olması gerekmektedir. Varlık sınıflarında eşleştikleri veri tabanı tablolarında bulunan alanlar, türleriyle birlikte tanımlanmaktadır. Bu sayede veri tabanı işlemlerini gerçekleştiren sınıflar, veri tabanından istenilen kayıtlar için nesneler halinde cevaplar alınmakta ve veri tabanında yapılmak istenen değişiklikler varlık nesneleri ile kolaylıkla uygulanabilmektedir. Sonuçlar gerekli işlemlerin yapılması için orta katmana gönderilmektedir.

3.3.Mobil Uygulama Tasarımı

Sistemin son parçası olan mobil uygulama kısmı kullanıcıyla etkileşim halinde olan bir ara yüzdür. Uygulama kullanıcının isteklerini toplayarak sunucu programına iletmektedir ve sunucu programından elde edilen sonuç bilgileri işlenerek tekrar kullanıcıya gösterilmektedir. Uygulamanın doğrudan veri tabanıyla ilişki halinde olmaması gerekmektedir. Bunun için sadece sunucu programı ile iletişim kurmaktadır. Uygulama sınırlı kapasitedeki mobil cihazlar üzerinde çalıştığından olabildiğince az işlem yapması gerekmektedir. Ancak sunucuyla iletişimin de maliyetli olduğu düşünülerek veri tabanı ile ilişkili olmayan basit işlemlerin uygulama içerisinde yapılması uygun bulunmuştur.

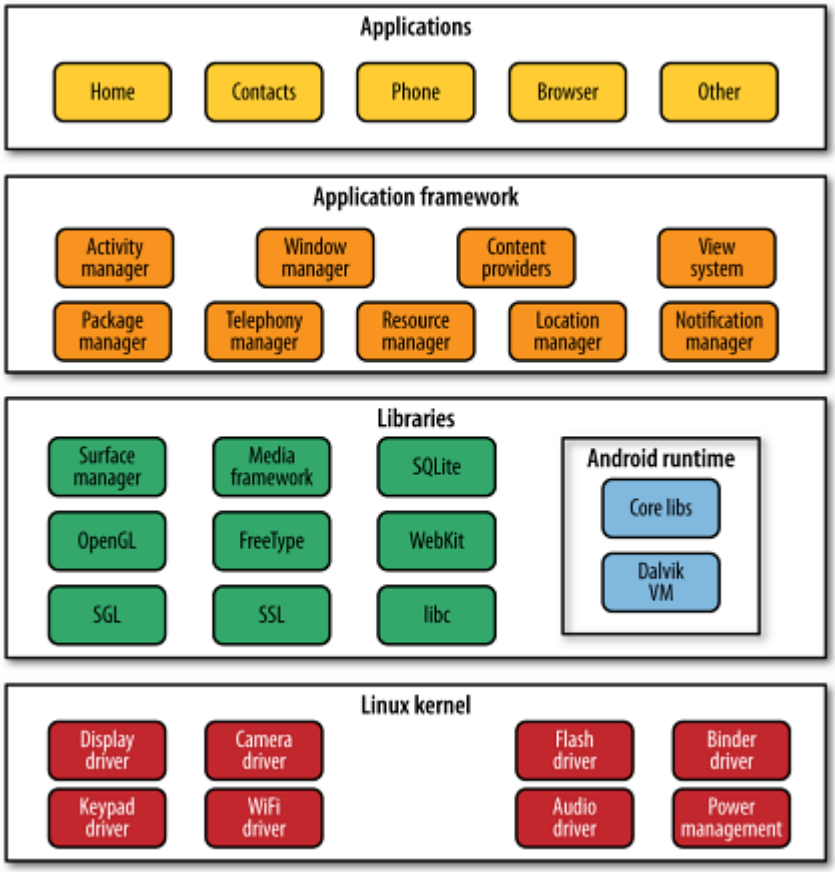
Mobil uygulama kendi içerisinde 2 katmandan oluşmaktadır. Üst katmanda kullanıcı arayüzünün tasarlandığı XML formatında dosyalar bulunmaktadır. Bu dosyalar uygulamadaki her sayfa için ayrı olarak oluşturulmakta ve sayfa tasarımları yapılmaktadır. Alt katmanda arayüz dosyaları ile eşleşen ve Java programlama dili ile yazılmış olan aktivite sınıfları bulunmaktadır. Bu sınıflar arayüz kısmında elde edilen kullanıcı isteklerini yorumlayarak cevap döndürürler. Kullanıcıdan gelen basit istekler aktivite sınıflarında işlenirken, veri tabanı ile iletişim gerektiren veya basit olmayan işlemler mobil cihazların sınırlı kaynak

kapasiteleri düşünüldüğünden ve kullanıcıların doğrudan veri tabanı ile doğrudan etkileşimi uygun olmadığından sunucu programında hesaplanmaktadır.

4. TEZ GENELİNDE KULLANILAN TEKNOLOJİLER

4.1.Android İşletim Sistemi

Android, bir mobil işletim sistemini de içinde barındıran açık kaynak kodlu bir yazılım yığınıdır. Bileşenleri yığın oluşturacak şekilde tasarlanmıştır. Bu yığının en alt katmanında Linux Çekirdeği bulunmakla beraber en üst katmanında ise Android uygulamaları bulunmaktadır [14].



Şekil 4.1. Android İşletim Sistemi Mimari Katmanları

Android güvenlik, bellek yönetimi, işlem yönetimi, ağ işlemleri ve sürücü modelleri gibi çekirdek sistemlerinin işletilmesi için Linux Çekirdeği'ni kullanır. Linux'un tercih edilmesinde açık kaynak olması ve güvenilirliğinin yanında birçok cihazla uyumlu olması etkili olmuştur [15].

Linux Çekirdeği katmanının bir üstünde çekirdek kütüphanelerinin, uygulamaların kullandığı diğer ana kütüphanelerin ve Dalvik adı verilen sanal

makine bulunur. Temelde JVM (Java Sanal Makinesi) ile aynı işlevi olan Dalvik mobil platformlara uygun olması için geliştirilmiştir.

Üstteki iki katmanda ise uygulamalar ve uygulamaların kullandığı kütüphaneler ve servisler bulunmaktadır. Bu katmanlarda hem uygulamaların geliştirilme aşamasında arkaplanda Java programlama dili hem de uygulamaların kullandığı kütüphanelerin Java kütüphaneleri olması açısından geliştiricilerine alışıldık ve etkili bir ortam sağlamaktadır.

4.2. Google Maps

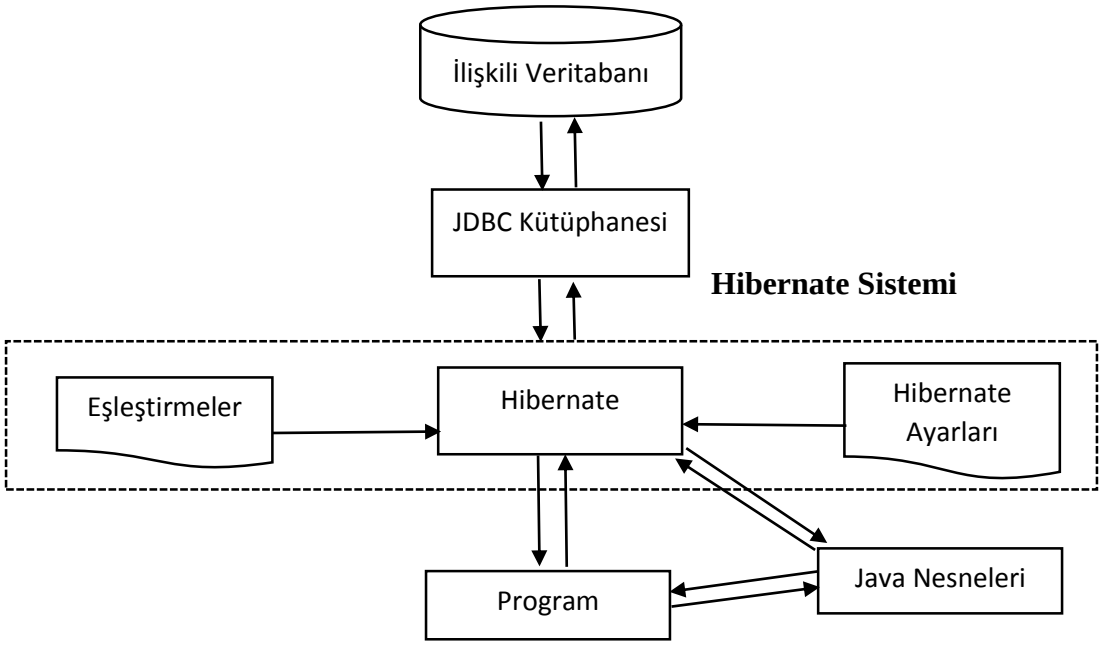
Google Maps içeriği ilk olarak Lars ve Jens Rasmussen adında iki Danimarkalı kardeş tarafından kendi kurdukları harita çözümleri sunan bir şirkette geliştirilmiştir. 2004 yılının Ekim ayında Google bünyesine katılan şirket çalışmalara devam etmiş ve 2005 yılının Şubat ayında Google tarafından Google Maps olarak tanıtılmıştır.

Google Maps için bir API(Application Programming Interface) geliştirilmeden önce bazı yazılımcıların kendi web sitelerinde bir yolunu bularak Google Maps kullanmaları sonucu, Google tarafından 2005 yılının Haziran ayında ilk API kullanıcılara sunulmuştur.

Google Maps yapı olarak incelendiğinde HTML, CSS ve JavaScript olmak üzere 3 programlama dilinin beraber çalışması sonucu oluştuğu görülmektedir. Google Maps genel olarak harita katmanlarından oluşmaktadır. Haritalar, arka planda Ajax çağrıları ile elde edilen ve HTML sayfası içinde etiketleri içine eklenen resimlerden oluşmaktadır. Harita üzerinde yönlendirilme yapıldığında yeni koordinat ve yakınlaştırma bilgileri API aracılığıyla Ajax çağrıları olarak gönderilmekte ve bu bilgilere göre yeni resimler cevap olarak dönmektedir. API temel olarak harita ile ilgili geliştiricinin kullanabileceği ve haritanın nasıl davranması gerektiğini beliryenen JavaScript sınıf ve metotlarından oluşmaktadır [16].

4.3.Hibernate

Bir programlama dilinde veritabanıyla ilişkili bir yazılım geliştirilirken, sıklıkla veritabanıyla iletişim kurulmaktadır. Bu iletişim yazılımın geliştirildiği platforma bağlı olarak farklı kütüphaneler aracılığıyla gerçekleştirilmektedir (JDBC, OLEDB vb.). Nesne tabanlı programlamanın yaygınlaşması ile birlikte ilişkili veritabanlarının programlardaki nesnelerle eşleştirilmesi fikrine dayanan Hibernate aracı geliştirilmiştir.

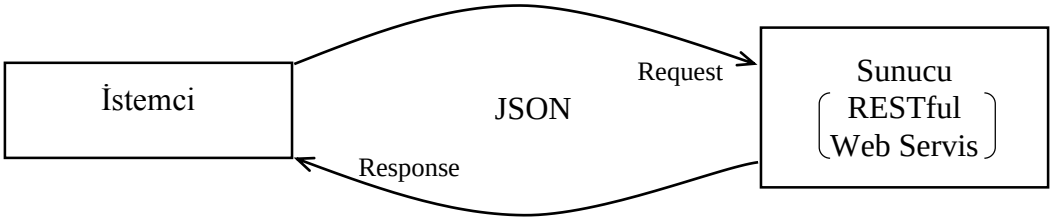


Şekil 4.2. Hibernate İşleyiş Şeması

Hibernate, Java platformunda geliştirilmiş bir ORM (Object-Relational Mapping) aracıdır. Temel olarak veritabanındaki varlıkları Java nesneleri ile ilişkilendirmektedir. Bu sayede veritabanı üzerinde yapılmak istenen değişiklikler Java nesnelerinde uygulanarak daha kolay ve daha kontrollü işlemler yapılabilmektedir. Hibernate aracının .NET platformunda kullanılmak üzere NHibernate adında başka bir türevi de mevcuttur [17, 18, 19].

4.4.RESTful Web Servisleri ve JSON

REST (Representational State Transfer), istemci-sunucu veri iletişimini HTTP protokolü üzerinden sağlamakta olan basit ve esnek bir mimaridir. REST mimarisini kullanan servislere de RESTful Servisler denmektedir. RESTful servisler, HTTP protokollerini kullandıkları için ve verileri XML formatında iletme zorunlulukları olmadığı için diğer web servislere göre daha basittir ve daha az boyutlarda transfer yapmaktadırlar. REST mimarisinin önemli özellikleri arasında platform bağımsız olması, programlama dillerinden bağımsız olması (istemci ve sunucu farklı programlama dillerinde yazılmış ve platformlarda çalışıyor olabilir), esnek ve kolay genişletilebilir olması gösterilebilir. Mimari içerisinde bir güvenlik yapısı barındırmamakla birlikte HTTP protokolünün kullandığı güvenlik mekanizmalarını rahatlıkla kullanabilmektedir [20, 21].



Şekil 4.3. JSON Formatı ile Haberleşme Sağlayan İstemci-Sunucu Modeli

JSON (JavaScript Object Notation) bir veri tanımlama formatıdır. Veriler bu şekilde saklanabilir ve transfer edilebilir. Çoğunlukla bir sunucu ve uygulama arasındaki veri transferini gerçekleştirme amaçlı kullanılır. JavaScript programlama dilinden türetilmiş olmasına rağmen, kullanım açısından programlama dillerinden bağımsızdır. Veri tanımında isim-değer çiftleri ve bu çiftlerden oluşan listeler kullanılır. Bütün modern programlama dillerinde bulunan yapıları kullandığından kullanım alanı geniştir. Veri tanımlamalarında toplam boyut diğer formatlara göre daha az olduğundan tercih edilmektedir [22].

5. ROTA HESAPLAMA PROBLEMİNDE KULLANILAN ARAMA ALGORİTMALARI

Geliştirilen mobil gezgin uygulamasının çalışması esnasında kullandığı en önemli özelliklerden birisi de rota hesaplamadır. Uygulama, ilk rotanın hesaplanmasının ardından kullanıcı seçimlerine bağlı olarak rotanın güncellenmesine imkân sağlamaktadır. Gezi esnasında birden fazla rota güncellemesi mümkün olduğundan rota hesaplama işleminin önemi oldukça fazladır. Uygulamanın mobil platformda hizmet vermesi ve kullanılış amacı gereği hareket halindeki kullanıcılar için de verimli bir şekilde çalışabilmesi için rota hesaplanması gibi maliyeti fazla olan bir işin doğru sonuçları vermesinin yanında performanslı çalışması da büyük önem taşımaktadır. Bu sebepten rota hesaplanması probleminde kullanılabilecek algoritmalar incelenmiş ve mobil gezgin uygulaması için uygunlukları araştırılmıştır. En kısa rotanın hesaplanması problemi yapısı itibariyle Gezgin-Satıcı Problemi'ne benzemektedir.

Gezgin-Satıcı Problemi: Bir satıcı bir şehir kümesindeki tüm şehirlerde satış yapmak istemektedir. Bir şehre birden fazla uğramadan ve en az mesafeyi kat ederek turunu tamamlamak istemektedir. Satıcı için en uygun rotanın bulunması problemine Gezgin-Satıcı Problemi denilmektedir. Problem üzerine çalışmalar ilk olarak 1930'lu yıllarda matematikçi Karl Menger tarafından yapılmıştır [23].

Gezgin-Satıcı Problemi'nde veri kümesi $G = (N, A)$ şeklinde gösterimi olan bir ağırlıklı tam çizge olarak gösterilebilmektedir. Bu gösterimde N, çizgedeki nokta kümesini ve A ise noktaları birbirine bağlayan kenarlar kümesini temsil etmektedir. Her bir $(i, j) \in A$ kenarı için d_{ij} şeklinde gösterilen ağırlık i ve j noktaları arasındaki uzaklığı temsil etmektedir. Gezgin-Satıcı Problemi ise bu çizgedeki en kısa Hamilton Devresi'ni bulmayı amaçlamaktadır. Hamilton Devresi ise bir çizgedeki tüm noktaların sadece bir kere ziyaret edilmesiyle sağlanmaktadır [24].

1950'li yılların ortalarından itibaren çözüm metotları türetilmeye başlanmıştır. 1972 yılında Richard M. Karp tarafından Hamilton Devresi'nin NP-Hard bir problem olduğunun ispatlanmasıyla Gezgin-Satıcı Problemi'nin de NP-Hard bir problem olduğu ispatlanmıştır [25]. İlk olarak 1954'te Dantzig ve ark.

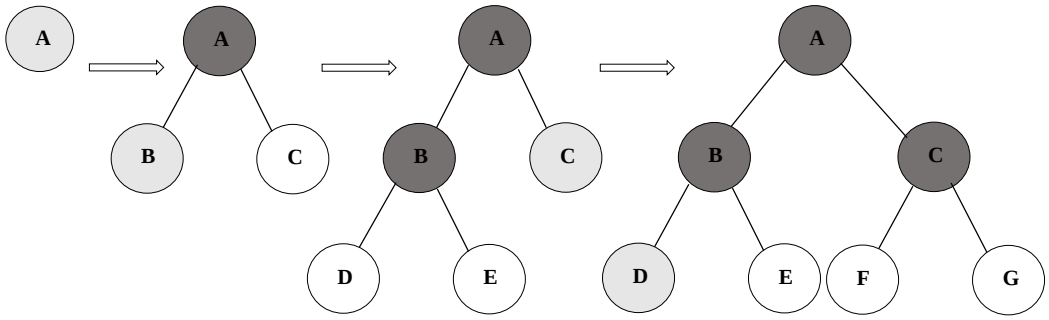
tarafından 49 şehir için çözümü türetilen problemin ilerleyen yıllarda artan şehir sayılarıyla farklı metotlar kullanılarak yeni çözümleri türetilmiştir [26]. Bu çözümlerden en yakın olanı Applegate ve ark. tarafından türetilmiş ve 85.900 elemanlı şehir kümesi için en uygun sonucu vermektedir [27].

5.1.Arama Algoritmaları

Bir başlangıç noktasından varış noktasına olan en kısa yolu bulmakta kullanılan arama algoritmaları; uninformed (kör) ve informed olmak üzere ikiye ayrılır. Uninformed arama algoritmalarında bulunulan noktanın varış noktasıyla ilişkisi bilinmemekte sadece ulaşabileceği komşu noktaların bulunulan noktaya uzaklığı bilinmektedir. Bu da bir sonraki nokta seçiminin toplam yola olan etkisi hakkında yeterli bilgiyi sağlamamaktadır. Informed arama algoritmalarında sezgisel metotlar kullanılarak bir sonraki noktanın seçilmesinde o noktaya kadar olan toplam maliyetle birlikte varış noktasına kadar olan yolun tahmini maliyeti de etkili olmaktadır. Bu açıdan informed algoritmalar daha kısa sürede sonuçlar verebilmekte ve daha etkili çalışabilmektedir [28].

Brute-Force Search Algoritması: Brute-Force algoritmalar olası tüm seçenekleri deneyerek en uygununu bulmak üzere tasarlanmaktadırlar. Bu metot kesin sonucu vermekle beraber artan değişken sayılarında çalışma zamanı fonksiyonel olarak artmaktadır. Bu sebepten çok değişkeni olan problemlerde sıklıkla tercih edilmemektedir.

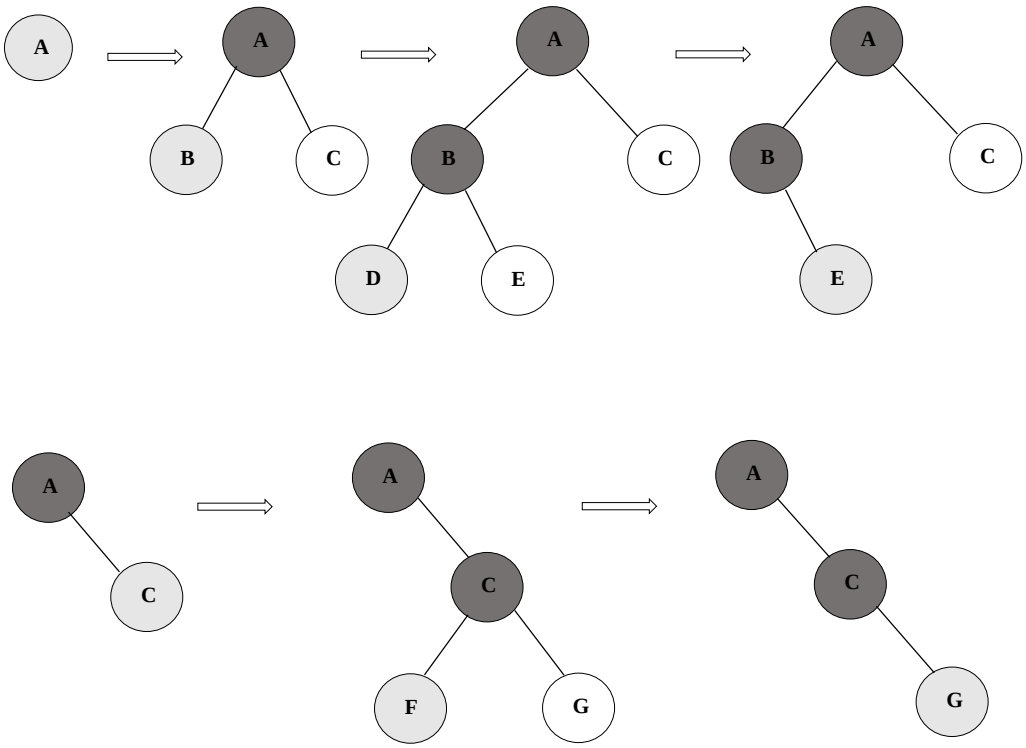
Breadth-First Search Algoritması: Breadth-First Search Algoritması'nın temelindeki strateji önce kaynak noktasının sonra kaynak noktasının komşusu olan noktaların daha sonra komşuların komşularının işlenmesine dayanmaktadır. Noktalar ağaç yapısında düşünülürse her adımda bir alt derinlikteki tüm noktalar işlenmektedir. Kör algoritmalardandır. Bir sonraki noktanın seçiminde sadece önceki noktanın komşusu olması yeterlidir. Programlama açısından incelenecek olursa noktalar "ilk giren ilk çıkar" kuralıyla çalışan bir kuyrukta başlangıç noktasından başlanarak ve ardından komşuları gelecek şekilde saklanmaktadır ve bu sıraya göre işlenmektedir.



Şekil 5.1. Breadth-First Search Algoritması

Şekil 5.1'de belirtilen örnekte A noktasından başlatılan aramada öncelikle A noktası işlenmektedir. Daha sonra A noktasının komşuları olan B ve C noktaları, daha sonra ise onların komşuları işlenmektedir. Bu işlem varılacak nokta işlenene kadar devam etmektedir.

Depth-First Search Algoritması: Depth-First Search Algoritması'nın çalışma prensibinde noktalar ağaç yapısında düşünülürse kaynak noktasından başlandığında her adımda bir alt derinlikteki ilk komşu noktanın işlenmesi bulunmaktadır. En alt derinliğe ulaşıldığında bir yandaki komşu nokta üzerinden aynı işlem gerçekleştirilmektedir. Bu işlem varış noktasına ulaşılıncaya kadar devam etmektedir. Kör algoritmalardandır. Seçilecek noktanın belirlenmesinde o noktanın varış noktasına uzaklığı veya rotaya uygunluğunun önemi bulunmamaktadır. Programlama açısından incelenecek olduğunda "son giren ilk çıkar" kuralıyla çalışan bir yığına öncelikle başlangıç noktası daha sonra en alt derinliğe ulaşana kadar tüm derinliklerdeki ilk komşu noktalar sırayla saklanmaktadır. Daha sonra son giren nokta incelenerek eğer işlenmemiş komşusu varsa komşu noktalar bu stratejiyle yığına eklenmekte yoksa yığından çıkarılmaktadır.



Şekil 5.2. Depth-First Search Algoritması

Dijkstra Algoritması: 1959 yılında Edsger W. Dijkstra tarafından geliştirilen bir en kısa yol bulma algoritmasıdır. Kör algoritmalardandır. Başlangıç noktasından herhangi bir n noktasına olan en kısa yolun değerini veren $g(n)$ fonksiyonunu kullanmaktadır. Başlangıç noktası için $g(n)$ fonksiyonunun değeri 0'dır. Algoritmanın çalışma prensibinde başlangıç noktasından itibaren varış noktası ve komşuları işlenene kadar nokta kümesindeki tüm noktaların $g(n)$ fonksiyonları hesaplanması bulunmaktadır.

Dijkstra algoritmasında noktaların işlenme metodu Breadth-First Search Algoritması ile aynıdır. Noktalar ağaç yapısında düşünülecek olursa her adımda bir alt derinlikteki komşular işlenmektedir. Bir n noktası için hesaplanan $g(n)$ fonksiyonu değeri n noktasına başka bir nokta üzerinden ulaşıldığında daha az bir değer veriyorsa n noktasının $g(n)$ değeri güncellenmektedir. Böylelikle algoritmanın çalışması sonlandığında, başlangıç noktasından sadece varış noktasına olan değil aynı zamanda algoritmanın çalışması esnasında işlenen tüm noktalara olan en kısa yol değerleri bilinmektedir [29, 30].

Best-First Search Algoritmaları: Best-First Search Algoritmaları Informed Search algoritmalarındandır. Yani yol bulma metotlarında seçilecek bir sonraki noktanın seçiminde o noktanın varış noktasına uzaklığı değerlendirilmektedir. Algoritmaların genel çalışma prensibinde nokta kümesindeki her nokta için bir değerlendirme fonksiyonu hesaplanmaktadır. Noktalar $f(n)$ şeklinde gösterimi olan değerlendirme fonksiyonlarına göre sıralanarak bir sonraki noktanın seçiminde fonksiyon değerleri göz önünde bulundurulmaktadır.

Best-First Search Algoritma grubuna ait farklı algoritmalarda değerlendirme fonksiyonunun içeriği de farklılıklar göstermektedir. Değerlendirme fonksiyonlarının önemli bileşenlerinden biri de sezgisel fonksiyondur. Gösterimi $h(n)$ şeklinde olan sezgisel fonksiyon standart yol bulma problemlerinde herhangi bir n noktasının varış noktasına olan tahmini uzaklığını göstermektedir. Informed algoritmaların en önemli özelliği noktaların seçiminde, varış noktasına olan uzaklıklarının tahmini değerlerinin göz önünde bulundurulması olduğundan bu algoritmalara sezgisel algoritmalar da denilmektedir.

A* (A-star) Algoritması: En bilinen Best-First Search algoritmalarındandır. Diğer Best-First Search algoritmalarında olduğu gibi A* algoritmasında da bir değerlendirme fonksiyonu kullanılmaktadır ve nokta kümesindeki noktalar bu değerlendirme fonksiyonu değerlerine göre sıralanmaktadırlar.

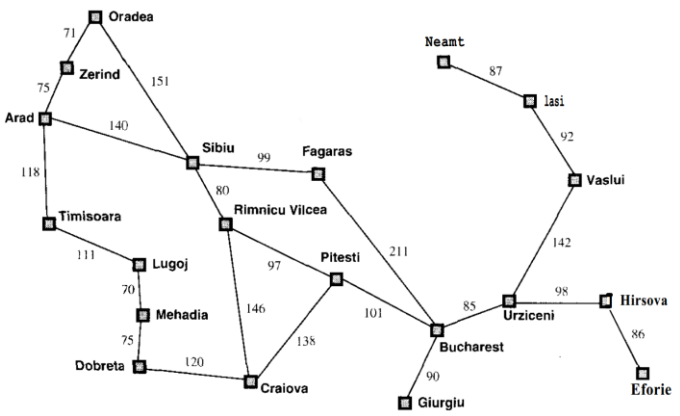
$$f(n) = g(n) + h(n) \tag{5.1}$$

A* algoritmasının kullandığı değerlendirme fonksiyonunda sezgisel fonksiyon olan $h(n)$ fonksiyonunun yanında bir g(n) fonksiyonu da kullanmaktadır. Dijkstra Algoritması'nda da kullanılan bu fonksiyon herhangi bir n noktası için başlangıç noktası ile n noktası arasındaki var olan en az maliyetli yolun değerini vermektedir. Bunun yanında A* algoritması sezgisel bir algoritma olduğundan değerlendirme fonksiyonu hesaplanırken sezgisel bir fonksiyondan da faydalanmaktadır. $h(n)$ şeklinde gösterimi olan sezgisel fonksiyon normalde herhangi bir n noktasının varış noktasına olan tahmini uzaklığını göstermekle

beraber farklı özel durumlarda tahmini uzaklıkla birlikte başka parametreler de barındırabilmektedir. A* algoritmasının değerlendirme fonksiyonunun genel gösterimi Denklem (5.1)'de gösterildiği şekildedir.

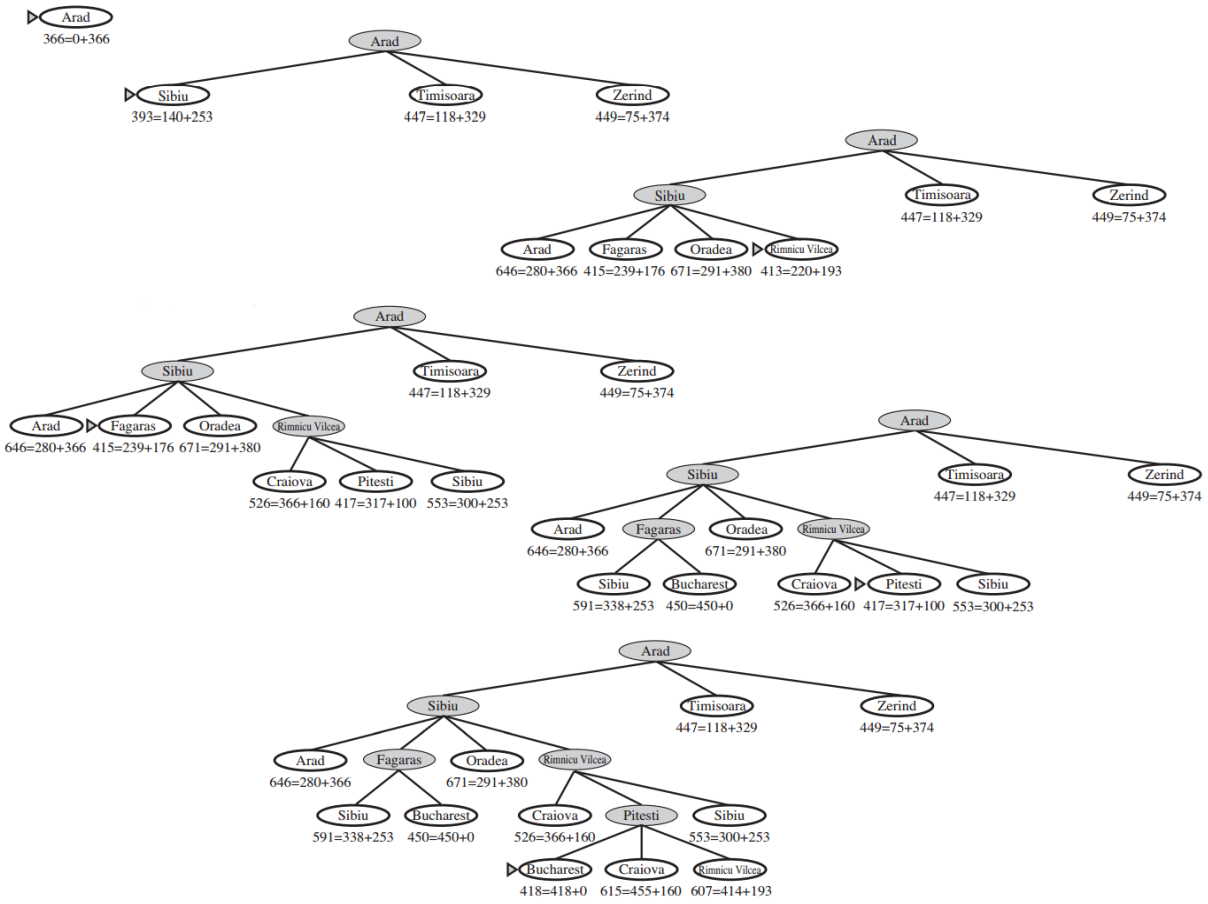
A* algoritması kabul edilebilir bir sezgisel fonksiyon kullanıldığında optimal sonuç veren bir en kısa yol bulma algoritmasıdır. Bu da sezgisel fonksiyonun iyimser bir yapıda olması ile sağlanabilmektedir. Örneğin standart bir kısa yol bulma probleminde $h(n)$ fonksiyonu n noktası ile varış noktası arasındaki kuş bakışı mesafe olarak hesaplandığında gerçek değerden daha kısa sonucu verdiğinden bu metotla hesaplanan $h(n)$ fonksiyonu iyimser yapıdadır. $g(n)$ fonksiyonu da başlangıç noktasından n noktasına olan gerçek uzaklık değerini verdiğinden, $f(n)$ fonksiyonu her zaman n noktasını kapsayan rotanın gerçek mesafe değerinden fazla bir değer vermemektedir.

A* algoritmasının çalışma prensibinde o anda ulaşılan ve ulaşılabilecek olan noktalar $f(n)$ fonksiyonu değerlerine göre sıralı bir şekilde bir yapıda saklanması ve bu sıraya göre değerlendirilmesi bulunmaktadır. Varış noktası bu sıralı yapının en üstünde olacak şekilde sıralanana kadar algoritma çalışmaktadır. Bu durumda varış noktasına olan en kısa mesafeyi sağlayacak rota bulunmuş olmaktadır.



Straight-line distance to Bucharest	
Arad	**366**
Bucharest	0
Craiova	**160**
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	**77**
Hirsova	151
Iasi	**226**
Lugoj	244
Mehadia	241
Neamt	**234**
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

(a)



(b)

Şekil 5.3. a) Romanya haritasının bir kısmı ve b) A* Algoritması

Şekil 5.3'te bir örnek üzerinde A* algoritmasının gerçek bir harita üzerinde çalışma prensibi gösterilmektedir. Örnekte Arad şehri başlangıç noktası kabul edilerek Bükreş şehrine olan en kısa yol bulma probleminin çözümünde A* algoritması kullanılmaktadır. Şekil 5.3a'da haritanın temsili halinde şehirlerin arası uzaklıklar çizgilerin üzerinde belirtilmektedir. Yan tarafındaki tabloda ise şehirlerin varış noktasına olan Bükreş şehrine kuş bakışı mesafeleri verilmektedir.

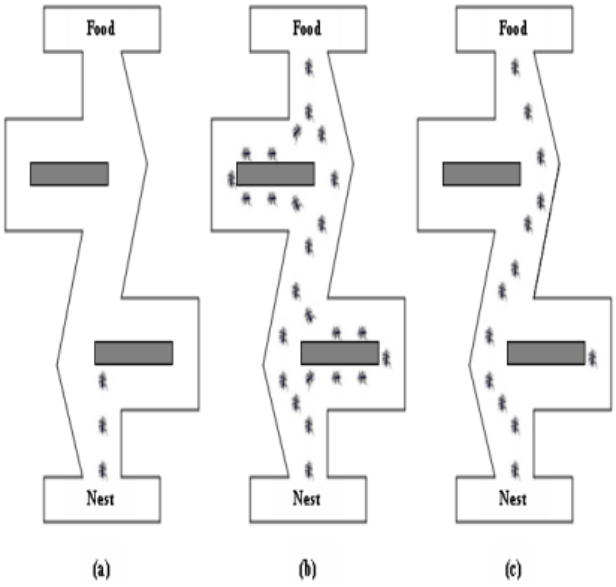
Şekil 5.3b'de A* algoritmasının çalışma prensibi verilen örnek üzerinde uygulanmaktadır. Arad şehri başlangıç noktası olarak kabul edilmekte ve bu şehir için $g(n)$ fonksiyonu 0 değerini almaktadır. Varış noktasına kuş bakışı uzaklık değerini veren $h(n)$ fonksiyonu bu şehir için 366 km. olduğundan $f(n)$ fonksiyonu 0+366=366 değerini vermektedir. Bu aşamada Arad şehrinin komşularının değerlendirme fonksiyonları hesaplanmaktadır. Hesaplanan değerlendirme

fonksiyonlarının değerlerine göre sıralı bir yapıda saklanan şehirlerden en düşük $f(n)$ fonksiyonuna sahip olan tercih edilerek rota sürdürülmektedir. Bir noktanın tüm komşularının değerlendirme fonksiyonları hesaplandığında bulunduğu sıralı yapıdan çıkarılmaktadır. Varış noktasına rastlandığında noktanın değerlendirme fonksiyonu ulaşılan noktaların değerlendirme fonksiyonları ile karşılaştırılmaktadır. Örneğin figürde sondan bir önceki adımda varış şekli olan Bükreş'e ulaşılmasına rağmen Pitesti şehrinin değerlendirme fonksiyonu daha az bir değere sahip olduğundan Bükreş sıralı yapıda en üste çıkamamakta ve gelinen rotanın en kısa rota olduğu anlaşılamamaktadır. Sonraki adımda Pitesti şehrinden Bükreş'e ulaşıldığında Bükreş sıralı yapının en üstünde bulunmakta ve o anda ulaşılmış olan noktalardan daha kısa bir şekilde varış noktasına ulaşılamayacağı anlaşılmaktadır. Algoritma bu esnada çalışmayı sonlandırır ve rotayı hesaplamış olur.

A* algoritması bir en kısa yol bulma algoritması olduğundan genellikle rota hesaplamalarında, ağ yönlendirmelerinde ve bilgisayar oyunlarının haritaları için yol bulma problemlerinde kullanılmaktadır [31, 32, 33, 34].

Karınca Kolonisi Optimizasyonu: 1992 yılında Marco Dorigo tarafından doktora tezi olarak sunulan Karınca Kolonisi Optimizasyonu (KKO), gerçek karıncaların davranışlarını örnek alarak Gezgin-Satıcı Problemi'ne çözüm sunmayı amaçlamaktadır.

Karıncalar sosyal canlılardır ve insanlar gibi gelişmiş türlerin kurduğu görsel ve işitsel iletişimden çok feromon adı verilen kimyasallarla iletişim kurarlar. Birbirlerini iz feromonu adı verilen özel feromonlar sayesinde takip etmenin karıncaların sosyal yaşantısında önemli bir payı vardır. Bu sayede yemek kaynaklarına giden yol hakkında diğer karıncaları bilgilendirebilmektedirler [35].



Şekil 5.4. Karınca kolonisinin yemek kaynağı ile yuva arasındaki yol haritası

Şekil 5.4a durumunda karıncaların yiyecek arayışı gösterilmektedir. Karıncalar Şekil 5.4b durumunda yiyecek kaynağını bulduktan sonra yuvaya eşit olasılıklarla farklı rotalarda yolculuk yapmaktadırlar. Bu esnada yola iz feromonlarından bırakmaktadırlar. Bu feromonlar uçucu olduklarından kısa yolda bulunan feromonların yoğunluğu daha fazla olmaktadır. Karıncalar Şekil 5.4c durumunda ise yoğun feromonları takip ederek kısa yoldan yiyecek kaynağına ulaşmaktadırlar. Ancak az miktarda karınca feromonların yoğun olduğu rotaları tercih etmeyerek rasgele rotalarda yolculuk yapmaktadır. Böylece daha kısa rotaları bulmayı hedeflemektedirler. Bu durum belirli bir süre devam ettikten sonra feromonların en yoğun olduğu rota en kısa yol olarak bulunmaktadır.

$$j = \begin{cases} \arg\max \left\{(\tau_{iu})^{\alpha}.(\eta_{iu})^{\beta}\right\} & q \leq q_0 \; ise \\ p_{ij}^{k} & değilse \end{cases} \tag{5.2}$$

$$p_{ij}^{k} = \begin{cases} \frac{(\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}}{\sum_{l \in J_i^k}(\tau_{il})^{\alpha}.(\eta_{il})^{\beta}} & j \in J_i^k \; ise \\ 0 & değilse \end{cases} \tag{5.3}$$

J_i^k: k karıncası için i noktasında iken uğrayabileceği noktalar kümesi

τ_{ij}: i ve j noktaları arasındaki feromon izi

η_{ij}: görünürlük değeri = $\frac{1}{d_{ij}}$ d_{ij}: i ve j noktaları arası uzaklık

α ve β: feromon izi ve görünürlük için ayar parametreleridir.

Karıncaların bu davranışı yol bulma algoritmaları için uyarlandığında karıncaların bir sonraki nokta seçimi için Denklem (5.2) uygulanmaktadır. Bu durum rasgele seçilen q değerinin başlangıçta belirlenen q_0 değerinden düşük olduğu durumlarda gerçekleşmektedir. Bunun sebebi yeni yerler keşfeden karıncalar için imkân sağlamaktır. Ancak yeni yerlerin keşfi sırasında da noktaların olasılık değerleri hesaplanmaktadır ve nokta seçimi bu olasılık değerine göre yapılmaktadır. Herhangi bir k karıncasının i noktasında iken seçeceği j noktasının olasılık değerinin hesaplanması için Denklem (5.3)'te belirtilen formül kullanılmaktadır [36].

Her turda m tane karıncadan hedefe ulaşanlar tarafından feromon değerleri güncellenmektedir. i ve j noktalarını birleştiren yolun feromon oranı değerinin formülü Denklem (5.4)'te gösterilmektedir.

$$\tau_{ij} = (1 - p).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{5.4}$$

p: feromonların buharlaşma oranı

m: toplam karınca sayısı

$\Delta\tau_{ij}^{k}$: i ve j noktaları arasındaki yolda k karıncası tarafından bırakılan feromon değeri

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L^k} & k\ karıncası\ i\ ve\ j\ noktaları\ arası\ yolu\ kullandıysa \\ 0 & kullanmadıysa \end{cases} \tag{5.5}$$

Q: başlangıçta belirlenen sabit değer

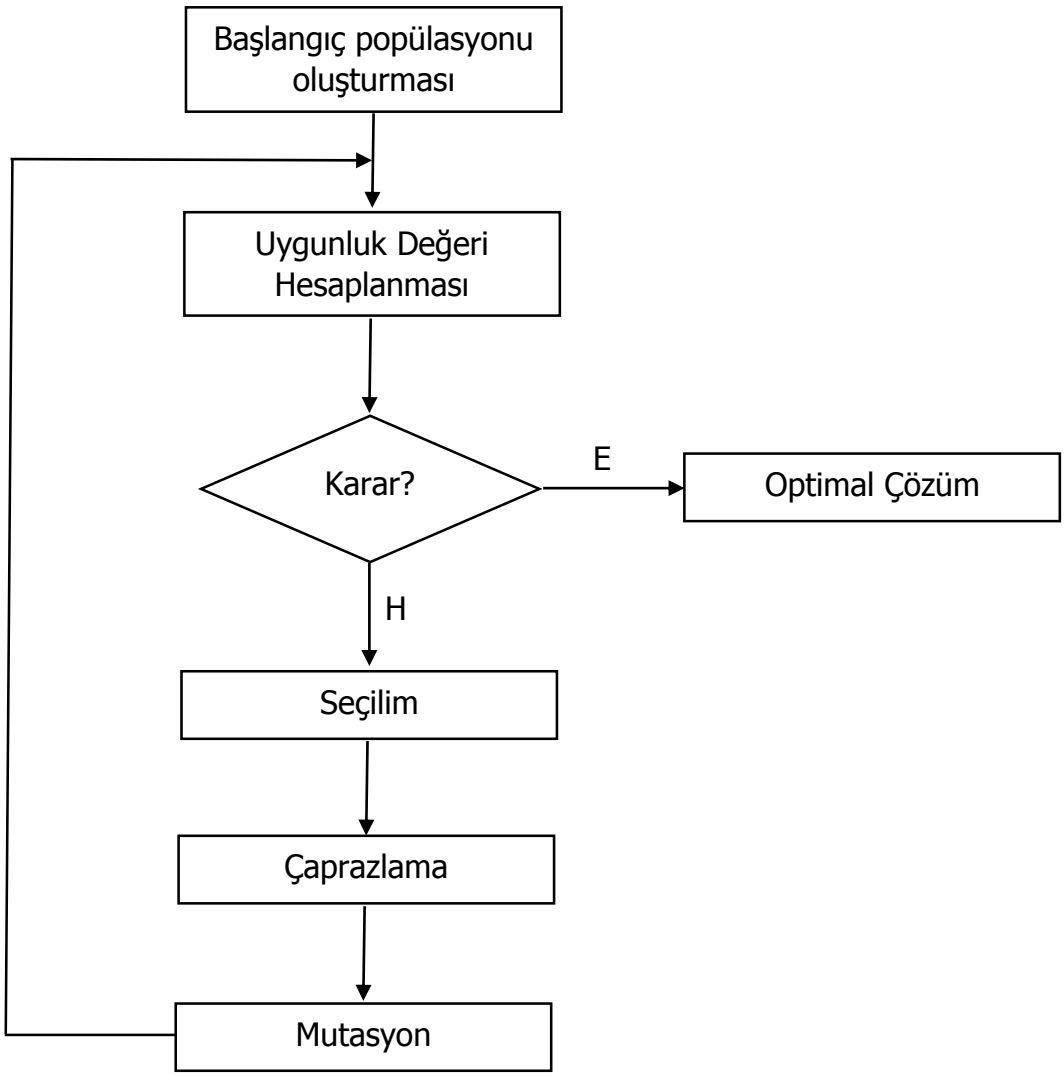
L^k: k karıncası tarafından tamamlanan turun uzunluğu

Genetik Algoritmalar: Genetik algoritmalar üzerinde birçok çalışma yapılmakla birlikte 1975 yılında Michigan Üniversitesi'nde John Holland, öğrencileri ve meslektaşları tarafından çıkarılan "Adaptation in Natural and Artificial Systems" adlı kitapla önem kazanmıştır.

Genetik algoritmalar evrimsel genetik ve doğal seleksiyon kuralları temel alınarak oluşturulmuştur. Başlangıçta oluşturulan bir çözüm kümesini en iyinin hayatta kalması kuralını uygulayarak sürekli iyileştirme çabasındadır. Genetik algoritmalarda genellikle 5 adım bulunmaktadır [37]:

- Belli bir çözüm kümesi üretilmesi
- Çözüm kümelerinin uygunluk değeri hesaplanması
- Uygunluk değerlerine göre ebeveyn seçilimi
- Seçilen ebeveynlerin çaprazlanarak yeni nesilleri üretmesi
- Yeni nesildeki çözümlerin mutasyona uğraması

Son adımda kriterlere uygun bir çözüm bulunmazsa algoritma yeni neslin ugyunluk değerlerini hesaplamak üzere ikinci adıma giderek diğer adımları da tekrarlamaktadır.



Şekil 5.5. Genetik Algoritma Akış Şeması

İlk adımda problemin çözümünde kullanılabilecek çözüm kümelerinden bir popülasyon oluşturulmaktadır. Kümedeki çözümlerin sayısını belirlemek için standart bir yol bulunmamaktadır. Genel olarak önerilen 10-160 çözüme sahip olan popülasyonlar olmakla beraber bu sayı yapılan işlemlerin karmaşıklığı ve arama derinliğine bağlı olarak değişebilmektedir [38].

Uygunluk değeri hesaplanması adımında bu çözüm kümeleri için belli bir uygunluk fonksiyonu çalıştırılmaktadır. Uygunluk fonksiyonu bir çözüm kümesinin sonuca olan yakınlığı ile orantılıdır. Uygunluk fonksiyonu değeri yüksek olan bir çözüm kümesi istenen sonuca daha benzerdir ve bir sonraki nesil için daha

faydalı ebeveynler olacakları düşünülmektedir. Genetik algoritmanın başarısı çoğunlukla bu fonksiyonun verimli ve hassas bir şekilde çalışmasına bağlıdır.

Tüm popülasyon için uygunluk değeri hesaplandığında bir karar mekanizması çalışmaktadır. Bu aşamada popülasyondaki en uygun çözümün optimal çözümü sağlaması durumunda veya başlangıçta belirlenen döngü sayısı tamamlandığında algoritma da tamamlanmış olur.

Karar aşamasından olumsuz yanıt alındığında çözümler uygunluk değerlerine göre ebeveyn olarak seçilirler ve yeni nesiller üretmek için eşleştirilirler. Ebeveynlerin seçilmesinde rulet tekeri, rütbe ve turnuva seçilim yöntemleri gibi yöntemler kullanılmaktadır. Bu yöntemlerde uygunluk fonksiyonu değeri yüksek olan çözümlerin ebeveyn olarak seçilme şansı yüksektir. Ancak bu durum kesin olmadığından elitizm denilen bir metotla yüksek uygunluk fonksiyonu değeri olan bir veya birkaç çözüm direkt olarak yeni nesil popülasyona aktarılabilmektedir [39].

Eşleşmiş olan ebeveyn çözümler yeni nesli oluşturmak için çaprazlanmaktadır. Çaprazlanma işlemi için ebeveyn dizilerini belirli bir veya iki noktadan ayırarak çaprazlayan veya Gezgin-Satıcı Problemi'nde kullanılması uygun olan ve yeni nesil çözümlerde eleman tekrarının olmasını önleyerek çaprazlayan PMX, OX, CX gibi metotlar kullanılmaktadır [40].

Mutasyon aşamasında yeni nesildeki çözümlerden bazılarındaki bir veya birkaç eleman değiştirilmektedir. Eşleşme ve çaprazlama aşamalarında var olan elemanlar kendi aralarında değiştirildiğinden bir süre sonra tekrarlanma durumuna düşülebilmektedir. Bunu önlemek ve çeşitliliği artırmak için mutasyon işlemi yapılmaktadır.

Tüm aşamalar sonlandığında döngü tekrar başa dönmekte ve uygunluk fonksiyonu değerleri hesaplanmaktadır. Makul çaprazlama ve mutasyon yöntemleriyle yeni nesiller istenilen sonuca daha yakın olmaktadır. Genetik algoritmaların kullanım alanları oldukça geniştir. Gezgin-Satıcı Problemi gibi rota hesaplama problemlerinin yanı sıra ekonomik sistem modellerinde, finans, pazarlama gibi birçok alanda genetik algoritmalardan faydalanılmaktadır [41].

6. YAPILAN ÇALIŞMALAR VE ALGORİTMA SEÇİMİ

Geliştirilen mobil uygulamanın kullanıcı seçimleri sonucunda oluşan gezi yerleri kümesini bir rota oluşturacak şekilde sıralaması beklenmektedir. Önceki bölümde açıklanan algoritmalar arasında Karınca Kolonisi Optimizasyonu ve Genetik Algoritmalar bu tür problemlerin çözümünde kullanılmaktadır. A* Algoritması ise bir en kısa yol bulma algoritmasıdır. Ancak kısa çalışma süresi sebebiyle A* algoritmasını temel alan bir algoritma geliştirilerek gezi yerlerinin en kısa rotayı oluşturacak şekilde sıralanması sağlanabilmektedir.

Öne çıkan 3 algoritma; koordinatları bilinen belirli gezi yerlerini, en kısa rotayı oluşturacak şekilde sıralamaları için kodlanmıştır. Algoritmalar başlangıçta verilen koordinatlara göre tüm noktaların birbirine olan coğrafik uzaklıklarının hesaplanarak saklandığı bir matris oluşturmaktadır. Algoritmaların karşılaştırılmasında kullanılacak nokta kümesinin enlem ve boylam koordinatları Çizelge 6.1.'de verilmiştir. Bu nokta kümesinden belirli sayılarda noktalar seçilerek algoritmalara girdi olarak verilmiş ve algoritmaların oluşturduğu rotalar, toplam yol uzunlukları ve algoritmaların çalışma süreleri gözlenmiştir. Algoritmaların çalışmaları esnasındaki adımlarının detayları, farklı girdiler için ürettikleri sonuçlar ve çalışma süreleri aşağıda anlatılmaktadır.

Çizelge 6.1. Nokta kümesinin enlem ve boylam koordinatları

Gezilecek Nokta	Yer Bilgisi	Enlem Koordinatları	Boylam Koordinatları
POI_1	Balmumu Müzesi	39,765100	30,521869
POI_2	Atatürk Kültür ve Sanat Merkezi	39,766772	30,533156
POI_3	Sazova Bilim Kültür ve Sanat Parkı	39,766389	30,475306
POI_4	Haller Gençlik Merkezi	39,781215	30,513590
POI_5	Bilim Merkezi	39,765653	30,473486
POI_6	Canlı Tarih Müzesi	39,765047	30,521947
POI_7	Kent Belleği Müzesi	39,765144	30,521858
POI_8	Masal Şatosu	39,760339	30,473342
POI_9	Cam Sanatları Müzesi	39,765399	30,521910
POI_{10}	Kentpark	39,775004	30,549225

6.1.Algoritmaların Kodlanması

6.1.1.Karınca Kolonisi Optimizasyonu

Algoritmanın ilk adımında parametrelere değerler atanmaktadır. Parametrelere atanan değerler algoritmanın çalışma hızını ve doğruluğunu da doğrudan etkilemektedir. Bu parametreler aşağıdaki gibidir:

α ve β: feromon izi ve görünürlük fonksiyonlarının ağırlıkları

ρ: feromonların buharlaşma değeri

q_0: karıncanın keşif yapma durumunu belirleyen kriter parametresi

m: karınca sayısı

Q: feromon güncellemesinde kullanılan parametre

Bu parametrelere atanması gereken değerler için literatürde araştırma yapılmış ve en uygun olan kombinasyonlar için alınan sonuçlar karşılaştırılmıştır. Parametrelerden feromon güncellemesinde kullanılan Q parametresinin değerinin sonuçtaki oluşturacağı fark önemsiz sayıldığından herhangi bir değer atanabilmektedir ve kodlamada bu parametre için 1 değeri kullanılmaktadır. Karınca sayısını temsil eden m değeri ise literatürdeki karşılaştırmalarda problemdeki nokta sayıları ile aynı olarak seçilmektedir. Bu sebeple karşılaştırmada kullanılan ve mobil uygulamada seçilebilecek nokta sayısına yakın bir değer olması açısından karınca sayısı 20 olarak belirlenmiştir [42, 43]. Diğer parametreler farklı değerlerle test edilmektedir.

```
baslangicFeromonlariAtama();
foreach karinca
{
    foreach nokta
    {
        if(q<q0)
            sonrakiNoktaSecimi(feromon, görünürlük değeri);
        else
            sonrakiNoktaSecimi(olasılık fonksiyonu);
        lokalFeromonGuncelle();
        rotaGuncelle();
    }
    if(rota_karinca < rota_enKisa)
        (rota_enKisa = rota_karinca
    globalFeromonGuncelle();
}
```

Şekil 6.1. Karınca Kolonisi Optimizasyonu Örnek Kodu

Algoritma başlangıçta bir karınca objesi oluşturarak başlangıç noktası belirli bu karıncanın bir sonraki nokta seçimi için rasgele seçilen bir q değeri ile $q \leq q_0$ eşitsizliğin sonucuna göre keşif yapıp yapmayacağına karar vermektedir. Bunun sonucunda keşif yapılmayan adımlar için henüz ziyaret edilmemiş noktaların feromon izi ve görünürlük değerleri hesaplanarak Denklem (5.2) koşulunu sağlayan nokta bir sonraki nokta olarak seçilmekte ve rotaya eklenmektedir. Keşif yapılan durumlarda ise ziyaret edilmemiş noktaların olasılık fonksiyonu değerleri Denklem (5.3) ile hesaplanmakta ve en yüksek değere sahip noktanın seçilme ihtimali daha yüksek olacak şekilde bir sonraki nokta rasgele seçilmekte ve rotaya eklenmektedir. Her eklenen nokta için bir önceki nokta ile arasında olan uzaklık, uzaklık matrisinden bulunarak toplam uzunluk değerine eklenmektedir. Her ziyaret edilen nokta için o noktanın feromon değerleri Denklem (5.4)'teki formül ile güncellenmektedir.

Tüm noktalar ziyaret edildiğinde bir karınca için rota ve toplam alınan yol değerleri saklanarak bir sonraki karınca için aynı işlemler tekrarlanmaktadır. Tüm karıncalar turlarını tamamladığında en az mesafede tüm noktaları ziyaret eden karınca sonucu bulmuş olmaktadır.

Karınca Kolonisi Optimizasyonu'nun çalışma verimliliğini artırmak için literatürde yapılan araştırmalar sonucu parametrelere atanabilecek uygunluktaki değerler test edilmiştir. Yapılan karşılaştırmalar sonucunda elde edilen sonuçlarda zaman bazında ağırlıklı olarak karınca sayısının etkisi gözlemlenirken doğruluk bazında ise karınca sayısının yanında α ve β değerlerinin etkili olduğu

gözlemlenmiştir (q_0=0.2, p=0.5 ve Q=0.1 değerleri için denenmiştir). Sonuçlar T.R.U. (Toplam Rota Uzunluğu) ve T.S. (Toplam Süre) başlıklarında Çizelge 6.2'de gösterilmektedir.

Çizelge 6.2. Karınca Kolonisi Optimizasyonu farklı parametre değerleri ile alınan sonuçlar

Parametre Değerleri		Deney Sırası	m=20		m=100		m=1000	
			T.R.U. (mt.)	T.S. (ms.)	T.R.U. (mt.)	T.S. (ms.)	T.R.U. (mt.)	T.S. (ms.)
α=1	β=1	1	**13.465**	44	**10.841**	52	10.227	72
		2	**13.899**	45	**10.234**	54	10.227	70
		3	**14.014**	44	**10.234**	50	10.227	72
	β=5	1	10.234	45	10.234	49	10.227	73
		2	10.234	45	10.234	49	10.227	71
		3	10.234	43	10.234	50	10.227	75
	β=9	1	10.234	47	10.234	51	10.227	71
		2	10.234	45	10.234	52	10.227	75
		3	10.234	49	10.234	49	10.234	72
α=5	β=1	1	**14.504**	44	**13.739**	51	10.234	72
		2	**14.872**	44	**14.007**	50	10.253	74
		3	**12.937**	45	**12.750**	50	10.234	73
	β=5	1	**13.673**	46	10.234	49	10.227	73
		2	**10.822**	45	10.234	52	10.234	74
		3	**11.717**	43	10.227	49	10.234	74
	β=9	1	10.720	43	10.234	49	10.227	73
		2	10.512	43	11.295	51	10.227	72
		3	10.234	45	10.234	50	10.227	72

Çizelge 6.2'deki deneylerden elde edilen sonuçlar göz önünde bulundurularak parametre değerlerindeki değişimin toplam rota uzunluğu ve toplam süreye olan etkileri hakkında yorum yapılabilmektedir. Bu sonuçlarda artan karınca sayısı ile beraber toplam sürede gözle görülür bir artış olmakla beraber algoritma daha stabil rota uzunluğu sonuçları vermektedir. α ve β parametrelerinin değişen değerleri yorumlandığında ise α değerinin, β değerinden az olduğu durumlarda en kısa rotanın stabil bir şekilde bulunması ancak yüksek sayıda karınca algoritmada görevlendirildiğinde mümkün olmaktadır. Ancak β değeri, α değerinden yüksek tutulduğunda algoritmada az sayıda karınca görevlendirilmesine rağmen en kısa rota değeri stabil bir şekilde kısa sürelerde bulunabilmektedir. Bu sonuçlardan yola çıkılarak en kısa rotanın stabil ve kısa sürede bulunabilmesi az

sayıda karınca görevlendirilerek ve görünürlük fonksiyonu feromon izine göre daha etkili kılınarak sağlanabilmektedir.

6.1.2.Genetik Algoritma

Algoritmada öncelikle verilen noktalardan belirli sayıda, nokta kümesinin sayısına da bağlı olarak, olası çözüm kümeleri oluşturulmaktadır. Bu çözüm kümeleri popülasyon olarak adlandırılmaktadır. Başlangıçta popülasyon içerisindeki noktalar başlangıç noktası sabit olacak şekilde rasgele karıştırılmaktadır. Daha sonra popülasyondaki çözüm kümelerinin uygunluk fonksiyonu değerleri hesaplanmaktadır. Rota hesaplama probleminde algoritmada kullanılan uygunluk fonksiyonu değeri rotada katedilen toplam mesafe ile ters orantılıdır.

Kodlaması yapılan algoritmada, algoritmanın tanımında da bahsedilen elitizm modeli uygulanmaktadır. Bu modele göre en yüksek uygunluk fonksiyonu değerine sahip çözüm kümesi bir sonraki nesile değişikliğe uğramadan aktarılmaktadır. Bu durum her nesilde en iyi olanların hayatta kalmasını garanti etmektedir. Elit olarak sınıflandırılan çözüm kümesi sayısı arttıkça sürekli en iyilerin hayatta kalmasından dolayı yüksek değerli sonuçlar elde edilse de elitizm çeşitliliği sınırlandırıldığından uygun değerleri bulabilmek için kodlama esnasında farklı değerler kullanılarak deneyler yapılmış ve sonuçlar karşılaştırılmıştır.

Çaprazlama işlemine katılması düşünülen çözüm kümelerini seçmek için turnuva modeli kullanılmaktadır. Bu modelde öncelikle turnuvaya katılması planlanan çözüm kümesi sayısı belirlenmektedir. Turnuva sayısınca elit olamayan çözüm kümeleri arasından rasgele seçilenlerden uygunluk fonksiyonu değeri en yüksek olan çözüm kümesi ebeveyn olarak seçilmektedir. Bu işlem tekrarlanarak bir sonraki ebeveyn de seçildikten sonra çaprazlama işlemi yapılmaktadır.

Çaprazlama işlemi aşamasında turnuva modeli ile ebeveyn olarak seçilen çözüm kümelerinin ilki için rasgele seçilen başlangıç ve bitiş noktaları arasındaki noktalar yeni oluşturulan ve yeni nesle aktarılacak olan çözüm kümesine eklenmektedir. Seçilen diğer ebeveynin noktaları içerisinde yeni nesilde bulunacak çözüm kümesinde bulunmayan noktalar eklenmektedir. Bu şekilde ebeveynler

çaprazlanarak yeni nesile aktarılacak olan ve aynı noktanın tekrarlanmaması sağlanan bir çözüm kümesi oluşturmaktadır. Önceki neslin elit ve çaprazlama işlemi sonucu oluşan çözüm kümeleri ile yeni popülasyon oluşturulur.

Yeni oluşturulan popülasyonda çeşitliliği artırmak için elit olamayn çözüm kümelerine mutasyon işlemi uygulanır. Mutasyon işleminde normalde çözüm kümesindeki bir veya birkaç eleman çözüm kümesinde bulunması zorunlu olmayan herhangi bir elemanla değiştirilirken bu problemin çözümünde seçili olmayan noktalar çözüme dahil edilemeyeceğinden aynı çözüm kümesindeki noktalar arasında yer değiştirme uygulanmaktadır. Bu işlemin amacı var olan çözüm kümelerine bağlı kalmadan değişik türler elde etmek iken bu problemde çaprazlamayla oluşturulan çeşitliliğin artmasında etkili olmaktadır. Çözüm kümesindeki her bir nokta başlangıçta belirlenen bir mutasyon oranı değeri sayesinde rasgele mutasyona uğramaktadır.

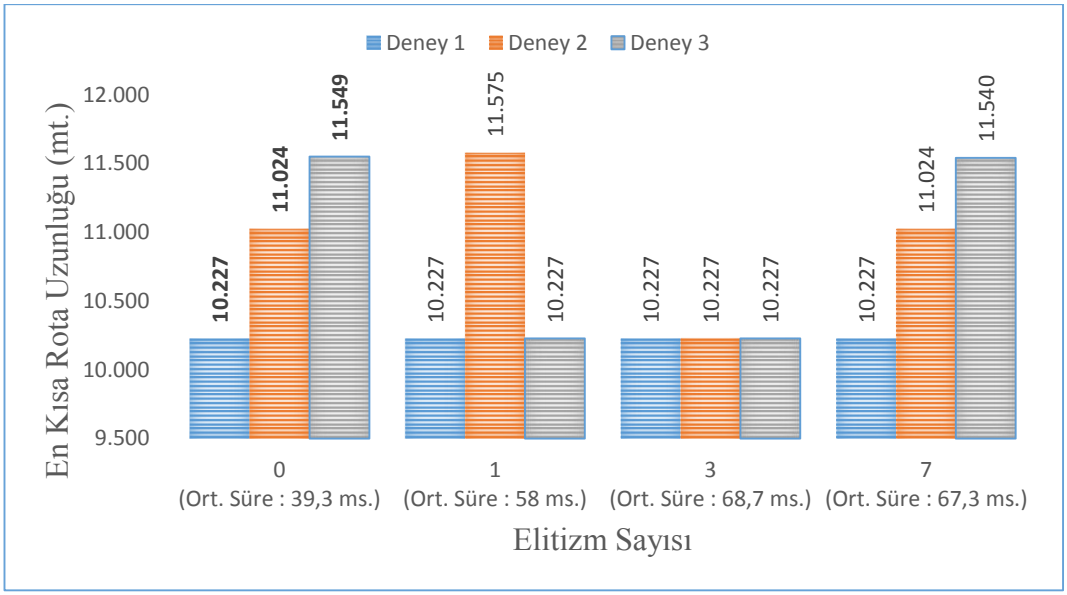
Mutasyon işleminin ardından yeni oluşturulan popülasyondaki çözüm kümeleri için uygunluk fonksiyonu değerleri hesaplanmakta ve yeni elit çözüm kümeleri seçilmektedir. Bu işlemler başlangıçta belirlenen nesil sayısı kadar döngü halinde devam etmektedir. Her yeni popülasyonun en iyileri bir sonraki nesle bozulmadan aktarıldığından en son nesilde bulunan en iyi uygunluk fonksiyonu değerine sahip çözüm kümesi aslında tüm nesiller arasında da en iyi değere sahip olan çözüm kümesi olmaktadır.

Algoritmanın hem verimli sonuç verebilmesi hem de çalışma süresinin makul bir seviyede olması için algoritmanın çalışmasını etkileyen parametrelere başlangıçta uygun değerler atanmalıdır.

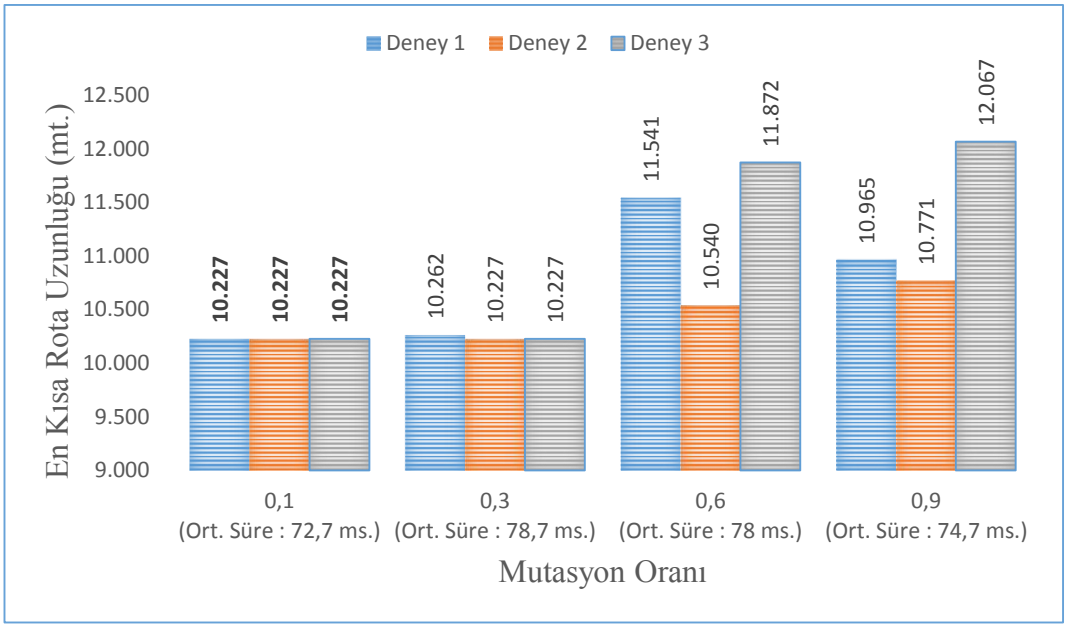
Algoritmanın performansını etkileyen parametrelerden bir tanesi olan popülasyondaki çözüm kümesi sayısı için literatürde yapılan araştırmalar sonucunda 100 değerinin atanması uygun görülmüştür. Popülasyondaki çözüm kümesi sayısının artması en iyi sonucun alınma ihtimalini artırmakla beraber çalışma süresini de artırmaktadır. Literatürde yapılan deneylerde çözüm kümesi sayısının 100'den fazla olduğu durumlarda algoritmanın çalışma süresi artarken alınan sonuçlarda daha az bir gelişim gözlenmektedir [44].

Mutasyon oranı, turnuvaya katılan çözüm kümesi sayısı,döngü sayısı ve elit çözüm kümesi sayısı algoritmanın performansını etkileyen faktörlerdendir.

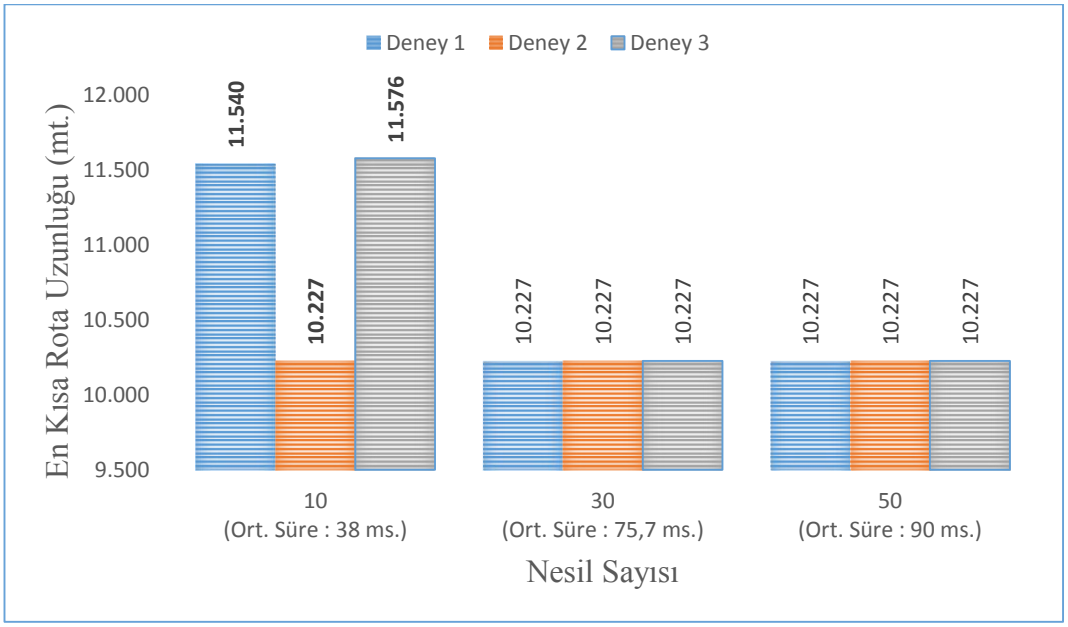
Mutasyon işlemi sırasında bir çözüm kümesindeki her nokta için 0 ile 1 arasında rasgele bir değer seçilerek bu değerin başlangıçta belirlenen mutasyon oranından düşük olması durumunda mutasyon işlemi yapılmaktadır. Ebeveynlerin seçilme aşamasında ise başlangıçta belirlenen sayıda çözüm kümesi turnuvaya katılarak aralarında uygunluk fonksiyonu değeri en yüksek olan çözüm kümesi ebeveyn olarak seçilmektedir. Her yeni oluşan nesilde daha iyi bir sonuç bulunma ihtimali artmaktadır. Nesil sayısının fazla olması bunu sağlamakla beraber değerleri algoritmanın çalışma süresiyle ters orantılı olarak değişmektedir. Elitizm ise her nesilde popülasyonun kalitesini artırmasına karşın seçilen elit çözüm kümesi sayısının fazla olması çeşitliliği azalttığından daha iyi sonuçların bulunma ihtimalini düşürmektedir. Bu parametrelerin aldığı farklı değerler için algoritmanın ürettiği sonuçlar ve performans karşılaştırması sonuçları Şekil 6.2, Şekil 6.3 ve Şekil 6.4'te gözlenmektedir. Parametreler algoritmaya birbirleriyle doğrudan bağlantılı olarak etki etmediğinden sonuçlar ayrı bir şekilde gözlenmiştir.



Şekil 6.2. Genetik Algoritmada Elitizm Etkisi



Şekil 6.3. Genetik Algoritmada Mutasyon Etkisi



Şekil 6.4. Genetik Algoritmada Nesil Sayısı Etkisi

Grafikler incelendiğinde algoritmanın kullandığı parametrelerin sonuçta belirleyici özellikte olan en kısa rota uzunluğu ve en kısa rota süresi değerlerine etkileri gözlemlenmektedir. Seçilen değerlerde en uygun sonucu verenler incelenmiş ve bu değerlere uygun olarak parametre değerleri ayarlanmıştır.

6.1.3.A* Algoritması

A* (A-star) algoritması bir kısa yol bulma algoritmasıdır. Başlangıç noktasından varış noktasına olan rotayı en kısa olacak şekilde bulmaktadır. Bu sebepten rota bulma probleminin çözümünde kullanılabilmesi için sezgisel fonksiyonunda ve algoritmanın işleyişinde bazı değişiklikler yapılması gerekmektedir.

A* algoritmasının hedefi varış noktasına ulaşmak olduğundan nokta kümesindeki tüm noktaları kullanmayarak en kısa yolu oluşturacak şekilde rotayı belirlemektedir. Ancak geliştirilen mobil uygulamada kullanıcının seçtiği tüm noktaların dahil olduğu en kısa veya kabul edilebilir uzunlukta bir rota hesaplanması gerekmektedir. A* algoritması amacı itibariyle bu problemin çözümüne hizmet etmese de en kısa yol bulma problemlerindeki başarısı ve çalışma zamanının oldukça düşük olması sebebiyle tercih edilmiştir [45,46].

Sezgisel fonksiyon A* algoritmasının çalışma süresi ve doğruluğunda önemli rol oynamaktadır. Dijkstra algoritması ile A* algoritması arasındaki en büyük fark A* algoritmasında kullanılan sezgisel fonksiyondur. Bu fonskiyon sayesinde bir sonraki nokta hakkında tahmin üretilerek sonuca yakın noktaların avantajlı olmaları sağlanmaktadır.

Algoritma her bir n noktası için hesaplanıp atanan Denklem (5.1)'deki f(n) fonksiyonunu baz alarak bir sonraki nokta seçimini yapmaktadır. Bu formüldeki h(n) fonksiyonu algoritmanın sezgisel fonksiyonunu simgelemektedir. Belirli bir başlangıç ve bitiş noktaları arasındaki en kısa yolun bulunması probleminde $h(n)$ fonksiyonu n noktasının bitiş noktasına olan kuş bakışı uzaklığı olarak hesaplanabilmektedir. Ancak nokta kümesindeki tüm noktalar ziyaret edilecek şekilde bir rota hesaplanmak istendiğinde farklı bir sezgisel fonksiyon kullanılmalıdır. Literatürde yapılan araştırmalarda kesin sonuç garantisi olan bir sezgisel fonksiyona rastlanılmamakta fakat benzer problemlerin çözümünde de kullanılan ve kabul edilebilir sonuçlar veren Minimum Spanning Tree (MST) algoritmasının kullanılması uygun bulunmuştur.

Herhangi bir yönsüz $G(V, E)$ çizgesinde tüm noktaları birbirine bağlayacak ve çevrim oluşturmayacak şekilde oluşan ağaç yapısındaki altçizgeye Spanning

Tree denilmektedir. Minimum Spanning Tree oluşan ağaç yapıları içerisinde noktalar arası kenarların ağırlıkları toplamı en düşük olan ağaç yapısıdır. Bu ağırlıklar genellikle noktalar arası kenarların uzunluklarıdır. Bağlı çizgelerde mutlaka bir spanning tree bulunmaktadır. Bir çizgede birden fazla MST bulunabilir. Bir çizgedeki MST yapı veya yapılarının bulunması problemine Minimum Spanning Tree Problem denilmektedir ve çözümü için algoritmalar üretilmiştir. Üretilenlerden en çok kullanılanlar Prim Algoritması ve Kruskal Algoritmasıdır [47].

Algoritmanın çalışması esnasında her adımda ziyaret edilmeyen tüm noktalar için sezgisel fonksiyon değerleri hesaplanmaktadır. Herhangi bir n noktasının sezgisel fonksiyon değeri hesaplanması aşamasında ziyaret edilmeyen noktalardan bir MST oluşturularak ortaya çıkan yol uzunluk değeri h(n) fonksiyonunun değeri olarak kabul edilmektedir. Aynı n noktasının $g(n)$ fonksiyon değeri, her noktanın başlangıç noktasından itibaren bulunduğu rota üzerindeki noktalar arası uzaklıklar toplanmakta ve mevcut nokta ile n noktası arasındaki uzaklık eklenerek hesaplanmaktadır. Hesaplanan $g(n)$ ve $h(n)$ fonksiyonları sayesinde n noktasının $f(n)$ fonksiyonu hesaplanmakta ve bu işlem tüm ziyaret edilmeyen noktalar için tekrarlanmaktadır.

```
 vStart, currentV; sortedOpenList, closedList, path[], cosr[];
closedList.add(vStart);
currentV=vStart;

while(!closedList.contains(vertices))
{
  foreach (v in (vertices- currentV))
  {
    if(!closedList.contains(v))
    {
       v.h = minSpanTree(vertices-closedList);
       v.g = path.cost + cost[currentV,v];
       v.f = v.g + v.h;
       sortedOpenList.add(v);
    }
  }
  currentV=sortedOpenList.pop();
  closedList.add(currentV);
  updatePath(closedList);
}
```

Şekil 6.5. A* Algoritması Örnek Kodu

Algoritmada ilk adımda içerisindeki noktaların $f(n)$ fonksiyonu değerlerine göre sıralı bir nokta listesi oluşturulmaktadır. Bununla beraber ziyaret edilen

noktaların ve rota bilgilerinin saklandığı listeler de oluşturulmaktadır. Bir sonraki adımda belirlenen başlangıç noktası ziyaret edilen noktalar listesine ve rotaya eklenerek bir sonraki noktanın seçimi yapılmaktadır. Seçim işlemi için tüm ziyaret edilmeyen noktalar incelenerek her bir nokta için diğer ziyaret edilmeyen noktalarla beraber bir MST oluşturularak sezgisel fonksiyon değeri, oluşan rotanın uzunluk değeri eklenerek maliyet fonksiyonu değeri ve bunların sonucunda değerlendirme fonksiyonu değeri hesaplanmaktadır.Değerlendirme fonksiyonu değeri en düşük olan nokta bir sonraki nokta olarak seçilerek rotaya eklenmektedir. Bu işlemler tüm noktalar ziyaret edilene kadar devam etmekte ve tüm noktalar ziyaret edildiğinde algoritma rotayı tamamlayarak sonlanmaktadır.

6.2.Algoritmaların Karşılaştırılması

Karınca Kolonisi Optimizasyonu, Genetik Algoritma ve A* Algoritması koordinatları belirli olan gezi yerlerini girdi olarak alacak ve çıktı olarak bu noktalardan oluşan en kısa rotayı bulmayı hedefleyecek şekilde kodlanmışlardır. Kodlama esnasında bir önceki kısımda yapılan deneyler sonucunda elde edilen veriler yorumlanarak algoritmaların içeriği tasarlanmış ve gerekli parametreler en uygun değerler atanmıştır.

Çalışmanın ilk sürecinde A* Algoritması ve Karınca Kolonisi Optimizasyonu farklı gezi yerleri için karşılaştırılmıştır. Karınca Kolonisi Optimizasyonu için parametre optimizasyonu olmadan yapılan karşılaştırmada Karınca Kolonisi Optimizasyonu'nun artan nokta sayılarında A* Algoritması'na kıyasla daha fazla oranlarda hesaplama süresi ile çalıştığı gözlemlenmiştir [48].

Karınca Kolonisi Optimizasyonu'nda gerekli optimizasyonların yapıldığı ve Genetik Algorima'nın da eklendiği farklı senaryolarda yapılan karşılaştırmaların olduğu diğer bir çalışmada Karınca Kolonisi Optimizasyonu'nun önceki karşılaştırmadaki sonuçlarına oranla daha düşük sonuçlar verdiği gözlemlenmiştir [49].

Algoritmalar belirli senaryolar için çalıştırılarak sonuçlar en kısa rota uzunluğu ve çalışma süresi bazında değerlendirilerek sonuçlar karşılaştırılmıştır. Mobil uygulamanın çalışma esnasında kullanıcıya belirli aralıklarda veya herhangi

bir anda yakın gezi yerlerini öneri olarak sunması ve kullanıcının seçimleri doğrultusunda rotanın dinamik olarak güncellenmesi düşünülmüştür. Bu sebepten başlangıçta Çizelge 6.1.'da belirlenen 10 gezi yerine ilave olarak öneri olarak sunulabilecek Çizelge 6.3.'da gösterilen 3 gezi yeri daha eklenmiştir ve bu gezi yerlerinin öneri olarak sunulması ve kullanıcı seçimleri farklı senaryoların oluşturulmasında kullanılmaktadır.

Çizelge 6.3. Öneri Nokta Kümesi Enlem ve Boylam Koordinatları

Gezilecek Nokta	Yer Bilgisi	Enlem Koordinatları	Boylam Koordinatları
Rec_1	Papağan Çibörek	39,775023	30,518881
Rec_2	Kocatepe Kahve Evi	39,762298	30,472288
Rec_3	Kurşunlu Külliyesi	39,763272	30,525695

Senaryo 1: Kullanıcı Çizelge 6.1'deki ilk 5 gezi yerini seçer ve rota sırasıyla POI_1, POI_2, POI_4, POI_3, POI_5 olacak şekilde oluşturulur. Kullanıcıya gezi esnasında POI_2, POI_4 noktaları arasında Rec_1 noktası öneri olarak sunulur. Kullanıcının öneriyi kabul etmediği durumda gezi aynı rota ile devam eder. Öneri kabul edildiğinde kullanıcının karşısına kalan yerlerin gezme süreleri ile bağlantılı olarak alternatif rotalar sunulur. İlk alternatif POI_3 ve POI_4 noktalarının rotadan çıkartılarak Rec_1 noktasının eklendiği $POI_1, POI_2, Rec_1, POI_5$ rotasıdır. İkinci alternatif olarak POI_5 noktasınıın rotadan çıkartılarak Rec_1 noktasının eklendiği $POI_1, POI_2, Rec_1, POI_4, POI_3$ rotasıdır. Kullanıcının seçimine göre güncellenen rota bilgileri tekrar hesaplanır.

Senaryo 2: Kullanıcı Çizelge 6.1'deki ilk 8 gezi yerini seçer ve rota POI_1, POI_7, POI_6, POI_2, POI_4, POI_3, POI_5, POI_8 olacak şekilde oluşturulur. İlk alternatifte kullanıcıya gezi esnasında POI_6 ve POI_4 noktaları arasında Rec_1 noktası öneri olarak sunulur. Kullanıcı öneriyi kabul ettiğinde rota $POI_1, POI_7, POI_6, Rec_1, POI_4, POI_3, POI_5, POI_8$ olarak güncellenir. Öneri kabul edilmediğinde gezi rotanın ilk haliyle devam eder. İkinci alternatif olarak POI_5 ve POI_8 noktaları arasında Rec_2 noktası öneri olarak sunulur. Kullanıcı öneriyi kabul ettiğinde rota POI_1, POI_7, POI_6, POI_2, POI_4, POI_3, POI_5, Rec_2 olarak güncellenir. Öneri kabul edilmediğinde rota ilk haliyle tamamlanır.

Senaryo 3: Kullanıcı Çizelge 6.1'deki 10 gezi yerini seçer ve rota POI_1, POI_7, POI_6, POI_9, POI_2, POI_{10}, POI_4, POI_3, POI_5, POI_8 olacak şekilde oluşturulur. İlk alternatifte kullanıcıya gezi esnasında POI_9 ve POI_2 noktaları

arasında Rec_3 noktası öneri olarak sunulur. Kullanıcı öneriyi kabul ettiğinde rota POI_1, POI_7, POI_6, POI_9, Rec_3, POI_2, POI_{10}, POI_3, POI_5 olarak güncellenir. Öneri kabul edilmediğinde gezi rotanın ilk haliyle devam eder. İkinci alternatif olarak POI_3 ve POI_5 noktaları arasında Rec_2 noktası öneri olarak sunulur. Kullanıcı öneriyi kabul ettiğinde rota POI_1, POI_7, POI_6, POI_9, POI_2, POI_{10}, POI_4, POI_3, Rec_2, POI_5 olarak güncellenir. Öneri kabul edilmediğinde rota ilk haliyle tamamlanır.

Çizelge 6.4'de algoritmaların tanımlanan senaryolarda ve bu senaryolarda sunulan öneriler sonucu oluşan alternatif durumlarda sergiledikleri performansları incelenmektedir.

Çizelge 6.4. Farklı Senaryolarla Algoritmaların Karşılaştırmaları

Senaryo No	Başlangıç Nokta Sayısı	Öneri Sonucu	Öneri Sonrası Nokta Sayısı	Toplan Rota Uzunluğu (mt.)	Algoritma Çalışma Süresi(ms.)		
					A*	K.K.O.	G.A.
1	5	Kabul Edilmedi	5	7.140	10	45	48
1	5	Alternatif 1	4	6.525	10	42	42
1	5	Alternatif 2	5	6.996	11	44	44
2	8	Kabul Edilmedi	8	7.744	13	44	66
2	8	Alternatif 1	8	6.412	12	44	67
2	8	Alternatif 2	7	7.500	12	43	63
3	10	Kabul Edilmedi	10	10.227	14	49	78
3	10	Alternatif 1	9	9.421	14	47	74
3	10	Alternatif 2	10	10.030	13	45	78

Çizelge 6.4'de görüldüğü üzere tablodaki tüm algoritmalar verilen noktalar için en kısa rotayı oluşturmaktadır. A* Algoritması mevcut senaryolarda ve alternatif durumlarda Karınca Kolonisi Optimizasyonu ve Genetik Algoritma'ya göre %70-85 oran aralığında daha az çalışma süresinde rotayı oluşturabilmektedir. Geliştirilen mobil uygulama için tahmin edilen ortalama girdi sayısı mevcut senaryolar ile örtüşmektedir. Uygulamanın hareket halinde kullanılması ve gerekli değişikliklerin kullanıcıya yansıma süresinin önemli olduğu düşünüldüğünde A* Algoritması uygulamada kullanılmak için uygun bulunmuştur.

7. SİSTEMİN GENEL ANLATIMI

7.1.Sistemin Anlatımı

Sistemin genel işleyişi işlevlerine göre 5 ana kısma ayrılmıştır. Bunlar kullanıcı girişi, gezi planı oluşturulması, gezi yeri seçimi, plan karar aşaması ve gezi takibidir.

7.1.1.Kullanıcı girişi

Uygulamaya giriş yapıldığında ana sayfada bir kullanıcı girişi ekranı bulunmaktadır. Arka planda Eskişehir'e ait görseller belirli aralıklarla değişmektedir. Sistem kullanıcı kaydı gerektirmektedir. E-posta adresi ve şifrenin gerekli olduğu basit bir kayıt olma aşamasından sonra sisteme giriş yapılabilmektedir. E-posta adresinin geçerli bir adres olması gerekmektedir. Sistem tarafından adresin geçerliliği kontrol edilmektedir. E-posta adresi kullanıcılar için tekillik içermektedir. Yani bir e-posta adresi ile sadece bir kullanıcı kayıt olabilmektedir. Kullanıcı girişi sağlandığında uygulamanın yaşam süresi boyunca kullanıcı kimliği sistem tarafından bilinmekte ve uygulama kullanıcıya özel olarak çalışmaktadır. Kullanıcının geçmiş planları ve hareketleri bu sayede kaydedilmekte ve ihtiyaç anında kullanılabilmektedir. Sistemin kayıt aşamasındaki bir diğer amaç da yapılan gezi planlarının ve seçilen gezi yerlerinin kullanıcıyla ilişkilendirilmesidir. Böylelikle kullanıcı tanınmakta ve öneriler kullanıcının geçmiş tercihleri göz önünde bulundurularak daha verimli olarak kullanıcıya sunulabilmektedir.

7.1.2.Plan oluşturma

Kullanıcı girişi gerçekleştirildikten sonra kullanıcının önceden planladığı ancak henüz tamamlamadığı planlar listelenmektedir. Tamamlanan planlar veri tabanında tamamlanmış olarak kaydedilerek sadece tamamlanmayan planların kullanıcıya listelenmesi sağlanmaktadır. Kullanıcı tamamlamadığı planlardan birini seçerek plan üzerinde güncellemeler yapabilmekte veya yeni bir plan

oluşturabilmektedir. Yeni plan oluşturmak istendiğinde kullanıcı plan oluşturma sayfasında yönlendirilmektedir.

Plan oluşturma sayfasında kullanıcıdan günlük oluşturulan gezi planına ait bilgileri seçmesi istenmektedir. Bunlardan ilki gezinin yapılmak istendiği ülke ve şehir seçenekleridir. Uygulama mevcut durumda yalnızca pilot il Eskişehir için olan bilgileri içerdiğinden ülke ve şehir seçeneklerinde sadece Türkiye ve Eskişehir bulunmaktadır. Ancak sistem bütünüyle genişletilmeye ve geliştirilmeye açık olarak tasarlandığından ülke ve şehir seçenekleri bulunmaktadır. Bir sonraki seçenek gezinin planlanan gün ve saatleridir. Veri kümesinde ülke ve şehir olarak mevcut Türkiye ve Eskişehir bulunduğundan bu seçenekler otomatik olarak seçili gelmektedir. Planın yapılmak istendiği gün seçeneği için mevcut gün bilgileri kullanılmaktadır. Ancak kullanıcı başka bir gün için plan yapmak istediğinde bu seöeneği değiştirebilmektedir. Planın yapılmak istendiği saat aralıkları sistem tarafından otomatik olarak 09:00 - 22:00 olarak atanmaktadır. Ancak bu seçenek de kullanıcı tarafından değiştirilebilmektedir. Sistemde mevcut gezi yerlerine ait gezi yerinin açık olduğu saatler, kapalı olduğu günler ve ideal gezme zamanı bilgileri bulunmaktadır. Sistem bu haliyle yeni özellikler eklenebilmesine imkân sağlayabilmektedir. Örneğin plan belirli gün ve saat aralığı için oluşturulduğunda, gezi yeri seçimi aşamasında planın gün ve saat aralığında açık olmayan gezi yerlerinin listeye eklenmemesi veya gezi yerlerinin sıralanarak gezi planının düzenlenmesi aşamasında gezi yerleri ideal gezme zamanları göz önünde bulundurularak bir plan düzenlenmesi özelliği için sistem gerekli altyapıya sahiptir. Plana ayırt edilmesi için bir isim verildikten sonra plan oluşturulmaktadır. İsim alanı boş bırakıldığında sistem otomatik olarak şehir ve gün bilgilerinden bir isim üretmektedir. Plan oluşturulduğunda veya kayıtlı bir plan seçildiğinde kullanıcı gezi yerlerini seçebildiği sayfaya yönlendirilmektedir.

7.1.3.Gezi yeri seçimi

Sistemdeki gezi yerleri kullanıcıya daha rahat seçim imkânı sağlamak ve tutarlı öneriler sunabilmek için kategorilere ayrılmıştır. Kültürel, eğlence, tarihi vb. kategori başlıklarının altında bu kategorilere ait gezi yerleri sıralanmaktadır.

Kategoriler arasında çoklu seçim yapılabilmektedir. Kullanıcı öncelikle bu kategorilerden çoklu seçim yaparak gezi yeri seçimi sayfasına yönlendirilmektedir.

Gezi yeri seçimi sayfasında kullanıcının seçtiği kategori başlıkları altında olan gezi yerleri tüm kullanıcılardan elde edilen oylama sonuçlarına göre sıralı bir şekilde listelenmektedir. Mevcut sistemde uygulama ticari amaçlarla tasarlanmadığından gezi yerlerine ait oylama puanları sistem yüneticisi tarafından belirlenmektedir. Ticari amaçlarla tasarlandığında, gezi yerlerinin puanlama sistemi kullanıcılar tarafından erişilebilir şekilde düzenlenerek bu özelliğin eklenebilmesine sistem olanak sağlamaktadır. Kullanıcı, gezi yerlerinin üstüne tıkladığında gezi yerinin tanıtıldığı detay sayfasına yönlendirilmektedir. Bu sayfada gezi yerine ait görsel veriler, açıklayıcı bilgiler, açık olduğu saatler ve kapalı olduğu günler, ideal gezme zamanı, ortalama gezme süresi ve diğer kullanıcıların tavsiyeleri bulunmaktadır.

Sistemde gezi yerine ait görsel veriler yüksek çözünürlüklü orijinal veriler ve aynı verilerin ait düşük çözünürlüklü ve düşük boyutta olan kopyaları olmak üzere iki çeşittir. Mobil uygulama veri tabanında bulunan bir bilgiye ihtiyaç duyduğunda veri transferi internet bağlantısı aracılığıyla gerçekleşmektedir. Bu sebepten mobil uygulama, sunucu programı ve veritabanı arasındaki veri transferi süresi, veri boyutu ile doğru orantılı olarak artmaktadır. Mobil uygulama ile sunucu programı arasındaki veri transferinde kullanılan JSON veri formatlama sistemi ile transfer edilen verinin boyutunda azalma sağlanmaktadır. Bu sayede çoğu zaman veri transferi esnasında geçen süre kullanıcı tarafından farkedilmemektedir. Ancak yüksek çözünürlüklü görsel verilerin boyutu fazla olduğundan ve bir gezi yerine ait birden fazla görsel veri bulunduğundan bu veriler transfer edilmek istendiğinde transfer süresi sayfanın yüklenme süresini oldukça artırmaktadır. Kullanıcının görsel verilere ihtiyaç duymadığı veya detaylı olarak incelemek istemediği durumlarda görsel verilerin transferi sebebiyle sayfanın yüklenmesini beklemesi bir sorun teşkil etmektedir. Bu sorunu gidermek amacıyla gezi detay sayfasında kullanıcıya öncelikli olarak gezi yerine ait düşük boyutlu kopya görselleri listelenmektedir. Bu sayede ihtiyaç duyulan veri boyutu daha az olmakta ve transfer hızlı bir şekilde gerçekleşmektedir. Kullanıcı görsel verilerin detaylarını incelemek istediğinde düşük çözünürlüklü görsel veriyi seçerek aynı görsel veriye ait yüksek

çözünürlüklü veriyi inceleyebilmektedir. Bu aşamada işlem başına bir yüksek boyutlu verinin transferi gerçekleştiğinden transfer süresi sorunu en aza indirgenmeye çalışılmıştır.

Gezi yeri detay sayfasında gezi yerine ait açıklayıcı metinler bulunmaktadır. Bunların çoğunluğu yerel resmi kurumlardan elde edilmiş açıklamalardır. Bu açıklamalarda gezi yerinin tarihçesi, gezi yeri hakkında tanıtıcı bilgiler ve önemi hakkında bilgiler bulunmaktadır. Sayfada bulunan gezi yerine ait açık olduğu saatler ve kapalı olduğu günler bilgileri kullanıcının gezi yeri seçiminde dikkat etmesi gereken bilgilerdendir. Kullanıcı bu bilgilere göre planına uygun gezi yerlerini plana ekleyebilmektedir. İdeal gezme zamanı bilgisi ise kullanıcıya gezi yerinden en verimli şekilde faydalanabilmek için hangi zaman aralığında orada bulunması hakkında bilgi vermektedir. Gezi yerine ait ortalama gezme süresi de gezi yerine ait kullanıcıyı bilgilendirici içerikler arasındadır. Kullanıcı bu sayede planı oluşturmayı tamamlamadan önce plan hakkında bir ön bilgiye sahip olabilmektedir. Gezi yeri hakkındaki tavsiyeler kısmında ise kullanıcı diğer kullanıcılar tarafından sisteme eklenen mevcut gezi yeri hakkındaki tavsiyeleri görebilmektedir. Sistem ticari amaçla geliştirilmediğinden sistemin sosyal becerileri ön planda tutulmamıştır. Bu sebeple mevcut sistemde kullanıcı tavsiyeleri sistem yöneticisi tarafından sisteme eklenmektedir. Ancak gerekli düzenlemeler yapıldığında tavsiyelerin kullanıcılarla ilişkilendirilmesi için gerekli şartlar sistemde bulunmaktadır.

Kullanıcı gezi yerleri hakkında detaylı bilgilere sahip olduğunda gezi planına eklemelerde bulunmak için gezi yeri seçim sayfasına yönlendirilmektedir. Bu sayfada gezi yerlerine ait kullanıcının çoklu seçim yapabildiği kutucuklar, gezi yerlerinin adları, gezi yerinin dahil olduğu kategori ve gezi yerinin puanı bulunmaktadır. Kullanıcı gerekli gezi yeri seçimlerini yaptığında plan hakkında ön bilgi almak ve bu bilgilere göre planda değişiklikler yapabilmek üzere plan karar aşaması sayfasına yönlendirilmektedir.

7.1.4.Plan karar aşaması

Mobil uygulamada gezi planı için seçilen gezi yerlerinin en kısa rotayı oluşturacak şekilde sıralanması gerekmektedir. Bunun için mobil uygulamada seçilen gezi yerleri liste halinde sunucu programına gönderilmektedir. Sunucu programında Bölüm 6'da anlatılan A* algoritmasından türetilen algoritma ile gezi yerleri koordinat bilgilerine göre en kısa rotayı oluşturacak şekilde sıralanmaktadır. Sunucu programında sıralanan gezi yerleri mobil uygulamada plan karar aşamasında sıralı bir biçimde listelenmektedir.

Sıralı gezi yerlerinden oluşturulan rotanın toplam yol uzunluğu ve ziyaret süreleri de dâhil gerekli toplam zaman bilgileri kullanıcıya gösterilmektedir. Google Maps kütüphanesi belirli formattaki bir URI adresini işleyerek rota oluşturan bilgileri XML formatında düzenleyerek cevap döndürmektedir. Oluşturulan URI adresinde içerik olarak başlangıç ve bitiş noktalarına ait enlem ve boylam koordinatları, ziyaret edilecek gezi yerlerine ait enlem ve boylam koordinatları ve seyahat türü gibi tercihe bağlı bilgiler bulunmaktadır. Başlangıç noktası koordinatları kullanıcının bulunduğu noktaya ait GPS koordinatları olarak kabul edilmiştir. Bitiş noktası ise sıralı gezi yerleri listesindeki son gezi yerine ait koordinat bilgileridir. Bu esnada listedeki son gezi yeri bitiş noktası olarak seçildiğinden gezi yerleri listesinden çıkarılarak tekrar durumu engellenmiş olmaktadır. Oluşturulan URI adresi ile Google Maps sunucularına bir istek gönderilmektedir. Sunuculardan dönen cevap XML formatındadır. Cevapta rotayı oluşturmak için gerekli noktalar farklı başlıklar altında ayrılmıştır. Her başlığın altında belirlenen noktaya ait enlem ve boylam koordinatları, bir sonraki nokta ile aralarında olan uzaklık ve tahmini varış süresi bulunmaktadır. Cevaptaki noktalar arası uzaklık ve tahmini varış süreleri toplanarak tüm gezi için alınması gereken toplam yol uzunluğu ve toplam süre hesaplanmaktadır. Kullanıcının plana eklemiş olduğu gezi yerlerinin ortalama gezme süreleri de toplam süreye eklenerek kullanıcıyı bilgilendirme amaçlı mevcut sayfada gösterilmektedir.

Kullanıcı plan karar aşamasında toplam yol uzunluğu ve toplam süre bilgilerini de göz önünde bulundurarak planı güncelleyebilmektedir. Kullanıcı bu bilgiler doğrultusunda planında gerekli gördüğü güncellemeleri plan başlamadan

yapabilmektedir. Kullanıcı toplam yol ve süre bilgilerinin fazla olduğunu düşündüğünde, sıralı liste içerisindeki istediği gezi yerlerini listeden çıkartarak plana devam edebilmektedir. Kullanıcı planın son halini oluşturduğuna emin olduğunda geziyi başlatabilmektedir. Bu durumda toplam yol uzunluğu ve toplam süre bilgileri güncel plana göre tekrar hesaplanarak plan güncellenmektedir.

7.1.5.Kullanıcı gezi takibi ve öneri sistemi

Gezi takibi ekranında Google Maps haritaları kullanılmaktadır. Google Maps sunucularına seçilen gezi yerleri ve başlangıç noktası bilgileri ile bir istek yapılmakta ve bu istek için bir cevap elde edilmektedir. Cevapta rota oluşturulmasına yardımcı olan nokta koordinat bilgileri bulunmaktadır. Bu noktalar harita üzerinde işlenerek ve noktalar arası çizgiler çizilerek plan görsel bir şekilde kullanıcıya sergilenmektedir. Ayrıca toplam yol uzunluğu ve toplam süre bilgileri ekranın üst kısmında bilgilendirme amacıyla gösterilmektedir.

Gezi başladığında rota çizgilendirilmiş bir şekilde harita üzerinde yer almaktadır. Kullanıcının seçmiş olduğu gezi yerlerine ve bulunduğu noktaya ait koordinatlarda farklı renklerde işaretçiler harita üzerinde konumlanmaktadır. Kullanıcı bu şekilde bütün planı harita üzerinde görüntüleyebilmektedir. Gezi yerlerine ait işaretçiler tıklandığında açılır bir pencerede seçili gezi yerine ait açıklayıcı bilgiler görüntülenmektedir. Bu bilgiler gezi detay sayfasındaki bilgilerle aynı olmakla beraber gezi yerine ait yüksek çözünürlüklü görseller de bu sayfada istenildiğinde görüntülenebilmektedir.

Kullanıcı geziye başladığı andan itibaren uygulama tarafından GPS yardımı ile takip edilmektedir. Kullanıcının hareketleri harita üzerinde konumlandırılarak kullanıcıya bilgi sağlanmaktadır. Gezi esnasında belirli aralıklarla kullanıcıya yakın yerler arka planda listelenerek, aralarında uygun kriterleri sağlayanlar öneri olarak sunulmaktadır. Kullanıcıya gezi esnasında öneriler dinlenme yerleri ve gezi yerleri olmak üzere iki farklı şekilde sunulmaktadır.

Kullanıcının planladığı gezi süresinin belirli bir süreden fazla olduğu durumlarda planın kullanıcının uygulayabileceği biçimde olabilmesi için kullanıcıya belirli aralıklarla dinlenme yerleri önerileri sunulmaktadır. Bu dinlenme

yerleri önceden belirlenen yemek saati aralıkları içerisinde bir restoran veya belirli zaman aralıklarında bir kafe olabilmektedir.

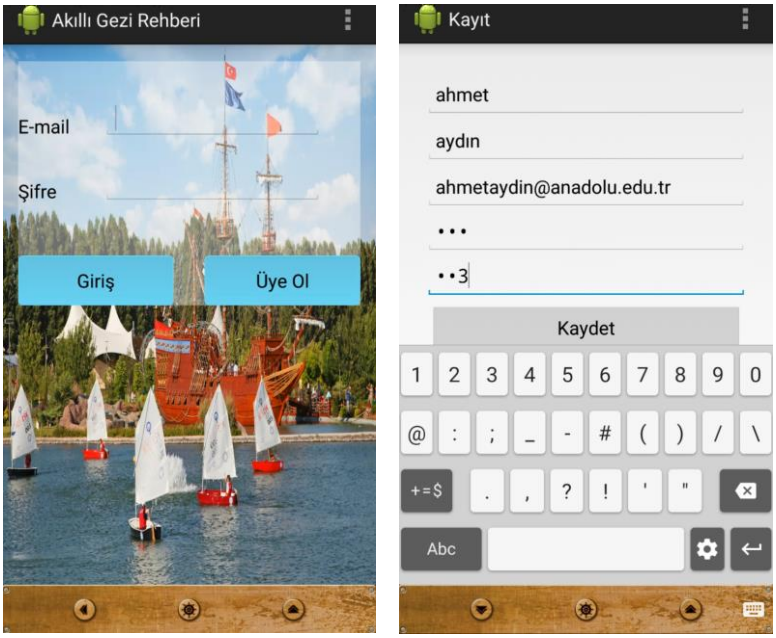
Kullanıcı önceden belirlemiş olduğu plana uygun rotasında devam etmekte iken önceki planları incelenerek seçtiği gezi yerleri ve kategorileri üzerinden kullanıcı profili oluşturulmaktadır. Oluşturulan profile göre kullanıcının gezmek isteyebileceği öngörülen kategorilerde, anlık konumuna belirli bir yakınlıkta bulunan ve belirli bir puanın üzerindeki gezi yerleri kullanıcıya öneri olarak sunulmaktadır.

Kullanıcı belirlenen rota üzerinde gezisine devam ederken belirli aralıklarla veri tabanındaki gezi yerleri sistem tarafından kontrol edilmektedir. Eğer anlık zaman, yemek veya dinlenme zamanına uygunsa veri tabanındaki gezi yerleri arasından yemek kategorisinde bulunan, kullanıcıya belirli yakınlıkta bulunan ve belirli bir puanın üzerinde bulunan yerler seçilmektedir. Seçilen yerler haritadaki mevcut işaretçi renklerinden farklı renkte bir işaretçiyle haritada konumlandırılmaktadır. Benzer işlem diğer zaman aralıklarında, yemek kategorisine ait olmayan ve kullanıcı profiline uygun kategorilere dahil gezi yerleri için uygulanmaktadır. Kullanıcı öneriyi farkettiğinde işaretçiye tıklayarak gezi yerinin detaylarını inceleyebilmektedir. Bu esnada kullanıcı öneriyi plana dahil etmek istemezse kolayca sayfayı terk edebilmektedir ve öneri noktası ile kullanıcının konumu arasında belirli uzaklık sağlandığında, öneri haritadan kaldırılmaktadır. Kullanıcının öneriyi plana dahil etmek istediği durumlarda ise "Plana Ekle" butonu tıklandığında, gezi yeri plana dahil edilmekte ve güncel plana göre yeni rota hesaplanmaktadır. Rota hesaplanması işleminde sistem, planda henüz mevcut gezi yerleri ile öneri gezi yerini sunucu programına ileterek gezi yerlerinin tekrar sıralanmasını sağlamaktadır. Sıralanan gezi yerleri ile tekrar Google Maps kütüphanesi aracılığıyla rota oluşturularak yeni rota harita üzerinde konumlandırılmaktadır. Bu esnada elde edilen güncel plana ait toplam yol uzunluğu ve tahmini toplam süre bilgileri, ortalama gezme süreleri de dahil edilerek, kullanıcıya gösterilmektedir. Kullanıcı bu andan itibaren güncel plan üzerinden geziye devam etmektedir. Aynı işlemler gezinin ilerleyen kısımlarında tekrar ederek gezi sürdürülmektedir.

Kullanıcı son gezi yerine ulaştığında “Planı Tamamla” seçeneği sunulmaktadır. Kullanıcı bu seçeneği seçtiğinde gezi, tamamlanmış olarak veri tabanında kaydedilmekte ve kullanıcı tamamlanmamış planların listelendiği sayfaya yönlendirilmektedir. Bu sayede uygulama, belirli plana ait yaşam döngüsünü sonlandırmaktadır.

7.2.Uygulamanın Kullanımı

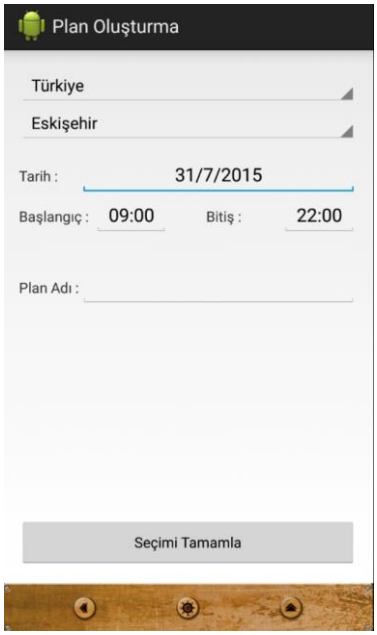
Kullanıcı uygulamayı başlattığında ilk olarak kullanıcı girişi yapması gereken ana sayfayla karşılaşmaktadır. Kullanıcı sistemde henüz kayıtlı olmadığından “Üye Ol” butonuna tıklayarak kayıt olmak üzere “Kayıt” sayfasına yönlendirilmektedir. Bu sayfada gerekli alanları doldurduktan sonra “Kaydet” butonuna tıklayarak kayıt işlemini gerçekleştirmektedir.



Şekil 7.1. Kullanıcı Giriş ve Kayıt Ekranları

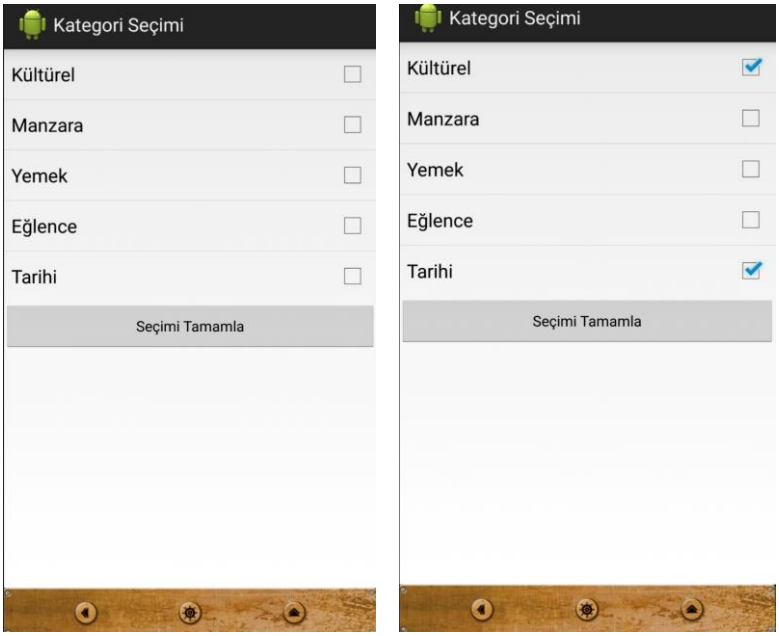
Kullanıcının sistemde kaydı bulunmakta ise kullanıcı girişi yaptıktan sonra “Kayıtlı Planlar” sayfasına yönlendirilmektedir. Kullanıcı bu sayfada geçmişte oluşturduğu ancak henüz tamamlamadığı planları görüntüleyebilmekte ve yeni plan oluşturabilmektedir. Kullanıcının henüz kayıtlı bir planı olmadığından doğrudan “Plan Oluşturma” sayfasına yönlendirilmektedir.

Kullanıcı, “Plan Oluşturma” sayfasında plana ait genel bilgileri doldurarak planı oluşturmaktadır. Türkiye’ye ait Eskişehir ilinde gezi planlamak isteyen kullanıcı, varsayılan gün ve saat bilgilerini değiştirmeyerek planını oluşturmaktadır. Kullanıcı, plana bir isim vermeyerek “Seçimi Tamamla” butununa tıkladığında sistem tarafından plana seçili il ve tarih bilgilerinden otomatik olarak bir isim verilmesine izin vermektedir.



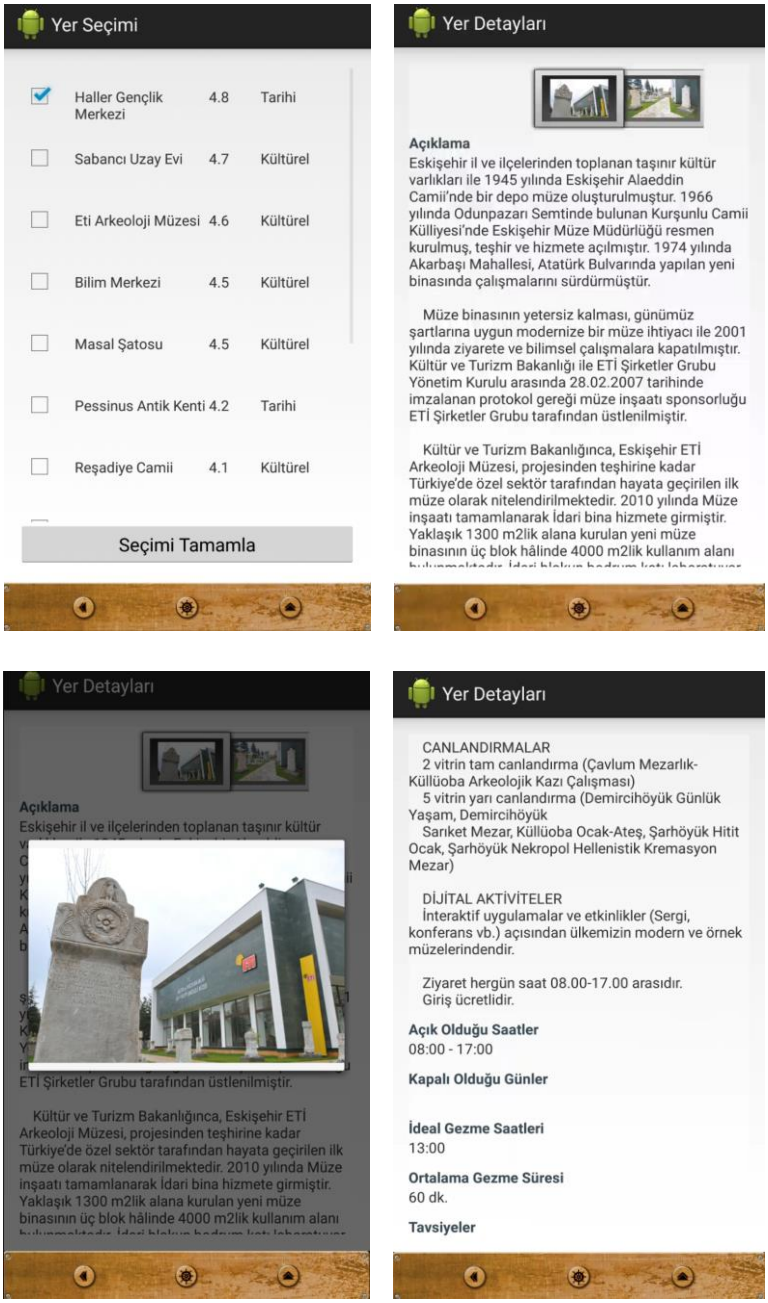
Şekil 7.2. Plan Oluşturma

Plan oluşturma aşamasından sonra kullanıcı “Kategori Seçimi” sayfasına yönlendirilmektedir. Kullanıcı bu aşamada aslında ne türde bir gezi planlamak istediğine karar vermektedir. Kullanıcı bu aşamada kültürel ve tarihi bir gezi yapmak istediğine karar vermekte ve “Seçimi Tamamla” butonuna tıklayarak bu karar doğrultusunda gezi yerlerini seçmek üzere “Yer Seçimi” sayfasına yönlendirilmektedir.



Şekil 7.3. Kategori Seçimi

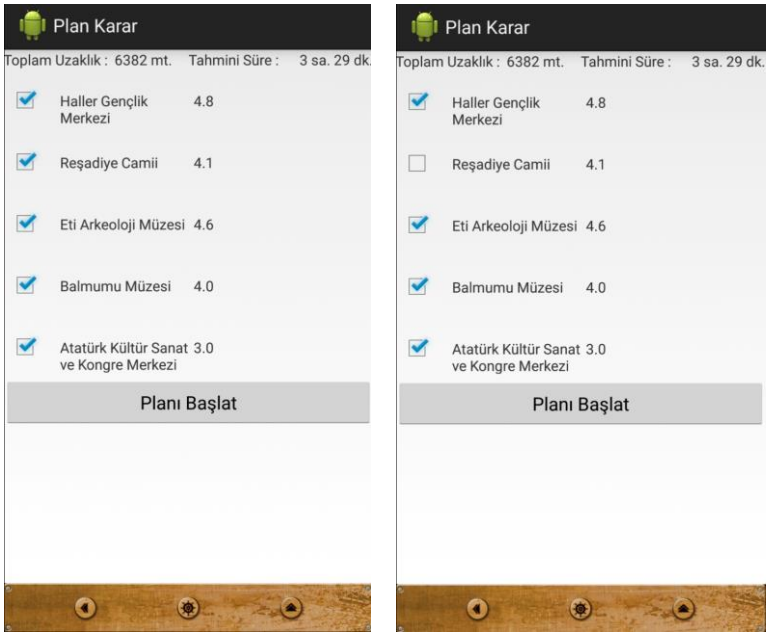
Kullanıcı, "Yer Seçimi" sayfasında seçmiş olduğu kategorilere ait gezi yerlerini görmekte ve listedeki gezi yerinin üzerine tıklayarak "Yer Detayları" sayfasında detaylarını incelemektedir. Kullanıcı bu aşamada gezi yerleri hakkında bilgi edinerek, ilgisini çeken gezi yerlerini plana eklemek üzere seçmektedir. Kullanıcı gezi yeri seçimini tamamladığında "Seçimi Tamamla" butonuna tıklayarak seçili gezi yerleri ile oluşturulan gezi planı hakkında bir ön bilgi almak üzere "Plan Karar" sayfasına yönlendirilmektedir.



Şekil 7.4. Gezi Yeri Seçim ve Detay Ekranları

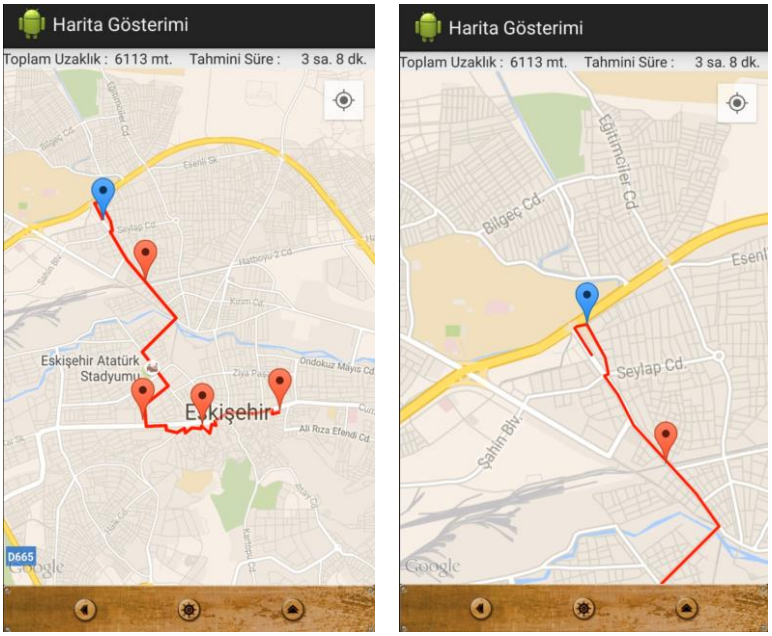
Kullanıcı plan karar aşamasında, seçtiği gezi yerlerini en kısa gezebilmesi için sıralanmış halde görüntülemektedir. Gezi yerleri bu sırada gezilmek üzere ve başlangıç noktası kullanıcının mevcut konumu olarak kabul edildiğinde oluşan gezinin toplam yol uzunluğu ve seçilen gezi yerlerinin ortalama süreleri de dahil tahmini süre bilgileri sayfanın üst kısmında yer almaktadır. Kullanıcı bu bilgileri de

değerlendirerek seçmiş olduğu gezi yerlerinden birini plandan çıkartmaya karar vermektedir. Kullanıcı plan üzerinde yapmış olduğu değişikliklerle "Planı Başlat" butonuna tıklayarak planı başlatabilmektedir. Bir sonraki aşamada plan, yapılan son değişiklikler doğrultusunda güncellenerek genel rota, toplam yol uzunluğu ve tahmini süre bilgileri tekrar hesaplanmaktadır.



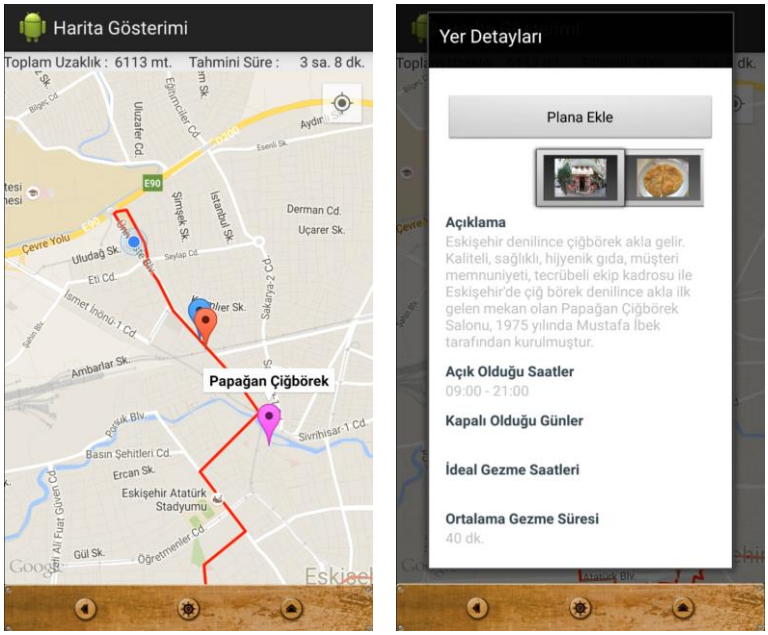
Şekil 7.5. Plan Karar Aşaması

"Harita Gösterimi" sayfasında plandaki güncel gezi yerleri harita üzerinde konumlandırılmaktadır. Üst kısımdaki plana ait bilgiler de mevcut plana göre güncellenmektedir. Kullanıcının anlık bulunduğu konum mavi işaretçi ile plana ait gezi yerlerinin konumları ise kırmızı işaretçilerle harita üzrinde konumlandırılmaktadır. Harita kullanıcının konumu odaklı yakınlaştırılarak kullanıcının konumuna yakın yerleri detaylı olarak inceleyebilmesi amaçlanmaktadır. Kullanıcı hareket etmeye başladığında mavi işaretçinin konumu da harita üzerinde güncellenmektedir.



Şekil 7.6. Planın Harita Üzerinde Gösterimi

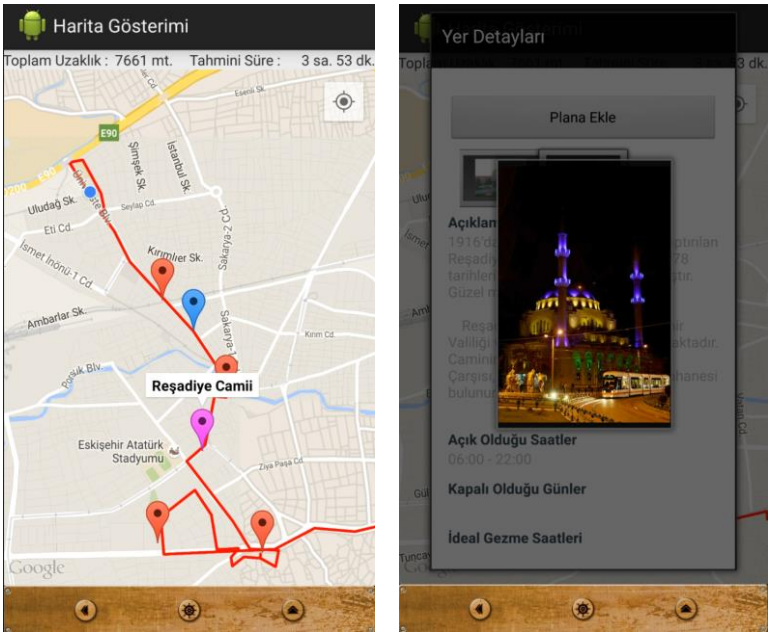
Gezi esnasında kullanıcıya belirli aralıklarla ve zamana bağlı olarak öneriler sunulmaktadır. Kullanıcının önerileri kabul etmesi durumunda önerilen gezi yeri plana eklenerek planın bilgileri güncellenmektedir.



Şekil 7.7. Önerilen Dinlenme Yerinin İncelenmesi ve Plana Eklenmesi

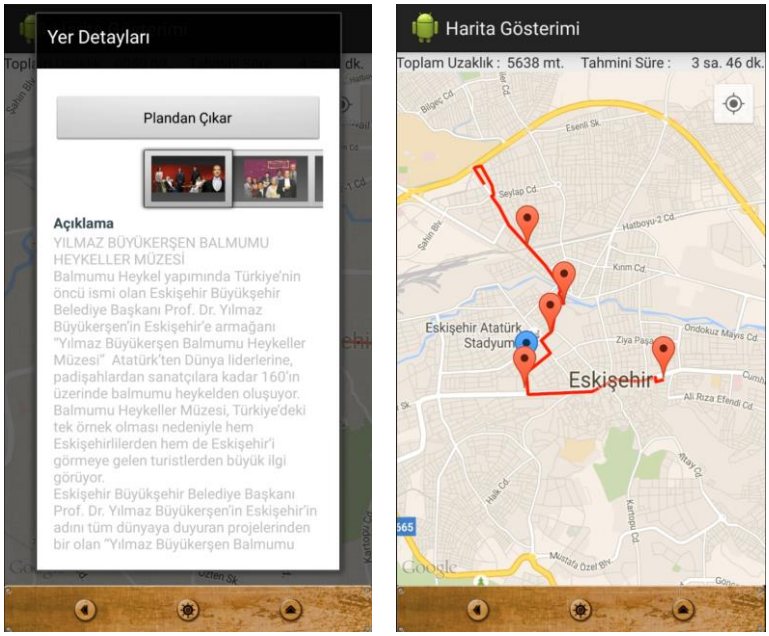
Kullanıcıya gezi esnasında ilk olarak Papağan Çiğbörek adlı yöresel bir restoran kullanıcıya öneri olarak sunulmaktadır. Kullanıcı önerilen gezi yerini inceleyerek hakkında bilgi sahibi olmakta ve plana eklemeye karar vermektedir. Gezi yeri plana eklendiğinde plan ve rota değişmekte, sayfanın üst kısmında bulunan bilgiler de güncel plana göre hesaplanarak güncellenmektedir. Kullanıcı gezisine güncel rota üzerinden devam etmektedir.

Kullanıcı geziye devam ederken yakınlarda bulunan Reşadiye Camii sistem tarafından kullanıcıya öneri olarak sunulmaktadır. Kullanıcının henüz bir uygulama geçmişi olmadığından öneriler, belirli yakınlıkta bulunan ve belirli bir puanın üzerindeki gezi yerlerinden oluşmaktadır. Kullanıcı önerilen gezi yerini inceleyerek plana ekledikten sonra plan tekrar güncellenmektedir.



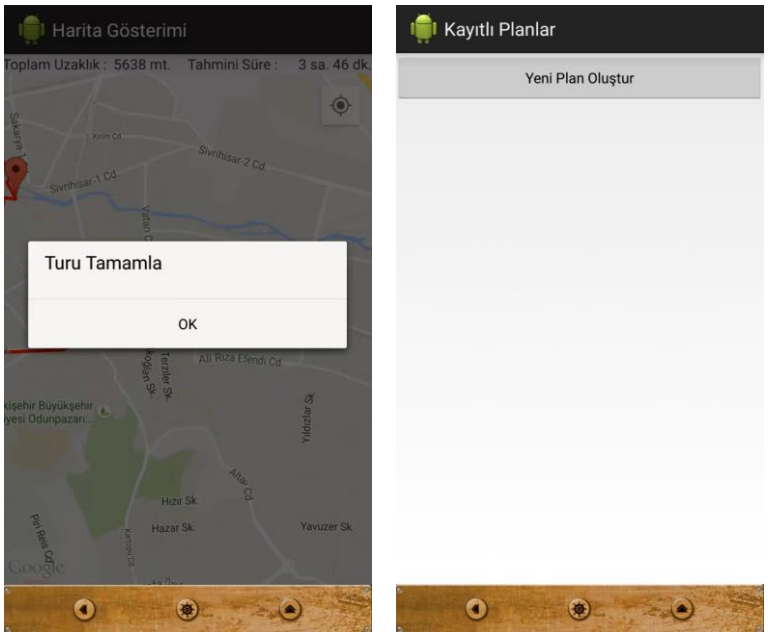
Şekil 7.8. Önerilen Gezi Yerinin İncelenmesi ve Plana Eklenmesi

Kullanıcı gezi esnasında harita üzerindeki işaretçilere tıklayarak gezi yerine ait detay bilgilerine erişebilmektedir. Planda henüz ziyaret edilmemiş gezi yerlerinin detayları incelendiğinde kullanıcı "Plandan Çıkar" butonu ile karşılaşmaktadır. Kullanıcı bu butona tıklayarak detayını incelediği gezi yerini plandan çıkartabilmektedir. Plan tekrar güncellenmekte ve yeni rota güncel gezi yerleriyle oluşturulmaktadır.



Şekil 7.9. Plandan Gezi Yeri Çıkarılması

Kullanıcı güncel rota üzerinden gezisine devam etmektedir. Son gezi yerine ulaştığında "Turu Tamamla" seçeneğini seçerek geziyi tamamlamakta ve tamamlamadığı planların olduğu "Kayıtlı Planlar" ekranına yönlendirilmektedir.



Şekil 7.10. Gezinin Tamamlanması

8. SONUÇ VE ÖNERİLER

Bu çalışmada, yapay zekâ arama algoritmaları ve öneri sistemleri kullanılarak bir mobil gezgin uygulaması geliştirilmiştir. Geliştirilen uygulama kullanıcı profili oluşturarak verimli öneriler sunmayı amaçlamaktadır. Literatürde yapılan araştırmalarda ve web ve mobil platformlardaki mevcut uygulamalarda gezi planı ve rota oluşturarak, kullanıcı takibi gerçekleştiren bir çalışmaya rastlanmamıştır. Geliştirilen uygulamanın özellikleri seyahatin planlı olmasını sağlamakta ve verimini artırmaktadır. Geliştirilen uygulamanın kullanıcılar için daha planlı ve etkili bir seyahat sunması ve şehirler için turizm alanında gelişme sağlaması amaçlanmaktadır.

Uygulamanın çalışma süresinde en kısa rotayı oluşturma ve kullanıcıya sunulan önerilere göre rota güncelleme özellikleri sıklıkla kullanılmaktadır. Bu amaçla rota hesaplama algoritmasının tutarlı ve hızlı çalışması gerekmektedir. Konuyla ilgili literatürde arama algoritmaları ile araştırmalar yapılmış ve çalışmalar incelenmiştir. Benzer problemlerde kullanılan algoritmalardan istenilen koşulları sağlayan A* Algoritması, Karınca Kolonisi Optimizasyonu ve Genetik Algoritma seçilerek üzerlerinde çalışmalar yapılmış ve uygulamanın kullanımına hazır olacak şekilde kodlanmışlardır. Uygulama ile uyumlu çalışan arama algoritmaları muhtemel senaryolarla test edilmiş ve tutarlılık ve çalışma süresi başlıklarında karşılaştırılmıştır. Uygulanan testler sonucunda farklı senaryolarda A* algoritmasından türetilen rota hesaplama algoritması, uygun özellikteki rotaları KKO ve GA'ya göre %70-85 oran aralığında daha kısa çalışma süresi sağlamıştır.

Geliştirilen mobil uygulama turizmi geliştirmeyi ve gezginlerin seyahatlerinden daha fazla verim almasını amaçlamaktadır. Ancak uygulamada gezi yerlerine ait görsel ve açıklayıcı içeriklerin yeterince açıklayıcı ve güvenilir olması gerekmektedir. Bu amaçla pilot şehir olarak seçilen Eskişehir ilinin yerel resmi kurumlarından gezi yerlerine ait koordinat bilgileri, görsel veriler ve açıklama metinleri temin edilmiştir. Bu veriler uygulamanın geliştirilmesi ve kullanılması için yeterli olsa da içeriklerin zenginleştirilmesi turizme sağlanması amaçlanan katkı ve geniş çaplı seyahatlerin verimi açısından daha etkili olacaktır.

İleri zamana yönelik çalışmalarda ticari amaçla geliştirilebilecek uygulamaya eklenebilecek bazı özelliklerin kullanımı kolaylaştırması ve uygulamanın sosyal becerilerini artırması öngörülmektedir. Bunlardan biri benzer kullanıcı profilleri için plan önerisidir. Bu özellikte kullanıcı kendi profiline benzer başka kullanıcıların önceden tamamlamış oldukları planları görüntüleyebilmekte ve aynı planı uygulayabilmektedir. Mevcut sistemde tahmini olarak sunulan toplam yol uzunluğu ve toplam süre bilgileri yerine önceki planların gerçek verileri kullanılarak kullanıcıya karşılaştırma yapabileceği veriler sağlanabilecektir. Diğer bir özellikte ise kullanıcı gerçekleştirdiği planları sosyal medya hesaplarında paylaşabilecek ve ziyaret etmiş olduğu gezi yerlerini önceden ziyaret eden sosyal platformdaki arkadaşlarını görüntüleyebilecektir.

KAYNAKLAR

[1] <http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694> Erişim Tarihi: Mayıs 2015.

[2] <http://www.idc.com/getdoc.jsp?containerId=prUS25450615> Erişim Tarihi: Mayıs 2015.

[3] <http://en.wikipedia.org/wiki/Google_Play> Erişim Tarihi: Mayıs 2015.

[4] <http://en.wikipedia.org/wiki/App_Store_(iOS)> Erişim Tarihi: Mayıs 2015.

[5] Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook* (pp. 73-105). Springer US.

[6] <http://www.eskisehir.bel.tr/> Erişim Tarihi: Mayıs 2015.

[7] <http://www.eskisehirkulturturizm.gov.tr/> Erişim Tarihi: Mayıs 2015.

[8] <http://www.eskisehirkentrehberi.com/> Erişim Tarihi: Mayıs 2015.

[9] <http://tr.wikipedia.org/wiki/Midas_An%C4%B1t%C4%B1_%28Yaz%C4%B1l%C4%B1kaya%29> Erişim Tarihi: Temmuz 2015.

[10] <http://www.eskisehirkulturenvanteri.gov.tr/detay.aspx?ID=19&turID=1> Erişim Tarihi: Temmuz 2015.

[11] <http://www.eskisehirkulturenvanteri.gov.tr/halkkulturudetay.aspx?ID=32> Erişim Tarihi: Temmuz 2015.

[12] <http://www.odunpazari.bel.tr/Projeler.aspx?ID=28> Erişim Tarihi: Temmuz 2015.

[13] Gözüdeli, Y. (2009), *Yazılımcılar için SQL Server 2008 & veritabanı programlama*. Seçkin Yayıncılık, pp. 99-102.

[14] Gargenta, M. (2011), *Learning android*, " O'Reilly Media, Inc.", pp. 7-14.

[15] Kuzmanovic, N., Maruna, T., Savic, M., Miljkovic, G., & Isailovic, D. (2010, June). Google's android as an application environment for DTV decoder system. In *Consumer Electronics (ISCE), 2010 IEEE 14th International Symposium on* (pp. 1-5). IEEE.

[16] Svennerberg, G. (2010). "*Beginning Google Maps API 3"*. Apress, pp. 1-6.

[17] Aquino, M. (2003). A Simple Data Access Layer using Hibernate. *Object Computing, Inc., http://www.ociweb.com/jnb/jnbNov2003.html,(Nov. 19, 2003), 13*.

[18] Bauer, C., & King, G. (2005). Hibernate in action.

[19] Linwood, J., & Minter, D. (2010), *Beginning Hibernate* (pp. 1-9), Apress.

[20] Burke, B. (2009). *RESTful Java with JaX-RS*. " O'Reilly Media, Inc.". pp. 1-13

[21] Hamad, H., Saad, M., & Abed, R. (2010). Performance Evaluation of RESTful Web Services for Mobile Devices. *Int. Arab J. e-Technol.*, *1*(3), pp. 72-78.

[22] <http://json.org/> Erişim Tarihi: Temmuz 2015.

[23] Crockford, D. (2008). *JavaScript: The Good Parts: The Good Parts*. " O'Reilly Media, Inc.". pp. 136-145

[24] Held, M., Hoffman, A. J., Johnson, E. L., & Wolfe, P. (1984). Aspects of the traveling salesman problem. *IBM journal of Research and Development*, *28*(4), 476-486.

[25] Maredia, A. (2010). *History, Analysis, and Implementation of Traveling Salesman Problem (TSP) and Related Problems* (Doctoral dissertation, University of Houston-Downtown).

[26] Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, *2*(4), 393-410.

[27] Applegate, D. L., Bixby, R. E., Chvátal, V., Cook, W., Espinoza, D. G., Goycoolea, M., & Helsgaun, K. (2009). Certification of an optimal TSP tour through 85,900 cities. *Operations Research Letters*, *37*(1), 11-15.

[28] Russell, S., Norvig, P., "A modern approach. Artificial Intelligence", *Prentice-Hall*, pp. 63-95, 1995.

[29] Joyner, D., Van Nguyen, M., & Cohen, N. (2010). Algorithmic Graph Theory.*Google Code,* pp. 72-76

[30] Zhan, F. B. (1997). Three fastest shortest path algorithms on real road networks: Data structures and procedures. *Journal of geographic information and decision analysis*, *1*(1), 69-82.

[31] Yao, J., Lin, C., Xie, X., Wang, A. J., & Hung, C. C. (2010, April). Path planning for virtual human motion using improved A* star algorithm. In Information Technology: New Generations (ITNG), 2010 Seventh International Conference on (pp. 1154-1158). IEEE.

[32] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. Systems Science and Cybernetics, IEEE Transactions on, 4(2), 100-107.

[33] Hu, J., gen Wan, W., & Yu, X. (2012, July). A pathfinding algorithm in real-time strategy game based on Unity3D. In Audio, Language and Image Processing (ICALIP), 2012 International Conference on (pp. 1159-1162). IEEE.

[34] Zou, H., Zong, L., Liu, H., Wang, C., Qu, Z., & Qu, Y. (2010, June). Optimized application and practice of A* algorithm in game map path-finding. In Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on (pp. 2138-2142). IEEE.

[35] Dorigo, M., Stützle, T. (2004). Ant Colony Optimization. MIT Press. Cambridge, MA.

[36] Bachir, B., Ali, A., & Abdellah, M. (2012). Multiobjective optimization of an operational amplifier by the ant colony optimisation algorithm. *Electrical and Electronic Engineering*, *2*(4), 230-235.

[37] Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

[38] Dianati, M., Song, I., & Treiber, M. (2002). *An introduction to genetic algorithms and evolution strategies*. Technical report, University of Waterloo, Ontario, N2L 3G1, Canada.

[39] Alabsi, F., & Naoum, R. (2012). Comparison of Selection Methods and Crossover Operations using Steady State Genetic Based Intrusion Detection System. *Journal of Emerging Trends in Computing and Information Sciences*,*3*(7), 1053-1058.

[40] Goldberg, D. E., & Lingle, R. (1985, July). Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications* (pp. 154-159). Lawrence Erlbaum Associates, Publishers.

[41] Emel, G. G., & TAŞKIN, Ç. (2002). Genetik Algoritmalar ve Uygulama Alanlari.*Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, *21*(1), pp. 129-152.

[42] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *26*(1), pp. 29-41.

[43] Gaertner, D., & Clark, K. L. (2005, June). On Optimal Parameters for Ant Colony Optimization Algorithms. In *IC-AI* (pp. 83-89).

[44] Roeva, O., Fidanova, S., & Paprzycki, M. (2013, September). Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on* (pp. 371-376). IEEE.

[45] Sinthamrongruk, T., Mahakitpaisarn, K., & Manopiniwes, W. (2013). A Performance Comparison between A* Pathfinding and Waypoint Navigator Algorithm on Android and IOS Operating System. *International Journal of Engineering and Technology, 5*(4), pp. 498-501.

[46] Santoso, L. W., Setiawan, A., & PRAJOGO, A. K. (2010). *Performance Analysis of Dijkstra, A* and Ant Algorithm for Finding Optimal Path Case Study: Surabaya City Map* (Doctoral dissertation, Petra Christian University).

[47] Mehlhorn, K., & Sanders, P. (2008). Minimum Spanning Trees. *Algorithms and Data Structures: The Basic Toolbox*, pp. 217-232.

[48] Aydin, A., Telceken, S., "The comparison of A* algorithm and Ant Colonial Optimization for mobile traveler application", *Signal Processing and Communications Applications Conference (SIU), 2015 23th*, pp. 491-494, 16-19 May 2015.

[49] Aydin, A., Telceken, S. "Artificial Intelligence Aided Recommendation Based Mobile Trip Planner For Eskisehir City", *Industrial Electronics and Applications(ICIEA), 2015 10th*, 15-17 June 2015.