

**T.C.**

**FIRAT ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ**

**AKILLI PARK SİSTEMİ**

BİTİRME PROJESİ

**HAZIRLAYAN**

Suat POLAT

Volkan DEMİRKOL

**BİTİRME YÜRÜTÜCÜSÜ**

Yrd. Doç. Dr. Fatih ÖZKAYNAK

**ELAZIĞ – 2017**

Bu çalışma Fırat Üniversitesi, Teknoloji Fakültesi, Yazılım Mühendisliği Bölümüne Bitirme Ödevi Yükümlülüğünü yerine getirmek amacıyla sunulmuştur.

21/01/2017

Suat POLAT - Volkan DEMİRKOL

**Proje Yöneticisi:**

Yrd. Doc. Dr. Fatih ÖZKAYNAK **Verdiği Not :** …

**Jüri Üyeleri:**

**Proje Yöneticisi Üye Üye**

Yrd. Doc. Dr. Fatih ÖZKAYNAK

**Bölüm Başkanı**

Prof. Dr. Asaf VAROL

İçindekiler

[Özet 4](#_Toc472775074)

[GenelBakış 5](#_Toc472775075)

[Projenin Amacı 6](#_Toc472775076)

[Projenin Hedef ve Başarı Kitleleri 6](#_Toc472775077)

[Projenin Kapsamı 6](#_Toc472775078)

[1.PROJE PLANLAMASI 7](#_Toc472775079)

[1.1 Proje İş-Zaman Grafiği 7](#_Toc472775080)

[1.2 Proje Ekibi 8](#_Toc472775081)

[1.3 Kullanılacak Ortam ve Araçlar 8](#_Toc472775082)

[1.4 Test Planı 9](#_Toc472775083)

[1.5 Bakım Planı 9](#_Toc472775084)

[1.6 Proje Yöntem ve Metotları 9](#_Toc472775085)

[1.7 Kalite Sağlama Planı 9](#_Toc472775086)

[1.8 Konfigürasyon Yönetim Planı 10](#_Toc472775087)

[2. SİSTEM ÇÖZÜMLEME 10](#_Toc472775088)

[2.1 Var Olan Donanım ve Yazılım Kaynakları 11](#_Toc472775089)

[2.2 Veri Modelleri 11](#_Toc472775090)

[2.2.1 Use Case Diyagramı(UML) 11](#_Toc472775091)

[3. SİSTEM TASARIMIN GERÇEKLEŞTİRİLMESİ 11](#_Toc472775092)

[3.1 Kullanıcı Arabirimi 12](#_Toc472775093)

[4. GERÇEKLEŞTİRİM 12](#_Toc472775094)

[4.1Yazılım Geliştirme Ortamları 12](#_Toc472775095)

[4.1.1 Android IDE 12](#_Toc472775096)

[4.1.2 Sublime Text 12](#_Toc472775097)

[4.1.3 Ubuntu 13](#_Toc472775098)

[4.2 Açıklama Satırı 13](#_Toc472775099)

[4.3 Projenin İşleyişi 13](#_Toc472775100)

[4.3.1 Kullanıcı Kayıt İşlemi 13](#_Toc472775101)

[4.3.2 Kullanıcı Giriş İşlemi 14](#_Toc472775102)

[4.3.3 Mesafelerin Hesaplanması 15](#_Toc472775103)

[4.3.4 En Yakın 4 Lokasyonun Getirilmesi 18](#_Toc472775104)

[4.3.5 Kamera Durum Güncellemesi 19](#_Toc472775105)

[4.3.6 Araç Tanıma 19](#_Toc472775106)

[4.4 Projenin Kullanımı 21](#_Toc472775107)

[4.4.1 Kayıt Ekranı 21](#_Toc472775108)

[4.4.2 Giriş Ekranı 21](#_Toc472775109)

[4.4.3 Önerilen Park Yerleri 22](#_Toc472775110)

[4.4.4 Seçilen Lokasyona Rota Çizimi 22](#_Toc472775111)

[5.DOĞRULAMA VE GEÇERLEME 23](#_Toc472775112)

[5.1 Giriş 23](#_Toc472775113)

[5.2 Sınama Kavramları 23](#_Toc472775114)

[5.3Doğrulama ve Geçerleme Yaşam Döngüsü 24](#_Toc472775115)

[5.3.1 Sınama Yöntemleri 24](#_Toc472775116)

[5.4 Sınama ve Bütünleştirme Stratejileri 24](#_Toc472775117)

[5.5 Sınama Belirtimleri 25](#_Toc472775118)

[5.6 Yaşam Döngüsü Boyunca Sınama Etkinlikleri 25](#_Toc472775119)

[6.BAKIM 26](#_Toc472775120)

[6.1 Giriş 26](#_Toc472775121)

[6.2 Kurulum 26](#_Toc472775122)

[6.3 Yerinde Destek Organizasyonu 26](#_Toc472775123)

[6.4 Yazılım Bakımı 27](#_Toc472775124)

[6.4.1 Bakım Süreç Modeli 27](#_Toc472775125)

[7.SONUÇ 28](#_Toc472775126)

[8.KAYNAK 28](#_Toc472775127)

# Özet

AKILLI PARK SİSTEMİ

YAZILIM MÜHENDİSLİĞİ

Suat POLAT - Volkan DEMİRKOL

FIRAT ÜNİVERSİTESİ TEKNOLOJİ FAKÜLTESİ

Sistem, bir problemi belirli disiplin ve kurallar çerçevesinde çözüme ulaştırmak isteyen yapılara denir. İnsanların arz ve talepleri sonrasında bir istek veya bir problem ortaya çıkmaktadır. Sistemler ise bu problem veya isteklere çözüm bulmaktadır.

Akıllı sistemler, günümüzün bir parçası haline gelmiş gündelik işleri kolaylaştıran yapılara denir. Akıllı sistemler 20.yüzyılın bir parçası haline gelmiş olup hayatın neredeyse vazgeçilmezi haline gelmiştir. Akıllı sistemlerin dünyada en çok kullanılanı akıllı telefonlardır. Dünyada 3,790 milyar kullanıcıya sahip olan bu akıllı sistem, insanlara taşınabilirlik, anında iletişim, anında veriye ulaşabilme ve bağlana bilirlik gibi özellikler sağlamaktadır. Akıllı sistemler insanların hayatını kolaylaştıran bir yapı haline gelmiş ve insanlar tarafından benimsenmiştir.

Bizde bu kapsamda Akıllı sistemleri kullanarak araç sahiplerinin, park sorununu kaldırmaya yönelik çözümler üretmeye çalıştık.

# ****Genel Bakış****

Akıllı park sistemi projesi, araç sahibi bireyler için park yeri bulmayı daha kolay ve çabuk bir biçimde gerçekleştirmesi için yapılan bir çalışmadır.

# Projenin Amacı

Teknolojinin hayatımızın her alanını kapsadığı bu günümüzde araç sahiplerinin hızlı bir şekilde park yeri bulabilmesi önemlidir. Bu proje sisteme kayıt yaptıran herkesin kendine en yakın park yerini bulması için geliştirilmekte olan bir projedir.

# Projenin Hedef ve Başarı Kitleleri

* Araç sahibi bireylerin daha çabuk park yeri bulmasını sağlamak.

# Projenin Kapsamı

* **Projenin Kapsamının Başlatılması.**

Bu kısımda kapsam yönetim planı gerçekleştirilerek yetki ve atamaların sağlanması

* **Kapsam Planlama**

Kapsamın sınırları oluşturularak, projede olması gerekenler ve olmaması gerekenleri belirlenir.

* **Kapsam Tanımlama**

Projenin ölçülebilir kurumsal değerlerini destekleyen proje ara ürünlerinin belirlenmesi sağlanır.

* **Kapsam Doğrulama**

Projenin kapsamının doğru, tam olduğunu ve ölçülebilir kurumsal değerleri desteklediğinin doğrulanmasının yapıldığı kısımdır.

* **Kapsam Değişimi Denetimi**

Proje kapsamı oluşturulduğunda, kapsam değişikliklerini yönetecek denetimlerin yerinde olmasını ve yordamların tüm paydaşlar tarafından değerlendirilmesini sağlamak amaçlı oluşturulan kısımdır.

# 1.PROJE PLANLAMASI

Projede yapılacak işlerin iş adımları, zamanlaması, ara ürünlerin neler olacağı ve ne zaman üretileceği türündeki bilgiler proje zaman-iş planı bölümünde yer alacaktır. Bu plan proje geliştirme döngüsü sürecinde neler yapılacağını zaman boyutuyla içerir. Bu planların nasıl gerçekleştirileceğini gösterir. Bu aşamada aşağıdaki maddeler yapılmıştır.

* Proje gereksinimlerini toplama
* Kapsam tanımlama
* İş döküm ağacı oluşturma
* Kapsam doğrulama ve geri bildirim alma
* Kapsam gözlemleme ve kontrol etme

Şekil 1:Proje Planlaması

## 1.1 Proje İş-Zaman Grafiği

Şekil 2:Projenin Geliştirme Süreçler

## 1.2 Proje Ekibi

Suat POLAT;

* + Projenin Android kısmından,
  + Projenin Web Service kısmından,
  + Projenin tasarımından,
  + Proje yönetiminden

Sorumludur.

Volkan DEMİRKOL;

* + Projenin Image Processing kısmından,
  + Projenin tasarımından

Sorumludur.

## 1.3 Kullanılacak Ortam ve Araçlar

* Proje dokümantasyon ve raporlama için MS Word araçları kullanılacaktır.
* Stackoverflow, Github, Tutorialspoint, Pyimagesearch gibi sitelerden yardım alındı.
* Ubuntu işletim sistemi üzerinde geliştirildi.

## 1.4 Test Planı

Proje her modülü oluşturulduktan sonra test edilir eğer doğrulama ve geçerlilik testlerinden başarısız olunursa tekrar başa dönülerek modül için sistem tasarımları tekrardan gözden geçirilir. Bu testler trafiğe açık alanda test aracıyla gerçekleştirilecektir. İstekler tekrardan tanımlanır ve modül gerekirse tekrardan yazılır. Eğer doğrulama ve geçerlilik testlerinden geçerse bir sonraki adıma geçilir.

## 1.5 Bakım Planı

Düzenli olarak yapılacak bakım sürecinde aşağıdaki maddelere uyulacaktır;

* Eklenecek yeni teknolojiler.
* Eklenecek yeni donanımlar.
* Sistem güncellemelerine uygunluk.

## 1.6 Proje Yöntem ve Metotları

Projemize Çevik Yazılım Yöntemini uygun gördük. Çünkü bu yöntemde takım çalışması çok ön plana çıkıyor. En büyük avantajının bu olduğunu düşünüyoruz. Değişikliklere sürekli olarak uyum sağlayabilme, adım adım ilerleme ve liderlik psikolojisi, sorumluluk, örgütlenme hep ön planda. Geliştirme ortamının uygunluğu açısından bu yöntemle projemize devam edeceğiz

## 1.7 Kalite Sağlama Planı

Kalite Uzmanı ve Teknik Ekip Liderine bağlı çalışan Kalite Güvence ve Test Ekibi tarafından yapılan kontrollerinden sonra yapılması gereken değişliklerle proje Kalite Sağlamlık onaylanır. Bu doğrultuda bu ekibin yapacağı işlemler sırasıyla;

Test senaryolarını iş süreçlerinin gerektirdiği koşulları dikkate alarak detaylı bir şekilde oluşturmak, dokümantere etmek ve Teknik Ekip Lideri’nebildirmek,

* + - Testler sırasında gerçekleşebilecek hataların belirli standartlar altında dokümanter edilmesi için, gerekli şablonları ve rapor formatlarını hazırlamak,
    - Test senaryoları çerçevesinde, yazılımı biten modüllerin Kalite Kontrol Testlerini gerçekleştirmek,
    - Kalite Kontrol Testlerinden geçen modüllerin Kullanıcı Kabul Testlerini, İş Süreç Analizi Sorumluları ile koordineli olarakgerçekleştirmek,
    - Yazılım Geliştirme Ekibi tarafından geliştirilen ve testleri tamamlanan yazılımın gerçek ortama geçiriliş aşamasında, Yazılım Geliştirme Ekibi ile birlikte planlamayı hazırlamak, bu konuda ilgili birimler ile koordinasyonu sağlamak,olmalıdır.

Bu olaylar sonucunda Kalite Uzmanın oluşturduğu rapordan yola çıkarak değiştirmesi olağan olan bölümler değiştirilir. Diğer bölümler ise Proje Yöneticisi ve Yazılım Ekip Liderinin vereceği karar ile düzenlemeler yapılır.

## 1.8 Konfigürasyon Yönetim Planı

Konfigürasyon yönetimi tasarım ve üretim süreçlerindeki işlemlerin kontrol altında bulundurulması, müşterinin talep ettiği bilgilere anında ve kolay ulaşılması, hataların azaltılması ve verimliliğin artırılması önemli katkılar sağlayacaktır.

Konfigürasyon yönetimi ürünü ömür döngüsünde ürünü oluşturan parçaların üretim hatalarına karşı kalite ve performans iyileştirmesine yönelik olarak izlenmesi, düzeltilmesi için gereken mali ,zaman ve iş gücü kaynaklarına önemli ölçüde azaltılmasına katkı sağlamaktadır.

# 2. SİSTEM ÇÖZÜMLEME

## 2.1 Var Olan Donanım ve Yazılım Kaynakları

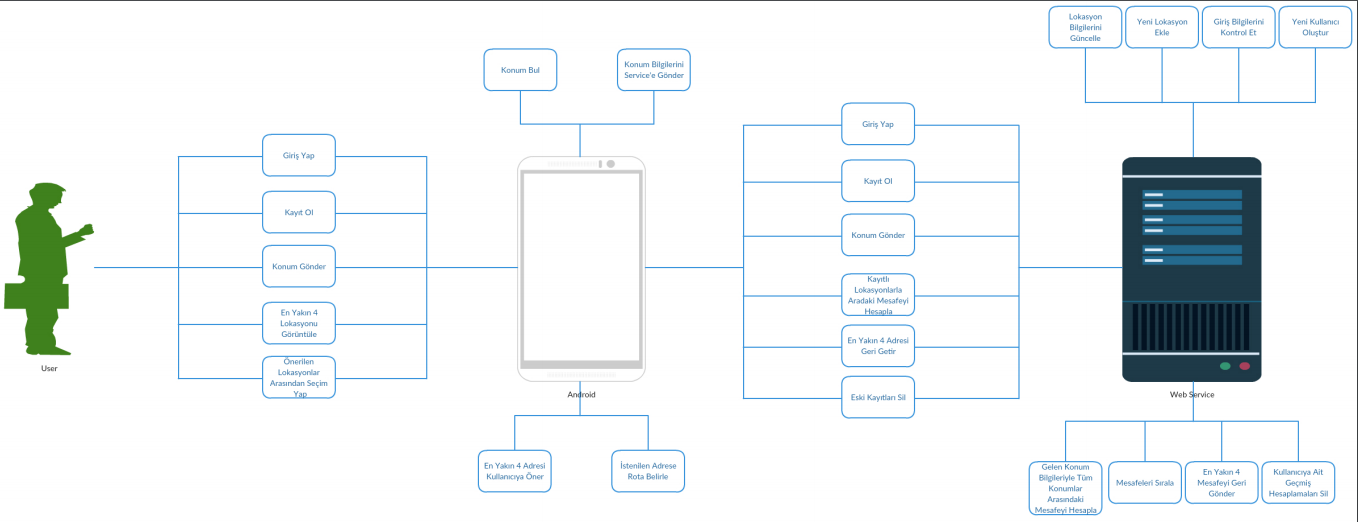
Projede kullanılacak olan yazılımlar ve donanımlar bir ücret teşkil etmektedir. Bu yazılım ve donanımlar tedarikçiler tarafından karşılanacaktır.

Ubuntu işletim sistemi kullanılıp proje bu işletim sistemi üzerinde geliştirilecektir. Ubuntu işletim sisteminin kullanılmasının sebebi ise güncel olmasıdır.

## 2.2 Veri Modelleri

### 2.2.1 Use Case Diyagramı(UML)

Case diyagramları ise sistemin ve sınıfların zamanla değişimini de içerecek biçimdeki diyagramlardır. Bir Use Case modeli Use Case diyagramları ve Use case açıklamaları dediğimiz senaryolardan oluşmaktadır. Use Case diyagramları ise Actors, Use Case ve Use Case'ler arasındaki ilişkilerden oluşmaktadır.



Şekil Use - Case Diyagramı

# 3. SİSTEM TASARIMIN GERÇEKLEŞTİRİLMESİ

Sistemin tasarımı teknoloji ve kullanılabilirliğin göstergesidir. İnsanların kullanımına bırakıldığında büyük ilgi gösterilmesi planlanmıştır. Önümüzde ki yıllar düşünüldüğünde park yeri bulmada problem yaşayan insanlar için gerekli olan sistem tasarım ve kullanım kolaylığı bakımından ön plana çıkması düşünülmektedir. Sistemin taşınabilir olması ve diğer sistemlerle uyumlu olarak çalışması gerekmektedir.

Sistemin temel işlevleri arasında bu ön görüyü sağlamak amacı ile sistem bu temel amaç üzerine geliştirilecektir.

## 3.1 Kullanıcı Arabirimi

Kullanıcı arabirimi Android işletim sistemi olarak belirlenmiştir.

# 4. GERÇEKLEŞTİRİM

Gerçekleştirim çalışması, tasarım sonucu üretilen süreç ve veritabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmalarını içerir. Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.

Söz konusu ortam, kullanılacak programlama dili ve yazılım geliştirme araçlarını içerir. Söz konusu ortamda belirli bir standartta geliştirilen programlar, gözden geçirilir, sınanır ve uygulamaya hazır hale getirilir. Üretilen kaynak kodların belirlenecek bir standartta üretilmesi yazılımın daha sonraki aşamalardaki bakımı açısından çok önemlidir. Tersi durumda kaynak kodların okunabilirliği, düzeltilebilirliği zorlaşır ve yazılımın işletimi süresince ortaya çıkabilecek sorunlar kolayca çözülemez.

## 4.1Yazılım Geliştirme Ortamları

### 4.1.1 Android IDE

Android işletim sistemi üzerinde çalışacak kısımların kodlanması için Android Studio kullanılmıştır.

### 4.1.2 Sublime Text

Uygulamanın diğer platformlar ile iletişime geçmesi, gerekli işlemlerin yapılması için Web Service Sublime Text üzerinden kodlanmıştır.

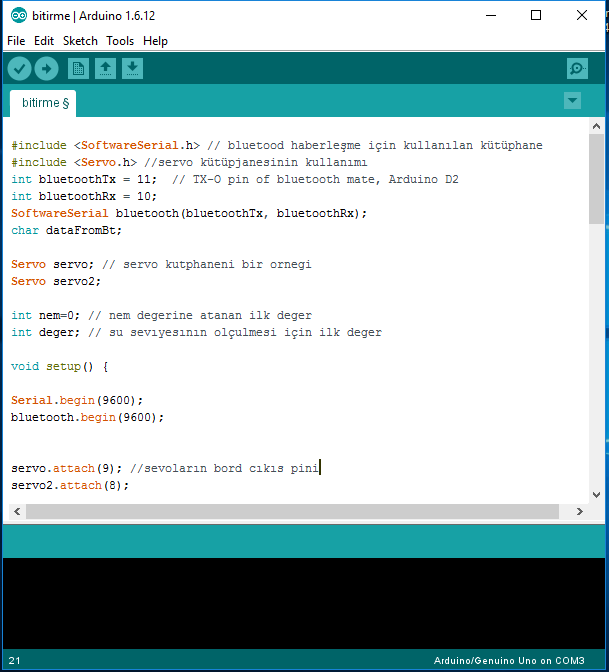
### 4.1.3 Ubuntu

Linux ailesinin son üyesi olan Ubuntu 16.04 işletim sistemi üzerine kurulmuş bir projedir. Ubuntu 'nun seçilmesinin sebebi Python kütüphanelerinin daha optimal çalışabilmesidir.

## 4.2 Açıklama Satırı

Yorum satıları geliştiricilere büyük katkılar sağlayan bir yapıdır. Projenin gerçekleştirme aşamasında kodlar açıklama satırları ile özetlenmiştir.

Program inşa edilirken program kodları arasına kod blokları ile ilgili açıklamalar ya da proje geliştiricilerinin projenin kişinin kodlarımıza baktığında müdahale edebilmesini kolaylaştıran açıklamalar içeren satırlar bütünü.



Şekil 4: Açıklama Satırı

## 4.3 Projenin İşleyişi

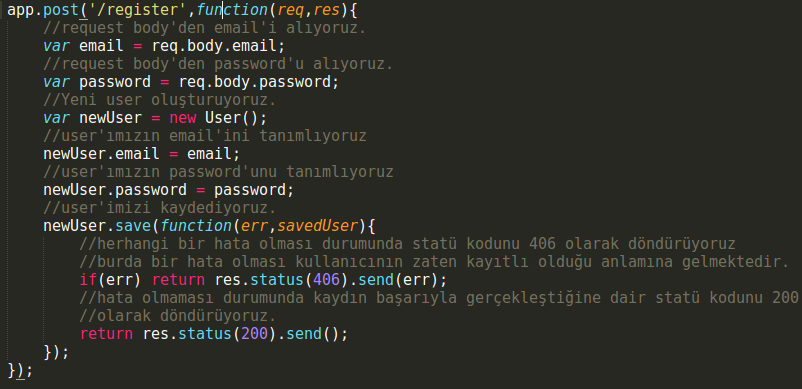
### 4.3.1 Kullanıcı Kayıt İşlemi

Kullanıcı sisteme kayıt olmak için "/register" endpoint'ine “email” ve “password” bilgilerini post request ile gönderdikten sonra parametreler request.body’den alınarak veri taanında kullanıcının kayıtlı olup olmadığı kontrol edilir.

Kullanıcı kayıtlı ise response statü kodu olarak 406 döndürülerek kullanıcının zaten kayıtlı olduğuna dair bilgilendirme yapılmaktadır.

Kullanıcı kayıtlı değilse “mongoose.Schema” ile kullanıcı için tanımlanan Schema modeline göre kullanıcı kayıt işlemi gerçekleştirilir. Kayıt işlemi aşağıdaki gibi gerçekleşmektedir:

* Kullanıcının göndermiş olduğu şifre veri tabanına kaydedilmeden önce şifrelenmektedir. Bunun yapılmasının sebebi bu tür önemli bilgilerin veri tabanında düz metin olarak tutulması yerine şifrelenmiş olarak tutulmasının daha güvenli oluşudur.
* Bcrypt modülünü kullanmamın sebebi ise MD5-SHA1 gibi algoritmaların Brute Force ile çok sayıda request’e maruz kalabilmesi. Ancak Bcrypt algoritmasının bize sunmuş olduğu özelliklerden bir tanesi Brute Force ile gelen istekleri sınırlayabilmesidir (saniyede/dakikada atılan istek).

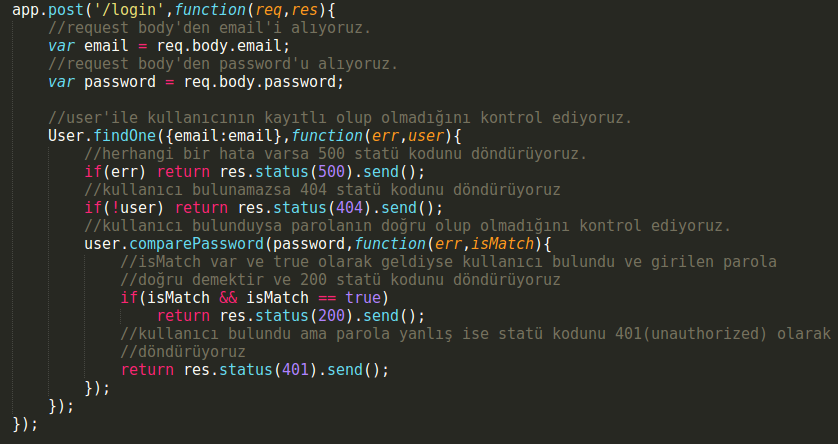


Şekil register Endpoint'i

### 4.3.2 Kullanıcı Giriş İşlemi

Kullanıcı sisteme giriş yaparken “/login” endpoint’ine “email” ve “password” bilgilerini gönderdikten sonra User Schema yardımıyla login kontrolü aşağıdaki gibi yapılmaktadır:

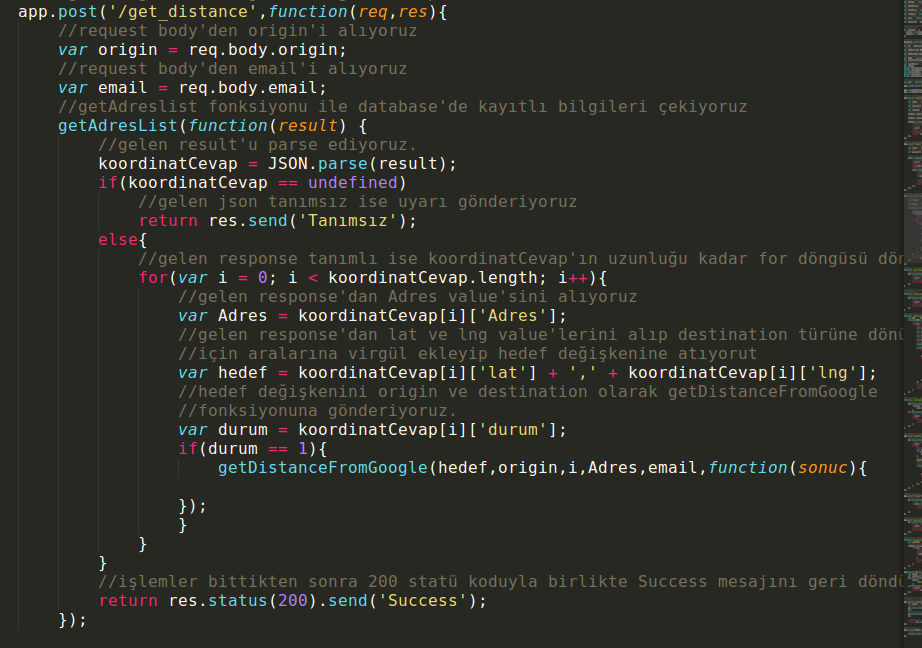
* Kullanıcı bulunamazsa (buraya kadar sadece email adresinin olup olmadığı kontrol edilmektedir) 404 statü kodu döndürülerek kullanıcının bulunmadığına dair bilgilendirme yapılmaktadır.
* Kullanıcı (email) database’de bulunduktan sonra şifre karşılaştırma işlemi için user modelimizin comparePassword fonksiyonuna request.body’den aldığımız “password” değeri gönderilerek şifrenin doğru olup olmadığı kontrol edilir.
* Girilen şifre yanlış ise 401 statü kodu geri döndürülerek kullanıcının girmiş olduğu şifrenin yanlış olduğuna dair bilgilendirme yapılmaktadır.
* Girilen şifre doğru ise 200 statü kodu geri döndürülerek kullanıcının başarılı bir şekilde giriş yaptığına dair bilgilendirme yapılmaktadır.



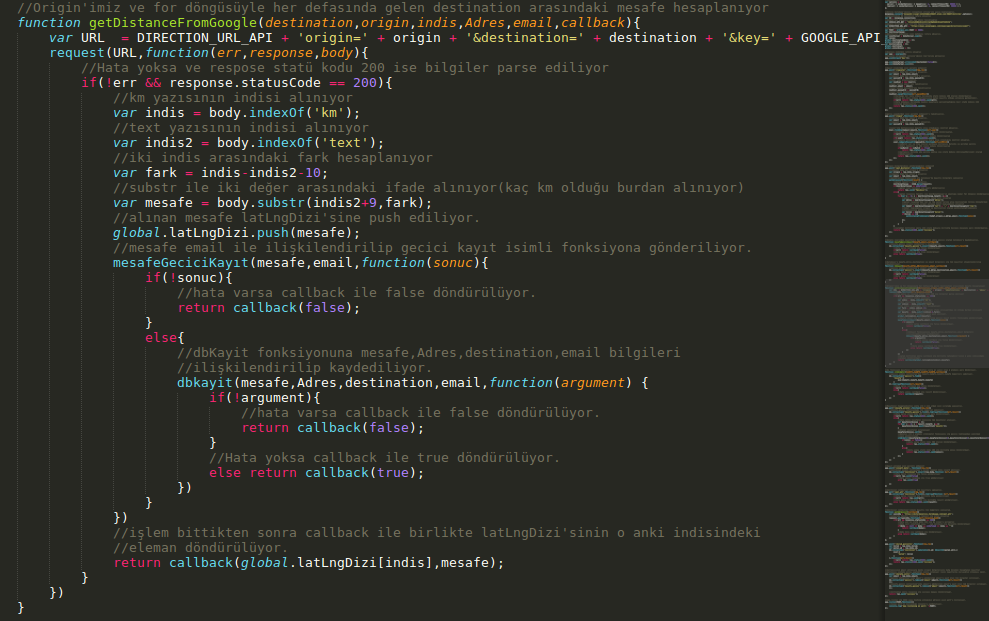
Şekil login Endpoint'i

### 4.3.3 Mesafelerin Hesaplanması

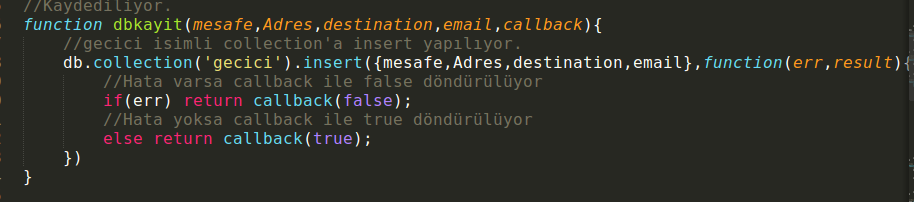
Bu kısımda konumumuzla veri tabanında kayıtlı lokasyonlar arasındaki mesafelerin ölçüm işlemleri yapılmaktadır.

Android uygulamamızdan “/get\_distance” endpoint’ine “origin” ve “email” bilgilerimizi göndererek öncelikle “getAdresList” isimli fonksiyonu çağırarak “/get\_all” endpoint’inden durumu uygun (durum:1) olan tüm lokasyonları çekiyoruz.

Şekil get\_distance Endpoint'i

* Uygun lokasyonları çektikten sonra gelen ham veriden (json data) Adres, lat (latitude/enlem) ve lng (longitude/boylam) bilgilerini parse ederek alıyoruz. Bu adımdan sonra “lat” ve “lng” değerlerini aralarında virgül olacak şekilde birleştirip google’a gönderilebilecek formata dönüştürdükten sonra “getDistanceFromGoogle” metoduna göndererek her bir lokasyon ile aramızdaki mesafeyi ölçüyoruz.
* getDistanceFromGoogle metodu: bu metod kendisine gelen origin ve destination değerlerine göre Google Direction Api Url’e request’de bulunarak tek tek mesafe hesaplaması yapmakta. Mesafeleri hesapladıktan sonra gelen her bi request’in içerisindeki km değerini alarak “mesafeGeciciKayıt” isimli metoda göndererek “mesafe\_gecici” isimli collection’a email ve mesafe bilgilerini ilişkilendirip kaydetmektedir (ilişki kurulmasının sebebi yapılan ölçümlerin hangi kullanıcıya ait olduğunu hesaplamak).

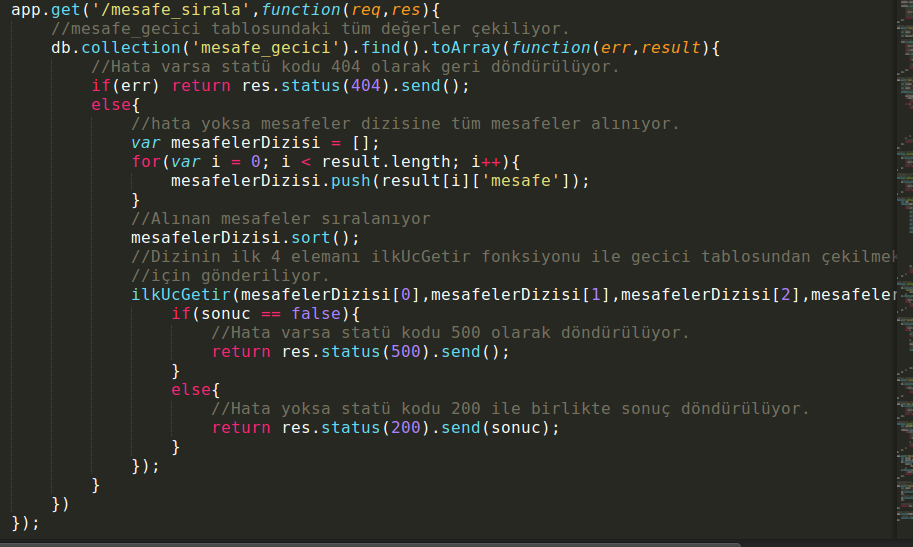
Şekil Get Distance From Google Metodu

* Bu işlemi yapıldıktan sonra tüm bilgiler ilişkilendirilip “dbkayit” fonksiyonu ile “gecici” isimli tabloya kaydedilmektedir.

Şekil dbKayit Endpoint'i

### 4.3.4 En Yakın 4 Lokasyonun Getirilmesi

Bu kısımda daha önce konumumuzla service üzerinde kayıtlı lokasyonlar arasındaki mesafelerin sıralanıp ilk 4 kaydın getirilmesi anlatılmaktadır.

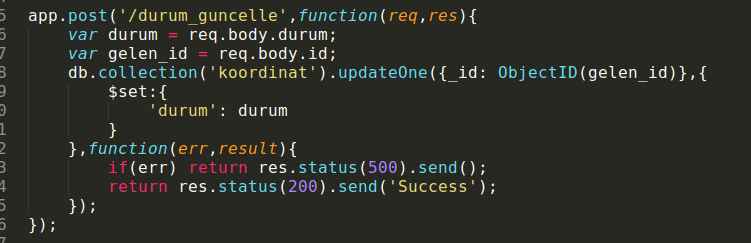
* Android uygulamamızdan “/mesafe\_sirala” isimli endpoint’imize istekte bulunup lokasyonumuza en yakın olan 4 kayıt çekilmektedir.

Şekil mesafe\_siralama Endpoint'i

* ”mesafe\_gecici” isimli collection’dan tüm kayıtları çekip “mesafe” kısmını parse edip değerlerimizi “mesafelerDizisi” isimli diziye push edip mesafelerDizisi.sort() ile dizimizi sıralıyoruz. Ardından dizinin ilk 4 elemanını “ilkUcGetir” isimli fonksiyona gönderip “gecici” tablosundan ilişkilendirilmiş kayıtları uygulamamıza gönderilmektedir.

### 4.3.5 Kamera Durum Güncellemesi

Bu kısımda durumu uygun olan kameranın kendi durumunu service üzerinden 1-0 olarak değiştirmesi anlatılmaktadır.

* Kamera değişiklik olduğu anda (boş park yeri olduğunda) service üzerinden kendi “id”sine göre “durum” sütununu uygunsa 1 değilse 0 olarak güncellemektedir. (“/durum\_guncelle” endpoint’i üzerinden işlemler yapılmaktadır.)

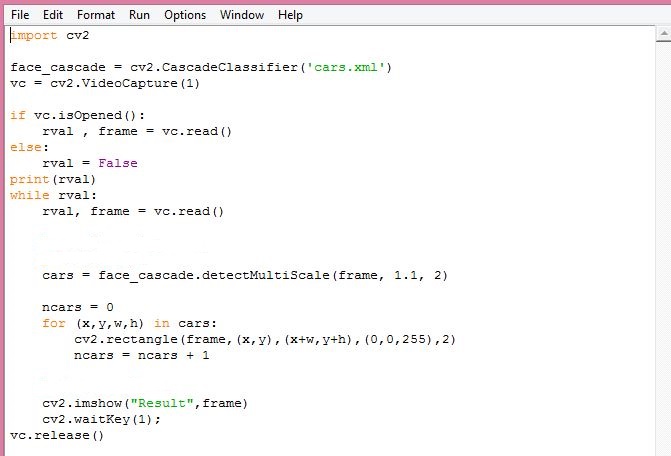
Şekil durum\_guncelle Endpoint'i

* req.body’den durum ve id parametreleri alınarak database’de koordinat collection’undan ilgili kayıt üzerinde güncelleme/update yapılmaktadır.

### 4.3.6 Araç Tanıma

Bu kısımda OpenCV kütüphanesi ile araç tanıma işlemi anlatılmaktadır.

* Kamera belirli aralıklarla uygun park yeri olup olmadığını denetlemektedir. Eğer uygun durum varsa servise bağlanıp durum sütununu 1 yapmaktadır. Uygun durum yoksa durum sütununu varsayılan değer olan 0 yapmaktadır.



Şekil Araba Tanıma İşlemi

## 4.4 Projenin Kullanımı

### 4.4.1 Kayıt Ekranı



Kullanıcıların sistemi kullanabilmeleri için kayıt yaptırmaları zorunludur. Kayıt işlemi aşağıdaki ekrandan kolayca yapılabilmektedir.

Şekil Kayıt Ekranı

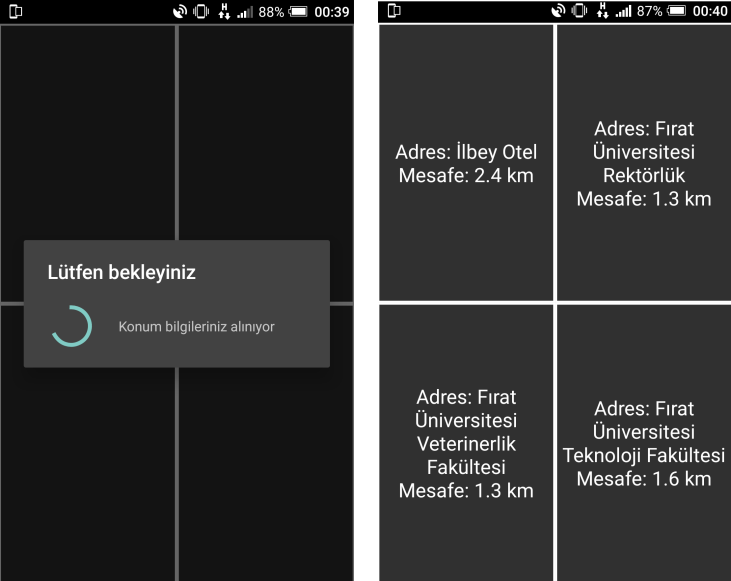
### 4.4.2 Giriş Ekranı



Şekil Giriş Ekranı

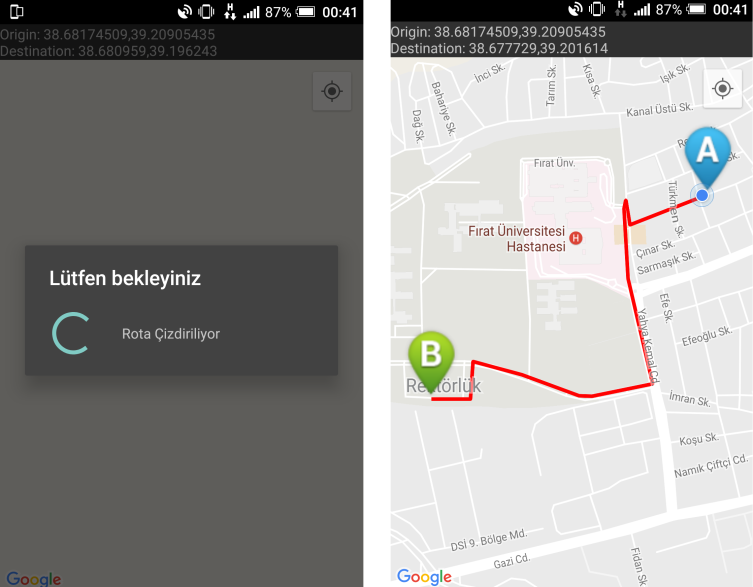
Kullanıcı sisteme kayıt olduktan sonra aşağıdaki ekrandan sisteme giriş yapabilmektedir.

### 4.4.3 Önerilen Park Yerleri

 Kullanıcı sisteme giriş yaptıktan sonra konum bilgileri alınana kadar bekletilir. Ardından kendisine önerilen en yakın 4 park yerinden birini seçer.

Şekil En Yakın 4 Lokasyon

### 4.4.4 Seçilen Lokasyona Rota Çizimi



Şekil Rota Hazırlanana Kadar Kullanıcının Bekletilmesi ve Rotanın Çizdirilmesi

Kullanıcı önerilen park yerlerinden birini seçtikten sonra aşağıdaki gibi konumu ile seçilen hedef arasında rota çizdirilir.

# 5.DOĞRULAMA VE GEÇERLEME

## 5.1 Giriş

Yazılım geliştirme karmaşık bir süreç olduğundan, hataların ortaya çıkması kaçınılmazdır.

• Yazılım, yaşam döngüsünün her aşamasında hatalara karşı sınanması,

• Gereksinimler arasındaki tutarsızlıkların giderilmesi,

• Çözümleme şeması ile uygulama alanı arasındaki uyumsuzlukların giderilmesi,

• Tasarım hatalarının düzeltilmesi,

• Çalışma anı hatalarının giderilmesi,

• Yaşam döngüsünde ilerledikçe, hataların düzeltilmesi,

• İyi bir sınama yaklaşım hataların erken belirlenmesine katkıda bulunacak ve yazılımın kalitesini arttıracaktır.

## 5.2 Sınama Kavramları

Sınama ve bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerinin seçilmesi gerekmektedir.

Sınama işlemleri dört ana sınıfta incelenebilir:

* Birim Sınama

Bağlı oldukları diğer sistem unsurlarından tümüyle soyutlanmış olarak birimlerin doğru çalışmalarının belirlenmesi amacıyla yapılır.

* Alt Sistem Sınama

Alt-sistemler modüllerin bütünleştirilmeleri ile ortaya çıkarlar. Yine bağımsız olarak sınamaları yapılmalıdır. Bu aşamada en çok hata ara yüzlerde bulunmaktadır. Bu yüzden ara yüz hatalarına doğru yoğunlaşmalıdır.

* Sistem Sınama

Üst düzeyde, bileşenlerin sistem ile olan etkileşiminde olan hatalar aranmaktadır. Ayrıca belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır

* Kabul Sınama

Çalıştırılmadan önce sistemin son sınamasıdır. Artık yapay veriler yerine gerçek veriler kullanılır. Bu sınama türü alfa sınaması veya beta sınaması olarak da bilinir.

Alfa sınamada; sistemin geliştirildiği yerde kullanıcıların gelerek katkıda bulunması sistemi test etmesi amaçlanmaktadır.

Beta sınamada; kullanıcı, geliştirilen sistemi kendi yerleşkesinde, bir gözetmen eşliğinde yapar.

## 5.3Doğrulama ve Geçerleme Yaşam Döngüsü

### 5.3.1 Sınama Yöntemleri

* Beyaz Kutu Sınaması

1. Yapılabilecek denetimler arasında:
2. Bütün bağımsız yolların en azından bir kere sınanması,
3. Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması,
4. Bütün döngülerin sınır değerlerinde sınanması,
5. İç veri yapılarının denenmesibulunur.

* Temel Yollar Sınaması

Daha önce çevrimsellik karmaşıklığı konusunda gördüğümüz hesap yöntemi ile bir programdaki bağımsız yollar bulunduktan sonra bu kadar sayıda sınama yaparak programın her birimini bir şekilde sınamalara dâhil etmiş oluruz. Bağımsız yolların saptanması için önce program çizgisel bir biçime çevrilir. Bunu yapmak için iş akış şemaları iyi bir başlangıç noktasıdır.

## 5.4 Sınama ve Bütünleştirme Stratejileri

Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendirilir. Ancak bazı sınama stratejileri bütünleştirme dışındaki tasaları hedefleyebilir. Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sıralandırmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya çıkarır.

## 5.5 Sınama Belirtimleri

Sınama belirtimleri, bir sınama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir.

Bu ayrıtılar temel olarak:

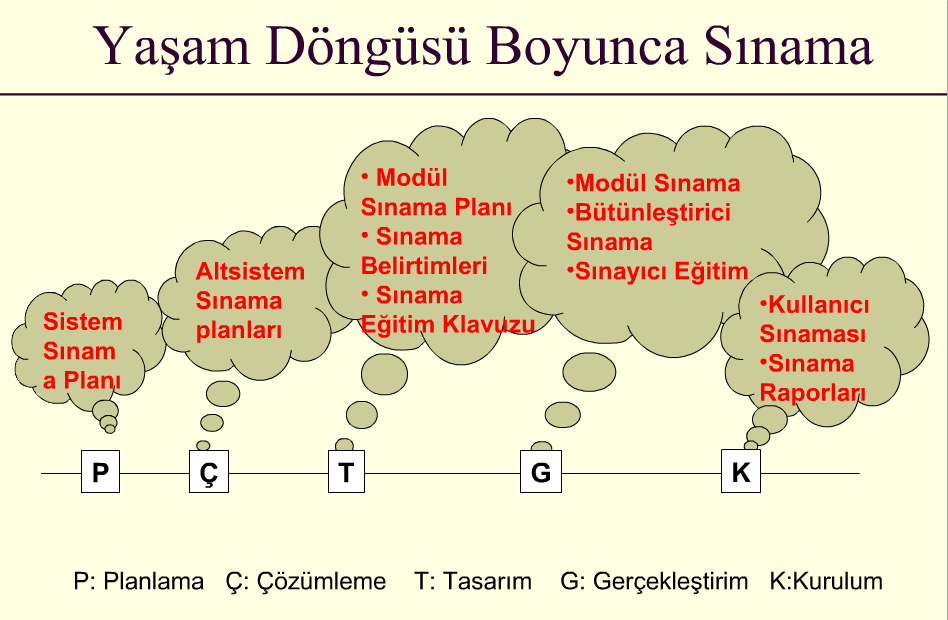
1. sınanan program modülü ya da modüllerinin adları,
2. sınama türü, stratejisi (beyaz kutu, temel yollar vb.),
3. sınama verileri,
4. sınama senaryoları türündeki bilgileri içerir.

Sınama işlemi sonrasında bu belirtimlere,

1. sınamayı yapan,
2. sınama tarihi,
3. bulunan hatalar ve açıklamaları

türündeki bilgiler eklenerek sınama raporları oluşturulur.

## 5.6 Yaşam Döngüsü Boyunca Sınama Etkinlikleri



Şekil 17 : Yaşam Döngüsü Boyunca Sınama

# 6.BAKIM

## 6.1 Giriş

Sınama işlemleri bitirilen yazılımın kullanıcı alanına yüklenmesi ve uygulamanın başlatılması gerekmektedir. Yazılım kullanıma geçtikten sonra, yaşam döngüsünün en önemli ve hiç bitmeyecek aşaması olan "bakım" aşaması başlar. İzleyen kesimlerde, kurulum ve bakım aşamasında yapılması gerekenler açıklanmaktadır. Bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı temel olarak alınmıştır.

## 6.2 Kurulu**m**

* Donanım kurulumu:

Proje için herhangi bir donanım kurulumu gerekmemektedir. Sistem sadece web ortamında çalışacaktır.

* Sistem yazılımı kurulumu:

Projenin çalışabilmesi veya kullanılabilmesi için herhangi bir internet tarayıcısı yeterli olacaktır.

* Veri tabanı kurulumu:

Sistemin çalışması için kullanıcı veya firma bilgisayarlarına herhangi bir veri tabanı kurulumu gerekmemektedir. Sunucuda barındırılacak olan ortak bir veri tabanı üzerinden işlemler yapılacaktır.

* Uygulama yazılımlarının kurulumu:

Sistemin çalışması için herhangi bir uygulama yazılımı kullanılmasına gerek yoktur.

* Eğitim:

Söz konusu eğitim; genel bilgisayar kullanımı eğitimi ve web sitesindeki ara yüzlerinin kullanımı hakkındaki eğitimdir.

## 6.3 Yerinde Destek Organizasyonu

Yerinde destek ekibi olarak yazılım geliştirme ekibimiz hizmet verecektir.

Bu ekibin temel görevleri:

* Kullanıcıları ziyaret ederek sorunlarını belirlemeye çalışmak,
* Giderilebilen kullanıcı sorunlarını gidermek ve giderilemeyenleri üretim sahasındaki uygulama yazılımı destek ekibine iletmek,
* Kullanıcıya işbaşında uygulama eğitimi vermek,
* Kullanıcı sınama günlüklerini toplamak
* Yapılan tüm işlemleri yapılandırma veri tabanına kaydetmekbiçimindedir.

## 6.4 Yazılım Bakımı

Bakım, işletime alınan yazılımın sağlıklı olarak çalışması ve ayakta tutulabilmesi için yapılması gereken çalışmalar bütünü olarak tanımlanır. Bakım yazılım yaşam döngüsünün en önemli ve en maliyetli aşamalarından biridir. Bakım süreci, yazılım yaşam döngüsünde "buzdağının görünmeyen kısmı" olarak adlandırılır. Bakım maliyetleri, zaman zaman üretim maliyetlerinin %60'ını geçer.

Bu bölümde, yazılım bakımı ile ilgili kavramlar açıklanmaktadır. Bu kavramların açıklanmasında IEEE 1219 standardı baz olarak alınmıştır.

### 6.4.1 Bakım Süreç Modeli

Bakım süreç modelinin süreçleri:

* Sorun Tanımlama/sınıflandırma
* Çözümleme
* Tasarım
* Gerçekleştirim
* Sistem Sınama
* Kabul Sunama
* Kurulum

# 7.SONUÇ

Akıllı Park Sistemi adlı projemizde gerçek kullanıcı ile yaptığımız testlerden aldığımız verilere göre %80 lik bir başarı sağlanmıştır. Bu başarımız ile belirlediğimiz hedeflerde ilk temel başarımızın bir kısmını sağlamış olduk.

# 8.KAYNAK

[1] https://github.com

[2] https://www.tutorialspoint.com/

[3] http://stackoverflow.com/

[4] http://www.pyimagesearch.com/

[5] https://nodejs.org/