

**ABANT İZZET BAYSAL ÜNİVERSİTESİ**  
**MÜHENDİSLİK FAKÜLTESİ**

**PARMAK İZİ OKUYARAK SUNUCUDAN KULLANICI GİRİŞİ**  
**VE KAYIT İŞLEMİ PROJESİ**

**Bedir Tuğra KARAABALI**

**LİSANS BİTİRME TEZİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ**

**BOLU, 2024**

**BOLU ABANT İZZET BAYSAL ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**PARMAK İZİ OKUYARAK SUNUCUDAN KULLANICI GİRİŞİ  
VE KAYIT İŞLEMİ PROJESİ**

**Bedir Tuğra KARAABALI**

**LİSANS BİTİRME TEZİ  
BİLGİSAYAR MÜHENDİSLİĞİ**

**BOLU, 2024**

### **DANIřMAN ONAY**

Bu tez alıřması Bolu Abant İzzet Baysal Üniversitesi Mühendislik Fakóltesi Bilgisayar Mühendislięi Bölümü Bitirme Projesi I-II dersleri kapsamında hazırlanmıştır. Lisans Bitirme Tezi olarak uygun bulunmuştur.

Tez Danışmanı

Tarih:

İmza:

## ÖNSÖZ VE TEŞEKKÜR

Teknoloji çağı henüz yeni yeni hayatımıza girdi derken, mühendislerin piyasayı alel acele ayaklandırmasıyla şimdi yeni bir çağa başlıyoruz: yapay zeka çağı. Yapay zeka gelişedursun, şöyle bir geriye baktığımız zaman bilmediğimiz birçok teknoloji ürünü veya altyapısı bizim için oldukça birikmiş ve öğrenilmeyi bekliyor. İlk internet sistemleri, henüz gelişmiş sayılmayan işletim sistemleri, bulut teknolojileri ve entegre sistemlerin çoğu bunlar arasında yer alıyor. Ben de bu projede yapay zekanın oturduğu teknolojinin çalışmasını örneklemek amacıyla ilgili işlemler kullandım. Projede yapay zeka adına bir ibare olmasa da altyapısı ona uygun hazırlandı. Bu proje, kullanıcı girişlerini kontrol etmek ve kayıtları tutmak amacıyla geliştirildi. Ortaya böyle bir projenin çıkması elbette çokça bilgi ve emek gerektirdi. Bu bilgileri edinirken bana akıl hocalığı yapan ve projemde ilham aldığım Doç. Dr. Mustafa Alper Akkaş hocama ve bugüne dek bana desteğini esirgemeyen, kendilerinden çok şey öğrendiğim tüm üniversite hocalarıma, en nihayetinde en büyük destekçim Karaabalı ailesine, özel olarak Fatma Karaabalı'ya sonsuz teşekkür ederim.

## İÇİNDEKİLER

DANIŞMAN ONAY.....	i
ÖNSÖZ ve TEŞEKKÜR.....	ii
İÇİNDEKİLER.....	iii
SİMGELER DİZİNİ VE KISALTMALAR.....	v
ŞEKİLLER LİSTESİ.....	vi
TABLolar DİZİNİ.....	vii
ÖZET.....	vii
ABSTRACT.....	ix
1. GİRİŞ.....	1
1.1 Projenin Amacı.....	1
1.2 Projenin Tanımı.....	2
1.3 Projenin Kapsamı.....	3
1.4 Çalışmanın Önemi.....	4
2. LİTERATÜR TARAMASI.....	4
2.1 Mevcut Teknolojiler ve Çözümler.....	4
2.2 Benzer Çalışmalar.....	5
3. YÖNTEM.....	7
3.1 Donanım Seçimi ve Özellikleri.....	7
3.2 Yazılım Geliştirme.....	9
3.2.1 Yazılım dillerinin önemi.....	9
3.3 Sistem Mimarisi.....	13
3.3.1 Sistem bileşenleri.....	13
3.3.2 Parmak izi sensörü.....	14
3.3.3 PHP tabanlı sunucu.....	15
3.3.4 Sistem iş akışı.....	16
3.3.5 CORS hataları ve çözümleri.....	16
3.3.6 CORS hatalarının nedenleri.....	17
3.3.7 CORS hatalarının çözümleri.....	18
3.3.8 OPTIONS isteklerini yönetme.....	19
3.3.9 Veri akış diyagramları ve sistem şemaları.....	19
3.4 Veritabanı Tasarımı.....	20
3.4.1 Veritabanı tasarımında dikkate alınması gereken husular.....	20
3.4.2 SQL ve NoSQL veritabanları.....	21
3.4.2.1 SQL veritabanları.....	21
3.4.2.2 NoSQL veritabanları.....	22

3.4.2.3 Neden SQL seçildi?.....	22
3.4.3 Tabloların tasarımı.....	23
3.4.4 İyi bir veritabanının tasarımı.....	24
3.4.5 Sonuç.....	25
4. TEST.....	25
4.1 Giriş Olayları.....	27
4.1.1 Giriş olmazsa.....	27
4.1.2 Giriş yanlış olursa.....	27
4.2 Kayıt Olayları.....	27
4.2.1 Kayıt yanlış olursa.....	27
4.2.2 Kayıt girilmezse.....	28
5. SONUÇLAR.....	28
5.1 Genel Değerlendirme.....	29
6. GELECEK ÇALIŞMALAR.....	29
6.1 Gelecekteki Iot Nesneler.....	31
7. KAYNAKLAR.....	33
8. EKLER.....	34
9. ÖZGEÇMİŞ.....	35

## **SİMGELER DİZİNİ VE KISALTMALAR**

ID : Bir veritabanındaki her kayıt için benzersiz bir numara veya dize.

IoT : IoT, internete bağlı olan fiziksel cihazlar, araçlar, ev aletleri ve diğer nesnelerin bir ağını ifade eder.

SQL : SQL veritabanları, verilerin yapılandırılmış tablolarda saklandığı, ilişkisel (relational) veritabanlarıdır.

NOSQL : NoSQL veritabanları, verilerin esnek, yapılandırılmamış veya yarı yapılandırılmış formatlarda saklanmasına izin veren veritabanlarıdır.

Cpp : "C++", genel olarak "C plus plus" olarak okunan ve genellikle "C++" olarak kısaltılan bir programlama dilidir.

## ŞEKİLLER LİSTESİ

Şekil 2.1 Arduino UNO.....	6
Şekil 2.2 RFID.....	6
Şekil 3.1 ESP32.....	7
Şekil 3.2 ESP82.....	8
Şekil 3.3 Parmak İzi Okuma Sensörü.....	9
Şekil 3.4 Javascript.....	10
Şekil 3.5 C++ (cpp).....	11
Şekil 3.6 ESP32 Main Fonksiyon.....	14
Şekil 3.7 Sunucu Ön Yüzü.....	15
Şekil 3.8 Node.js/Express ve Cors.....	18
Şekil 3.8 CORS.....	19
Şekil 3.10 Proxy.....	19
Şekil 3.11 Akış Diyagramı.....	20
Şekil 3.12 SQL Tablo Oluşturma.....	24



## **TABLÖLAR DİZİNİ**

Tablo 3.1 Veritabanı Örneği.....	23
----------------------------------	----

# **PARMAK İZİ OKUYARAK SUNUCUDAN KULLANICI GİRİŞİ VE KAYIT**

## **İŞLEMİ PROJESİ**

Bedir Tuğra Karaabalı

Anahtar Kelimeler: Parmak izi sensörü, sunucu, entegre kart

Özet: Bu makalede, kullanıcı girişi ve kayıt ol kısmı bulunan bir sunucuda parmak izi okuyarak kullanıcı doğrulama ve kayıt işlemlerinin nasıl gerçekleştirileceği incelenmektedir. Proje, entegre karta bağlı parmak izi sensörü kullanarak kullanıcıların kimlik doğrulama işlemlerini yürütmektedir. Kullanıcı girişi seçildiğinde, parmak izi sensörü kullanıcıdan parmak izini alır ve bu izi hafızadaki diğer parmak izleriyle karşılaştırır. Eğer eşleşme sağlanırsa, sunucu ilgili kullanıcının ID'sini geri gönderir ve kullanıcının bilgileri sunucudan çekilerek ekranda gösterilir. Kayıt ol işlemi ise, kullanıcıdan bir fotoğraf çekilmesi ve parmak izinin iki kez okutulması istenir. Bu adımlar, kullanıcının kimlik doğrulama işlemini tamamlamasını sağlar ve başarılı bir şekilde kaydedilir.

# **USER LOGIN AND REGISTRATION PROJECT USING FINGERPRINT RECOGNITION AND SERVER**

Bedir Tuğra Karaabalı

**Keywords:** Fingerprint sensor, server, embedded board

**Abstract:** In a server with user login and registration functionalities, when a user opts to log in, a fingerprint sensor connected to an embedded board captures the user's fingerprint and compares it with the fingerprints stored in its memory. If a match is found, the sensor sends the corresponding user ID back to the server. The server then retrieves and displays the user's information from its database, thereby confirming the user's identity. If the user chooses to register, they are prompted to take a photo and scan their fingerprint twice using the sensor. The registration process involves similar steps as the login process. If all steps are completed correctly, the user is successfully registered.

# 1.GİRİŞ

## 1.1 Projenin Amacı

Bu projenin amacı, parmak izi okuyucu kullanarak kullanıcıların giriş ve kayıt işlemlerini güvenli ve verimli bir şekilde gerçekleştirmektir. Proje, fiziksel bir parmak izi sensörü ile entegre edilmiş bir sunucu altyapısı kullanarak kimlik doğrulama sürecini otomatikleştirmeyi hedeflemektedir.

Kullanıcılar, parmak izlerini kullanarak hızlı ve güvenli bir şekilde sunucuya giriş yapabilecek veya yeni bir hesap oluşturabileceklerdir. Bu sistem, mevcut yöntemlere kıyasla daha güvenli ve kullanışlı bir kimlik doğrulama sağlar, çünkü parmak izi gibi biyometrik veriler, çalınması veya taklit edilmesi zor olan benzersiz özelliklerdir.

Projenin özel hedefleri şunlardır:

**Güvenlik Artışı:** Parmak izi gibi biyometrik veriler kullanarak yetkisiz erişim riskini en aza indirmek.

**Kullanıcı Deneyimini İyileştirme:** Kullanıcıların parmak izlerini kullanarak hızlı ve kolay bir şekilde giriş yapmalarını sağlamak.

**Veri Bütünlüğü ve Doğruluğu:** Kullanıcıların kimlik bilgilerinin doğru ve güvenilir bir şekilde kaydedilmesini ve saklanmasını sağlamak.

**Otomatikleştirilmiş Süreçler:** Parmak izi verilerinin toplanması, karşılaştırılması ve doğrulanması süreçlerini otomatikleştirerek insan hatasını en aza indirmek ve operasyonel verimliliği artırmak.

**Geliştirilebilir Altyapı:** Gelecekte ek biyometrik veriler veya güvenlik katmanları ile kolayca genişletilebilecek esnek ve ölçeklenebilir bir sistem oluşturmak.

Bu hedefler doğrultusunda, parmak izi sensörü, entegre kart ve sunucu arasında güvenli bir iletişim sağlanarak kullanıcının kimlik doğrulama işlemleri güvenilir ve etkin bir şekilde gerçekleştirilmiş olacaktır. Bu proje, hem bireysel hem de kurumsal düzeyde güvenlik gereksinimlerini karşılamaya yönelik önemli bir adım olarak görülmektedir.

## **1.2 Projenin tanımı**

Bu proje, parmak izi teknolojisi kullanılarak kullanıcıların güvenli giriş ve kayıt işlemlerini gerçekleştirmeyi amaçlamaktadır. Proje, bir sunucu üzerinde çalışan bir yazılım ve bu yazılımla entegre edilmiş bir parmak izi sensörü tarafından desteklenmektedir. Kullanıcılar, parmak izlerini kullanarak sunucuya erişim sağlayabilecek ve gerektiğinde yeni bir hesap oluşturabileceklerdir.

Projenin ana işlevleri şunlardır:

**Kullanıcı Girişi:** Kullanıcılar, sunucuya erişmek için parmak izi kimlik doğrulama yöntemini kullanacaklardır. Parmak izi sensörü, kullanıcının parmak izini alacak, sunucuya iletecek ve karşılaştırma işlemi sonucunda kullanıcının kimliğini doğrulayacaktır. Doğrulama başarılı olduğunda, ilgili kullanıcı bilgileri sunucudan alınacak ve kullanıcıya gösterilecektir.

**Kullanıcı Kaydı:** Kullanıcılar, sunucuda yeni bir hesap oluşturmak için parmak izi sensörünü kullanacaklardır. Kayıt işlemi sırasında, kullanıcıdan bir fotoğraf çekilip parmak izi alınacak ve bu veriler sunucuya iletilerek kayıt işlemi tamamlanacaktır. Kayıt işlemi başarılı olduğunda, kullanıcının bilgileri sunucuda kaydedilecek ve ilgili kullanıcıya bildirilecektir.

Proje, kullanıcıların giriş ve kayıt işlemlerini daha güvenli ve kullanıcı dostu bir şekilde gerçekleştirmelerini sağlayarak hem kullanıcıların hem de sistem yöneticilerinin güvenliğini artırmayı hedeflemektedir. Bu proje ayrıca, biyometrik kimlik doğrulama teknolojilerinin pratik uygulamalarından biri olarak da öne

çıkmaktadır.

### **1.3 Projenin Kapsamı**

Bu proje, parmak izi teknolojisinin kullanıldığı kullanıcı girişi ve kayıt işlemlerini içermektedir. Projenin kapsamı aşağıdaki unsurları içermektedir:

**Parmak İzi Sensörü Entegrasyonu:** Proje, bir parmak izi sensörünün sunucu sistemine entegrasyonunu içermektedir. Bu entegrasyon, parmak izi sensöründen alınan

verilerin sunucuya aktarılmasını ve kullanıcı kimlik doğrulama işlemlerinin gerçekleştirilmesini sağlar.

**Sunucu Yazılımı Geliştirme:** Proje, kullanıcı kimlik doğrulama ve kayıt işlemlerini yönetmek için sunucuda çalışan bir yazılımın geliştirilmesini içermektedir. Bu yazılım, parmak izi sensöründen gelen verileri işleyecek, kullanıcı kimlik doğrulama işlemlerini gerçekleştirecek ve kullanıcı verilerini sunucuda saklayacaktır.

**Kullanıcı Arayüzü Tasarımı:** Proje, kullanıcıların giriş ve kayıt işlemlerini gerçekleştirebilmeleri için bir kullanıcı arayüzü tasarımını içermektedir. Bu arayüz, kullanıcıların parmak izlerini okutmalarını sağlayacak ve giriş veya kayıt işlemlerini yönlendirecektir.

**Veritabanı Yönetimi:** Proje, kullanıcıların kayıt bilgilerini saklamak için bir veritabanı yönetimini içermektedir. Bu veritabanı, kullanıcıların kimlik bilgilerini, parmak izlerini ve diğer ilgili verileri güvenli bir şekilde depolayacaktır.

**Güvenlik ve Veri Koruma:** Proje, kullanıcıların kişisel verilerini korumak için gerekli güvenlik önlemlerini içermektedir. Bu önlemler, veri şifreleme, erişim kontrolü ve diğer güvenlik protokollerini içerebilir.

## **1.4 Çalışmanın Önemi**

**Güvenlik:** Bu çalışma, biyometrik kimlik doğrulamanın bir örneği olan parmak izi teknolojisinin kullanılmasıyla güvenlik düzeyini artırır. Geleneksel kimlik doğrulama yöntemlerine kıyasla, parmak izi kullanımı daha güvenlidir çünkü parmak izleri benzersizdir ve kolayca kopyalanamaz. Bu, sistemlerde yetkisiz erişimi önlemeye yardımcı olur.

**Kullanıcı Dostu Deneyim:** Parmak izi tabanlı kimlik doğrulama, kullanıcılar için daha kullanıcı dostu bir deneyim sunar. Kullanıcılar, karmaşık şifreleri hatırlamak yerine sadece parmak izlerini kullanarak sistemlere erişebilirler. Bu, kullanıcıların deneyimini kolaylaştırır ve zaman kazandırır.

**Hız ve Verimlilik:** Parmak izi tabanlı kimlik doğrulama, diğer kimlik doğrulama yöntemlerine kıyasla daha hızlı ve verimlidir. Kullanıcılar, parmak izlerini okutarak anında erişim elde ederler. Bu, özellikle yoğun kullanıcı trafiği olan sistemlerde işlem sürelerini azaltır.

**Hassasiyet ve Doğruluk:** Parmak izi teknolojisi, diğer biyometrik kimlik doğrulama yöntemlerine göre daha yüksek doğruluk ve hassasiyet sağlar. Bu, yanlış kimlik doğrulama işlemlerinin azalmasını ve sistem güvenilirliğinin artmasını sağlar.

**Geleceğe Yönelik Teknolojik Uygulama:** Bu çalışma, geleceğin teknolojik trendlerine uygun olarak biyometrik kimlik doğrulamanın pratik uygulamalarından birini sunar. Biyometrik teknolojilerin kullanımı giderek artmakta ve bu çalışma, bu trende ayak uydurarak geleceğe yönelik bir çözüm sunar.

## **2. LİTERATÜR TARAMASI**

### **2.1 Mevcut Teknolojiler ve Araştırmalar**

Teknolojinin gelişimine paralel olarak, mevcut teknolojiler arasında en ayırt edici

özellik fiyat olarak belirlenmektedir. Teknoloji ne kadar ilerlese de, fiyatlandırma ve maliyet fonksiyonu her zaman mühendisliğin sınırlarından biri olacaktır. Büyük projelerde kullanılan parmak izi sensörleri, hassasiyet ve hız açısından farklılık gösterecektir. Ayrıca, hafıza, enerji verimliliği, uzun süre

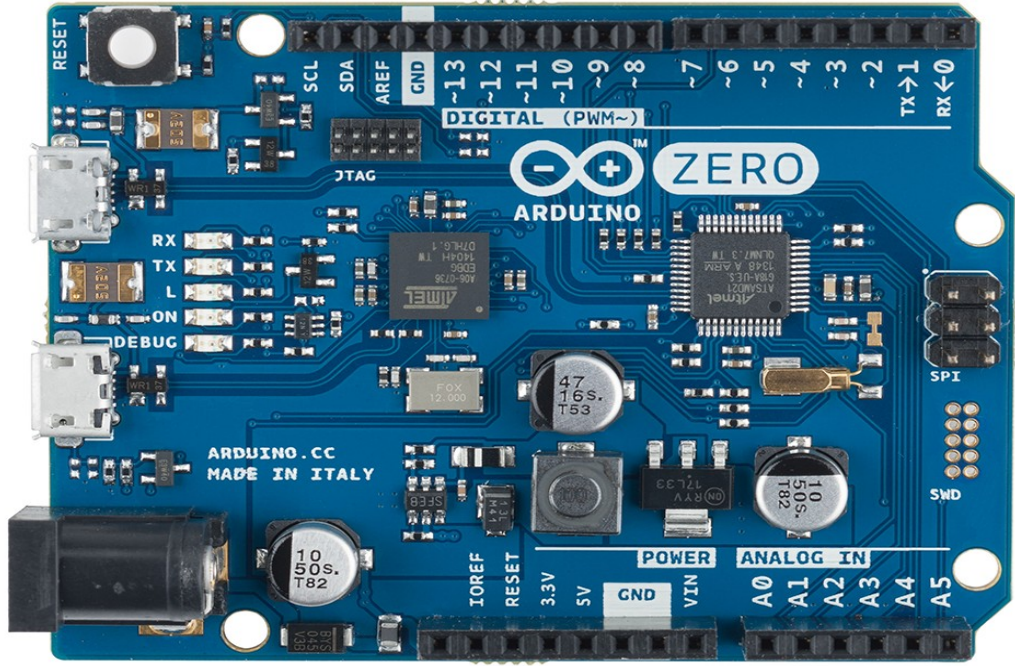
kullanılabilirlik, veri doğruluğu gibi diğer özellikler bakımından da çeşitlilik gösterir. Fiyatın yanı sıra, kullanım amacına göre de farklılıklar bulunmaktadır. Bazı parmak izi sensörleri, parmak izini bir resim formatına dönüştürmeyi tercih ederken, bazıları sadece hafızada bir kimlik adresi tutmayı hedefler [1].

Mevcut piyasalar, kullanıcıların isteklerine ve bütçelerine uygun parmak izi sensörlerini sunmaktadır. Parmak izi sensörleri üzerine yapılan araştırmalara göz attığımızda, bu uygulamalarda RFID sistemlerin popüler olduğunu ve bazı durumlarda android tabanlı telefonlardan izleme özelliği eklenmesinin tercih edildiğini görebiliriz. Diğer bir deyişle, projenin, RFID kullanılarak bir çevrim içi ağa doğrudan bağlanmadan da kullanılması, projede python, sqlite gibi dillerin tercih edilmesi ve daha düşük güç tüketen kartların [2] , örneğin arduino gibi, kullanılması oldukça yaygındır. Ayrıca, bu teknolojiler çeşitli mühendislik alanlarında geliştirilmeye devam etmektedir.

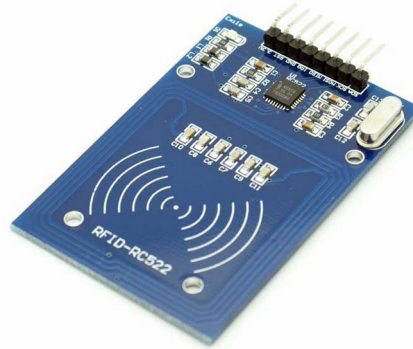
## **2.2 Benzer Çalışmalar**

IoT sistemlerinin hayatımızın içinde bu kadar yaygın olması nedeniyle, projeye benzer çalışmaların bulunması oldukça muhtemeldir. Burada, ihtiyaçlara göre benzer çalışmalar şekillenecektir. Öncelikle, çalışmada kullanılan kartın güç tüketimi ve ek sensör kullanımına göre değişecektir. Bazıları doğrudan internet bağlantısı kullanabilirken, daha küçük veri tabanları ile daha düşük güç tüketimi sergileyen entegre kartlar da projeyi tamamlayabilir. [2]





Şekil 2.1 Arduino UNO



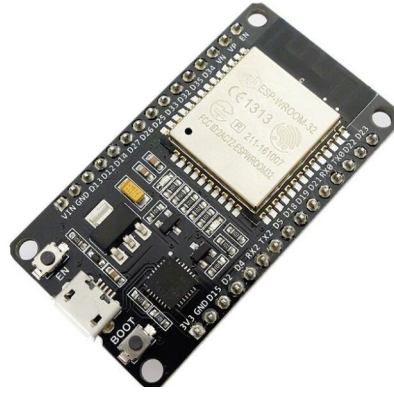
Şekil 2.2 RFID

Çalışmaların benzerliği, amaçlara göre şekillense bile, projenin çerçevesi gereği bir entegre sistem, bilgi aktarımı için sunucu ve bir veri tabanı yönünden ortak olduğu için nihayetinde benzer işler ortaya çıkmaktadır.

### 3. YÖNTEM

#### 3.1 Donanım Seçimi ve Özellikleri

Böyle bir sistemin donanım olarak, 180 kişilik orta çaplı bir işletmenin ihtiyaçlarını kolayca karşılayabilmesi ve herhangi bir karmaşıklığa sebep olmaması için orta kalite bir kart seçilmiştir: ESP32. Bu kart, internete kendi içerisindeki entegre modülden bağlanabilmenin yanı sıra Arduino UNO kartlardan daha yüksek bir performans sunmaktadır.



Şekil 3.1 ESP32

Bu projede daha yüksek verimlilik isteniyorsa, Tang Nano gibi bir kartta devre gerçekleştirilip ardından bastırılarak doğrudan sisteme monte edilebilirdi. Ancak, böyle bir proje, hem geliştirme ortamının zorluğu, hem projenin uzun soluklu olması ve yatırım gerektirmesi açısından büyük bir emek istediği açıktır. Peki, daha fazla güç

gerekmesi durumunda ESP82 tercih edilebilirdi; çünkü bu kart, diğer ESP32'ye kıyasla daha güçlüdür.



Şekil 3.2 ESP82

Bu orta ölçekli projede ESP32 oldukça yeterli olmaktadır. Eğer entegre kartımızı seçtiysek, parmak izi modülünü seçmek gerekir. Bu sensör, hem ardı ardına birçok girişi destekleyecek kadar stabil olmalı, hem de giriş yapan kişileri uzun süreler bekletmeyecek kadar hızlı çalışmalıdır. Daha sonra parmak izi saklama işlemi yapmak, daha fazla gerekli izin ve güvenlik sorunu ortaya çıkarabileceği için parmak izi sensörü, parmak izlerini okuduktan sonra onları bir ID şeklinde hafızada saklar ve parmak izlerini dışarıya vermez.



Şekil 3.3 Parmak İzi Okuma Sensörü

Genellikle piyasada bulunan bu sensörler, kablolar yardımıyla entegre kartının ilgili portlarına bağlanacak şekilde tasarlanmıştır ve kablolar dışarıda kalır. Bu sensörler, seri iletişim yoluyla entegre kart ile iletişim kurar.[3]

## **3.2 Yazılım Geliştirme**

### **3.2.1 Yazılım dillerinin önemi**

Birçok farklı proje için uygun olabilecek çeşitlilikte yazılım dili mevcuttur. Yazılım dili seçimi, projenin başarısını doğrudan etkileyen kritik bir faktördür. Bu seçim, projenin gereksinimlerine, ekibin yetkinliklerine, kullanılacak teknolojilere ve uzun vadeli bakım ve destek imkanlarına bağlı olarak yapılır. Yazılım dillerinin önemini ve bu dillerin projelerde nasıl bir rol oynadığını daha

ayrıntılı incelemek gereklidir. Bir yazılım dilinin seçimi, o dilde sağlanan desteklerin ve ekosistemin zenginliğine bağlıdır. Örneğin, Python ve JavaScript gibi diller, geniş topluluklar, çok sayıda kütüphane ve framework'ler ile güçlü destek sağlar.

```
function enEdition(){
    /* Ne rien faire mode edit + preload */
    if( encodeURIComponent(document.location).search(/%26preload%3D/) != -1 ) return;
    // /&preload=

    if ( !wgPageName.match(/Discussion.*\Traduction/) ) return;
    var diff = new Array();
    var status; var pecTraduction; var pecRelecture;
    var avancementTraduction; var avancementRelecture;

    /* ***** Parser ***** */
    var params = document.location.search.substr(1, document.location.search.length).split('&');
    var i = 0;
    var tmp; var name;
    while ( i < params.length )
    {
        tmp = params[i].split('=');
        name = tmp[0];
        switch( name ) {
            case 'status':
                status = tmp[1];
                break;
            case 'pecTraduction':
```

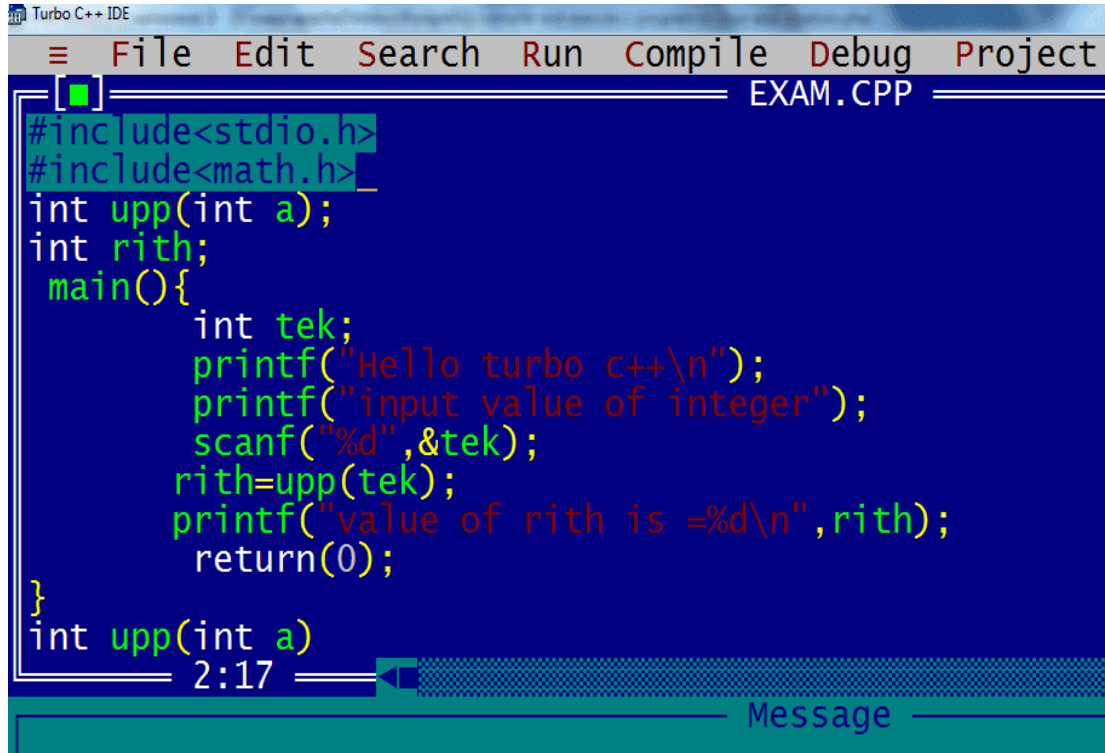
Şekil 3.4 Javascript

Bu durum, geliştiricilerin karşılaştıkları sorunları daha hızlı çözmelerine ve projelerini daha etkin bir şekilde geliştirmelerine olanak tanır. Ayrıca, geniş bir

topluluğa sahip diller, sürekli güncellenen ve geliştirilen araçlarla desteklenir, bu da projenin uzun vadede sürdürülebilirliğini artırır. Yazılım dili seçimi, projeyi gerçekleştirecek ekibin yetkinlikleri ve deneyimleriyle de doğrudan ilişkilidir. Bir ekip, belirli bir dilde uzmanlaşmışsa, o dili kullanarak daha hızlı ve daha verimli çalışabilir. Bu durum, projenin geliştirme sürecini hızlandırır ve maliyetleri düşürür. Ekibin aşına olduğu bir dilde çalışması, hataların daha hızlı tespit edilip düzeltilmesini de sağlar. Bununla birlikte, yeni bir dil öğrenmek ve bu dilde

ustalaşmak zaman ve kaynak gerektirebilir, bu da projenin başlangıç maliyetlerini artırabilir. Yazılım dilinin verimliliği ve performansı, projenin başarısı için kritik öneme sahiptir. Bazı diller, belirli görevler için daha yüksek performans sunarken, diğerleri daha fazla esneklik ve hızlı prototipleme imkanı sağlar.

Örneğin, C++ ve Rust gibi diller, yüksek performans gerektiren sistemler ve uygulamalar için idealdir,



```
#include<stdio.h>
#include<math.h>
int upp(int a);
int rith;
main(){
    int tek;
    printf("Hello turbo c++\n");
    printf("input value of integer");
    scanf("%d",&tek);
    rith=upp(tek);
    printf("value of rith is =%d\n",rith);
    return(0);
}
int upp(int a)
2:17
```

Şekil 3.5 C++ (cpp)

çünkü düşük seviyeli bellek yönetimi ve optimize edilmiş çalışma zamanı sunarlar. Diğer yandan, Python ve Ruby gibi diller, hızlı geliştirme döngüleri ve kolay okunabilirlik sunarak, hızlı prototipleme ve veri bilimi projelerinde tercih edilir. Yazılım dilinin seçimi, projenin uzun vadeli bakım ve geliştirme ihtiyaçlarını da etkiler. İyi belgelenmiş ve geniş topluluk desteğine sahip diller, projenin yaşam süresi boyunca bakımını ve geliştirilmesini kolaylaştırır. Bunun yanı sıra, seçilen dilin gelecekte de destekleneceğine ve geliştirileceğine dair güvence, projenin uzun

ömürlü olmasını sağlar. Örneğin, Java ve C# gibi diller, kurumsal uygulamalar için

uzun vadeli destek ve sürekli güncellemeler sunar, bu da bu dillerde yazılan projelerin uzun yıllar boyunca sorunsuz bir şekilde çalışmasını sağlar. Yazılım dilinin önemini belirleyen bir diğer faktör, yazılım tasarım desenlerinin kullanımıdır. Tasarım desenleri, yazılım geliştirme sürecinde karşılaşılan yaygın problemlere çözümler sunar ve kodun yeniden kullanılabilirliğini, bakımını ve genişletilebilirliğini artırır. Yazılım dilinin, belirli tasarım desenlerini destekleyip desteklememesi veya bu desenleri uygulamanın ne kadar kolay olduğu, dil seçimini etkileyen önemli bir faktördür. Örneğin, nesne yönelimli programlama dillerinde (Java, C++, C#), tasarım desenlerinin kullanımı oldukça

yaygındır ve bu diller, bu desenlerin uygulanmasını kolaylaştıran özelliklere sahiptir. Yazılım dili

seçimi, hedef platform ve uyumluluk gereksinimleriyle de ilgilidir. Bazı diller, belirli platformlarla daha iyi entegrasyon sağlar ve bu platformlarda daha verimli çalışır. Örneğin, Swift, Apple ekosisteminde iOS ve macOS uygulamaları geliştirmek için optimize edilmiştir, bu da Swift kullanan geliştiricilerin Apple platformlarında daha iyi performans elde etmelerini sağlar. Aynı şekilde, Java, platformdan bağımsız olarak çalışabilen uygulamalar geliştirmek için tasarlanmıştır ve bu nedenle geniş bir yelpazede cihaz ve işletim sistemlerinde kullanılabilir. Son olarak, bir yazılım dilinin seçimi, o dil için mevcut topluluk ve eğitim kaynaklarına da bağlıdır. Geniş ve aktif bir topluluğa sahip diller, geliştiricilerin sorunlarını çözmelerine ve yeni yetenekler öğrenmelerine yardımcı olacak çok sayıda kaynak sunar. Bu kaynaklar, online eğitimler, kitaplar, konferanslar ve topluluk forumları gibi çeşitli biçimlerde olabilir. Bu tür kaynaklar, özellikle yeni başlayanlar için öğrenme sürecini hızlandırır ve projelerin daha hızlı ilerlemesini sağlar. Özetlemek gerekirse, yazılım dili seçimi, projenin başarısını belirleyen kritik bir faktördür. Yazılım dilinin seçimi, destek ve ekosistem, ekibin yetkinlikleri, verimlilik ve performans, uzun vadeli bakım ve geliştirme, yazılım tasarım desenleri, platform ve



uyumluluk, topluluk ve eğitim kaynakları gibi birçok faktöre bağlıdır. Bu faktörlerin her biri, projenin gereksinimlerine ve hedeflerine uygun bir dil seçimi yapılmasını sağlar ve bu seçim, projenin başarısı için hayati öneme sahiptir.

### **3.3 Sistem Mimarisi**

İlk başlıkta incelencek kısım sistem bileşenleridir.

#### **3.3.1 Sistem bileşenleri**

Sistem, üç ana bileşenden oluşmaktadır: ESP32 mikrodnetleyici, parmak izi sensörü ve PHP tabanlı sunucu. ESP32 Mikrodnetleyici, Wi-Fi ve Bluetooth yeteneklerine sahip, yüksek performanslı ve düşük güç tüketimli bir mikrodnetleyicidir. Bu projede, ESP32 şu görevleri yerine getirir: Parmak izi sensöründen veri almak, parmak izi sensöründen gelen verilerin bütünlüğünü değerlendirmek, kullanıcı verilerini sunucuya iletmek ve sunucudan gelen yanıtları işlemek. Ek olarak, sürekli sunucu ile iletişimi korumak da önemlidir. ESP32 kodları tamamen nesne yönelimli (OOP) olarak yazılmıştır.



```

#include <Arduino.h>
#include <wifiConnect.h>
#include <fServer.h>
#include <Adafruit_GFX.h>

const String SSID = "Kilicarslan";
const String PASSWD = "14531453";

WifiConnect connect(SSID, PASSWD);
fServer server(80);

void setup() {
    Serial.begin(9600);
    connect.Initialize();
    Serial.println(connect.get_IP());
    Serial.println(connect.getIP());
    server.Initialize("/post", HTTP_POST);
    delay(20);
}

void loop() {
    server.listen();
}

```

Şekil 3.6 ESP32 Main Fonksiyon

Ana fonksiyon oldukça sadece ve anlaşılabilir şekilde bırakılmıştır. Bu ilerde projede değişiklik yapmak isteyen geliştiriciler için açıkça bir dökümantasyon oluşturmaktadır.

### 3.3.2 Parmak izi sensörü

Parmak izi sensörü, kullanıcıların parmak izlerini algılar ve dijital veri olarak ESP32'ye iletir. Sensörün temel işlevleri şunlardır: Parmak izi algılama, Parmak izi verilerini dijital formata dönüştürme, Parmak izi verilerini hafızada karşılaştırdıktan sonra ilgili ID'yi ESP32'ye iletme.

Sensörün özellikleri:

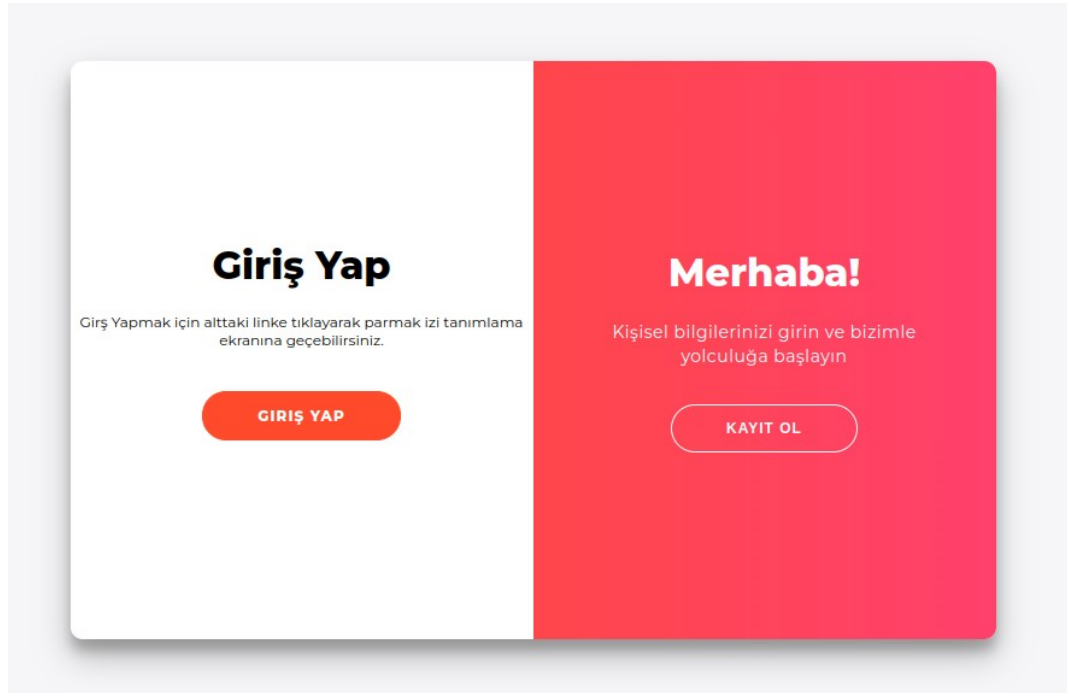
**Yüksek Hassasiyet:** Parmak izi verilerini doğru ve güvenilir bir şekilde algılar.

**Hızlı Yanıt Süresi:** Parmak izi verilerini hızlı bir şekilde işler ve iletir.

**Entegre Bellek:** parmak izi verilerini saklayabilir.

### 3.3.3 PHP tabanlı sunucu

PHP tabanlı sunucu, kullanıcılara ait verileri saklamak ve işlemek için kullanılır. Sunucu şu işlevleri yerine getirir: Kullanıcı kayıt ve doğrulama işlemlerini yönetmek. Veritabanı işlemlerini gerçekleştirmek. ESP32'den gelen istekleri işlemek ve yanıt vermek. Web arayüzü sağlayarak kullanıcıların kayıt ve giriş işlemlerini kolaylaştırmak. Sunucu arayüzü html ve css kullanılarak oluşturulmuştur



Şekil 3.7 Sunucu Ön Yüzü

Sunucu özellikleri: PHP ve MySQL: Güçlü ve yaygın olarak kullanılan web teknolojileri. Veritabanı Yönetimi: Kullanıcı bilgilerini güvenli ve organize bir şekilde saklar. API Desteği: ESP32 ile iletişim için RESTful API'lar sunar. İncelenmesi gereken ikinci kısım sistemin ilerleyiştir.

### **3.3.4 Sistem iş akışı**

Sistemin iş akışı iki ana süreci içerir: kullanıcı girişi ve kullanıcı kaydı. Kullanıcı girişi ve Parmak izi alma. Kullanıcı, parmak izi sensörüne parmağını yerleştirir. Sensör, parmak izini algılar ve veriyi ESP32'ye gönderir. ESP32, alınan parmak izini hafızasındaki mevcut parmak izi verileriyle karşılaştırır. Eğer eşleşme bulunursa, ilgili kullanıcının ID'sini sunucuya gönderir. Sunucu, aldığı kullanıcı ID'sine göre veritabanında sorgulama yapar. Sunucu eğer kullanıcıyı bulmuşsa sunucu önyüz ekranında onu doğrular. Kullanıcı, kayıt işlemi için sistem tarafından fotoğraf çekmesi istenir. Daha sonra kullanıcı parmak izi sensörüne iki kez parmağını yerleştirir birincisi parmak izini algılamak ikincisi ise doğrulamak içindir. Alınan parmak izi verileri ESP32 tarafından doğrulanır ve sunucuya iletilir. Sunucu, gelen verileri veritabanına kaydeder ve kullanıcıyı başarılı bir şekilde kayıt eder. En önemli kısımlardan biri de iletişim protokolleridir. ESP32 ve PHP tabanlı sunucu arasındaki iletişim, HTTP protokolü üzerinden gerçekleştirilir. İletişim adımları şunlardır:

Veri Gönderme: ESP32, parmak izi verilerini ve kullanıcı bilgilerini HTTP POST istekleri kullanarak sunucuya gönderir.

Veri Alma: Sunucu, gelen isteklere yanıt verir ve gerekli kullanıcı bilgilerini ESP32'ye geri gönderir.

### **3.3.5 CORS hataları ve çözümleri**

Web geliştirme sürecinde, güvenlik ve işlevsellik arasında bir denge kurmak önemlidir. Bu bağlamda, Cross-Origin Resource Sharing (CORS) politikaları, farklı kaynaklardan gelen web sayfaları arasındaki kaynak paylaşımını kontrol eder.

CORS, modern tarayıcılar tarafından uygulanır ve web sayfalarının güvenliğini artırmayı amaçlar. Ancak, CORS hataları, geliştiriciler için yaygın bir engel olabilir. Bu makalede, CORS hatalarının nedenlerini ve bu hataları çözmek için uygulanabilecek çeşitli yöntemleri inceleyeceğiz. CORS, bir kaynağın (genellikle bir web sayfası) başka bir kaynaktan (dış bir etki alanından) veri isteğinde bulunmasına izin veren bir mekanizmadır. Aynı Kaynak Politikası (Same-Origin Policy), bir kaynağın sadece aynı etki alanından gelen istekleri kabul etmesini sağlar. CORS, bu politikayı genişleterek, güvenli bir şekilde çapraz kaynak isteklerine izin verir.

### **3.3.6 CORS hatalarının nedenleri**

CORS hataları, bir web sayfasının, farklı bir etki alanından kaynak isteği yaptığı ve bu isteğin sunucu tarafından uygun şekilde işlenmediğinde ortaya çıkar. Yaygın CORS hataları şunlardır:

Eksik veya Yanlış Ayarlanmış CORS Başlıkları: Sunucu, gerekli CORS başlıklarını eklemediğinde veya yanlış eklediğinde.

Ön Uç ve Arka Uç Uyumsuzlukları: İstemci tarafı (ön uç) ile sunucu tarafı (arka uç) arasında uyumsuzluk olduğunda.

Yanıtın Hatalı Olması: Sunucunun yanıtı, istemcinin beklediği formatta olmadığında.

OPTIONS İsteği Problemleri: Tarayıcı, bazı istekleri yapmadan önce OPTIONS isteği gönderir ve sunucu bu isteğe uygun şekilde yanıt vermezse.

#### **CORS Hatalarını Tespit Etme**

CORS hatalarını tespit etmenin en yaygın yolu tarayıcı konsoludur. Tarayıcılar, CORS hatalarını açık bir şekilde bildirir. Örneğin, Google Chrome geliştirici konsolunda şu hata mesajını görebilirsiniz:

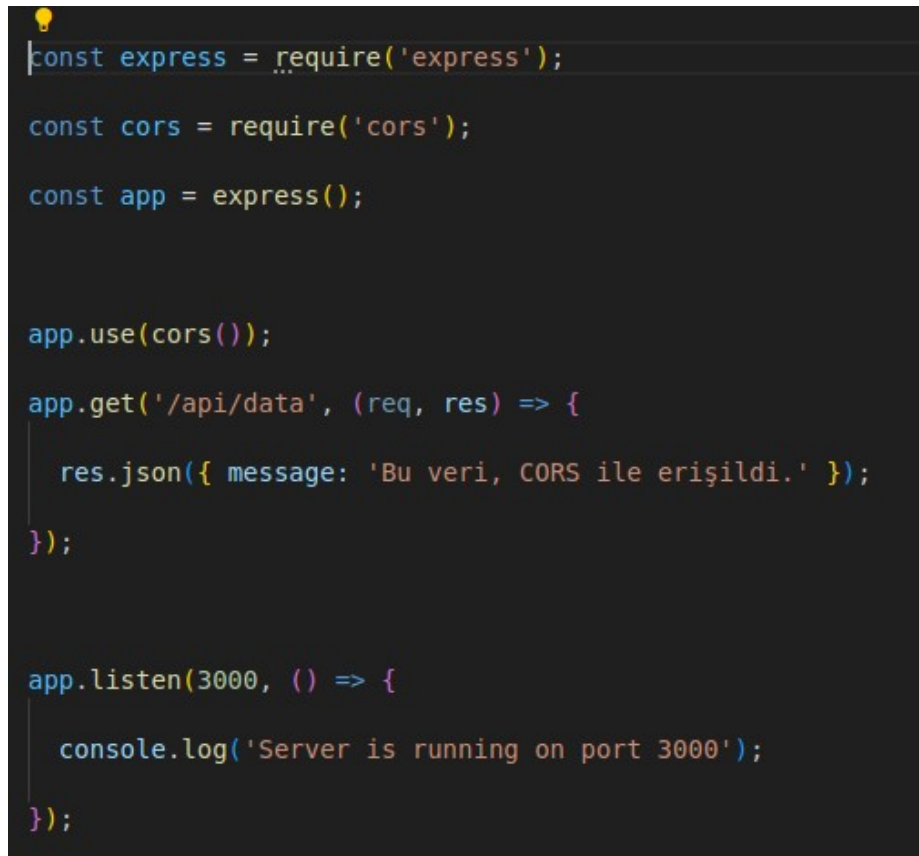
Access to fetch at 'http://example.com/api/data' from origin 'http://yourdomain.com' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

### 3.3.7 CORS hatalarının çözümleri

CORS hatalarını çözmek için aşağıdaki yöntemler kullanılabilir:

#### Sunucu Konfigürasyonunu Düzenleme

Sunucu, uygun CORS başlıklarını eklemelidir. Örneğin, bir Node.js/Express sunucusunda CORS'u şu şekilde yapılandırabilirsiniz:



```
const express = require('express');
const cors = require('cors');
const app = express();

app.use(cors());

app.get('/api/data', (req, res) => {
  res.json({ message: 'Bu veri, CORS ile erişildi.' });
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Şekil 3.8 Node.js/Express ve Cors

Access-Control-Allow-Origin: Bu başlık, hangi etki alanlarının kaynağa erişebileceğini belirler. Genel erişim için \* kullanılabilir, ancak güvenlik riskleri içerir.

### 3.3.8 OPTIONS isteklerini yönetme

Sunucu, tarayıcıdan gelen preflight OPTIONS isteklerine uygun yanıt vermelidir. Örneğin, Express.js'de:

```
app.options('/api/data', cors()); |
```

Şekil 3.9 CORS

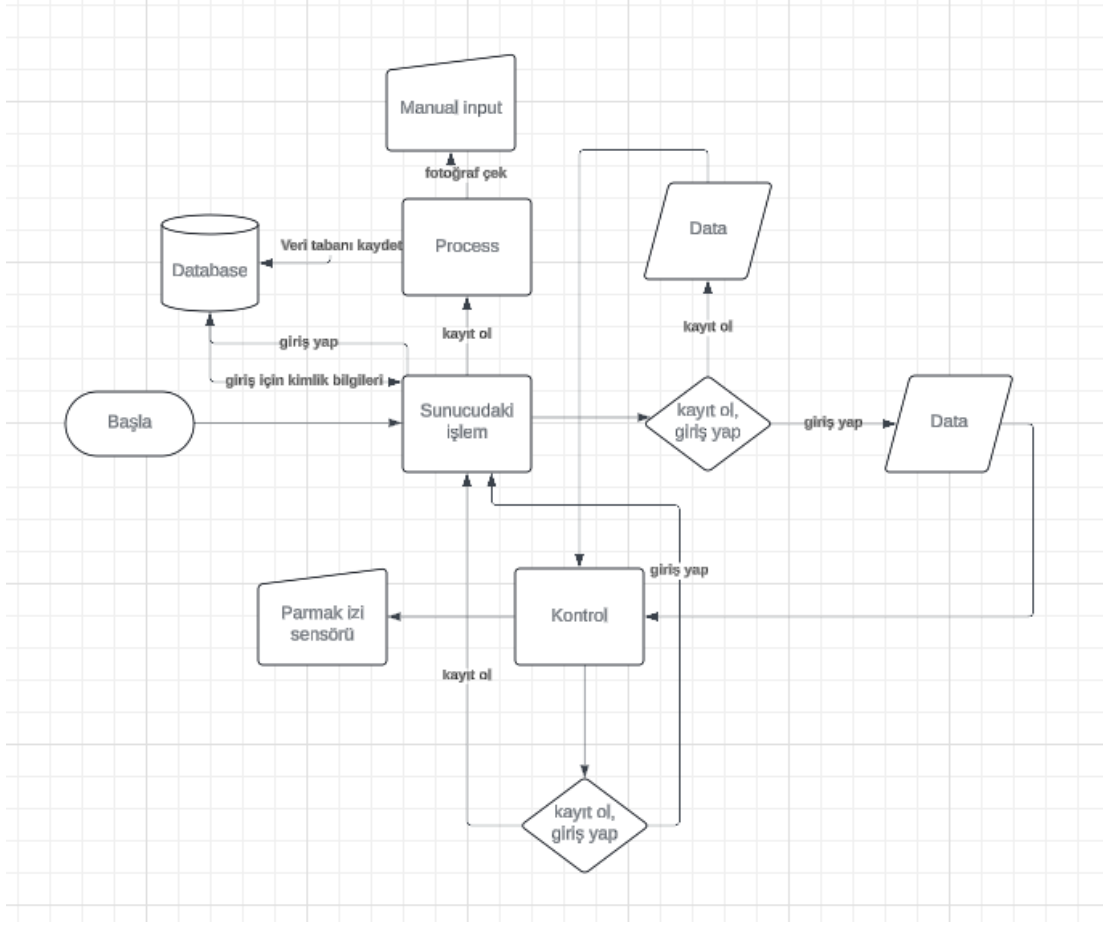
CORS sorunlarını aşmak için bir proxy sunucusu kullanabilirsiniz. Örneğin, geliştirme aşamasında, http-proxy-middleware paketiyle bir proxy ayarlayabilirsiniz:

```
const { createProxyMiddleware } = require('http-proxy-middleware');  
app.use('/api', createProxyMiddleware({ target: 'http://example.com', changeOrigin: true })); |
```

Şekil 3.10 Proxy

### 3.3.9 Veri akış diyagramları ve sistem şemaları

Sistemin nasıl çalıştığını görselleştirmek için veri akış diyagramları ve sistem şemaları kullanılmıştır. Bu şemalar, bileşenler arasındaki etkileşimleri ve veri akışını ayrıntılı olarak gösterir.



Şekil 3.11 Akış Diyagramı

### 3.4 Veritabanı Tasarımı

Veritabanı tasarımı, bir uygulamanın başarısı ve performansı açısından kritik bir rol oynar. Bu bölümde, veritabanı tasarımında dikkate alınması gereken önemli hususları, SQL ve NoSQL veritabanları arasındaki farkları [4] ve veritabanı tasarımıımızda SQL kullanmanın nedenlerini detaylı bir şekilde ele alacağız. Ayrıca, iyi bir veritabanının nasıl tasarlanacağına dair genel prensipler üzerinde duracağız

#### 3.4.1 Veritabanı tasarımında dikkate alınması gereken hususlar

Veri tabanı birçok kullanıcı tarafından kullanılan büyük veri kümelerini düzenleyip belirli bir formatta depolar[6].

Veritabanı tasarımında dikkate alınması gereken başlıca hususlar şunlardır:

Veri Yapısı ve Karmaşıklığı: Verilerin nasıl yapılandırıldığı ve aralarındaki ilişkiler.

Kullanıcı Sayısı: Sistemin kaç kullanıcıya hizmet vereceği.

Veri Çeşitliliği: Saklanacak veri türlerinin çeşitliliği.

Performans Gereksinimleri: Veri okuma ve yazma hızları ile sorgu performansı.

Güvenlik ve Bütünlük: Verilerin güvenliği ve bütünlüğünün sağlanması.

Ölçeklenebilirlik: Veritabanının gelecekte artacak veri hacmini ve kullanıcı sayısını karşılayabilme kapasitesi.

### **3.4.2 SQL ve NoSQL veritabanları**

Veritabanları genel olarak iki ana kategoriye ayrılır: SQL (Structured Query Language) ve NoSQL (Not Only SQL). Her iki veritabanı türü de farklı ihtiyaçları karşılamak üzere tasarlanmıştır ve belirli avantaj ve dezavantajlara sahiptir.

#### **3.4.2.1 SQL veritabanları**

SQL veritabanları, verilerin yapılandırılmış tablolarda saklandığı, ilişkisel veritabanlarıdır. SQL, bu veritabanlarıyla etkileşim kurmak için kullanılan standart bir sorgulama dilidir. SQL veritabanlarının temel özellikleri şunlardır:

Yapılandırılmış Veri: SQL veritabanları, verilerin belirli bir şema ile tanımlandığı yapılandırılmış veri modeline dayanır.

İlişkisel Model: Veriler, tablolar arasında ilişkilerle bağlanır ve bu ilişkiler yabancı anahtarlar (foreign keys) ile yönetilir.



ACID Özellikleri: SQL veritabanları, atomicity, consistency, isolation ve durability (ACID) özelliklerini sağlar, bu da veri bütünlüğünü garanti eder.

SQL Dili: Verilere erişim ve yönetim, SQL dili kullanılarak gerçekleştirilir.

#### **3.4.2.2 NoSQL veritabanları**

NoSQL veritabanları, verilerin esnek, yapılandırılmamış veya yarı yapılandırılmış formatlarda saklanmasına izin veren veritabanlarıdır. NoSQL, "Not Only SQL" ifadesinin kısaltmasıdır ve SQL dışında farklı veri modellerini ifade eder. NoSQL veritabanlarının temel özellikleri şunlardır:

Esnek Şema: NoSQL veritabanları, belirli bir şema gerektirmez, bu da veri modellerinin dinamik olarak değişmesine izin verir.

Çeşitli Veri Modelleri: NoSQL veritabanları, çeşitli veri modellerini destekler, örneğin belge tabanlı, anahtar-değer tabanlı, grafik tabanlı ve sütun tabanlı veritabanları.

Yatay Ölçeklenebilirlik: NoSQL veritabanları, büyük veri kümelerini dağıtık sistemler üzerinde yatay olarak ölçeklendirme yeteneğine sahiptir.

BASE Özellikleri: NoSQL veritabanları genellikle basic availability, soft state ve eventual consistency (BASE) özelliklerine dayanır, bu da esneklik sağlar.

#### **3.4.2.3 Neden SQL seçildi?**

Veritabanı tasarımımda SQL tercih edilmesinin birkaç önemli nedeni bulunmaktadır:

Veri Yapısının Belirginliği: Sistemimde id, isim, soyisim ve fotoğraf gibi belirli veri türleri bulunmaktadır. Bu verilerin yapısı ve aralarındaki ilişkiler belirgindir ve SQL veritabanları,

yapılandırılmış veri yönetimi konusunda mükemmel bir uyum sağlar.

**Veri Bütünlüğü ve Güvenlik:** SQL veritabanlarının ACID özellikleri, veri bütünlüğü ve güvenliği açısından büyük avantaj sağlar. Bu özellikler, verilerin tutarlılığını ve güvenliğini garanti eder.

**Kolay Yönetilebilirlik:** SQL, veritabanı yönetimi ve sorgulama için güçlü ve yaygın olarak kullanılan bir dildir. Veritabanı yönetimi, veri sorgulama ve güncelleme işlemleri SQL ile kolayca gerçekleştirilebilir.

**Geniş Destek ve Topluluk:** SQL veritabanları, geniş bir destek ve topluluk ağına sahiptir. MySQL gibi popüler SQL veritabanları, kapsamlı dokümantasyon ve destek kaynaklarına sahiptir.

#### Veritabanı Tasarımı

Veritabanımızda kişi bilgileri id, isim, soyisim ve fotoğraf olarak tutulmaktadır. Bu veritabanı tasarımı için MySQL kullanılmıştır. MySQL, açık kaynaklı ve güçlü bir SQL veritabanı yönetim sistemidir.

#### 3.4.3 Tabloların tasarımı

Veritabanı tasarımında, her bir kullanıcı için temel bilgiler tutan bir tablo oluşturulmuştur. Bu tablo, aşağıdaki yapıya sahiptir:

Tablo 3.1 Veritabanı Örneği

id	isim	soyisim	fotoğraf
1	ahmet	yilmaz	Ahm_yil.jpg
2	ayşe	demir	Ays_dem.jpg

id: Her kullanıcı için benzersiz bir kimlik numarası (primary key).

isim: Kullanıcının adı.

soyisim: Kullanıcının soyadı.

fotograf: Kullanıcının fotoğrafının dosya adı veya yolu.

Tablonun Oluşturulması

Tabloyu oluşturmak için kullanılan SQL sorgusu şu şekildedir:

```
CREATE TABLE Kullanicilar (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    isim VARCHAR(100) NOT NULL,  
    soyisim VARCHAR(100) NOT NULL,  
    fotograf VARCHAR(255) NOT NULL  
); |
```

Şekil 3.12 SQL Tablo Oluşturma

#### 3.4.4 İyi bir veritabanının tasarımı

İyi bir veritabanı tasarımı, aşağıdaki prensiplere dayanır:

Normalizasyon: Veritabanındaki verilerin tekrarlanmasını önlemek ve veri bütünlüğünü sağlamak için normalizasyon kurallarına uyulmalıdır. Normalizasyon, veritabanı performansını artırır ve veri tutarlılığını sağlar.

Uygun Veri Tipleri: Her bir veri alanı için uygun veri tipi seçilmelidir. Bu, veritabanının performansını ve veri bütünlüğünü etkiler.

İndeksleme: Sık kullanılan sorguların performansını artırmak için uygun indeksler oluşturulmalıdır. İndeksler, veri erişim hızını artırır.

İlişkilerin Tanımlanması: Tablolar arasındaki ilişkiler doğru bir şekilde tanımlanmalıdır. Yabancı anahtarlar kullanılarak, ilişkisel veritabanlarında veri tutarlılığı sağlanabilir.

Güvenlik: Veritabanı güvenliği için kullanıcı yetkilendirmeleri ve veri şifreleme yöntemleri kullanılmalıdır.

Yedekleme ve Geri Yükleme: Veri kaybını önlemek için düzenli yedekleme planları oluşturulmalı ve gerektiğinde veri geri yükleme süreçleri belirlenmelidir.

### **3.4.5 Sonuç**

Veritabanı tasarımı, bir uygulamanın verimli ve güvenli çalışması için temel bir unsurdur. SQL ve NoSQL veritabanları, farklı veri yapıları ve gereksinimleri için çeşitli çözümler sunar. Bu projede, yapılandırılmış veri yapısı, veri bütünlüğü ve yönetim kolaylığı gibi nedenlerle SQL veritabanı tercih edilmiştir. MySQL kullanılarak tasarlanan veritabanı, kullanıcı bilgilerini güvenli ve verimli bir şekilde yönetmek için gerekli tüm özelliklere sahiptir. İyi bir veritabanı tasarımı, veri normalizasyonu, uygun veri tipi seçimi, indeksleme ve güvenlik gibi prensiplere dayanarak gerçekleştirilmiştir. Bu prensipler, veritabanının performansını, güvenliğini ve sürdürülebilirliğini artırır.

## **4. TEST**

Yazılım geliştirme sürecinde test etme, kodun doğruluğunu, güvenilirliğini ve performansını sağlamak için kritik bir adımdır. Bir yazılım projesinde testlerin atlanması, ciddi hatalara, güvenlik açıklarına ve kötü kullanıcı deneyimlerine yol açabilir. Bu bölümde, kodlama projelerinin test edilmesinin önemini detaylı bir şekilde ele alacağız. Testler, yazılım geliştirme sürecinin erken aşamalarında hataları

tespit etmeye yardımcı olur. Bu, hataların daha az maliyetle ve daha hızlı bir şekilde düzeltilmesini sağlar. Hataların üretim ortamında ortaya çıkması durumunda, düzeltme maliyeti ve zamanı genellikle çok daha yüksektir. Erken tespit, bu maliyetleri minimize eder ve geliştirme sürecini hızlandırır. Testler, kodun kalitesini artırır. Bir yazılım projesinde birim testleri,

entegrasyon testleri ve sistem testleri gibi çeşitli test türleri uygulanır. Bu testler, kodun beklenen şekilde çalışıp çalışmadığını kontrol eder ve kod kalitesinin yüksek olmasını sağlar. Kaliteli kod, bakımın daha kolay yapılmasını ve gelecekteki geliştirmelerin daha sorunsuz olmasını sağlar. Yazılım projelerinde güvenlik, büyük bir öneme sahiptir. Güvenlik testleri, yazılımın olası güvenlik açıklarını tespit eder ve bu açıkların kapatılmasını sağlar. Güvenlik testleri yapılmayan yazılımlar, siber saldırılara açık hale gelir ve ciddi veri ihlallerine yol açabilir. Güvenlik testleri, yazılımın hem kullanıcı verilerini korumasını hem de güvenilir bir şekilde çalışmasını sağlar. Performans testleri, yazılımın farklı yük koşullarında nasıl davrandığını değerlendirmeye yardımcı olur. Bu testler, yazılımın hızını, yanıt verme süresini ve genel performansını ölçer. Performans sorunları, kullanıcı deneyimini olumsuz etkileyebilir. Bu nedenle, performans testleri, yazılımın optimum düzeyde çalışmasını sağlamak için kritik öneme sahiptir. Kullanıcı kabul testleri (UAT), yazılımın son kullanıcıların beklentilerini karşılayıp karşılamadığını değerlendirir. Bu testler, yazılımın kullanıcı dostu olup olmadığını ve kullanıcıların ihtiyaçlarına uygun olup olmadığını belirler. İyi test edilmiş bir yazılım, kullanıcılar için daha az hata içerir ve daha sorunsuz bir deneyim sunar. Modern yazılım geliştirme süreçlerinde, sürekli entegrasyon (CI) ve sürekli dağıtım (CD) önemli bir rol oynar. Testlerin otomasyonu, CI/CD süreçlerinde kritik bir bileşendir. Otomatik testler, her kod değişikliğinden sonra yazılımın otomatik olarak test edilmesini ve dağıtılmasını sağlar. Bu, yazılımın sürekli olarak güncel ve hatasız olmasını garanti eder. Yazılım projelerinde testlerin ihmal edilmesi, uzun vadede maliyetlerin artmasına neden olur. Hataların geç aşamalarda tespit edilmesi, düzeltme maliyetlerini artırır ve geliştirme sürecini yavaşlatır. Testlerin erken aşamalarda yapılması, bu maliyetleri azaltır.

## **4.1 Giriş Olayları**

Bir sisteme giriş yaparken yetkisiz girişleri önlemek oldukça önemli olmaktadır. Bu işlemi garanti altına almanın en etkili yollarından biri hataları test etmek ve beklenmeyen durumları oldukça ele almaktır.

### **4.1.1 Giriş olmazsa**

Eğer kullanıcı girişi seçilmiş ama bir giriş için parmak izi verilmemişse sistem bir parmak izi alana kadar beklemeye devam eder. Burda ikinci kişi önce parmak izini okutmalı ve yanlış bilgisi geldikten sonra kendi işlemini tekrar seçmelidir.

### **4.1.2 Giriş yanlış olursa**

Eğer kullanıcı yanlış bir giriş yaparsa entegre karttan sunucuya kullanıcının yanlış girdi yaptığını bildiren bir mesaj yollanır. Bu kısımda entegre kart tekrar sunucudan cevap bekler ve sunucu kullanıcıyı giriş kısmına yönlendirir.

## **4.2 Kayıt Olayları**

Bir sisteme kayıt yaparken bir kişinin birden fazla kaydını önlemek oldukça önemli olmaktadır. Bu işlemi garanti altına almanın en etkili yollarından biri hataları test etmek ve beklenmeyen durumları oldukça ele almaktır.

### **4.2.1 Kayıt yanlış olursa**

Kayıt yanlış olursa entegre kart sunucuya kaydın yanlış olduğunu bildiren bir mesaj yollıyacaktır. Sunucu ise kullanıcıyı kayıt sayfasına yönlendirir.

#### 4.2.2 Kayıt girilmezse

Eğer kullanıcı kayıt ol seçmiş ama bir kayıt için parmak izi verilmemişse sistem bir parmak izi alana kadar beklemeye devam eder. Burda ikinci kişi önce parmak izini okutmalı ve yanlış bilgisi geldikten sonra kendi işlemini tekrar seçmelidir.

### 5. SONUÇLAR

Bu proje kapsamında geliştirilen parmak izi okuyarak sunucudan kullanıcı girişi ve kayıt işlemi sistemi, başarılı bir şekilde hayata geçirilmiştir. Aşağıda, proje boyunca elde edilen ana bulgular ve sonuçlar özetlenmiştir:

**Parmak İzi Okuyucu Sensör ve Sunucu Entegrasyonu:** Parmak izi okuyucu sensör, entegre sistem kartı ile başarılı bir şekilde çalışmıştır. Kullanıcıdan alınan parmak izi verileri, sensör hafızasındaki diğer parmak izleriyle karşılaştırılarak doğru bir şekilde analiz edilmiştir. Kullanıcı giriş ve kayıt işlemleri sunucu üzerinden etkin bir şekilde yönetilmiştir. Parmak izi sensörü tarafından alınan veriler sunucuya iletilmiş ve sunucu, kullanıcının kimliğini doğrulamak için bu verileri işlemiştir.

**Kullanıcı Girişi ve Kayıt İşlemleri:** Kullanıcı giriş işlemlerinde, sistemin hafızasında kayıtlı olan parmak izleri ile yeni alınan parmak izleri doğru bir şekilde eşleştirilmiş ve kullanıcı kimlik doğrulaması başarılı bir şekilde gerçekleştirilmiştir. Kullanıcı kayıt işlemlerinde, yeni kullanıcıların parmak izleri ve fotoğrafları başarıyla alınmış ve sistemin veritabanına kaydedilmiştir. Kayıt işlemi sırasında, parmak izlerinin iki kez okutulmasıyla doğruluk artırılmıştır.

**Sistem Performansı ve Güvenilirlik:** Sistem, kullanıcı kimlik doğrulama ve kayıt işlemlerini hızlı ve doğru bir şekilde gerçekleştirmiştir. Parmak izi eşleştirme süreci oldukça etkili olup, yanlış pozitif ve yanlış negatif oranları minimize edilmiştir.

**Kullanıcı verilerinin güvenliği sağlanmış ve tüm işlemler güvenli bir şekilde gerçekleştirilmiştir.**  
**Kullanıcı Deneyimi:** Kullanıcı dostu bir arayüz sayesinde, kullanıcılar giriş ve kayıt işlemlerini kolayca gerçekleştirebilmiştir. Sistem,

kullanıcıların parmak izi okutma ve fotoğraf çekme işlemlerini sorunsuz bir şekilde tamamlamıştır.

### **5.1 Genel Değerlendirme**

Proje, belirlenen hedeflere ulaşmada başarılı olmuştur. Parmak izi okuyucu sensör ile sunucu arasındaki entegrasyon ve veri işlemleri, projenin temel amacı olan güvenli ve hızlı kullanıcı kimlik doğrulama ve kayıt işlemlerinin gerçekleştirilmesini sağlamıştır. Bu çalışma, ileriye dönük olarak yapay zeka ve IoT teknolojileri ile entegre edilebilecek daha karmaşık güvenlik sistemlerinin temelini oluşturmuştur.

Projenin geliştirilmesi sırasında elde edilen bilgiler ve deneyimler, benzer sistemlerin tasarımında ve uygulanmasında önemli bir referans olacaktır. Proje, kullanıcı güvenliği ve veri bütünlüğü açısından yüksek bir performans sergilemiş ve başarılı bir şekilde tamamlanmıştır.

## **6. GELECEK ÇALIŞMALAR**

Nesnelerin İnterneti (IoT), birbirine bağlı cihazların ve sistemlerin oluşturduğu geniş kapsamlı bir ekosistem olarak, teknoloji dünyasında devrim yaratmaya devam etmektedir. IoT'nin geleceği, hızla gelişen bağlantı teknolojileri, yapay zeka entegrasyonu, güvenlik ve gizlilik önlemleri, akıllı şehir çözümleri, Endüstri 4.0, sağlık hizmetlerinde yenilikler ve tarım sektöründeki uygulamalar gibi pek çok alanda devam eden çalışmalarla şekillenecektir.

Gelecek yıllarda, 5G ve sonrasında 6G gibi gelişmiş bağlantı teknolojileri, IoT cihazlarının daha hızlı, daha güvenilir ve daha geniş kapsamlı bir şekilde iletişim kurmasını sağlayacaktır. Bu teknolojiler, daha yüksek veri hızları ve daha düşük gecikme süreleri sunarak, IoT'nin potansiyelini daha da artıracaktır. Özellikle akıllı şehirler, endüstriyel otomasyon ve sağlık hizmetleri gibi alanlarda, bu gelişmiş bağlantı teknolojilerinin önemli etkileri olacaktır.

Yapay zeka (AI) ve makine öğrenimi (ML), IoT cihazlarının yeteneklerini önemli ölçüde artırmaktadır. AI ve ML entegrasyonu, IoT cihazlarının daha akıllı, özerk ve



veri analizi konusunda daha yetenekli hale gelmesini sağlamaktadır. Örneğin, akıllı ev sistemleri, kullanıcıların alışkanlıklarını öğrenerek daha kişiselleştirilmiş hizmetler sunabilirken, endüstriyel IoT sistemleri, üretim süreçlerini optimize edebilir ve verimliliği artırabilir.

Güvenlik ve gizlilik, IoT'nin büyümesiyle birlikte daha da kritik hale gelmiştir. IoT ekosisteminin korunması için daha gelişmiş güvenlik protokolleri ve şifreleme yöntemleri geliştirilmektedir. Aynı zamanda, veri gizliliği ve kullanıcı mahremiyeti konularında daha katı düzenlemeler ve standartlar oluşturulmaktadır. Bu önlemler, IoT cihazlarının ve kullanıcı verilerinin güvenliğini sağlamak için hayati öneme sahiptir.

Akıllı şehirler ve altyapı çözümleri, IoT'nin sunduğu en heyecan verici gelişmeler arasındadır. Trafik yönetimi, atık yönetimi, enerji tasarrufu ve kamu güvenliği gibi alanlarda IoT cihazları kullanılarak şehirler daha sürdürülebilir ve yaşanabilir hale getirilmektedir. Bu uygulamalar, şehirlerin daha verimli ve çevre dostu olmasını sağlayarak, vatandaşların yaşam kalitesini artırmaktadır.

Endüstri 4.0 devrimi, sanayi ve üretim süreçlerinde IoT'nin kullanımını merkezine almaktadır. Akıllı fabrikalar, üretim süreçlerinin otomasyonu ve optimizasyonu sayesinde daha verimli ve esnek hale gelmektedir. IoT sensörleri ve cihazları, üretim hattındaki her adımı izleyerek verimliliği artırmakta ve maliyetleri düşürmektedir. Bu da endüstriyel üretimin daha rekabetçi ve sürdürülebilir olmasını sağlamaktadır.

Sağlık sektöründe IoT uygulamaları, uzaktan hasta izleme, giyilebilir sağlık cihazları ve akıllı hastane çözümleri gibi alanlarda büyük ilerlemeler sağlamaktadır. Bu yenilikler, hastaların daha iyi izlenmesini ve sağlık hizmetlerinin daha etkin bir şekilde sunulmasını mümkün kılmaktadır. Özellikle kronik hastalıkların yönetimi ve acil durum müdahalelerinde, IoT teknolojilerinin önemi giderek artmaktadır.

Tarım sektöründe IoT, verimliliği artırmak ve doğal kaynakları korumak için kullanılmaktadır. Akıllı tarım uygulamaları, sulama sistemlerinin optimize edilmesi, toprak ve hava koşullarının izlenmesi gibi konularda çiftçilere yardımcı olmaktadır.

Ayrıca, çevre izleme sistemleri, iklim değişikliği ve çevresel etkiler konusunda daha hassas veri toplama ve analiz yapma imkanı sunmaktadır.

Sonuç olarak, IoT alanındaki çalışmalar hız kesmeden devam etmekte ve bu teknoloji, hayatımızın birçok alanında devrim yaratacak potansiyele sahiptir. Gelecekte, daha akıllı, daha bağlı ve daha verimli bir dünya yaratmak için IoT'nin kullanımı ve gelişimi sürecektir. Bu, yalnızca teknolojik ilerlemelerle sınırlı kalmayacak, aynı zamanda toplumsal ve ekonomik dönüşümlere de yol açacaktır.

### **6.1 Gelecekteki IOT Nesneler**

Nesnelerin İnterneti (IoT), teknoloji dünyasında hızla yayılan bir kavram haline gelmiştir. Gelişen teknolojiyle birlikte, IoT cihazlarının sayısı her geçen gün artmakta ve bu da gelecekteki IoT'nin daha da etkili olacağını işaret etmektedir. Özellikle, akıllı ev sistemleri ve şehir çözümleri gibi alanlarda IoT'nin kullanımının yaygınlaşması beklenmektedir [5]. En temelde Iot nesneleri bu kadar önemli kılan etmenlerden biri nano aygıtlardır[8].

Akıllı ev sistemleri, ev sahiplerine daha konforlu, güvenli ve enerji verimli bir yaşam sunmak amacıyla tasarlanmıştır. Gelecekte, evlerdeki IoT cihazları ve sensörlerin daha yaygın hale gelmesiyle birlikte, akıllı ev sistemlerinin daha da gelişmesi ve yaygınlaşması beklenmektedir.

Akıllı şehir çözümleri ise, şehirlerin daha sürdürülebilir, verimli ve yaşanabilir hale gelmesini sağlamak için IoT'nin kullanıldığı alanlardan biridir. IoT cihazları ve sensörleri, trafik yönetimi, çevre izleme, enerji yönetimi ve güvenlik gibi birçok alanda şehirlerin altyapısını iyileştirecek ve şehir yaşamını daha kolay hale getirecektir.

Bunun yanı sıra, endüstriyel otomasyon ve üretim alanında da IoT'nin etkisi oldukça büyüktür. Fabrikalardaki sensörler ve akıllı cihazlar, üretim süreçlerini izleyerek,

optimize ederek ve verimliliği artırarak endüstriyel otomasyonun gelişmesine katkı sağlayacaktır.

Sağlık ve tıp alanında da IoT'nin önemi giderek artmaktadır. Giyilebilir sağlık cihazları ve akıllı tıbbi cihazlar[7], hastaların sağlık durumlarını izlemek ve tedavi etmek için kullanılacaktır. Bu da sağlık hizmetlerinin daha erişilebilir ve etkili hale gelmesini sağlayacaktır.

Sonuç olarak, IoT'nin geleceği oldukça parlak görünmektedir. Akıllı evler, akıllı şehirler, endüstriyel otomasyon, sağlık ve tıp gibi alanlarda IoT'nin kullanımı giderek artacak ve hayatımızı daha da kolaylaştıracaktır.

## 7. KAYNAKLAR

- [1]. Hameed, S., Saquib, S.M.T., ul Hassan, M., and Junejo, F., “Radio frequency identification (RFID) based attendance & assessment system with wireless database records”, *ProcediaSocial and Behavioral Sciences*, 2015, 195, 2889-2895.
- [2]. Sinan Uğuz, S.U, Şafak Turan, Ş.F, “Parmak İzine Dayalı Taşınabilir Özellikli Öğrenci Yoklama Sistemi Geliştirilmesi”, *El-Cezerî Fen ve Mühendislik Dergisi*, 8, 1, 2021 (36-44)
- [3]. Fatma Meltem Aksakallı, F.M.A., Zehra Gül, Z.G., Parmak İzi Sensörü İle Kimlik Tanımlama , “*Erzurum Öğrenci Dergisi*”, 2024
- [4]. Serdar Öztürk, S.Ö., Hatice Ediz Atmaca, H.E.A , “İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim Bilişim Sistemleri Kapsamında İncelenmesi”, *Bilişim Teknolojileri Dergisi*, 10, 2, 2017, 202-203.
- [5]. Erdal Erdal, E.E., Atilla Ergüzen, A.E., “Nesnelerin İnterneti (IoT)”, “*Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi*”, 12, 3, 2020, 29-32
- [6]. Bekir Eray Katı, B.E.K, Ecir Uğur Küçükşille, E.U.K ,“ORACLE VE MS SQL SERVER Veri Tabanları İçin Veri Tabanı Yönetim Sistemleri Güvenlik Kontrol İlkelerinin Takip Edilmesi Ve Uygulanması”, “*SDU International Journal of Technological Sciences*”, 10, 2, 2021, 22-23
- [7].Bircan Çiftçi, B.Ç, Zeynep Şen, Z.Ş, Mustafa Alper Akkaş, M.A.A, “Nesnelerin İnterneti Tabanlı Kablosuz Taşınabilir EKG Cihazı”, “*Avrupa Bilim ve Teknoloji Dergisi*”, 26, 91-95.
- [8]. Emre Şahin, E.Ş., Orhan Dağdeviren, O.D., Mustafa Alper Akkaş, M.A.A, “Nano-Nesnelerin İnternetinin Gelecekteki Uygulamalarına Yönelik Bir Yol Haritası”, “*Avrupa Bilim ve Teknoloji Dergisi*”, 26, 2021, 174-179.

## **8. EKLER**

## 9. ÖZGEÇMİŞ

Bedir Tuğra Karaabalı 2001 yılında Ümraniye’de doğdu ilk bilgisayarla 2006 yılında Üsküdar’da tanıştı bilgi evlerinin ve internet kafelerin kapılarından ayrılmadı. Henüz yurt ile tanıştığında 11 yaşındaydı, öğrenmeye ve bu alanda çalışmalarına lise yıllarında başlayıp lise’de satranç ile ilgilendi. Lise eğitiminden sıyrılıp açıktan tamamladı Üniversite’de Ayvaz Elektromobil takımında arkadaşıyla beraber araç kontrol sistemi tasarladı şimdiye kadar tüm projeleri açık kaynak , açık kaynağın çok büyük bir destekçisi, yapay zeka ve veri bilimi alanlarında ilerlemek istiyor. Abant İzzet Baysal Üniversitesinde eğitim görmeye, yurttta kalmaya ve bilgisayarlara hayranlık duymaya devam ediyor.

