Umuttepe Turizm Otobüs Bileti Satış Sitesi
Umuttepe Tourism Bus Ticket Sales Site

1. Betül Erdik
211307033
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
betuler5301@gmail.com

2. Melisa Ceylan
191307022
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
melisacy22@outlook.com

3. Seher Köroğlu
211307016
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
211307016@kocaeli.edu.tr

***Özet*— Bu çalışmada, dört farklı şehre sefer düzenleyen bir otobüs bileti satış sitesi geliştirilmiştir. Proje ekibi olarak, CodeIgniter Framework'ü kullanarak PHP tabanlı bir web uygulaması tasarlanmış ve geliştirilmiştir. Veri yönetimi için MySQL veri tabanı kullanılmıştır. Otobüs bileti satış sitesi, kullanıcıların farklı şehirler arasında seyahat planlaması yapmalarını sağlayan bir platform olarak tasarlanmıştır. Projenin ana hedefleri arasında, kullanıcı dostu bir ara yüz sağlamak, seyahat planlamasını kolaylaştırmak, güvenli ödeme yöntemleri sunmak ve kullanıcıların seyahatlerini verimli bir şekilde yönetmelerine yardımcı olmak bulunmaktadır.**

Anahtar Kelimeler — CodeIgniter Framework; otobüs bilet satışı; otobüs bileti; bilet kiralama; koltuk seçimi; PHP ; MySQL.

I. GİRİŞ

Teknolojinin gelişimiyle birlikte seyahat planlaması ve bilet satın alma süreçleri büyük ölçüde dijitalleşmiştir. İnsanlar artık seyahatlerini internet üzerinden kolayca planlayıp bilet satın alabilmektedirler. Bu bağlamda, otobüs seyahatlerinde de çevrimiçi bilet satışı oldukça yaygınlaşmış ve kullanıcıların tercih ettiği bir yöntem haline gelmiştir. Bu proje, dört farklı şehir arasında sefer düzenleyen bir otobüs bileti satış sitesi geliştirmeyi amaçlamaktadır. İnternet tabanlı bu platform, kullanıcıların seyahat planlarını kolaylaştırmak ve bilet satın alma sürecini daha verimli hale getirmek için tasarlanmıştır. Proje, CodeIgniter Framework'ü kullanarak PHP tabanlı bir web uygulaması olarak geliştirilmiş ve MySQL veri tabanıyla entegre edilmiştir. Giriş bölümünde, projenin amacı ve kapsamı hakkında genel bir açıklama yapılmıştır. İlerleyen bölümlerde, projenin detaylarına ve geliştirme sürecine daha detaylı bir şekilde değinilecektir.

II. CODEIGNİTER

Projeyi gerçekleştirirken CodeIgniter frameworkunu kullandık. CodeIgniter PHP dilinde geliştirilen, MySQL veri tabanını destekleyen bir frameworktur.

A. Özellikler

- CodeIgniter diğer frameworklardan farklı olarak minimal bir yapılandırma gerektirir. Önceden yapılandırılmış çok fazla kütüphane ve yardımcı fonksiyonla beraber gelir. Bu sayede kullanımı da kolaylaşır. Kullanımı olduğu gibi kurulumu da oldukça kolaydır.
- Hafif ve performans odaklı bir yapıya sahiptir. Daha az bellek tüketip daha hızlı çalışabilir. Bu da uygulamaların daha hızlı bir şekilde geliştirilmelerini sağlar.
- Gelişmiş güvenlik önlemlerine sahiptir. SQL enjeksiyonlarına, CSRF saldırılarına karşı koruma sağlar.
- Büyük ve aktif bir topluluğa sahip olmasından dolayı yardım ve kaynak bulmak kolaydır.

1) MVC

Temel özelliklerinden biri de MVC mimarisini kullanmasıdır. MVC yani Model-View-Controller uygulamanın kullanıcı arayüzü, veri ve işleyiş mantığı olmak üzere farklı katmanlarını ayırarak uygulamanın daha düzenli, bakımı kolay, genişletilebilir ve daha kolay yönetilebilir hale gelmesini sağlar.

- MVC'de bulunan "M" yani model, veri işleme kısmını temsil eder. Bu katmanda veri tabanından veri çekme ve iş mantığı işlemleri gibi veriyle ilgili tüm işlemler gerçekleştirilir. Örnek olarak genelde veri tabanı tablolarını temsil eden sınıflar veya veri işleme işlevlerinden oluşur.
- "V" yani View (Görünüm), kullanıcı arayüzünü temsil eder. Bu katmanda kullanıcıya sunulacak olan içerik ve arayüz öğeleri bulunur. Kullanıcı arayüzü HTML, CSS JavaScript kullanılarak oluşturulur. View (Görünüm) genelde sunucu tarafında oluşturulan HTML ya da şablon dosyalarıdır.
- "C" yani Controller (Denetleyici), kullanıcının etkileşime girip uygulama mantığının kontrol edildiği katmandır. Kullanıcının isteğine göre yönlendirme işlemlerini de gerçekleştirebilir. Kullanıcının isteğini alır, uygun Modeli çağırarak veri işleme işlemlerini gerçekleştirir ve sonuçları Viewe aktarır.

B. Kurulum

- Ilk olarak CodeIgniter'ı resmi web sitesinden indirip sunucumuza yüklüyoruz *KAYNAKLAR[1]*
- Web projelerini çalıştırması için indirdiğimiz Xampp uygulamasının bulunduğu klasöre -xampp\htdocs- CodeIgniter dosyalarını aktarıyoruz. Ve hazır gelen bu MVC projesi üzerinde projemizi geliştirmeye başlıyoruz.
- Temel yapılandırma dosyalarını yani Database.php, .env ve config.inc.php dosyalarındaki bilgileri sunucuyu yansıtacak şekilde (kullanıcı adı, şifre, hostname) güncelliyoruz.

III. VERİ TABANI TASARIMI - MYSQL

Projenin veri tabanını tasarlamak için MySql veri tabanı yönetim sistemini kullanmayı tercih ettik. Projemizin veri tabanında ilk olarak biletler, koltuklar, otobussirketi, olmak üzere üç tane tablo oluşturduk. Bu tablolarda sırasıyla şu bilgiler yer almaktadır: Biletler tablosunda; biletin, kullanıcının, koltuğun ve seyahatin ID'leri, toplam bilet fiyatı, ödeme yöntemi (kredi kartı veya bakiye), biletin iptal edilip edilmediği bilgisi, rezervasyonun sona erme zamanı, PNR kodu ve satış zamanı bilgileri yer almaktadır. Koltuklar tablosunda; koltuğun ve seyahatin ID'leri, koltuk numarası, koltuğun kullanılabilir olup olmadığı, koltuğun rezerve edilip edilmediği, koltuğun satılıp satılmadığı, seyahat türü (tek yön veya gidiş dönüş), cinsiyet tercihi, yaş ve indirim türü (öğrenci, yaşlı, çalışan ve güvenlik) bilgileri yer almaktadır. Otobüs şirketi tablosunda ise otobüs şirketinin ID bilgisi ve otobüs şirketinin adı yer almaktadır.

Daha sonrasında bazı bilgileri depolamak amacıyla seyahatlertablosu, terminaller ve kullanıcılar adlı üç tablo oluşturduk. Seyahatler tablosunu seyahat bilgilerini depolamak için kullandık. Bu tabloda; seyahatlerin, kalkış terminalinin, varış terminalinin ID'leri, seyahatin başlangıç ve varış zamanı, seyahat türü (tek yön veya gidiş dönüş), otobüs şirketinin ID bilgisi ve otobüsün plaka bilgileri yer almaktadır. Terminaller tablosunu otogar terminallerinin bilgilerini saklama amacıyla kullandık. Bu tabloda; terminallerin ID bilgisi, terminalin adı, terminalin bulunduğu şehir ve terminaldeki peron harfleri bilgileri saklanmaktadır. Kullanıcı tablosunda ise kayıtlı kullanıcıların bilgileri saklanmaktadır. Bu tabloda kullanıcıların ID bilgisi, kullanıcı adı, şifre ve e-posta bilgileri yer almaktadır.

Oluşturduğumuz veri tabanındaki tabloların birbirleri ile ilişkileri şu şekildedir:

- Biletler tablosundaki kullanıcı ID bilgisi, kullanıcılar tablosundaki ID alanına referans oluşturur. Bu sayede bir biletin hangi kullanıcıya ait olduğu belirtilmiş olur.
- Biletler tablosundaki koltuğun ID bilgisi ile koltuklar tablosundaki koltuk ID bilgisi ilişkilendirilmiştir. Bu ilişkilendirme bir biletin hangi koltuğa atanmış olduğunu gösterir.
- Biletler tablosundaki seyahat ID bilgisi seyahatler tablosundaki seyahat ID bilgisi alanına referans oluşturur. Bu referans sayesinde bir biletin hangi seyahate ait olduğu belirtilmiş olur.
- Koltuklar tablosundaki seyahat ID bilgisi seyahatler tablosundaki seyahat ID bilgisi ilişkilendirilmiştir. Bu ilişki bir koltuğun hangi seyahate ait olduğunu belirtir.
- Seyahatler tablosundaki kalkış terminali ID bilgisi terminaller tablosundaki terminal ID bilgisi alanına referans oluşturur. Bu sayede bir seyahatin kalkış terminali belirlenmiş olur.
- Seyahatler tablosundaki varış terminali ID bilgisi, terminal tablosundaki terminal ID bilgisi ile ilişkilendirilmiştir. Bu ilişki sayesinde bir seyahatin varış terminali belirlenmiş olur.
- Seyahatler tablosundaki otobüs şirketi ID bilgisi otobüs şirketi tablosundaki şirket ID bilgisi alanına referans oluşturur. Bu referans sayesinde bir seyahatin hangi otobüs şirketi tarafından sağlandığı belirlenmiş olur.

IV. KULLANILAN YAZILIMSAL MİMARİ, YÖNTEM VE TEKNİKLER

Projemizi gerçekleştirme sürecinde projenin gereksinimlerini karşılayan çeşitli yazılımsal mimariler, yöntemler ve teknikler kullandık. Bu yöntemler arasında frontend ve backend teknolojileri, veri tabanı yönetimi, harita tabanlı hizmetler ve ödeme entegrasyonu gibi çeşitli alanlar bulunmaktadır. Kullandığımız teknolojiler şu şekildedir:

A. Payment

Payment, genel olarak bir ödeme işleme çözümü sağlayıcısı olarak adlandırılır. Ödeme, genellikle farklı bir ödeme yöntemlerini destekleyen bir API olarak sunulur. Ödeme işlemlerini yönetmek için kullanılır. Bu işlemleri yönetirken kullanılan çeşitli çözümleri ifade eder. Sitemizde kredi kartı ve bakiye olmak üzere iki ayrı ödeme sistemi olduğundan dolayı Payment teknolojisini kullanmak bize kolaylık sağladı. Paystack gibi belirli hizmet sağlayıcısının yanı sıra başka ödeme platformlarını da bulundurmaktadır.

B. Paystack

Paystack, ödeme işlemlerini kolaylaştırmak için geliştirilmiş bir ödeme platformudur. Web ve mobil uygulamalarda ödeme kabul etmek için geliştiricilere API'ler sunar. Bu API'ler, geliştiricilere ödeme işlemlerini yönetmek için gereken araçları sağlar. Paystack farklı ödeme yöntemlerini kabul etme, güvenlik önlemleri gibi özelliklere sahiptir. Payment genel bir terim olup Paystack özel bir ödeme platformu sağlayıcısıdır. *KAYNAKLAR[2]*

C. Google Maps API

Geliştiricilere harita ve konum tabanlı hizmetleri kullanma imkanı sunan bir platformdur. Bu API, web ve mobil

uygulamalara harita görselleştirilmesi eklemek ve konum tabanlı hizmetler oluşturmak için kullanılır. Biz projemizde kalkış ve varış noktası seçildikten sonra kullanıcıların hangi güzergahtan gideceklerini görmelerini sağlaması amacıyla Google Maps API'nin Directions API olarak adlandırılan bölümünü kullandık. *KAYNAKLAR[3]*

D. *Google Maps Directions API*

Google Maps API'nin "Directions API" olarak adlandırılan bölümü, belirli bir başlangıç noktasından belirli bir varış noktasına olan yol tariflerini almayı sağlar. Bu API, kullanıcılara navigasyon hizmetleri sunmak için kullanılır. Directions API sayesinde belirli bir başlangıç ve varış noktası arasında bir rota oluşturabildik. Bu rota boyunca da geçilecek noktaları, toplam mesafeyi görüntüleyebildik. Bilet satın aldıktan sonra kişi harita üzerinde hangi güzergahtan geçeceğini bu API sayesinde görüntüleyebilecektir. *KAYNAKLAR[4]*

V. WEB SİTESİ TASARIMI

A. *Ana Sayfa*

Web sitesine girişte config/routes.php dosyasında belirlediğimiz home controller altındaki index fonksiyonu çalışmaktadır. Bu index fonksiyonu da bize views/welcome_message.php sayfasını çağırmaktadır. Giriş ekranını bu sayfa üzerine yapacağız. Projemizde bootstrap yapısı kullanıyoruz. Geliştirmelerimizi de bunun üzerinden yapıyoruz. Ana sayfada giriş ve üye ol başlıkları olan bir sayfa tasarladık. Web sitesinde dolaşıp işlem gerçekleştirebilmek için üye olma şartımız aranmaktadır. Bu yüzden ana ekranda önce login ekranına geçiş yapmak için açıklama ve giriş yap butonumuz bulunuyor. Giriş yap butonundaki href bağlantısına Kod1'deki php kodunu yazıyoruz. Bu URL'e /login'i ekletecektir. URL'de yazılı olan adresin izleyeceği yolu ayarlamak için yine config/routes.php dizinine geliyoruz. Kod2'de göründüğü gibi login URL'ini Auth Controller'ın altındaki index methodunu çağıracak şekilde tanımlıyoruz. Bu sırada Controller dizini altına Auth.php sayfasını ekliyoruz. Ve extends olarak BaseController'ı ekliyoruz (Kod3). Tanımladığımız index fonksiyonunu views'deki auth klasöründe login.php sayfasını çağıracak şekilde ayarlıyoruz (Kod4).

```
<?= site_url('/login'); ?>
```

Kod1

```
$routes->get('/login','Auth::index');
```

Kod2

```
class Auth extends BaseController
```

Kod3

```
return view('auth/login')
```

Kod4

B. *Login*

Login sayfasında kullanıcıdan e-mail ve şifre isteyen bir form oluşturuyoruz. Daha önceden kaydı olmayanlar için alt kısmına üye ol bağlantısını ekliyoruz. Bu bağlantı bizi register sayfasına götürecek şekilde ayarlıyoruz.

Üye ol sayfasında isim, e-mail, şifre bilgilerini istiyoruz. Üye ol tuşuyla beraber verileri form üzerinden post tipiyle beraber auth controller'ındaki save fonksiyonuna yolluyoruz. Save fonksiyonunda gönderilen verilerin kontrollerini sağlıyoruz ve hata mesajları döndürüyoruz. Herhangi bir sorun olmaması durumunda post üzerinden aldığımız verileri (şifre verisi de şifrelenmiş şekilde Kod5) değişkenlere atıyoruz. Daha sonrasında veri tabanından oluşturduğumuz modelden yeni nesne oluşturup verileri buraya ekliyoruz ve kontrollerini sağlıyoruz Kod6. Daha sonrasında login sayfasına geri yönlendiriyoruz.

```
'password' => Hash::make($password)
```

Kod5

```
$userModel = new \App\Models\UsersModel();
$query = $userModel->insert($values);
    if(!$query){
        return redirect()->back()->with('fail', 'An error occured');
    }else{
        return redirect()->to('/register')->with('success', 'You are now registered');
    }
```

Kod6

Login sayfasında girdiğimiz form verilerini yine gerekli rotayı çizdikten sonra auth controller'ındaki check fonksiyonuna yolluyoruz. Burada girilen verilerin doğruluğunu kontrol ettikten sonra doğruysa dashboard, hatalıysa yine login ekranını çağırıyoruz.

C. *Bilet Bul*

Dashboard/index.php dizininde ana başlığa biletlerim sayfası ekliyoruz. Bilet aramak için bir form oluşturuyoruz. Bu formda nereden nereye gidileceğini hangi tarihte ve yönde olacağını seçiyoruz. Girilen bilgileri post ile bilet controller'daki biletAra fonksiyonuna gönderiyoruz. Bu fonksiyonda girilen verileri veri tabanında sorgulatıp ilgili bilet varsa bulup getiriyoruz Kod7. Sonuçları dashboard/searchResults view'ine gönderiyoruz.

```
$seyahatler = $model->table('seyahatlertablosu')
    ->select('tripID, tripType, departureTime, arrivalTime, departure.city AS departureCity, arrival.city AS arrivalCity, departure.terminalName AS departureTerminal, arrival.terminalName AS arrivalTerminal, busCompanyID, busPlate')
    ->join('terminaller AS departure', 'departure.terminalID = seyahatlertablosu.departureTerminalID')
```

```
        ->join('terminaller AS arrival', 'arrival.terminalID = seyahatlertablosu.arrivalTerminalID')
        ->where('departure.city', $inputFrom)
        ->where('arrival.city', $inputTo)
        ->where('DATE(departureTime)', $inputDate)
        ->whereIn('tripType', [$inputStatus])
        ->findAll();
    $data['seyahatler'] = $seyahatler;
```

Kod7

SearchResults sayfasına gelen verileri Kod8'de olduğu gibi ekrana yazdırıyoruz.

```
<?php  foreach ( $seyahatler as $seyahat) { ?>
    <h5 class="card-title
        ">Select a Bus Trip</h5>
        <form action="<?= base_url('bilet/biletBul'); ?>" method="post">
            <input type="hidden" name="tripID" value="<?= $seyahat['tripID']; ?>">
            <p>Trip ID: <?=$seyahat['tripID']; ?> | Trip Type: <?= $seyahat['tripType']; ?></p>
            <p>Departure Terminal: <?= $seyahat['departureTerminal']; ?> (<?= $seyahat['departureCity']; ?>) | Arrival Terminal: <?= $seyahat['arrivalTerminal']; ?> (<?= $seyahat['arrivalCity']; ?>)</p>
            <p>Departure Time: <?= $seyahat['departureTime']; ?> | Arrival Time: <?= $seyahat['arrivalTime']; ?> </p>
            <button type="submit" class="btn btn-primary">Select Trip</button>
            <br><br>
        </form>
    <?php } ?>
```

Kod8

D. *Bilet Detay*

Seçmek istenilen biletin select trip butonuna basıldığında ilgili bilete ait bilgiler bilet/biletBul fonksiyonuna gönderiliyor. Burada biletin ID'si ile biletin tüm bilgilerine erişim sağlanıyor. Ayrıca kullanılacak harita için de veri tabanında koordinatları bulunan seyahatin varış-kalkış terminalleri alınıyor. Ardından dashboard/biletBul view'ine aktarılıyor. Burada seyahat bilgileri daha ayrıntılı bir şekilde gösteriliyor.

1) *MAPS*

Alt kısımda da çekmiş olduğumuz kalkış-varış terminali bilgilerini kullanarak maps üzerinde sürüş moduyla rota çizimi yaptırıyoruz Kod9.

```
function initMap() {
        var map = new google.maps.Map(document.getElementById('map'), {
            zoom: 7
        });
        var directionsService = new google.maps.DirectionsService();
        var directionsRenderer = new google.maps.DirectionsRenderer({
            map: map
        });
        var start = {lat: <?= $departureTerminal['latitude'] ?>, lng: <?= $departureTerminal['longitude'] ?>};
        var end = {lat: <?= $arrivalTerminal['latitude'] ?>, lng: <?= $arrivalTerminal['longitude'] ?>};
        var request = {
            origin: start,
            destination: end,
            travelMode: 'DRIVING'
        };
        directionsService.route(request, function(response, status) {
            if (status === 'OK') {
                directionsRenderer.setDirections(response);
            } else {
                window.alert('Directions request failed due to ' + status);
            }
        });
    }
```

Kod9

2) *Koltuk Seçim*

Koltuk seçme görünümü için for döngüsü ile 32 adet koltuk oluşturuyoruz Kod10. Oluşturduğumuz bu koltuk düzeninin altına yolcu sayısı, cinsiyet, yaş, indirim türü seçeneklerini ekliyoruz. Bilgiler girildikten sonra da altına koltuk rezerve et, koltuk satın al ve hemen satın al butonlarını ekliyoruz. Koltuklar üzerinde seçim yapabilmek için bir script kodu yazıyoruz. Bu script kodunda koltuk numarası, koltuk seçimi, renk değişimi, seçim iptali, yeterli koltuk seçili değilse ve seçilen koltuk yeniyse renk değiştirme, yan yana farklı cins oturumu kontrolü gibi birçok kontrol sağlanıyor Kod11.

```
for ($row = 1; $row <= 4; $row++) {
        for ($column = $row; $column <= 32; $column += 4) {
            echo '<div class="seat';
```

```
                // Check the selected gender and add appropriate CSS class
                if (isset ($_POST['genderPreference'])) {
                  if ($_POST['genderPreference'] == 'Male') {
                    echo ' male';
                  }  elseif  ($_POST['genderPreference']  == 'Female') {
                    echo ' female';
                  }
                }
                echo '">' . $column . '</div>';
              }
              echo '<br>'; // Her satırın sonunda bir alt satıra geçmek için bir satır ekle
            }
```

Kod10

```
    selectedGender = document.getElementById('genderPreference').value;

        const numPassengers = parseInt(numPassengersInput.value);

        // Koltuğun numarasını al

        const seatNumber = parseInt(seat.textContent);

        // Koltuğun rengini değiştir veya geri al

        if (selectedSeats.includes(seatNumber)) {

          // Seçili koltukları geri al

          seat.classList.remove(selectedGender.toLowerCase());

          selectedSeats = selectedSeats.filter(num => num !== seatNumber);

        } else if (selectedSeats.length < numPassengers) {

          // Yeterli koltuk seçilmediyse ve seçilen koltuk yeni ise rengini değiştir

          // ve yan yana oturan farklı cinsiyetteki yolcuları kontrol et

          if (!selectedSeats.length || !selectedSeats.find(selectedSeat => Math.abs(selectedSeat - seatNumber) === 1) ||

            !selectedSeats.find(selectedSeat => {

              const selectedSeatElement = document.querySelector(`.seat:nth-child(${selectedSeat})`);

              const selectedSeatGender = selectedSeatElement.classList.contains('Male') ? 'Male' : 'Female';

              return selectedSeatGender !== selectedGender.toLowerCase();

            })) {

            seat.classList.add(selectedGender.toLowerCase());

            selectedSeats.push(seatNumber);

          }

        }
```

Kod11

E. Ödeme

Altta yer alan hemen satın al butonuyla OdemeController altındaki payment fonksiyonunda gidiyoruz. Bu controller'da ödeme için Iyzico kütüphanesi kullanıyoruz ve belirtilen fonksiyonda ödeme bilgilerine göre bir ödeme formu oluşturuluyor. Buradan da dashboard/odeme view'ine gönderiliyor. Bu view'de kullanıcıdan mail adresi, adı ve soyadı bilgileri alınıyor. Pay butonuna basıldığında ise payWithPayStack() fonksiyonu çalıştırılıyor Kod12. PayStack platformu online ödeme işlemlerini oldukça kolaylaştırıyor. Bu fonksiyonda bir popup açılıyor, önce ödeme formu yapılandırılıp API ile erişim sağlanıyor. Sonrasında e-mail ve ödeme miktarı bilgileri kullanılıyor. Açılan popupta giriş doğrulaması yapılıyor. Ardından ödeme miktarı bilgisi gösteriliyor. İşlem onay butonuyla da ödeme işlemi tamamlanıyor ve kullanıcının mail adresine bilgilendirme mesajı gönderiliyor. Ödeme işlemi tamamen bittikten sonra da belirli saniye sonra bilet bul sayfasına geri yönlendiriliyor.

```
    function payWithPaystack() {
        let handler = PaystackPop.setup({
            key: 'pk_test_f282c6fe1981b4b52220c218bf987211a997f39c', // Replace with your public key
            email: document.getElementById("email").value,
            amount: document.getElementById("amount").value * 100, // Convert to kobo (100 kobo = 1 ZAR)
            currency: "ZAR", // Specify currency as ZAR
            ref: '' + Math.floor((Math.random() * 1000000000) + 1), // Generate a pseudo-unique reference
            onClose: function () {
              alert('Window closed.');
            },
            callback: function (response) {
              let message = 'Payment complete! Reference: ' + response.reference;
```

```
                alert(message);


                setTimeout(function(){
                window.location.href = "/dashboard"; // Yönlendirme yapılacak sayfa
            }, 1000); // Bekleme süresi: 3000 milisaniye (3 saniye)
                }
            });
            handler.openIframe();
        }
```

Kod12

KAYNAKLAR

[1] https://codeigniter.com/download

[2] https://paystack.com/docs/api/

[3] https://developers.google.com/maps

[4] https://developers.google.com/maps/documentation/directions/?_gl=1*1qli6qp*_ga*ODUwNDM1NDY1LjE3MTEzMTM2MDY.*_ga_NRWSTWS78N*MTcxMTMxMzYwOC4xLjEuMTcxMTMxMzYyMS4wLjAuMA..

[5] https://getbootstrap.com/docs/5.3/getting-started/introduction/