



KOCAELİ ÜNİVERSİTESİ

Mühendislik Fakültesi  
Yazılım Mühendisliği Bölümü

## Programlama Laboratuvarı III

### FİNAL PROJE RAPORU

Uçak Bileti Rezervasyon Sistemi

**Ad - Soyad:** Sena Nur Dülger      **No:** 220229053

**Ad - Soyad:** Serhat Can Bakır      **No:** 220229036

**Tarih:** 8 Ocak 2026

Bu rapor, vize raporunda belirtilen temel analizlere ek olarak projenin tamamlanma sürecine ve final isterlerine odaklanmaktadır.

## 1. Sistemin Tamamlanması ve Entegrasyon

Bu proje, bağımsız olarak geliştirilen kullanıcı, rezervasyon ve yönetim modüllerinin ASP.NET Core MVC mimarisi altında tek bir çatı altında entegre edilmesiyle tamamlaşmıştır.

### 1.1. Entegrasyon Süreci

Projenin başlangıcında ayrı ayrı tasarlanan veritabanı tabloları (Users, Flights, Planes vb.) SqliteDbHelper sınıfı ve Entity modelleri aracılığıyla birbirine bağlanmıştır. Örneğin, bir rezervasyon (Reservation) oluştururken hem Kullanıcı (User), hem Uçuş (Flight) hem de Koltuk (Seat) tablolarından veriler doğrulanarak bütünlük sağlanmıştır.

### 1.2. Uçtan Uca Akış

Sistem, bir misafir kullanıcının siteye girip üye olması (Account/Register), giriş yapması, uçuş araması, koltuk seçmesi ve rezervasyonu tamamlaması sürecini kesintisiz olarak yürütmektedir.

## 2. Kullanıcı Arayüzü (UI) Geliştirmeleri

Final sürümde kullanıcı deneyimini artırmak ve modern bir görünüm sağlamak amacıyla arayüzde önemli iyileştirmeler yapılmıştır.

- **Front-End Teknolojileri:** Proje arayüzü HTML5, CSS3 ve JavaScript kullanılarak geliştirilmiştir. Razor View Engine sayesinde sunucu taraflı veriler dinamik olarak HTML içerisinde aktarılmaktadır.
- **Responsive Tasarım:** Sayfalar, mobil ve masaüstü cihazlarla uyumlu olacak şekilde izgara (grid) sistemi kullanılarak tasarlanmıştır.
- **Kullanıcı Geri Bildirimleri:** Hatalı girişlerde (örneğin yanlış şifre) veya başarılı işlemlerde (örneğin rezervasyon onayı) kullanıcıya TempData ve ViewData mekanizmaları aracılığıyla anlık bildirimler gösterilmektedir.

## 2.1. Ortak Sayfalar

(a) Ana sayfa

(b) Kayıt ekranı

(c) Hata gösterimi

(d) Uçuş Ara

Figure 1: Ortak sayfalara ait ekran görüntüleri

## 2.2. Kullanıcı Sayfaları

(a) Profil Bilgilerim

(b) Koltuk Seçimi

(c) Rezervasyonlarım

Figure 2: Kullanıcı sayfalarına ait ekran görüntüleri

## 2.3. Admin Sayfalar

Panel	İçerik / Özellikler
(a) Rezervasyonlar	Rezervasyonlar listesi (ID, Adres, Uygun, Adet, Fiyat, Durum, İşlemler)
(b) Kullanıcılar	Kullanıcılar listesi (ID, TC No, Adı - Soyadı, E-posta, Doğum Tarihi, Telefon, Durum, İşlemler)
(c) Uçaklar	Uçaklar listesi (ID, Uçak Adı, Koltuk Sayısı, İşlemler)
(d) Havaalanları	Havaalanları listesi (ID, Adı, Şehir, Ülke, İşlemler)
(e) Uçuşlar	Uçuşlar listesi (ID, Uçak Numarası, Havalimanı, Ülkeyi, Hangi, Kalkış Zamanı, Varış Zamanı, Uçuş Tipi, İşlemler)
(f) Koltuklar	Koltuklar listesi (ID, Sıra No, Saat, Uçak, İşlemler)

Figure 3: Admin paneline ait ekran görüntüleri

## 3. Kullanıcı Doğrulama ve Rol Yönetimi (Authentication & Authorization)

Sistemin güvenliği, cookie tabanlı kimlik doğrulama (*Cookie Authentication*) mekanizması ile sağlanmıştır.

### 3.1. Giriş ve Kayıt

- Kullanıcılar, e-posta ve şifre bilgileri ile sisteme giriş yapmaktadır.
- **Register (Kayıt)** işlemi sırasında aşağıdaki kontroller sunucu tarafında gerçekleştirilmektedir:
  - TC Kimlik Numarası doğrulaması,
  - Yaş kontrolü (18 yaş sınırı),
  - E-posta adresinin benzersiz (unique) olup olmadığı kontrolü.

```

// ----- REGISTER -----
[HttpGet]
[AllowAnonymous]
public IActionResult Register()
{
    return View();
}

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public IActionResult Register(
    string tcNo,
    string firstName,
    string lastName,
    DateTime? birthDate,
    string email,
    string phone,
    string password)
{
    // Boş alan kontrolü
    if (string.IsNullOrEmpty(tcNo) ||
        string.IsNullOrEmpty(firstName) ||
        string.IsNullOrEmpty(lastName) ||
        string.IsNullOrEmpty(email) ||
        string.IsNullOrEmpty(phone) ||
        string.IsNullOrEmpty(password) ||
        !birthDate.HasValue)
    {
        ViewData["Error"] = "Tüm alanları doldurun.";
        return View();
    }

    // TC kimlik kontrolü
    if (tcNo.Length != 11)
    {
        ViewData["Error"] = "TC Kimlik Numarası 11 haneli olmalıdır.";
        return View();
    }

    // 18 yaş kontrolü
    var today = DateTime.Today;
    var age = today.Year - birthDate.Value.Year;
    if (birthDate.Value.Date > today.AddYears(-age))
    {
        age--;
    }

    if (age < 18)
    {
        ViewData["Error"] = "Sisteme kayıt olabilmek için en az 18 yaşında olmalısınız.";
        return View();
    }

    bool success = UserModel.Register(
        tcNo,
        firstName,
        lastName,
        birthDate.Value,
        email,
        phone,
        password);

    if (success)
    {
        TempData["Success"] = "Kayıt başarılı. Lütfen giriş yapın.";
        return RedirectToAction("Login");
    }

    ViewData["Error"] = "Kayıt sırasında bir hata oluştu veya bu e-posta zaten kayıtlı.";
    return View();
}

```

Figure 4: Register metodunda sunucu tarafı doğrulamalar (TC kimlik, yaş kontrolü vb.).

### 3.2. Rol Yönetimi

- Sistemde **Admin** ve **Customer (Müşteri)** olmak üzere iki temel rol bulunmaktadır.
- Kullanıcının rol bilgisi, giriş işlemi sırasında `ClaimTypes.Role` olarak cookie içeriğine yazılmaktadır.
- **Admin Paneli:** Sadece `[Authorize(Roles = "Admin")]` niteliğine (attribute) sahip kullanıcılar /Admin rotasına erişebilmektedir.

```

// Admin tarafından tüm yönetimsel işlemleri yöneten controller.
// Uçak, havayolu, uçuş, koltuk, kullanıcı ve rezervasyon yönetimi burada yapılır.
[Authorize(Roles = "admin")]
public class AdminController : Controller
{
    public IActionResult Index()
    {
        return View();
    }

    // --- USERS ---
    public IActionResult Users()
    {
        var users = SqliteDbHelper.ExecuteQuery("SELECT * FROM users", null);
        return View(users);
    }

    // --- PLANES ---
    public IActionResult Planes()
    {
        var planes = SqliteDbHelper.ExecuteQuery("SELECT * FROM planes", null);
        return View(planes);
    }

    // GET: Admin/AddPlane
    public IActionResult AddPlane()
    {
        return View();
    }
}

```

(a) Rol bazlı yetkilendirme (Admin)

```

// Cookie içine yazılacak claim'ler
var claims = new List<Claim>
{
    new Claim(ClaimTypes.Name, user.Username),
    new Claim(ClaimTypes.Email, user.Email),
    new Claim(ClaimTypes.Role, user.Role == UserModel.RoleAdmin ? "Admin" : "Customer"),
    new Claim("UserId", user.Id.ToString())
};

var claimsIdentity = new ClaimsIdentity(claims, "CookieAuth");
var authProperties = new AuthenticationProperties
{
    IsPersistent = true
};

await HttpContext.SignInAsync(
    "CookieAuth",
    new ClaimsPrincipal(claimsIdentity),
    authProperties);

// ADMİN İse admin pipeline
if (user.Role == UserModel.RoleAdmin)
{
    return RedirectToAction("Index", "Admin");
}

// Normal kullanıcı: varsa geçerli returnUrl'e, yoksa Booking/Index'e
if (!string.IsNullOrEmpty(returnUrl) && Url.IsLocalUrl(returnUrl))
{
    return Redirect(returnUrl);
}

return RedirectToAction("Index", "Booking");
}

```

(b) Cookie Authentication

Figure 5: Yetkilendirme ve cookie tabanlı kimlik doğrulama kod parçaları

## 4. Veri Yönetimi ve Raporlama Fonksiyonları

Yönetici (Admin) modülü, yalnızca temel veri giriş işlemleriyle sınırlı kalmayıp, veri bütünlüğünü koruyan mantıksal kontroller ve detaylı raporlama yetenekleri içerecek şekilde tasarlanmıştır. Bu sayede sisteme tutulan verilerin doğruluğu, tutarlılığı ve izlenebilirliği sağlanmıştır.

### 4.1. Veri Tutarlılığı ve Kayıt İşlemleri

Admin paneli üzerinden gerçekleştirilen ekleme ve silme işlemleri, veritabanındaki ilişkisel bütünlüğü koruyacak şekilde yapılandırılmıştır. Sistemde veri tutarsızlığına yol açabilecek durumlar, işlem aşamasında kontrol edilerek engellenmektedir.

#### 4.1.1. Akıllı Uçuş Planlama

Uçuş ekleme (AddFlight) işlemi sırasında, bir uçağın aynı zaman diliminde iki farklı uçuşta yer alamayacağı varsayıyı temel almıştır. Bu kapsamda, yeni bir uçuş eklenmeden önce seçilen uçağın belirtilen tarih ve saat aralığında başka bir uçuşa atanmadığı kontrol edilmektedir.

Bu kontrol aşağıdaki SQL sorgusu aracılığıyla gerçekleştirilmektedir:

```

// Aynı uçak için çakışma kontrolü
var overlaps = SqliteDbHelper.ExecuteScalar<int>(
    @"SELECT COUNT(*) FROM flights
    WHERE plane_id = @pid
    AND id <> @fid
    AND (departure_time < @arr AND arrival_time > @dep)",
    cmd =>
    {
        cmd.Parameters.AddWithValue("@pid", flight.PlaneId);
        cmd.Parameters.AddWithValue("@fid", flight.Id);
        cmd.Parameters.AddWithValue("@dep", departureTime.ToString("s"));
        cmd.Parameters.AddWithValue("@arr", arrivalTime.ToString("s"));
    });

```

Figure 6: Aynı uçak için çakışma kontrolü sorgusu ve parametreler.

Eğer sorgu sonucunda ilgili uçağın aynı zaman aralığında başka bir uçuşu olduğu tespit edilirse, sistem uçuş ekleme işlemini iptal etmekte ve yöneticiye uyarı mesajı

göstermektedir.

#### 4.1.2. Güvenli Silme

Sistem, bağlı verilerin kontolsüz şekilde silinmesini engelleyerek yetim kayıtların olmasını önlemektedir. Bu kapsamda aşağıdaki kurallar uygulanmaktadır:

- **Uçak Silme:** Eğer bir uçağa atanmış aktif uçuşlar bulunuyorsa, ilgili uçak silinemez.
- **Uçuş Silme:** Eğer bir uçuşa ait aktif rezervasyonlar mevcutsa, uçuş silme işlemi engellenir ve öncelikle rezervasyonların iptal edilmesi istenir.

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult DeletePlane(int id)
{
    try
    {
        // Bu uçağa bağlı uçuş var mı?
        var fcount = SqliteDbHelper.ExecuteScalar<int>(
            "SELECT COUNT(*) FROM flights WHERE plane_id=@id",
            cmd => cmd.Parameters.AddWithValue("@id", id));
    }
    if (fcount > 0)
    {
        TempData["Error"] = "Bu uçak için uçuşlar mevcut: silmeden önce uçuşları kaldırın.";
        return RedirectToAction(nameof(Planes));
    }
}
```

Figure 7: Uçak silme işleminde bağlı uçuş kontrolü

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult DeleteFlight(int id)
{
    try
    {
        // Aktif rezervasyon var mı?
        var activeCount = SqliteDbHelper.ExecuteScalar<int>(
            @"SELECT COUNT(*)
              FROM reservations
             WHERE flight_id=@id
               AND (status IS NULL OR UPPER(status) <> 'CANCELLED'),
            cmd => cmd.Parameters.AddWithValue("@id", id));
    }
    if (activeCount > 0)
    {
        TempData["Error"] = "Bu uçuş için aktif rezervasyonlar mevcut: silmeden önce bu rezervasyonları iptal edin.";
        return RedirectToAction(nameof(Flights));
    }
}
```

Figure 8: Uçuş silme işleminde aktif rezervasyon kontrolü

Bu yaklaşım sayesinde sistemdeki ilişkisel yapı korunmakta ve veri kaybı riski azaltılmaktadır.

## 4.2. Raporlama ve Anlık İzleme

Yöneticiler, Admin/Reservations ekranı üzerinden sistemdeki rezervasyonları anlık olarak görüntüleyebilmektedir. Raporlama modülü, reservations, users, flights ve seats tablolarının birleştirilmesiyle oluşturulmuştur.

### 4.2.1. Rezervasyon Detay Raporu

Sistem, her rezervasyon için kullanıcı bilgisi, uçuş bilgisi, bilet fiyatı ve rezervasyon durumunu tek bir satırda sunmaktadır.

### 4.2.2. İptal Yönetimi

Admin, rezervasyonları silmek yerine durumunu CANCELLED olarak güncelleyebilmektedir. Bu sayede veriler korunmakta ve geçmişe dönük raporlama yapılmaktadır.

## 5. Vize Kısmından Sonra Eklenen Özellikler

Vize sonrasında projenin işlevsellliğini artırmak ve daha gerçekçi bir simülasyon sunmak adına sisteme Dinamik Fiyatlandırma ve İndirim Kuponu modülleri eklenmiştir.

## 5.1. Dinamik Fiyatlandırma Algoritması

Her uçuş sorgusunda sistem, ilgili uçağın mevcut rezervasyon sayısını toplam koltuk sayısına bölerek bir Doluluk Oranı (Occupancy Ratio) hesaplamaktadır. FlightModel içerisinde tanımlanan CalculateDynamicPrice metodu, bu oranı baz alarak taban bilet fiyatı üzerinde dinamik bir fiyat güncellemesi gerçekleştirmektedir. Uçak boşken taban fiyat geçerli olurken, doluluk oranı arttıkça bilet fiyatı otomatik olarak yükselmektedir.

```
// Uçuşa kalan gün + doluluk + koltuk tipi + sezon
// occupancyRatio : 0.0-1.0 arası doluluk oranı
public double CalculateDynamicPrice(int seatClass, double occupancyRatio, DateTime? nowOverride = null)
{
    var now = nowOverride ?? DateTime.Now;

    // 1) Zaman faktörü
    var timeToDeparture = DepartureTime - now;
    var days = timeToDeparture.TotalDays;

    double timeFactor;
    if (days < 0)
    {
        // Geçmiş uçuş, zaman faktörü 1 ve direkt baz fiyat
        timeFactor = 1.0;
    }
    else if (days >= 30)
    {
        // 30+ gün varsa %40 indirim
        timeFactor = 0.60;
    }
    else
    {
        // 0-30 gün arasında lineer olarak 0.6 => 1.0
        // days=30 => 0.6, days=0 => 1.0
        timeFactor = 1.0 - (days / 30.0) * 0.4;
    }

    // 2) Doluluk faktörü
    // 0-30% : 0.9 (indirim)
    // 30-70% : 1.0 (normal)
    // 70-90% : 1.2 (artış)
    // 90-100% : 1.4 (yüksek artış)
    double occ = Math.Clamp(occupancyRatio, 0.0, 1.0);
    double occupancyFactor;
    if (occ < 0.30)
        occupancyFactor = 0.90;
    else if (occ < 0.70)
        occupancyFactor = 1.00;
    else if (occ < 0.90)
        occupancyFactor = 1.20;
    else
        occupancyFactor = 1.40;

    // 3) Koltuk tipi faktörü
    // Economy : 1.0
    // Business: 1.5
    double seatFactor = seatClass == 1 ? 1.50 : 1.00;

    // 4) Sezon faktörü (ay bazlı)
    // Yüksek sezon: Haziran-Eylül (6-9)      => 1.20
    // Ortal sezon : Mayıs, Ekim, Aralık (5,10,12)=> 1.05
    // Düşük sezon : diğer aylar                 => 0.95
    int m = now.Month;
    double seasonFactor;
    if (m >= 6 && m <= 9)
    {
        seasonFactor = 1.20;
    }
    else if (m == 5 || m == 10 || m == 12)
    {
        seasonFactor = 1.05;
    }
    else
    {
        seasonFactor = 0.95;
    }

    double finalFactor = timeFactor * occupancyFactor * seatFactor * seasonFactor;
    double result = Price * finalFactor;

    return Math.Round(result, 2, MidpointRounding.AwayFromZero);
}
```

Figure 9: Doluluk oranı, zaman, koltuk tipi ve sezon faktörlerine göre bilet fiyatı hesaplama algoritması.

## 5.2. İndirim Kuponu Sistemi

Rezervasyon sırasında promosyon kodu kullanılarak indirim uygulanmasını sağlayan bir kupon sistemi geliştirilmiştir. CouponModel ve coupons tablosu üzerinden yönetilen bu yapı, kuponun geçerlilik tarihi ve indirim oranını kontrol ederek, geçerli olması durumunda toplam tutar üzerinden indirim uygulamakta ve güncel fiyatı kullanıcıya göstermektedir.

```
using System;
using System.Collections.Generic;
using prgmlab3.data;

namespace prgmlab3.Models
{
    public class CouponModel : BaseModel
    {
        public int Id { get; set; }
        public string Code { get; set; } = "";
        public int DiscountPercent { get; set; }
        public DateTime? ExpirationDate { get; set; }

        public static void EnsureTableExists()
        {
            SqliteDbHelper.ExecuteNonQuery(@""
                CREATE TABLE IF NOT EXISTS coupons (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    code TEXT UNIQUE NOT NULL,
                    discount_percent INTEGER NOT NULL,
                    expiration_date TEXT
                );
            ");

            // Kupon oluşturma
            var count = SqliteDbHelper.ExecuteScalar<long>("SELECT COUNT(*) FROM coupons;", null);
            if (count == 0)
            {
                SqliteDbHelper.ExecuteNonQuery(@""
                    INSERT INTO coupons (code, discount_percent, expiration_date)
                    VALUES ('YENIYIL20', 20, '2025-12-31');
                ");

                SqliteDbHelper.ExecuteNonQuery(@""
                    INSERT INTO coupons (code, discount_percent, expiration_date)
                    VALUES ('INDIRIM10', 10, '2026-06-01');
                ");
            }
        }

        public static CouponModel? GetByCode(string code)
        {
            if (string.IsNullOrWhiteSpace(code)) return null;

            var rows = SqliteDbHelper.ExecuteQuery(
                "SELECT * FROM coupons WHERE code=@code COLLATE NOCASE", // Case insensitive match
                cmd => cmd.Parameters.AddWithValue("@code", code.Trim())
            );

            if (rows.Count == 0) return null;

            var r = rows[0];
            return new CouponModel
            {
                Id = Convert.ToInt32(r["id"]),
                Code = Convert.ToString(r["code"]) ?? "",
                DiscountPercent = Convert.ToInt32(r["discount_percent"]),
                ExpirationDate = DateTime.TryParse(Convert.ToString(r["expiration_date"]), out var dt) ? dt : null
            };
        }

        public bool IsValid()
        {
            if (ExpirationDate.HasValue && ExpirationDate.Value < DateTime.Now)
                return false;
            return true;
        }
    }
}
```

Figure 10: Kupon yönetimi ve doğrulama işlemlerini gerçekleştiren CouponModel sınıfı.

## 6. Sonuç ve Değerlendirme

Uçuş Rezervasyon Sistemi projesi, belirlenen analiz ve tasarım isterlerine uygun olarak başarıyla tamamlanmıştır. Sistem, güvenli üyelik yapısı, dinamik rezervasyon yönetimi ve kapsamlı yönetici paneli ile gerçek hayat senaryolarını simüle edebilecek yetkinliğe ulaşmıştır.