

T.C.

KONYA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

Moment Projesi Detaylı Raporu

1. Genel Bilgi

- Proje Adı: Moment
- Ders: Web Teknolojileri
- Geliştirici: Kerim Serkan Şahin
- Danışman: Dr. Öğr. Üyesi Burak Yılmaz
- Dönem: 2024-2025 Güz Dönemi
- Amaç:
 - Moment, kullanıcıların anılarını paylaşımlarına, fotoğraf yüklemelerine, diğer kullanıcılarla etkileşim kurmalarına ve birbirlerini takip etmelerine olanak tanıyan bir sosyal medya platformu olarak geliştirilmiştir.
 - Platform, kullanıcı dostu bir arayüz ve güçlü bir backend ile desteklenerek kolay kullanılabilirlik ve yüksek performans sağlamayı amaçlamaktadır.
 - Kullanıcılar, fotoğraflarını yükleyerek ve açıklamalar ekleyerek dijital albümler oluşturabilir, diğer kullanıcıların içeriklerini beğenebilir ve yorum yapabilir.
 - Anlık bildirimler ve takip sistemleri ile etkileşim artırılarak, sosyal medya deneyimi iyileştirilmektedir.
 - Kimlik doğrulama ve güvenlik önlemleri ile kullanıcı verilerinin güvenliği sağlanmıştır.

2. Kullanılan Teknolojiler

- Backend: .NET Core, ASP.NET MVC
 - .NET Core, Microsoft tarafından geliştirilen açık kaynaklı bir platformdur ve yüksek performanslı, ölçeklenebilir web uygulamaları geliştirmek için kullanılır.
 - ASP.NET MVC (Model-View-Controller), katmanlı mimariyi destekleyerek kodun daha düzenli ve yönetilebilir olmasını sağlar.
 - Entity Framework Core (EF Core), veritabanı işlemlerini kolaylaştıran bir ORM (Object-Relational Mapping) aracıdır.
- Frontend: HTML, CSS, Bootstrap, JavaScript

- **HTML (HyperText Markup Language):** Sayfa yapısını oluşturmak için kullanılmıştır.
- **CSS (Cascading Style Sheets):** Sayfa tasarımını ve stil yönetimini sağlamak için kullanılmıştır.
- **Bootstrap:** Responsive ve modern tasarımlar için kullanılmıştır. Hazır bileşenler sayesinde geliştirme süreci hızlandırılmıştır.
- **JavaScript:** Dinamik içerik yönetimi ve kullanıcı etkileşimleri için kullanılmıştır.

Aşağıda anasayfa , profil ve etkileşim sayfalarından görüntüler bulunmaktadır. Bu görüntülerde bildirim, fotoğraf yükleme, fotoğraf silme , fotoğrafları favorilere ekleme , favorilerden silme , diğer kullanıcıları takip etme , takip ettiğin kullanıcıları takipten çıkarma , takip ettiğin kullanıcıların fotoğraflarını görme ve bu fotoğraflara yorum ve beğeni gibi özellikler

Moment [Home](#) [Profile](#) [Favorites](#) [Logout](#) [Feed](#)

Profile

User Information

Name: Kerim Serkan

Email: kerim.sahin@surpay.com

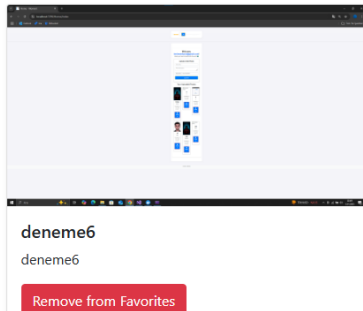
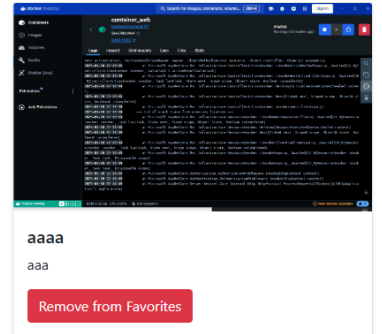
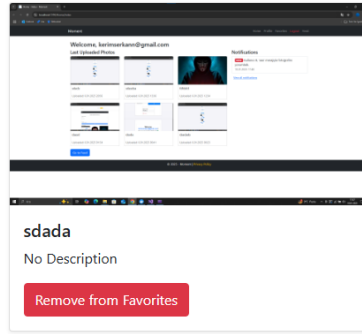
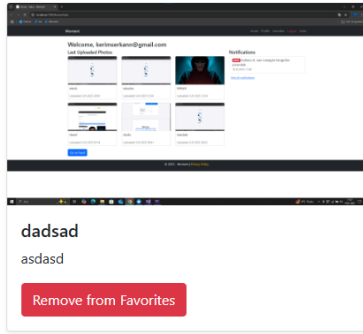
Follow New Users

Kerim Serkan	kerimserkann1@gmail.com	Follow
Kerim Serkan	kesdas@adsdada	Follow
deneme	deneme@gmail.com	Follow
DASD	kerimserkjhhjann@gmail.com	Follow
dassdadasdas	kerimserkandsasdn@gmail.com	Follow
DASD	kerimserkann@gmail	Follow

Following (Takip Ettiğin Kullanıcılar)

Name	Email	
Kerim Serkan	kerimserkann@gmail.com	Unfollow

Your Favorite Photos



Feed

Yorumlar



Kerim Serkan ?? \$"User ID (comment.UserId)" (12.01.2025 14:51)
dasdasdasd

dddd
dddd

Uploaded by Kerim Serkan
12 Ocak 2025 Pazar 15:17

Likes: 0 [Like](#)

Comments: 0

Add a comment...

Submit Comment

deneme4
deneme4

Uploaded by Kerim Serkan
10 Ocak 2025 Cuma 20:24

Likes: 1 [Unlike](#)

Comments: 1 [\(Yorumları Gör\)](#)


Add a comment...


Submit Comment


Feed


Welcome, kerimserkan@gmail.com


Last Uploaded Photos
























Notifications

 kullanıcı 5, yeni fotoğraf yükledi: 'deneme4'

[View all notifications](#)

deneme4

deneme4


Uploaded by Kerim Serkan

12 Ocak 2025 Pazar 15:17

Likes: 0 [Like](#)

Comments: 0

[Submit Comment](#)



deneme4

deneme4

Uploaded by Kerim Serkan

10 Ocak 2025 Cuma 20:24

Likes: 1 [Unlike](#)

Comments: 1 [\(Yorumları Gör\)](#)

[Submit Comment](#)

deneme4

deneme4

Uploaded: 12.01.2025 15:17

[Delete](#) [Add to Favorites](#)

deneme6

deneme6

Uploaded: 10.01.2025 20:24

[Delete](#) [Add to Favorites](#)

NEW kullanıcı 5, yeni bir fotoğraf yükledi: 'ddddd'

12.01.2025 15:17

NEW kullanıcı 5, yeni bir fotoğraf yükledi: 'deneme4'

10.01.2025 20:24

NEW kullanıcı 5, yeni bir fotoğraf yükledi: 'deneme6'

10.01.2025 20:24

NEW kullanıcı 5, yeni bir fotoğraf yükledi: 'deneme5'

10.01.2025 20:24

NEW kullanıcı 5, yeni bir fotoğraf yükledi: 'deneme4'

10.01.2025 20:24

[View all](#)

Upload a New Photo

Dosya Seç

Dosya seçilmedi

[Upload](#)

ddddd

ddddd

Uploaded: 12.01.2025 15:17

[Delete](#) [Add to Favorites](#)

aaaa

aaaa

Uploaded: 12.01.2025 14:22

[Delete](#) [Add to Favorites](#)

sdada

sdada

Uploaded: 12.01.2025 13:36

[Delete](#) [Add to Favorites](#)

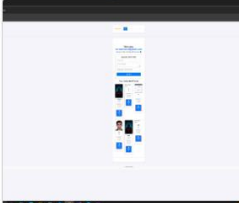


deneme4

deneme4

Uploaded: 10.01.2025 20:24

[Delete](#) [Add to Favorites](#)

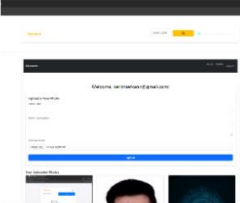


deneme6

deneme6

Uploaded: 10.01.2025 20:24

[Delete](#) [Add to Favorites](#)

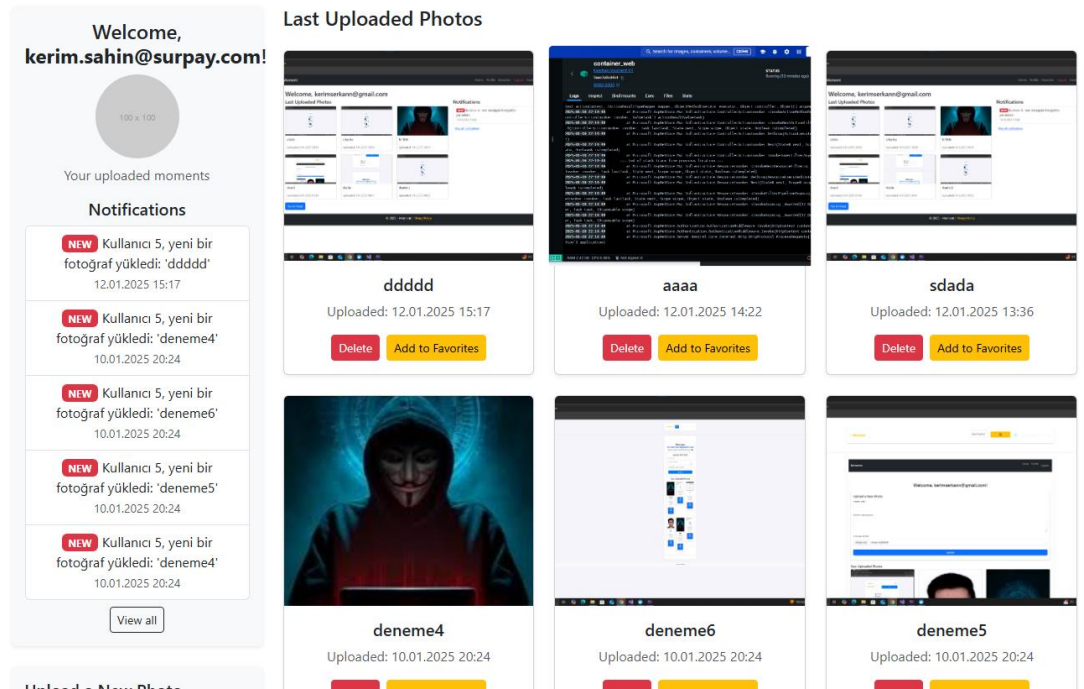


deneme5

deneme5

Uploaded: 10.01.2025 20:24

[Delete](#) [Add to Favorites](#)



- **Veritabanı: SQL Server, Entity Framework Core**
 - **SQL Server:** Microsoft'un geliştirdiği ilişkisel veritabanı yönetim sistemidir.
 - **Entity Framework Core:** Veritabanı işlemlerini yönetmek için kullanılmış olup, Code-First yaklaşımıyla veritabanı tabloları oluşturulmuştur.
 - Kullanıcı, fotoğraf ve yorum gibi veriler ilişkisel veri yapısında saklanmaktadır.
- **API: Web API**
 - **Web API,** istemci ve sunucu arasındaki veri alışverişini JSON formatında sağlayan bir RESTful API yapısıdır.
 - **API,** giriş yapma, fotoğraf yükleme, yorum ekleme ve kullanıcı etkileşimlerini yönetme gibi işlevleri gerçekleştirmektedir.
 - **API,** CORS (Cross-Origin Resource Sharing) desteği ile farklı istemcilerden gelen istekleri güvenli bir şekilde yönetmektedir.
- **Docker: Uygulamanın konteynerleştirilmesi**
 - **Docker,** uygulamanın bağımlılıklarıyla birlikte taşınabilir bir ortamda çalışmasını sağlayan bir konteyner teknolojisidir.
 - **Docker Compose** ile hem web uygulaması hem de veritabanı senkronize bir şekilde çalışmaktadır.
 - Geliştirme ve dağıtım süreçlerinde platform bağımsız bir çözüm sunmaktadır.
- **Kimlik Doğrulama: JWT Authentication**

- JWT (JSON Web Token), kullanıcı oturumlarını yönetmek ve güvenli kimlik doğrulama sağlamak için kullanılmıştır.
- Kullanıcı giriş yaptığında, güvenli bir token üretilir ve kullanıcı her isteğinde bu token ile kimliğini doğrular.
- Token, şifrelenmiş bir yapıya sahiptir ve belirli bir süre sonra otomatik olarak süresi dolar.

```

namespace Moment.Models
{
    public class Comment
    {
        public int Id { get; set; }

        public int PhotoId { get; set; }
        public Photo Photo { get; set; }

        public int UserId { get; set; }
        public User User { get; set; }

        public string Content { get; set; }
        public DateTime CreatedAt { get; set; } = DateTime.Now;
    }
}

```

```

namespace Moment.Data
{
    public class ApplicationDbContext : DbContext
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }

        public DbSet<User> Users { get; set; }
        public DbSet<Photo> Photos { get; set; }
        public DbSet<Favorite> Favorites { get; set; }
        public DbSet<Follower> Followers { get; set; }
        public DbSet<Comment> Comments { get; set; }
        public DbSet<Like> Likes { get; set; }
        public DbSet<Notification> Notifications { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            // Follower -> User ilişkisi
            modelBuilder.Entity<Follower>()
                .HasOne(f => f.Followers) // Follower tablosundaki Follower özelliği
                .WithMany(u => u.Following) // User'ın takip ettiği kişiler
                .HasForeignKey(f => f.FollowerId) // FollowerId User tablosuna referans eder
                .OnDelete(DeleteBehavior.Cascade);

            // Followed -> User ilişkisi
            modelBuilder.Entity<Follower>()
                .HasOne(f => f.Followed) // Follower tablosundaki Followed özelliği
                .WithMany(u => u.Followers) // User'ın takipçileri
                .HasForeignKey(f => f.FollowedId) // FollowedId User tablosuna referans eder
                .OnDelete(DeleteBehavior.Cascade);
        }
    }
}

```

```

namespace Moment.Models
{
    public class Photo
    {
        [Key]
        public int Id { get; set; }

        [Required]
        public int UserId { get; set; }

        [Required]
        public string Title { get; set; }

        public string Description { get; set; }

        [Required]
        public byte[] ImageData { get; set; } // Fotoğraf binary olarak saklanıyor

        public string ContentType { get; set; } // MIME türü

        public DateTime UploadDate { get; set; } = DateTime.Now; // Yükleme tarihi

        public ICollection<Comment> Comments { get; set; }
        public ICollection<Like> Likes { get; set; }

        [ForeignKey("UserId")]
        public virtual User User { get; set; } // Kullanıcı ile ilişki
    }
}

1 using System;
2 using System.ComponentModel.DataAnnotations;
3
4 namespace Moment.Models
5 {
6     public class User
7     {
8         [Key]
9         public int Id { get; set; }
10
11         [Required]
12         public string Name { get; set; }
13
14         [Required, EmailAddress]
15         public string Email { get; set; }
16
17         [Required]
18         public string Password { get; set; } // ● Şifre için doğru alan ismi
19
20         public DateTime CreatedAt { get; set; } = DateTime.UtcNow; // Kullanıcının kayıt tarihi
21
22         // Navigation properties
23         public ICollection<Follower>? Followers { get; set; } // Kullanıcının takipçileri
24         public ICollection<Follower>? Following { get; set; } // Kullanıcının takip ettikleri
25     }
26 }
27

```

4. Veritabanı Yapısı ve İlişkiler

Aşağıdaki ilişkiler **Entity Framework Core** tarafından yönetilmektedir:

- **User - Photo (1-N)** → Bir kullanıcı birden fazla fotoğraf yükleyebilir.
- **User - Comment (1-N)** → Bir kullanıcı birçok yorum yapabilir.
- **User - Like (1-N)** → Kullanıcılar fotoğrafları beğenebilir.
- **User - Follower (N-N)** → Kullanıcılar birbirini takip edebilir.
- **User - Notification (1-N)** → Kullanıcılara birden fazla bildirim gönderilebilir.

Örnek DbContext sınıfı:

```
public class ApplicationDbContext : DbContext
{
    public DbSet<User> Users { get; set; }
    public DbSet<Photo> Photos { get; set; }
    public DbSet<Comment> Comments { get; set; }
}
```

Tablolar ve tablo yapılarından birkaç tane örnek:

Tables_in_memory_site
Comments
Favorites
Followers
Likes
Notifications
Photos
Users
__EFMigrationsHistory


```
mysql> DESC Photos;
```

Field	Type	Null	Key	Default	Extra
Id	int	NO	PRI	NULL	auto_increment
UserId	int	NO	MUL	NULL	
Title	varchar(255)	NO		NULL	
Description	text	YES		NULL	
ImageData	longblob	NO		NULL	
ContentType	varchar(50)	YES		NULL	
UploadDate	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
mysql> DESC Likes;
```

Field	Type	Null	Key	Default	Extra
Id	int	NO	PRI	NULL	auto_increment
PhotoId	int	NO	MUL	NULL	
UserId	int	NO	MUL	NULL	
Content	text	YES		NULL	
CreatedAt	datetime	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
mysql> DESC Users;
```

Field	Type	Null	Key	Default	Extra
Id	int	NO	PRI	NULL	auto_increment
Name	longtext	NO		NULL	
Email	longtext	NO		NULL	
Password	longtext	NO		NULL	
CreatedAt	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

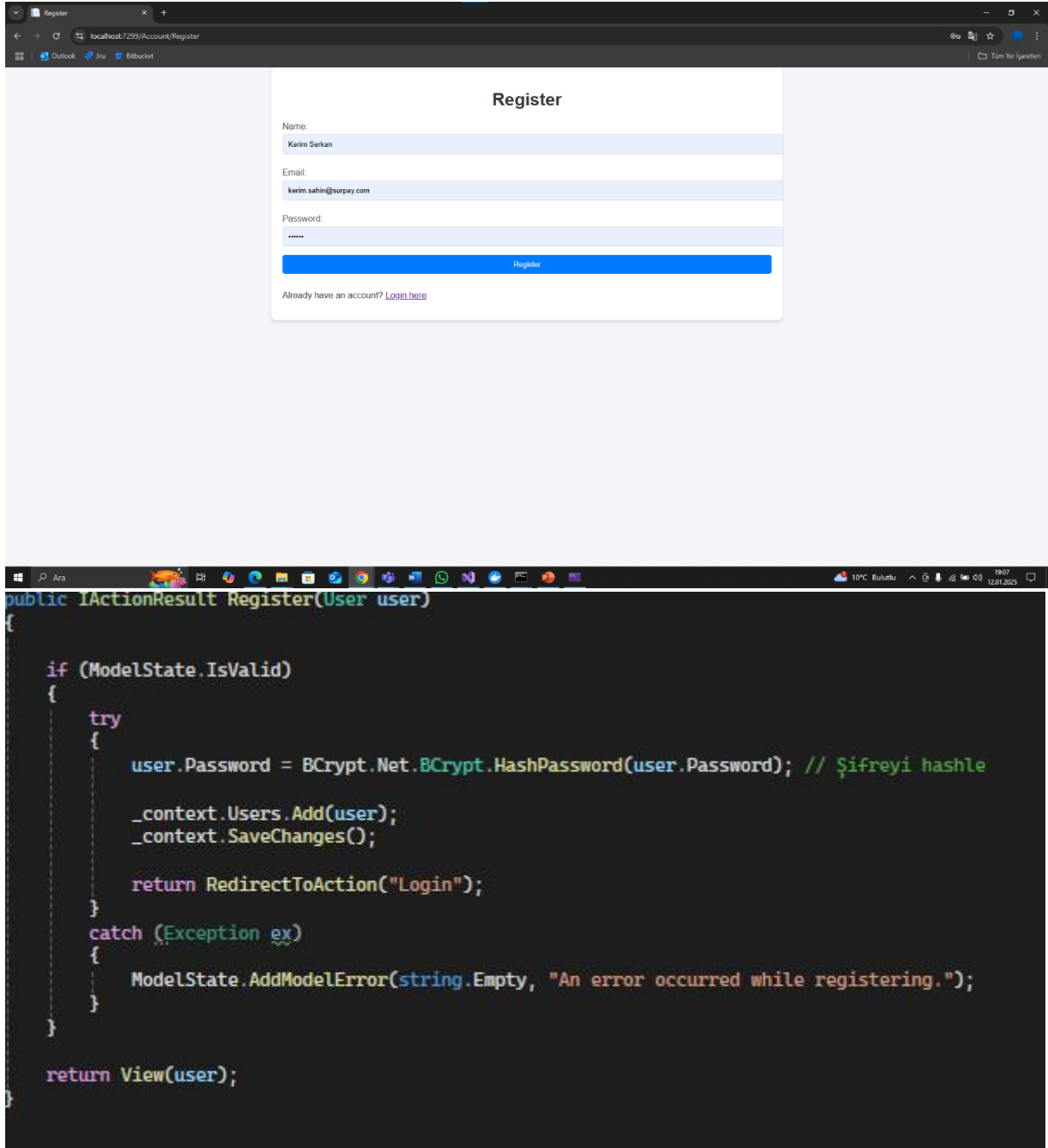
5. Uygulama Çalışma Mantığı

5.1.1. Kullanıcı Kayıt Süreci

- Kullanıcı, Register.cshtml sayfasına yönlendirilir ve aşağıdaki bilgileri içeren kayıt formunu doldurur:
 - Ad Soyad
 - E-posta adresi
 - Şifre (Güçlü bir şifre belirlemesi için kullanıcıya yönergeler verilir.)
 - Şifre Tekrarı (Girilen şifrenin doğruluğunu kontrol etmek için)
- Kullanıcı formu gönderdikten sonra, backend tarafında aşağıdaki adımlar gerçekleştirilir:
 - Girilen e-posta adresinin sistemde daha önce kayıtlı olup olmadığı kontrol edilir.
 - Kullanıcının girdiği şifre, BCrypt hashing algoritması ile güvenli şekilde hashlenerek saklanır.
 - Kullanıcı bilgileri veritabanına kaydedilir ve başarılı kayıt durumu kullanıcıya geri bildirilir.

3. Kullanıcı başarılı şekilde kayıt olduğunda, sisteme giriş yapabilmesi için Login.cshtml sayfasına yönlendirilir.

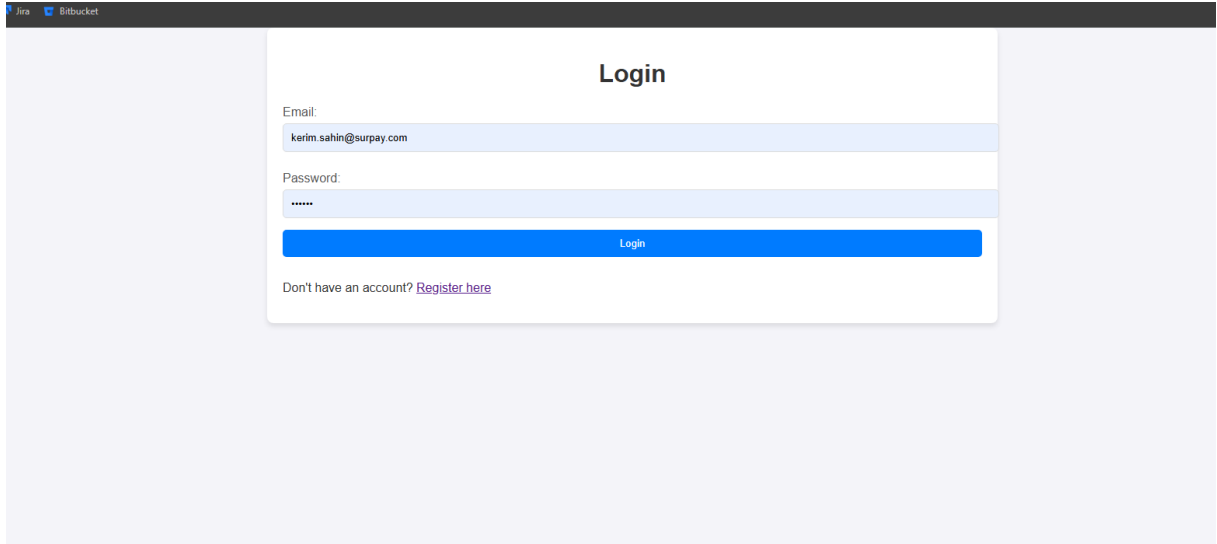
Örnek Kullanıcı Kayıt Metodu:



5.1.2. Kullanıcı Giriş Süreci

1. Kullanıcı, Login.cshtml sayfasına yönlendirilir ve aşağıdaki bilgileri içeren giriş formunu doldurur:
 - E-posta adresi
 - Şifre
2. Kullanıcı giriş yaptıktan sonra, aşağıdaki doğrulama işlemleri gerçekleştirilir:
 - Girilen e-posta adresine sahip bir kullanıcı var mı? kontrol edilir.

- Kullanıcının girdiği şifre, veritabanında hashlenmiş şekilde saklanan şifre ile karşılaştırılır.
 - Şifre doğruysa, kullanıcının oturum açmasına izin verilir ve JWT token üretilir.
3. Başarılı giriş yapan kullanıcıya, JWT token oluşturularak yanıt verilir. Bu token, kullanıcı kimlik doğrulaması için her istekte kullanılır.

A screenshot of a web application's login page. The page has a light blue background. At the top, there's a dark blue header with the text 'Jira' and 'Bitbucket'. The main content area is white and contains a 'Login' form. The form has two input fields: 'Email:' with the value 'kerim.sahin@surpay.com' and 'Password:' with masked characters '*****'. Below the password field is a blue 'Login' button. At the bottom of the form, there's a link that says 'Don't have an account? Register here'.

5.2.1. Fotoğraf Yükleme Süreci

1. Kullanıcı, **Photo/Upload.cshtml** sayfasına yönlendirilir ve aşağıdaki bilgileri içeren formu doldurur:
 1. **Fotoğraf** (JPEG, PNG vb. formatlarda desteklenir)
 2. **Başlık** (Opsiyonel, kullanıcının fotoğrafa ekleyebileceği kısa bir açıklama)
 3. **Açıklama** (Opsiyonel, daha detaylı bir açıklama eklemek için)
2. Kullanıcı, **“Yükle”** butonuna bastığında aşağıdaki işlemler gerçekleşir:
 1. Fotoğraf **istemci tarafında doğrulama sürecinden geçirilir** (dosya türü ve boyut kontrolü yapılır).
 2. Backend tarafında, **fotoğrafın binary (byte[]) formatına dönüştürülmesi** sağlanır.
 3. Fotoğraf veritabanına kaydedilmeden önce **dosya boyutu ve türü tekrar kontrol edilir**.
 4. Veritabanına kaydedilerek **ilgili kullanıcı ile ilişkilendirilir**.

Upload a New Photo

```
[ValidateAntiForgeryToken]
public async Task<ActionResult> Upload(IFormFile photo, string title, string description)
{
    try
    {
        if (photo == null || photo.Length == 0)
        {
            ModelState.AddModelError("", "Please select a photo to upload.");
            return RedirectToAction("Index", "Home");
        }

        var userIdClaim = User.FindFirstValue(ClaimTypes.NameIdentifier);
        if (string.IsNullOrEmpty(userIdClaim) || !int.TryParse(userIdClaim, out int userId))
        {
            return RedirectToAction("Login", "Account");
        }

        using (var memoryStream = new MemoryStream())
        {
            await photo.CopyToAsync(memoryStream);
            var newPhoto = new Photo
            {
                UserId = userId,
                Title = title,
                Description = description,
                ImageData = memoryStream.ToArray(),
                ContentType = photo.ContentType,
                UploadDate = DateTime.Now
            };

            _context.Photos.Add(newPhoto);
            await _context.SaveChangesAsync();

            //Notifications
            var followerIds = _context.Followers
                .Where(f => f.FollowedId == userId) // followedId = bu kullanıcı
                .Select(f => f.FollowerId)
                .ToList();

            // 1) Mesajı hazırlayın
            var message = $"Kullanıcı {userId}, yeni bir fotoğraf yükledi: '{title}'";

            // 2) Her takipçiye notification kaydı ekleyin
            foreach (var fid in followerIds)
            {
                var notification = new Notification
                {
                    UserId = fid,
                    Message = message,
                    CreatedAt = DateTime.Now,
                    IsRead = false
                };
                _context.Notifications.Add(notification);
            }
            await _context.SaveChangesAsync();

            return RedirectToAction("Index", "Home");
        }
    }
    catch (Exception ex)
    {
        ModelState.AddModelError("", $"An error occurred: {ex.Message}");
        return RedirectToAction("Index", "Home");
    }
}
```

5.2.2. Fotoğrafların Veritabanına Kaydedilmesi

- Fotoğraflar **byte[]** formatında saklanır, bu sayede doğrudan veritabanında tutulur.
- Veritabanında yer kaplamaması için alternatif olarak fotoğrafların bulut depolama servislerine yüklenmesi önerilebilir (AWS S3, Firebase, Azure Blob Storage vb.).
- Fotoğraflar, ilgili kullanıcı ID'si ile ilişkilendirilerek saklanır.

```

namespace Moment.Models
{
    public class Photo
    {
        [Key]
        public int Id { get; set; }

        [Required]
        public int UserId { get; set; }

        [Required]
        public string Title { get; set; }

        public string Description { get; set; }

        [Required]
        public byte[] ImageData { get; set; } // Fotoğraf binary olarak saklanıyor

        public string ContentType { get; set; } // MIME türü

        public DateTime UploadDate { get; set; } = DateTime.Now; // Yükleme tarihi

        public ICollection<Comment> Comments { get; set; }
        public ICollection<Like> Likes { get; set; }

        [ForeignKey("UserId")]
        public virtual User User { get; set; } // Kullanıcı ile ilişki
    }
}

```

5.2.3. Fotoğraf Görüntüleme & Listeleme

1. Kullanıcı, **Photo/List.cshtml** sayfasına gittiğinde aşağıdaki işlemler gerçekleşir:
 - Kullanıcının yüklediği fotoğraflar veritabanından çekilir.
 - Fotoğraflar, **binary formatından Base64 formatına dönüştürülerek** frontend tarafında gösterilir.
 - Fotoğraflar, **yüklenme tarihi baz** alınarak sıralanır.

5.3. Kullanıcı Takip İşlemleri

1. Kullanıcı, FollowController üzerinden başka bir kullanıcıyı takip edebilir.
2. Takip edilen kullanıcıların fotoğrafları Home/Feed.cshtml üzerinden görüntülenir.

```

[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Follow(int followedId)
{
    var userIdClaim = User.FindFirstValue(ClaimTypes.NameIdentifier);
    if (!int.TryParse(userIdClaim, out int userId))
    {
        return RedirectToAction("Login");
    }

    // Takip ekleme işlemi
    var newFollow = new Follower
    {
        FollowerId = userId,
        FollowedId = followedId
    };
    _context.Followers.Add(newFollow);
    _context.SaveChanges();

    // Profile sayfasına geri dön
    return RedirectToAction("Profile");
}

[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Unfollow(int followedId)
{
    var followerIdClaim = User.FindFirstValue(ClaimTypes.NameIdentifier);
    if (!int.TryParse(followerIdClaim, out int followerId))
    {
        return RedirectToAction("Login", "Account");
    }

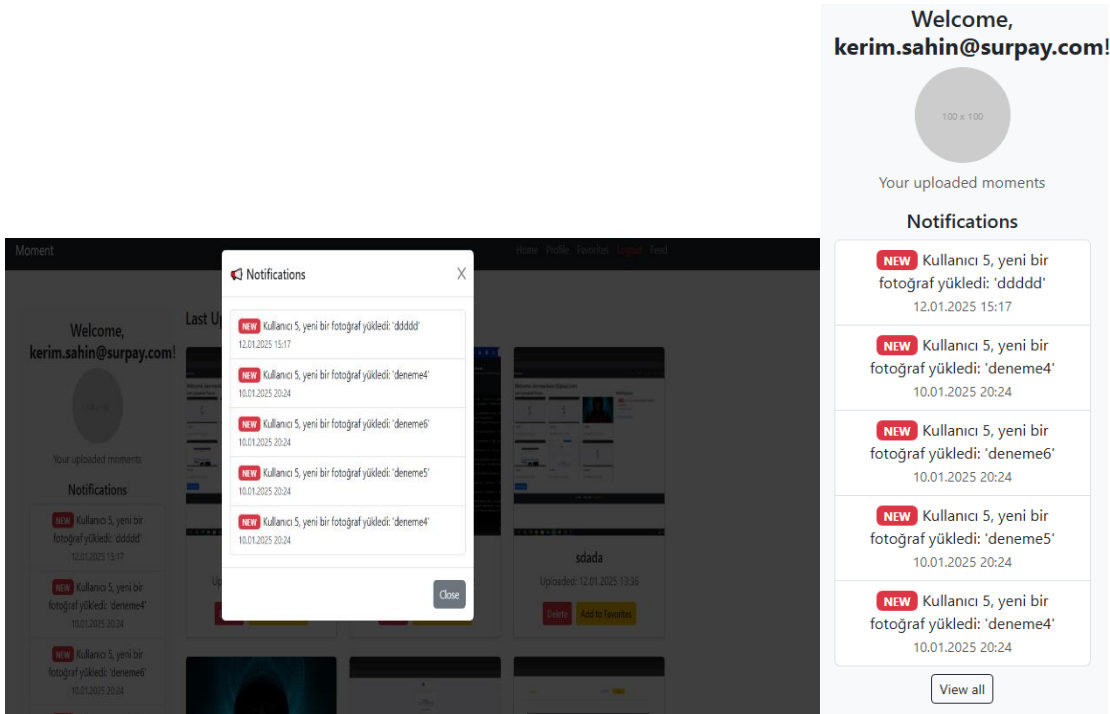
    var existingFollower = _context.Followers.FirstOrDefault(f => f.FollowerId == followerId && f.FollowedId == followedId);
    if (existingFollower != null)
    {
        _context.Followers.Remove(existingFollower);
        _context.SaveChanges();
    }

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

```

5.4. Bildirim Sistemi

1. Kullanıcı etkileşimleri sonucunda NotificationController çalışarak bildirim oluşturur.



2. Bildirimler veritabanına kaydedilir ve Home/Notifications.cshtml sayfasında görüntülenir.

