BLM4537

IOS İle Mobil Uygulama Geliştirme I

Habibe Aleyna Taşdemir

22290583

Proje Tanıtım Videosu:
https://drive.google.com/drive/folders/19KqttHyOJO4hVA9I7C0W6gpO3nN8EIKb?usp=drive_link

GitHub:

github.com/aleynatasdemir/Digi-Mem-Flutter

ÖZET

Bu çalışmada, kullanıcıların dijital anılarını mobil cihazlar üzerinden güvenli ve etkili bir şekilde yönetebilmelerini amaçlayan **DigiMem Mobil Uygulaması** geliştirilmiştir. Günümüzde bireyler, mobil cihazları aracılığıyla yoğun biçimde fotoğraf, video ve ses kaydı üretmekte; ancak bu içeriklerin düzenlenmesi, saklanması ve anlamlandırılması çoğu zaman yetersiz kalmaktadır. DigiMem mobil istemcisi, bu problemi çözmek amacıyla kullanıcıların dijital anılarını tek bir platformda toplayan ve yapay zeka destekli görsel üretimi ile bu anıları anlamlandıran bir çözüm sunmaktadır.

Mobil uygulama, **Flutter cross-platform framework'ü** kullanılarak geliştirilmiş olup Android ve iOS platformlarında tek bir kod tabanı ile çalışacak şekilde tasarlanmıştır. Uygulama, **ASP.NET Core tabanlı RESTful Web API** ile haberleşmekte ve tüm veri yönetimi, kimlik doğrulama ve yapay zeka işlemleri sunucu tarafında gerçekleştirilmektedir. Mobil istemci, JWT tabanlı kimlik doğrulama mekanizması kullanarak backend ile güvenli bir iletişim kurmakta; kullanıcıya ait erişim belirteçleri yerel depolama alanında saklanmaktadır.

Uygulama mimarisinde **MVVM benzeri Service Layer Pattern** benimsenmiş ve durum yönetimi için **Provider (ChangeNotifier)** yaklaşımı kullanılmıştır. Bu sayede kullanıcı arayüzü ile iş mantığı birbirinden ayrılmış, reaktif ve sürdürülebilir bir mobil mimari elde edilmiştir. Mobil istemci, kamera ve galeri erişimi gibi cihaz yeteneklerinden yararlanarak medya dosyalarını seçebilmekte ve bu dosyaları **multipart/form-data** protokolü üzerinden backend'e güvenli şekilde aktarabilmektedir.

DigiMem mobil uygulamasının öne çıkan özelliklerinden biri, **Google Gemini API** kullanılarak gerçekleştirilen yapay zeka destekli görsel üretim sürecidir. Kullanıcılar, mobil uygulama üzerinden belirli bir zaman aralığı seçerek backend'e istek göndermekte; bu anılara ait içeriklerin analizi sonucunda tek ve özgün bir yapay zeka görseli üretilmektedir. Üretilen görsel, mobil uygulama içerisinde kullanıcıya anında sunulmaktadır.

Sonuç olarak DigiMem mobil istemcisi; mobil cihazların sunduğu donanımsal avantajlardan yararlanan, güvenli, kullanıcı odaklı ve yapay zeka destekli bir dijital anı yönetim uygulaması olarak geliştirilmiştir. Çalışma kapsamında mobil uygulama geliştirme, REST API entegrasyonu, durum yönetimi ve yapay zeka servislerinin kullanımı gibi konular uygulamalı olarak ele alınmış ve teorik bilgilerin gerçek bir mobil sistem üzerinde hayata geçirilmesi sağlanmıştır.

İçindekiler Tablosu

1. GİRİŞ

1.1. Problem Tanımı

Mobil cihazların yaygınlaşmasıyla birlikte bireyler günlük yaşamlarında yoğun biçimde fotoğraf, video, ses kaydı ve metin üretmektedir. Bu içerikler çoğunlukla kişisel deneyimleri temsil eden dijital anılar niteliği taşımaktadır. Ancak mobil platformlarda kullanılan mevcut uygulamaların büyük bir bölümü, bu anıları yalnızca depolama odaklı ele almakta; kullanıcıların geçmiş deneyimlerini düzenli bir biçimde incelemesine ve anlamlandırmasına yeterli imkân sunmamaktadır.

Dijital anıların zaman içerisinde artması, özellikle belirli bir döneme ait içeriklerin manuel olarak derlenmesini ve özetlenmesini zorlaştırmaktadır. Yüzlerce medya içeriği arasından anlamlı bir bütün oluşturmak, mobil cihazlar üzerinde zaman ve dikkat gerektiren bir süreç haline gelmektedir. Bu durum, kullanıcıların dijital anılarını etkin bir şekilde değerlendirememesine yol açmaktadır.

1.2. Çözüm Yaklaşımı

Bu çalışmada geliştirilen **DigiMem mobil uygulaması**, kullanıcıların dijital anılarını mobil cihazlar üzerinden güvenli, düzenli ve anlamlı bir biçimde yönetebilmelerini amaçlayan bir çözüm sunmaktadır. Uygulama; fotoğraf, video, ses ve metin gibi farklı medya türlerini destekleyerek kullanıcıların anılarını tek bir yapı altında toplamaktadır.

DigiMem mobil istemcisinin en önemli özelliklerinden biri, yapay zeka destekli görsel üretim mekanizmasıdır. Kullanıcılar, mobil uygulama üzerinden belirli bir zaman aralığı seçerek bu döneme ait anılarını otomatik olarak analiz ettirebilmekte ve bu anıları temsil eden tek bir görsel çıktıyı elde edebilmektedir. Bu yaklaşım, kullanıcıların anılarını manuel olarak derleme gereksinimini ortadan kaldırmakta ve anıların anlamlandırılmasını kolaylaştırmaktadır.

1.3. Çalışmanın Amacı

Bu çalışmanın temel amacı, mobil cihazların sunduğu donanımsal ve yazılımsal olanaklardan yararlanarak kullanıcı odaklı, güvenli ve yapay zeka destekli bir dijital anı yönetim uygulaması geliştirmektir. Geliştirilen mobil uygulama ile kullanıcıların:

- Dijital anılarını mobil ortamda kolayca oluşturabilmesi,
- Farklı medya türlerini tek bir uygulama üzerinden yönetebilmesi,
- Yapay zeka destekli görsel üretimi ile anılarını anlamlandırabilmesi

hedeflenmiştir.

1.4. Kullanılan Teknolojiler ve Yaklaşım

DigiMem mobil uygulaması, **Flutter cross-platform framework'ü** kullanılarak geliştirilmiştir. Flutter'ın sunduğu tek kod tabanı ile çoklu platform desteği, uygulamanın hem Android hem de iOS ortamlarında çalışmasını mümkün kılmıştır. Uygulama mimarisinde, kullanıcı arayüzü ile iş mantığının ayrıştırılmasına önem verilmiş ve **MVVM benzeri Service Layer Pattern** uygulanmıştır.

Durum yönetimi için **Provider (ChangeNotifier)** yaklaşımı tercih edilmiş; bu sayede reaktif bir kullanıcı deneyimi elde edilmiştir. Mobil uygulama, backend servisleri ile RESTful API üzerinden iletişim kurmakta ve kimlik doğrulama süreçlerinde JWT tabanlı güvenlik mekanizmalarını kullanmaktadır.

1.5. Çalışmanın Kapsamı

Bu rapor, DigiMem mobil uygulamasının tasarımı ve geliştirilmesi sürecini kapsamaktadır. Çalışma kapsamında mobil uygulamanın mimari yapısı, durum yönetimi yaklaşımı, backend servisleri ile etkileşimi, medya dosyası yükleme süreçleri ve yapay zeka destekli görsel üretim mekanizması ele alınmaktadır. Web uygulaması ve backend servislerinin detaylı geliştirme süreçleri bu raporun kapsamı dışında bırakılmıştır.

2. MOBİL SİSTEM MİMARİSİ VE GENEL TASARIM

2.1. Genel Mimari Yaklaşım

DigiMem mobil uygulaması, güvenli, sürdürülebilir ve genişletilebilir bir yapı sunmak amacıyla **istemci–sunucu (client–server)** mimarisi temel alınarak tasarlanmıştır. Mobil istemci, yalnızca kullanıcı etkileşimi ve veri sunumundan sorumlu tutulmuş; veri yönetimi, kimlik doğrulama ve yapay zeka işlemleri sunucu tarafında merkezi olarak gerçekleştirilmiştir. Bu yaklaşım, mobil uygulamanın hafif ve performanslı bir yapıda kalmasını sağlamıştır.

Mobil uygulama mimarisinde temel hedef, kullanıcı arayüzü ile iş mantığının net biçimde ayrılması ve mobil cihazlara özgü kullanım senaryolarının etkin biçimde desteklenmesidir. Bu doğrultuda, uygulama katmanlı bir yapı ile ele alınmış ve her katmana belirli sorumluluklar atanmıştır.

2.2. İstemci–Sunucu İletişim Modeli

Mobil istemci, sunucu ile iletişimini **RESTful API** üzerinden gerçekleştirmektedir. Tüm veri alışverişi HTTP protokolü kullanılarak yapılmakta ve veri formatı olarak JSON tercih edilmektedir. Mobil uygulama, her istekte gerekli kimlik doğrulama bilgilerini göndererek **stateless** bir iletişim modeli sunmaktadır.

Bu iletişim modelinde mobil istemci:

- Kullanıcıdan aldığı girdileri sunucuya iletir,
- Sunucudan dönen yanıtları işler,
- Yanıtları kullanıcı arayüzünde görselleştirir.

Bu yapı, farklı platformlarda çalışan mobil uygulamalar için ortak ve yeniden kullanılabilir bir servis altyapısı sunmaktadır.

2.3. Katmanlı Mimari Yapı

DigiMem mobil uygulamasında **katmanlı mimari** benimsenmiştir. Bu mimari yaklaşım, kodun okunabilirliğini ve bakımını kolaylaştırmakta, aynı zamanda genişletilebilir bir yapı sunmaktadır. Uygulama temel olarak aşağıdaki katmanlardan oluşmaktadır:

- **Sunum Katmanı (UI Layer):**
  Flutter widget'ları kullanılarak oluşturulan ekranlar bu katmanı oluşturmaktadır. Kullanıcı etkileşimi, navigasyon ve görsel bileşenler bu katmanda ele alınmaktadır.
- **İş Mantığı Katmanı (Service Layer):**
  services/ dizini altında yer alan sınıflar, API çağrılarını ve iş kurallarını içermektedir. Bu katman, UI katmanından bağımsız olarak çalışmakta ve uygulamanın temel davranışlarını tanımlamaktadır.
- **Veri Modeli Katmanı (Model Layer):**
  Backend servislerinden dönen JSON verilerinin Dart karşılıkları bu katmanda tanımlanmıştır. Model sınıfları, veri tutarlılığını sağlamak ve tip güvenliğini artırmak amacıyla kullanılmıştır.

Bu yapı, **separation of concerns** ilkesine uygun olarak tasarlanmış ve her katmanın sorumluluğu net bir şekilde belirlenmiştir.

2.4. MVVM Benzeri Mimari Yaklaşım

Mobil uygulama mimarisinde, **MVVM (Model–View–ViewModel)** yaklaşımından esinlenen bir yapı kullanılmıştır. Flutter ekosisteminde doğrudan ViewModel kavramı bulunmamakla birlikte, bu rol **service sınıfları** ve **Provider** aracılığıyla üstlenilmiştir.

Bu yaklaşımda:

- **View:** Flutter widget'ları (ekranlar)
- **ViewModel:** Provider ile yönetilen servis sınıfları
- **Model:** Veri sınıfları

olarak konumlandırılmıştır. Bu sayede kullanıcı arayüzü ile iş mantığı arasında gevşek bir bağ kurulmuş, test edilebilir ve sürdürülebilir bir yapı elde edilmiştir.

2.5. Durum Yönetimi ve Reaktif Tasarım

Uygulamada durum yönetimi için **Provider (ChangeNotifier)** yaklaşımı kullanılmıştır. Provider, uygulama durumundaki değişikliklerin kullanıcı arayüzüne otomatik olarak yansıtılmasını sağlayan reaktif bir yapı sunmaktadır.

Kimlik doğrulama durumu, anı listeleri ve kullanıcı bilgileri gibi kritik veriler Provider üzerinden yönetilmekte; notifyListeners() çağrısı ile UI güncellenmektedir. Bu yaklaşım, mobil uygulamada kullanıcı deneyimini kesintisiz ve tutarlı hale getirmiştir.

2.6. Platform Bağımlılıkları ve Mobil Odaklı Tasarım

DigiMem mobil uygulaması, Android ve iOS platformlarını destekleyecek şekilde tasarlanmıştır. Platforma özgü farklılıklar (örneğin Android emülatör için API adresi) uygulama içerisinde kontrol edilerek yönetilmiştir. Ayrıca mobil cihazların kamera, galeri ve dosya sistemi gibi donanımsal özelliklerinden yararlanacak şekilde bir tasarım benimsenmiştir.

Bu yaklaşım, uygulamanın mobil kullanım senaryolarına uygun, esnek ve genişletilebilir bir mimari kazanmasını sağlamıştır.

3. BACKEND (SUNUCU TARAFI) TASARIMI VE UYGULAMASI

3.1. ASP.NET Core Web API Mimarisi

DigiMem uygulamasının sunucu tarafı, **ASP.NET Core Web API** mimarisi kullanılarak geliştirilmiştir. Uygulama, RESTful prensiplere uygun olarak tasarlanmış olup istemci ile sunucu arasındaki tüm iletişim HTTP protokolü üzerinden JSON tabanlı veri formatı ile gerçekleştirilmektedir. Backend mimarisinde iş mantığı, veri erişimi ve güvenlik sorumlulukları ayrıştırılarak sürdürülebilir ve genişletilebilir bir yapı elde edilmiştir.

Uygulama başlatıldığında veritabanı şemasının güncel kalması amacıyla migration işlemleri otomatik olarak çalıştırılmaktadır:

```
using (var scope = app.Services.CreateScope())
{
    var db = scope.ServiceProvider.GetRequiredService<AppDbContext>();
    db.Database.Migrate();
}
```

Bu yaklaşım, özellikle geliştirme ve dağıtım süreçlerinde manuel veritabanı müdahalelerini ortadan kaldırmaktadır.

3.2. Controller Katmanı ve Endpoint Tasarımı

Controller katmanı, istemci tarafından gönderilen HTTP isteklerini karşılayan ve bu istekleri ilgili servis katmanına yönlendiren bileşenlerden oluşmaktadır. DigiMem uygulamasında controller'lar, tek sorumluluk ilkesine uygun olarak tasarlanmıştır.

3.2.1. Kimlik Doğrulama ve Yetkilendirme (AuthController)

AuthController, kullanıcı kayıt ve giriş işlemlerini yönetmektedir. Kimlik doğrulama sürecinde **ASP.NET Core Identity** altyapısı kullanılmış ve JWT tabanlı bir yetkilendirme mekanizması uygulanmıştır.

Kullanıcı giriş süreci aşağıdaki şekilde gerçekleştirilmektedir:

```
[HttpPost("login")]
public async Task<IActionResult> Login(LoginRequest request)
{
    var user = await _userManager.FindByEmailAsync(request.Email);
    if (user == null || user.IsBanned)
        return Unauthorized();

    var result = await _signInManager.CheckPasswordSignInAsync(
        user, request.Password, false);

    if (!result.Succeeded)
        return Unauthorized();
```

```
    var token = GenerateJwtToken(user);
    return Ok(new { token });
}
```

JWT token içerisine kullanıcı kimliği ve rol bilgileri eklenerek her istekte **stateless** bir kimlik doğrulama sağlanmıştır.

3.2.2. Anı Yönetimi (MemoriesController)

MemoriesController, kullanıcıların dijital anılarını yönetmesini sağlayan temel controller'dır. Fotoğraf, video, ses, metin ve müzik türündeki anılar bu controller üzerinden oluşturulmakta ve yönetilmektedir.

Anı listeleme işlemlerinde kullanıcıya ait kayıtlar filtrelenmekte ve performans amacıyla sıralama uygulanmaktadır:

```
var query = _context.Memories
    .Where(m => m.UserId == userId)
    .Where(m => m.CreatedAt >= fromDate && m.CreatedAt <= toDate)
    .OrderByDescending(m => m.CreatedAt);
```

Dosya tabanlı anılar için yükleme işlemi multipart/form-data formatında alınmakta ve dosyalar fiziksel dosya sisteminde kullanıcı bazlı dizinler altında saklanmaktadır:

```
var fileName = "{Guid.NewGuid()}{Path.GetExtension(file.FileName)}";
var filePath = Path.Combine("wwwroot/uploads", userId, fileName);

using var stream = new FileStream(filePath, FileMode.Create);
await file.CopyToAsync(stream);
```

Bu süreçte kullanıcı kimliği JWT token üzerinden elde edilerek **sahiplik kontrolü (ownership)** sağlanmaktadır.

3.2.3. Yapay Zeka Destekli Görsel Üretimi (SummariesController)

SummariesController, kullanıcının belirli bir zaman aralığında oluşturduğu anılardan yola çıkarak **tek bir yapay zeka üretimi görsel** oluşturulmasını sağlamaktadır. Bu noktada sistem, birden fazla görseli birleştiren kolaj yaklaşımı yerine, yapay zekanın içerik analizi yaparak **yeni ve özgün bir görsel üretmesi** prensibiyle çalışmaktadır.

Controller, istemciden gelen isteği alarak ilgili servis katmanına yönlendirmektedir:

```
[HttpPost("ai-image")]
[Authorize]
public async Task<IActionResult> GenerateAiImage(AiImageRequest request)
{
    var userId = GetUserId();
    var result = await _geminiService
```

```
    .GenerateAiImageAsync(userId, request.StartDate, request.EndDate);


  return Ok(result);
}
```

Bu yaklaşım, kullanıcının anılarını birebir görsel olarak birleştirmek yerine, anıların bağlamsal analizine dayalı yaratıcı bir çıktı üretmeyi hedeflemektedir.

3.3. Service Katmanı ve İş Mantığı

Service katmanı, uygulamanın temel iş kurallarını içermektedir. Controller'lar yalnızca yönlendirme yaparken, karmaşık işlemler bu katmanda soyutlanmıştır.

3.3.1. Yapay Zeka Entegrasyonu (GeminiService)

GeminiService, Google Gemini API ile entegrasyonu sağlayan servis bileşenidir. Servis, belirli bir kullanıcı ve tarih aralığı için veritabanında bulunan fotoğraf anılarını analiz etmekte ve bu içeriklerden **tek bir yapay zeka üretimi görsel** oluşturulmasını sağlamaktadır.

Görseller Gemini API'ye gönderilmeden önce base64 formatına dönüştürülmektedir:

```
var bytes = File.ReadAllBytes(photo.FileUrl);

var base64 = Convert.ToBase64String(bytes);
```

Gemini API'ye multimodal bir istek gönderilmekte ve yapay zekadan dönen sonuç, sistem tarafından dosya olarak saklanmaktadır. Üretilen bu görsel, kullanıcının seçtiği dönemi temsil eden soyut ve anlamlı bir çıktı niteliği taşımaktadır.

3.3.2. Güvenlik Amaçlı Şifreleme Mekanizmaları (EncryptionService)

Sistemde saklanan hassas bilgiler (örneğin Spotify refresh token'ları), **AES-256** algoritması kullanılarak şifrelenmektedir:

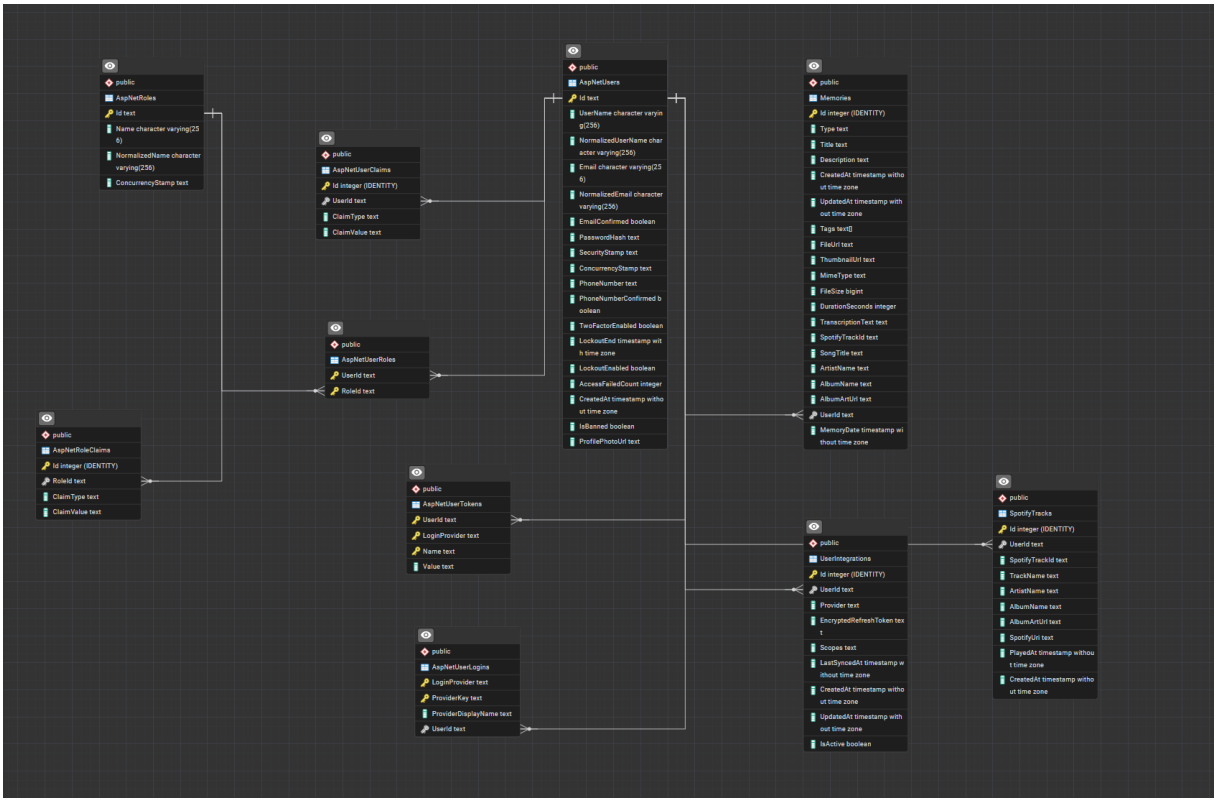
```
using var aes = Aes.Create();

aes.KeySize = 256;

aes.GenerateIV();
```

Bu yaklaşım, veritabanında düz metin hassas veri tutulmasını engelleyerek güvenlik seviyesini artırmaktadır.

3.4. Kimlik Doğrulama ve Güvenlik Mekanizmaları

Backend mimarisinde JWT doğrulama, rol ve sahiplik kontrolleri, dosya türü ve boyut doğrulamaları birlikte uygulanmıştır. Böylece sistem, kullanıcı verilerinin gizliliğini koruyan ve yetkisiz erişimleri engelleyen bütüncül bir güvenlik yapısına sahip olmuştur.

4. VERİTABANI TASARIMI VE VERİ YÖNETİMİ



Şekil 4. Veri Tabanı Tabloları

4.1. Veritabanı Modelleme Yaklaşımı

DigiMem uygulamasında veritabanı tasarımı, **ilişkisel veritabanı modeli** temel alınarak gerçekleştirilmiştir. Tasarım sürecinde veri bütünlüğü, ölçeklenebilirlik ve performans kriterleri ön planda tutulmuştur. Kullanıcıya ait dijital anılar, medya içerikleri ve üçüncü parti entegrasyonlar arasında anlamlı ilişkiler kurulmuş; gereksiz veri tekrarının önüne geçmek amacıyla normalizasyon prensipleri uygulanmıştır.

Veritabanı yapısı, kullanıcı merkezli bir yaklaşımla kurgulanmış ve tüm temel varlıklar (anı, medya, entegrasyon verileri) kullanıcı kimliği üzerinden ilişkilendirilmiştir. Bu sayede hem güvenli veri erişimi sağlanmış hem de kullanıcı bazlı sorguların performansı artırılmıştır.

4.2. Entity Framework Core ve ORM Kullanımı

Veri erişim katmanında **Entity Framework Core** ORM aracı kullanılmıştır. ORM yaklaşımı sayesinde veritabanı işlemleri nesne yönelimli programlama prensipleriyle gerçekleştirilmiş; doğrudan SQL yazımına olan bağımlılık azaltılmıştır. Bu yaklaşım, kod okunabilirliğini artırmakla birlikte bakım ve geliştirme süreçlerini de kolaylaştırmıştır.

Uygulamada kullanılan veritabanı context'i, ASP.NET Core Identity altyapısını da kapsayacak şekilde yapılandırılmıştır:

```
public class AppDbContext : IdentityDbContext<ApplicationUser>
{
```

```
    public DbSet<Memory> Memories { get; set; }
    public DbSet<UserIntegration> UserIntegrations { get; set; }
    public DbSet<SpotifyTrack> SpotifyTracks { get; set; }

    public AppDbContext(DbContextOptions<AppDbContext> options)
        : base(options) { }
}
```

Bu yapı sayesinde kimlik yönetimi ile uygulamaya özel veriler tek bir context üzerinden yönetilebilmiştir.

4.3. Varlıklar (Entities) ve İlişkiler

4.3.1. Kullanıcı ve Kimlik Yönetimi

Kullanıcı verileri, ASP.NET Core Identity tarafından sağlanan AspNetUsers tablosu üzerinden yönetilmiştir. ApplicationUser sınıfı, bu tabloyu genişleterek uygulamaya özgü alanlar eklemiştir:

```
public class ApplicationUser : IdentityUser
{
    public DateTime CreatedAt { get; set; }
    public bool IsBanned { get; set; }
    public ICollection<Memory> Memories { get; set; }
}
```

Bu yapı, kimlik doğrulama ve yetkilendirme süreçlerini standart bir çerçevede yürütürken, uygulamaya özgü kullanıcı bilgilerini de saklamaya olanak tanımaktadır.

4.3.2. Anı (Memory) Veri Modeli

Dijital anılar, sistemin temel veri varlığını oluşturmaktadır. Her anı, belirli bir kullanıcıya ait olacak şekilde tasarlanmıştır ve farklı medya türlerini desteklemektedir:

```
public class Memory
{
    public int Id { get; set; }
    public string Type { get; set; } // photo, video, audio, text, music
    public string Title { get; set; }
    public string Description { get; set; }
    public DateTime MemoryDate { get; set; }
    public DateTime CreatedAt { get; set; }
```

```
    public string FileUrl { get; set; }
    public string MimeType { get; set; }
    public long? FileSize { get; set; }

    public string UserId { get; set; }
    public ApplicationUser User { get; set; }
}
```

Bu tasarım sayesinde hem medya tabanlı hem de metin tabanlı anılar tek bir yapı altında toplanmış, sistemin esnekliği artırılmıştır.

4.3.3. Entegrasyon ve Müzik Verileri

Üçüncü parti servislerle entegrasyon amacıyla ayrı tablolar oluşturulmuştur. Spotify entegrasyonu buna örnek teşkil etmektedir. Kullanıcıya ait Spotify bağlantı bilgileri ayrı bir varlıkta tutulmuş ve güvenlik gerekçesiyle token bilgileri şifrelenmiştir:

```
public class UserIntegration
{
    public int Id { get; set; }
    public string Provider { get; set; }
    public string EncryptedRefreshToken { get; set; }
    public string UserId { get; set; }
}
```

Ayrıca kullanıcının dinlediği şarkılar, analiz ve özetleme süreçlerinde kullanılmak üzere ayrı bir tabloda saklanmıştır.

4.4. İlişkiler, Kısıtlar ve İndeksleme

Veritabanı tasarımında **one-to-many** ilişkiler yoğun olarak kullanılmıştır. Bir kullanıcının birden fazla anısı ve entegrasyon kaydı olabilmesi sağlanmıştır. Kullanıcı silindiğinde ilişkili verilerin de silinmesi için **cascade delete** davranışı uygulanmıştır.

Performansı artırmak amacıyla sık kullanılan sorgular için indeksler tanımlanmıştır. Özellikle kullanıcıya ait anıların tarih bazlı sorgulanması için aşağıdaki indeks kullanılmıştır:

```
modelBuilder.Entity<Memory>()
    .HasIndex(m => new { m.UserId, m.CreatedAt });
```

Bu yaklaşım, büyük veri kümelerinde sorgu sürelerini önemli ölçüde azaltmaktadır.

4.5. Migration ve Sürüm Yönetimi

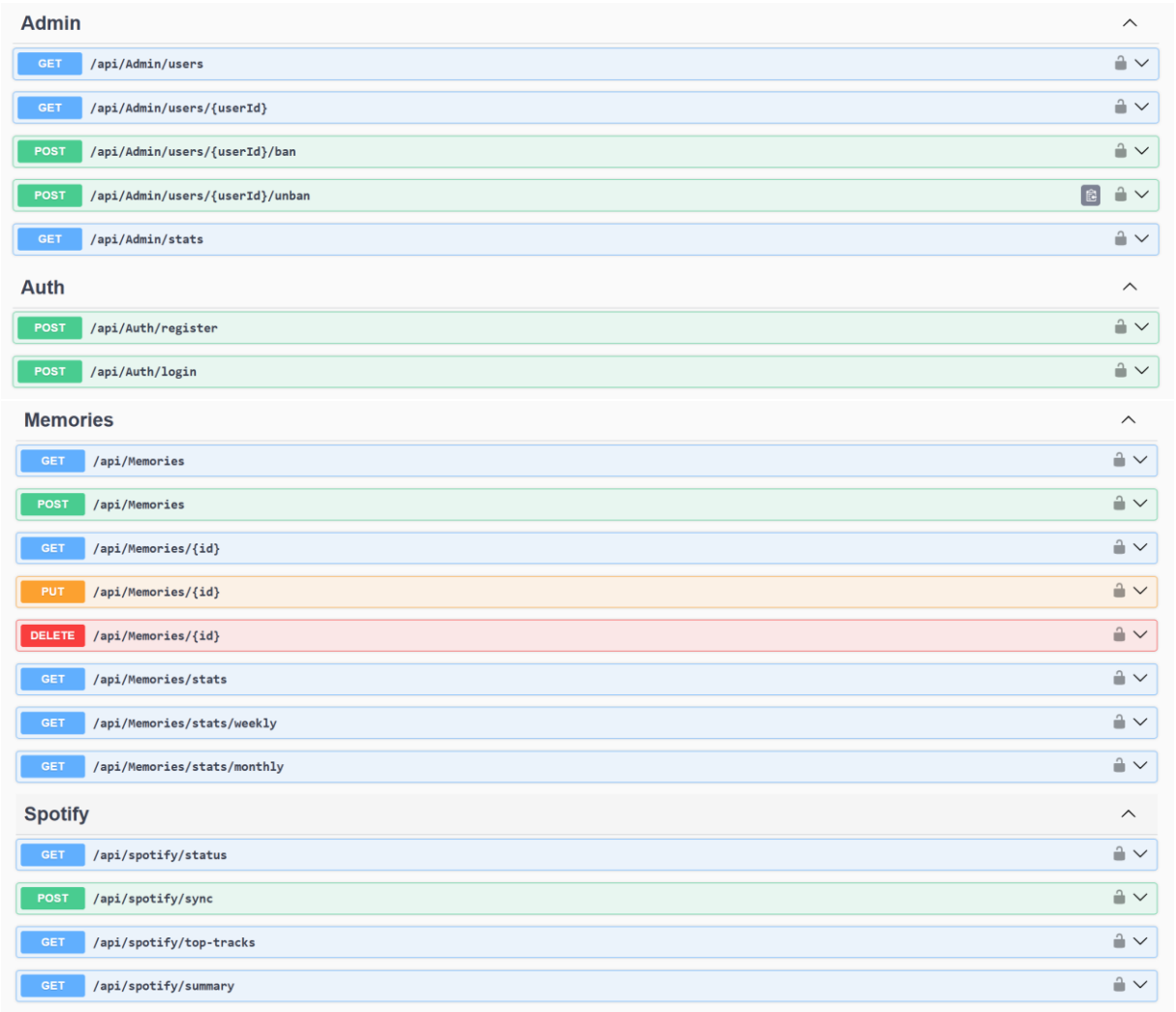
Veritabanı şemasındaki değişiklikler **migration** mekanizması ile yönetilmiştir. Her yapısal değişiklik için ayrı bir migration oluşturulmuş ve sürüm kontrolü sağlanmıştır:

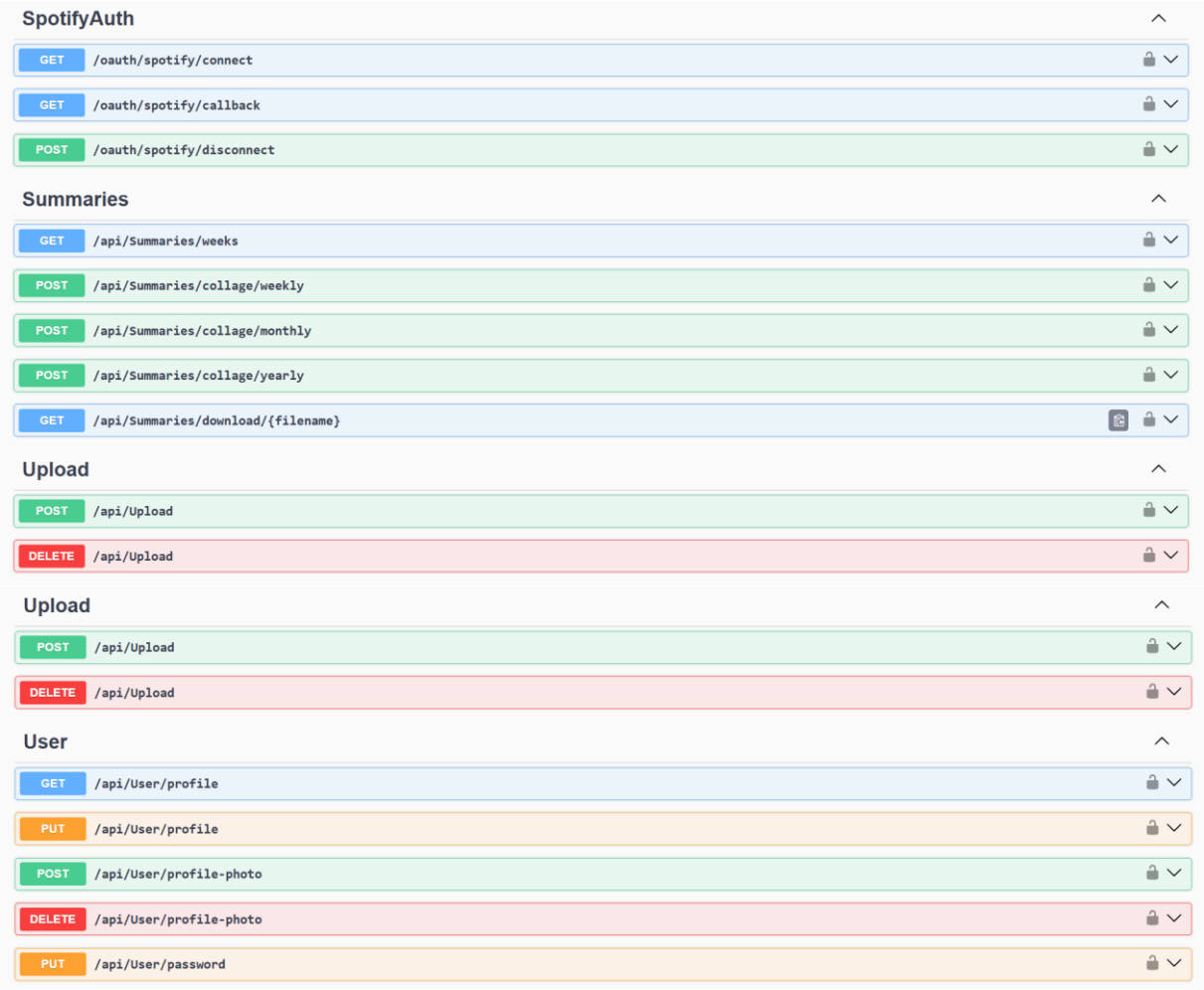
dotnet ef migrations add AddMemories

dotnet ef database update

Uygulama başlatılırken migration'ların otomatik olarak uygulanması, farklı ortamlar arasında şema uyumsuzluklarını önlemiştir. Bu yaklaşım, özellikle ekip çalışması ve üretim ortamına geçiş süreçlerinde önemli bir avantaj sağlamıştır.

5. API VE İSTEMCİ ARASI İLETİŞİM





Şekil 5. Swagger ekran görüntüsü

5.1. API İletişim Akışının Genel Yapısı

DigiMem uygulamasında istemci ve sunucu arasındaki iletişim, **RESTful Web API** mimarisi temel alınarak tasarlanmıştır. İstemci tarafı (Next.js tabanlı web uygulaması), sunucu tarafında geliştirilen ASP.NET Core Web API'ye HTTP protokolü üzerinden istekler göndermekte; sunucu ise bu isteklere JSON formatında yanıtlar üretmektedir.

İstemci–sunucu iletişiminde **stateless** bir yapı benimsenmiş olup, her istek gerekli kimlik doğrulama bilgilerini kendi içerisinde taşımaktadır. Bu yaklaşım, sunucu tarafında oturum bilgisinin tutulmasını gereksiz kılarak ölçeklenebilirliği artırmaktadır.

5.2. JWT Tabanlı Kimlik Doğrulama ve Yetkilendirme Akışı

Kimlik doğrulama süreci, **JSON Web Token (JWT)** kullanılarak gerçekleştirilmiştir. Kullanıcı giriş yaptıktan sonra backend tarafından üretilen JWT token, istemci tarafında saklanmakta ve yetki gerektiren tüm API isteklerinde HTTP header üzerinden sunucuya iletilmektedir.

İstemci tarafında token, aşağıdaki şekilde her isteğe otomatik olarak eklenmektedir:

const token = localStorage.getItem('token')

```
const headers = {
  'Content-Type': 'application/json',
}

if (token) {
  headers['Authorization'] = `Bearer ${token}`
}
```

İstemci tarafında token saklama yöntemi platforma göre değişmekte olup, web ortamında localStorage, mobil ortamda ise yerel depolama çözümleri kullanılmaktadır.

Sunucu tarafında ise JWT doğrulama middleware aracılığıyla yapılmakta; token'ın imzası, geçerlilik süresi ve içerdiği claim'ler kontrol edilmektedir. Yetkilendirme gerektiren endpoint'ler [Authorize] attribute'u ile korunmaktadır:

```
[Authorize]
[HttpGet("memories")]
public async Task<IActionResult> GetMemories()
{
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    ...
}
```

Bu yapı sayesinde kullanıcılar yalnızca kendilerine ait verilere erişebilmekte ve yetkisiz erişimler engellenmektedir.

5.3. Dosya Yükleme Süreci (Multipart/Form-Data)

Fotoğraf, video ve ses gibi medya içeriklerinin yüklenmesi sırasında **multipart/form-data** veri formatı kullanılmıştır. Bu yöntem, ikili (binary) dosya verileri ile birlikte metadata bilgilerin tek bir HTTP isteği içerisinde iletilmesine olanak tanımaktadır.

İstemci tarafında dosya yükleme işlemi şu şekilde gerçekleştirilmektedir:

```
const formData = new FormData()
formData.append('file', file)
formData.append('type', 'photo')
formData.append('date', '2025-01-12')
formData.append('title', 'İstanbul Gezisi')

await fetch(`${API_URL}/api/memories`, {
  method: 'POST',
```

```
  headers: {
    'Authorization': `Bearer ${token}`,
  },
  body: formData,
})
```

Sunucu tarafında ise dosya, IFormFile aracılığıyla alınmakta ve gerekli doğrulamalar yapıldıktan sonra fiziksel dosya sistemine kaydedilmektedir:

```
[FromForm] IFormFile file
```

Bu süreçte dosya türü, boyutu ve kullanıcı sahipliği kontrol edilerek güvenli bir yükleme mekanizması oluşturulmuştur.

5.4. API Yanıtları ve Veri Dönüş Formatı

API, istemciye döndüğü tüm yanıtları **JSON** formatında üretmektedir. Başarılı işlemlerde, ilgili kaynağa ait veriler veya işlem sonucu döndürülürken; hatalı durumlarda standart HTTP durum kodları kullanılmaktadır.

Örnek bir başarılı yanıt:

```
{
  "id": 42,
  "fileUrl": "/uploads/user123/photo_abc.jpg",
  "createdAt": "2025-01-12T10:30:00Z"
}
```

Bu yapı, istemci tarafında tip güvenli veri modelleri ile kolayca işlenebilmektedir.

5.5. Hata Yönetimi ve HTTP Durum Kodları

Hata yönetimi sürecinde, hem istemci hem de sunucu tarafında tutarlı bir yaklaşım benimsenmiştir. Sunucu tarafında doğrulama hataları, yetkisiz erişimler ve sunucu hataları uygun HTTP durum kodları ile istemciye iletilmektedir.

Örnek hata durumları:

- **400 Bad Request:** Geçersiz veri gönderimi
- **401 Unauthorized:** Kimlik doğrulama başarısız
- **403 Forbidden:** Yetkisiz erişim
- **404 Not Found:** Kaynak bulunamadı
- **500 Internal Server Error:** Sunucu hatası

İstemci tarafında ise bu hatalar yakalanmakta ve kullanıcıya anlamlı geri bildirimler sunulmaktadır:

```
if (!response.ok) {
```

```
  throw new ApiError(response.status, await response.text())
}
```

Bu yaklaşım, kullanıcı deneyimini iyileştirirken hata ayıklama süreçlerini de kolaylaştırmaktadır.

5.6. İstemci–Sunucu Etkileşiminin Değerlendirilmesi

Uygulanan API iletişim yapısı, güvenli, performanslı ve sürdürülebilir bir istemci–sunucu etkileşimi sunmaktadır. JWT tabanlı kimlik doğrulama sayesinde stateless bir yapı elde edilmiş; multipart dosya yükleme mekanizması ile farklı medya türlerinin sisteme entegre edilmesi sağlanmıştır. Bu tasarım, DigiMem uygulamasının hem mevcut gereksinimleri karşılamasına hem de gelecekteki genişletmelere açık olmasına katkı sağlamaktadır.

6. KULLANICI ARAYÜZÜ (UI) TASARIMI

6.1. Tasarım Yaklaşımı ve Kullanıcı Deneyimi

DigiMem mobil uygulamasının kullanıcı arayüzü tasarlanırken, **kullanılabilirlik**, **erişilebilirlik** ve **mobil odaklı etkileşim** ilkeleri temel alınmıştır. Arayüz tasarımında kullanıcıların anı oluşturma, görüntüleme ve yapay zeka destekli özet üretme işlemlerini minimum adım ile gerçekleştirebilmesi hedeflenmiştir. Bu doğrultuda sade, anlaşılır ve tutarlı bir arayüz yapısı benimsenmiştir.

Mobil cihazların sınırlı ekran alanı göz önünde bulundurularak, karmaşık görsel yapılardan kaçınılmış; içerik öncelikli ve kullanıcıyı yormayan bir tasarım dili tercih edilmiştir. Uygulama genelinde sezgisel ikonlar, açık etiketler ve tutarlı navigasyon yapısı kullanılmıştır.

6.2. Ekran Yapısı ve Navigasyon Tasarımı

Mobil uygulama, farklı işlevleri temsil eden ekranlardan oluşmaktadır. Bu ekranlar screens/ dizini altında modüler bir şekilde yapılandırılmıştır. Başlıca ekranlar aşağıdaki gibidir:

- **Giriş Ekranı (Login Screen):** Kullanıcı kimlik doğrulama işlemlerinin gerçekleştirildiği ekrandır.
- **Ana Sayfa (Home / Dashboard Screen):** Kullanıcının anı ekleme, arşivlere erişme ve özetleme işlemlerine yönlendirildiği ana kontrol ekranıdır.
- **Anı Ekleme Ekranı (Add Memory Screen):** Fotoğraf, video veya diğer anı türlerinin seçilerek sisteme yüklendiği ekrandır.
- **Arşiv ve Detay Ekranları:** Kullanıcıya ait anıların listelendiği ve detaylarının görüntülendiği ekranlardır.
- **Özetler Ekranı (Summaries Screen):** Yapay zeka destekli görsel üretimin tetiklendiği ve sonuçların gösterildiği ekrandır.

Ekranlar arası geçişler, Flutter'ın yerleşik navigasyon mekanizmaları kullanılarak gerçekleştirilmiş ve kullanıcı akışının kesintiye uğramaması sağlanmıştır.

6.3. Bileşen Tabanlı Arayüz Yapısı

Kullanıcı arayüzünde tekrar eden görsel yapılar için **bileşen tabanlı tasarım** yaklaşımı uygulanmıştır. Bu amaçla, yeniden kullanılabilir UI bileşenleri widgets/ dizini altında toplanmıştır. Örneğin, anıların listelendiği kart yapısı ayrı bir widget olarak tanımlanmıştır:

```
class MemoryCard extends StatelessWidget {
 final Memory memory;
 const MemoryCard({required this.memory});
 @override
 Widget build(BuildContext context) {
  return Card(
   child: ListTile(
```

```
      title: Text(memory.title ?? ''),
      subtitle: Text(memory.memoryDate),
    ),
  );
 }
}
```

Bu yaklaşım sayesinde kod tekrarının önüne geçilmiş, arayüzde tutarlılık sağlanmış ve bakım maliyeti azaltılmıştır.

6.4. Form Tasarımı ve Kullanıcı Etkileşimi

Anı ekleme ve giriş gibi işlemlerde kullanıcıdan veri alınan formlar, mobil kullanım senaryolarına uygun olarak tasarlanmıştır. Form alanları açık etiketler ile desteklenmiş, kullanıcı hatalarını azaltmak amacıyla doğrulama kontrolleri uygulanmıştır.

Dosya seçimi gibi işlemler için mobil cihazların doğal etkileşimleri kullanılmış; galeri erişimi image_picker paketi aracılığıyla gerçekleştirilmiştir. Bu sayede kullanıcılar, alışık oldukları mobil etkileşimler üzerinden uygulamayı kullanabilmiştir.

6.5. Yapay Zeka Sonuçlarının Görselleştirilmesi

Yapay zeka destekli görsel üretimi sonucunda elde edilen çıktılar, kullanıcıya sade ve odaklanmış bir arayüz ile sunulmuştur. Üretilen görsel, ağ üzerinden alınarak doğrudan arayüzde gösterilmektedir:

```
Image.network(imageUrl)
```

Görselin yanında, yapay zeka tarafından üretilen açıklama metni (varsa) kullanıcıya sunularak anıların daha anlamlı bir biçimde yorumlanması sağlanmıştır. Bu tasarım, kullanıcıların yapay zeka çıktısını hızlı ve anlaşılır bir şekilde değerlendirebilmesine olanak tanımaktadır.

6.6. Tema ve Görsel Tutarlılık

Uygulama genelinde tutarlı bir renk paleti, yazı tipi ve boşluk kullanımı benimsenmiştir. Varsayılan olarak açık tema kullanılmış olup, arayüz bileşenleri Material Design prensiplerine uygun şekilde tasarlanmıştır. Yazı tipleri ve ikonlar, okunabilirlik ve erişilebilirlik kriterleri göz önünde bulundurularak seçilmiştir.

Tema ayarları merkezi bir yapı üzerinden yönetilerek uygulama genelinde görsel bütünlük sağlanmıştır.

6.7. UI Tasarımının Değerlendirilmesi

DigiMem mobil uygulamasının kullanıcı arayüzü, mobil cihazların kullanım alışkanlıklarına uygun, sade ve işlevsel bir yapı sunmaktadır. Bileşen tabanlı tasarım yaklaşımı, ekranlar arası tutarlılığı artırmış; kullanıcı etkileşimi açısından anlaşılır ve sezgisel bir deneyim sağlamıştır. Yapay zeka destekli görsel üretim çıktılarının sade bir arayüz ile sunulması, uygulamanın temel amacını destekleyen önemli bir tasarım unsuru olmuştur.

7. MOBİL UYGULAMADA YAPAY ZEKA DESTEKLİ GÖRSEL ÜRETİM SÜRECİ

7.1. Yapay Zeka Kullanım Amacı

DigiMem mobil uygulamasında yapay zeka, kullanıcıların belirli bir zaman aralığında oluşturduğu dijital anıları otomatik olarak analiz ederek bu dönemi temsil eden **tek ve özgün bir görsel çıktı** üretmek amacıyla kullanılmıştır. Bu yaklaşımda hedef, kullanıcıların çok sayıda medya içeriği arasından manuel olarak özet çıkarmasını gereksiz kılmak ve anıların daha soyut, tematik ve anlamlı bir biçimde temsil edilmesini sağlamaktır.

Mobil istemci, yapay zeka sürecinde yalnızca tetikleyici ve sonuçları gösteren bir rol üstlenmekte; tüm analiz ve üretim işlemleri sunucu tarafında gerçekleştirilmektedir. Bu tasarım, mobil uygulamanın performansını korurken karmaşık işlemlerin merkezi olarak yönetilmesine olanak tanımaktadır.

7.2. Mobil Tarafta Yapay Zeka Sürecinin Tetiklenmesi

Mobil uygulamada yapay zeka destekli görsel üretimi, kullanıcı etkileşimi ile başlatılmaktadır. Kullanıcı, uygulama içerisindeki özetler ekranında haftalık veya aylık bir zaman aralığı seçerek ilgili işlemi tetikleyebilmektedir. Seçilen tarih bilgisi, RESTful API aracılığıyla sunucuya iletilmektedir.

Mobil istemci tarafından gerçekleştirilen bu istek, yalnızca gerekli parametreleri içermekte; yapay zeka modeli, medya dosyalarının analizi ve görsel üretimi gibi işlemler tamamen sunucu tarafında yürütülmektedir. Bu yaklaşım, istemci–sunucu sorumluluklarının net biçimde ayrılmasını sağlamaktadır.

7.3. Üretilen Görselin Mobil Arayüzde Sunulması

Sunucu tarafından üretilen yapay zeka görseli, mobil uygulamaya bir URL olarak iletilmektedir. Mobil istemci, bu görseli ağ üzerinden alarak kullanıcıya doğrudan arayüz içerisinde sunmaktadır. Flutter'da bu işlem, yerleşik Image.network bileşeni kullanılarak gerçekleştirilmiştir.

Üretilen görselin sade bir arayüz ile sunulması, kullanıcının yapay zeka çıktısına odaklanmasını kolaylaştırmakta ve uygulamanın temel amacını desteklemektedir. Gerekli durumlarda görsel ile birlikte açıklayıcı bir metin de kullanıcıya gösterilerek anıların daha iyi yorumlanması sağlanmıştır.

7.4. Mobil Yapay Zeka Sürecinin Değerlendirilmesi

Mobil uygulamada uygulanan yapay zeka destekli görsel üretim yaklaşımı, kullanıcı deneyimini zenginleştiren önemli bir bileşen olarak öne çıkmaktadır. Kullanıcının yalnızca birkaç etkileşim ile geçmiş anılarını özetleyebilmesi, mobil cihazlar için uygun ve etkili bir kullanım senaryosu sunmaktadır. Bu yapı, DigiMem mobil uygulamasını yalnızca bir anı saklama aracı olmaktan çıkararak anıların anlamlandırılmasına katkı sağlayan bir sistem haline getirmiştir.

SONUÇ

Bu çalışmada, kullanıcıların dijital anılarını mobil cihazlar üzerinden güvenli, düzenli ve anlamlı bir biçimde yönetebilmelerini amaçlayan **DigiMem mobil uygulaması** geliştirilmiştir. Uygulama, Flutter cross-platform framework'ü kullanılarak tasarlanmış ve mobil kullanım senaryolarına uygun, performanslı ve sürdürülebilir bir mimari sunmuştur.

Çalışma kapsamında, mobil uygulama geliştirme sürecinde modern yazılım mimarileri benimsenmiş; kullanıcı arayüzü, iş mantığı ve veri yönetimi katmanları birbirinden ayrılarak temiz ve yönetilebilir bir yapı oluşturulmuştur. Provider tabanlı durum yönetimi sayesinde reaktif bir kullanıcı deneyimi sağlanmış; mobil istemci ile sunucu arasındaki iletişim JWT tabanlı güvenlik mekanizmaları ile güvence altına alınmıştır.

DigiMem mobil uygulamasının en önemli katkılarından biri, yapay zeka destekli görsel üretim sürecinin mobil ortama başarıyla entegre edilmesidir. Kullanıcıların belirli bir zaman aralığında oluşturduğu anıların analiz edilerek tek bir yapay zeka görseli ile temsil edilmesi, mobil uygulamaya yenilikçi ve katma değerli bir özellik kazandırmıştır. Bu yaklaşım, dijital anıların yalnızca depolanmasını değil, aynı zamanda anlamlandırılmasını da mümkün kılmıştır.

Akademik açıdan değerlendirildiğinde bu çalışma; mobil uygulama geliştirme, RESTful API entegrasyonu, durum yönetimi ve yapay zeka servislerinin kullanımı gibi konuların uygulamalı bir örneğini sunmaktadır. Proje süreci, teorik bilgilerin gerçek bir mobil sistem üzerinde hayata geçirilmesini sağlamış ve geliştiriciye hem teknik hem de problem çözme açısından önemli kazanımlar kazandırmıştır.

Sonuç olarak DigiMem mobil uygulaması, mobil cihazların sunduğu olanaklardan etkin biçimde yararlanan, kullanıcı odaklı ve yapay zeka destekli bir dijital anı yönetim sistemi olarak başarıyla geliştirilmiştir.