ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM4537/BLM4531 PROJE RAPORU

SmartReceipt Mobil Uygulaması

Enes Yıldız
22290180

Video Linki:
https://drive.google.com/drive/folders/1oAZYXNM1WuJS5CaL9y6ugSoaKLVzxGzC?usp=sharing

https://github.com/enesoglu/smart-receipt-api

https://github.com/enesoglu/smart-receipt-mobile

https://github.com/enesoglu/smart-receipt-web

1. ÖZET

SmartReceipt, kullanıcıların market fişlerini dijitalleştirmelerini ve harcamalarını takip etmelerini sağlayan kapsamlı bir sistemdir. Sistem, Flutter ile geliştirilmiş cross-platform mobil uygulama ve ASP.NET Core MVC ile geliştirilmiş web uygulaması olmak üzere iki farklı istemci uygulamasından oluşmaktadır. Her iki uygulama da .NET Web API ile RESTful servis mimarisi üzerinden çalışmakta ve Azure Computer Vision OCR servisi entegrasyonu ile fiş fotoğraflarından otomatik veri çıkarma özelliği sunmaktadır.

Mobil uygulama, iOS ve Android platformlarında native performans sağlayan Flutter framework'ü ile geliştirilmiştir. Web uygulaması ise responsive tasarım ve modern UI/UX prensipleri ile tüm cihazlardan erişilebilir şekilde tasarlanmıştır. Her iki uygulama da JWT/Cookie tabanlı kimlik doğrulama sistemi ile güvenli veri erişimi sağlamakta, kullanıcı bazlı harcama analizi, grafiksel raporlama ve etiket bazlı arama gibi özellikler sunmaktadır.

2. GİRİŞ

2.1. Problem Tanımı

Günlük hayatta alışveriş yaparken elde edilen fiziksel fişlerin kaybolması, yıpranması ve takibinin zor olması gibi sorunlar bulunmaktadır. Kullanıcıların harcamalarını takip etmek, bütçe planlaması yapmak ve vergi iadeleri için fiş saklamak gibi ihtiyaçları bulunmaktadır. Bu sorunları çözmek için hem mobil hem de web platformlarında erişilebilir dijital bir çözüm gereklidir.

2.2. Amaç ve Kapsam

Bu projenin amacı, kullanıcıların: Mobil ve web platformlarından fiş yönetimi yapabilmesi, fiş fotoğraflarını çekerek/yükleyerek otomatik veri çıkarma (OCR), harcamalarını kategorize etme ve etiketleme, aylık/günlük harcama istatistiklerini görselleştirme, mağaza bazlı harcama analizi yapma, fiş arama ve filtreleme işlemlerini gerçekleştirme özelliklerini içeren kapsamlı bir sistem geliştirmektir.

2.3. Proje Kapsamı

Sistem aşağıdaki bileşenlerden oluşmaktadır:

2.3.1. Backend API (.NET Web API)

- RESTful API servisleri
- JWT tabanlı kimlik doğrulama
- Azure Computer Vision OCR entegrasyonu
- SQL Server veritabanı yönetimi
- Repository Pattern ile veri erişimi

2.3.2. Mobil Uygulama (Flutter)

- Cross-platform (iOS/Android) desteği
- Kullanıcı kayıt ve giriş sistemi
- Onboarding ekranları
- Dashboard ile istatistik görüntüleme
- Fiş listeleme, ekleme, düzenleme ve silme
- OCR ile fiş tarama
- Arama ve filtreleme
- Profil yönetimi

2.3.3. Web Uygulaması (ASP.NET Core MVC)

- Responsive tasarım (Bootstrap 5)
- Cookie-based authentication
- Dashboard ile istatistik görüntüleme
- Fiş listeleme, ekleme, düzenleme ve silme
- OCR ile fiş tarama
- Detaylı raporlama sayfası
- Interaktif grafikler (Chart.js)

3. TEKNOLOJİLER

3.1. Flutter Framework

Flutter, Google tarafından geliştirilen açık kaynaklı bir UI framework'üdür. Dart programlama dili kullanılarak geliştirilir ve tek bir kod tabanı ile hem iOS hem de Android platformlarında native performans sağlar. Widget tabanlı mimarisi sayesinde hızlı geliştirme ve tutarlı UI/UX deneyimi sunar. Hot reload özelliği ile geliştirme sürecini hızlandırır.

3.2. ASP.NET Core MVC

ASP.NET Core MVC, Microsoft tarafından geliştirilen açık kaynaklı bir web framework'üdür. Model-View-Controller (MVC) mimari desenini kullanarak, test edilebilir ve bakımı kolay web uygulamaları geliştirmeyi sağlar. Cross-platform desteği ile Windows, Linux ve macOS üzerinde çalışabilir. Razor view engine ile server-side rendering yaparak dinamik web sayfaları oluşturur.

3.3. RESTful API Mimarisi

REST web servisleri için bir mimari stildir. HTTP protokolü üzerinden JSON formatında veri alışverişi yaparak platform bağımsız entegrasyon sağlar. Stateless yapısı sayesinde ölçeklenebilir ve bakımı kolay sistemler oluşturulmasını sağlar.

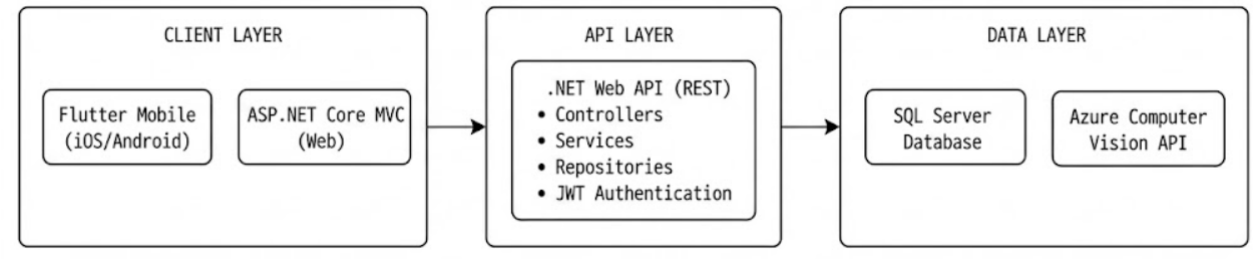
3.4. OCR Teknolojileri

Optical Character Recognition (OCR), görüntülerden metin çıkarma teknolojisidir. Azure Computer Vision API, Microsoft'un bulut tabanlı OCR servisidir ve yüksek doğruluk oranı ile Türkçe karakter desteği sunmaktadır. Receipt API özelliği ile fiş formatlarına özel optimizasyon sağlar.

4. SİSTEM TASARIMI

4.1. Genel Mimari

Sistem, üç katmanlı bir mimari yapıda tasarlanmıştır:



4.2. Veri Modeli

4.2.1. User Modeli

```
public class User
{
    public int Id { get; set; }
    public string Username { get; set; }
    public string PasswordHash { get; set; }
    public List<Receipt> Receipts { get; set; }
}
```

4.2.2. Receipt Modeli

```
public class Receipt
{
    public int Id { get; set; }
    public string StoreName { get; set; }
    public DateTime Date { get; set; }
    public decimal TotalAmount { get; set; }
    public string? ImagePath { get; set; }
    public string? Tags { get; set; }
    public int UserId { get; set; }
    public User? User { get; set; }
    public List<ReceiptItems> Items { get; set; }
}
```

4.2.3. ReceiptItems Modeli

```
public class ReceiptItems
{
    public int Id { get; set; }
    public string ProductName { get; set; }
    public decimal Price { get; set; }
    public int ReceiptId { get; set; }
    public Receipt? Receipt { get; set; }
}
```

4.3. API Endpoint Yapısı

4.3.1. Authentication Endpoints

- POST /api/auth/register - Kullanıcı kaydı
- POST /api/auth/login - Kullanıcı girişi (JWT token döner)

4.3.2. Receipt Endpoints

- GET /api/receipts - Kullanıcının tüm fişlerini listele
- GET /api/receipts/{id} - Belirli bir fişi getir
- POST /api/receipts - Yeni fiş ekle
- PUT /api/receipts/{id} - Fiş güncelle
- DELETE /api/receipts/{id} - Fiş sil
- POST /api/receipts/scan - OCR ile fiş tara
- GET /api/receipts/search?query=... - Fiş ara

4.3.3. Statistics Endpoints

- GET /api/receipts/stats - Dashboard istatistikleri
- GET /api/receipts/store-stats - Mağaza bazlı istatistikler
- GET /api/receipts/daily-spending?year=...&month=... - Günlük harcama
- GET /api/receipts/dashboard - Kapsamlı dashboard verileri

4.4. Güvenlik Tasarımı

4.4.1. Mobil Uygulama (JWT)

- JWT token SharedPreferences'ta saklanır
- Her API çağrısında Authorization: Bearer <token> header'ı eklenir
- Token süresi: 60 dakika
- Token süresi dolduğunda otomatik logout

4.4.2. Web Uygulaması (Cookie)

- Cookie-based authentication
- HttpOnly ve Secure flag'leri
- Sliding expiration: 60 dakika
- CSRF token koruması

5. UYGULAMA VE İMPLEMENTASYON

5.1. Backend API (.NET Web API)

5.1.1. Proje Yapısı

```
smart-receipt-api/
├── Controllers/
│   ├── AuthController.cs
│   └── ReceiptsController.cs
├── Services/
│   ├── IUserService.cs / UserService.cs
│   ├── IReceiptService.cs / ReceiptService.cs
│   └── IOcrService.cs / OcrService.cs
├── Repositories/
│   ├── IRepository.cs
│   └── Repository.cs
├── Models/
│   ├── User.cs
│   ├── Receipt.cs
│   └── ReceiptItems.cs
├── DTOs/
│   └── DtoModels.cs
├── AppDbContext.cs
└── Program.cs
```

5.1.2. Repository Pattern

Generic repository pattern kullanılarak veri erişimi soyutlanmıştır:

```
public interface IRepository<T> where T : class
{
    Task<IEnumerable<T>> GetAllAsync();
    Task<T?> GetByIdAsync(int id);
    Task AddAsync(T entity);
    Task UpdateAsync(T entity);
    Task DeleteAsync(int id);
    Task SaveAsync();
}
```

5.1.3. Service Katmanı

Business logic, service katmanında implemente edilmiştir:

```
public class ReceiptService : IReceiptService
{
    private readonly IRepository<Receipt> _receiptRepository;

    public async Task<IEnumerable<Receipt>> GetUserReceiptsAsync(int
userId)
    {
        var receipts = await _receiptRepository.GetAllAsync();
        return receipts.Where(r => r.UserId == userId).ToList();
    }

    // Diğer metodlar...
}
```

5.1.4. OCR Servisi

Azure Computer Vision API entegrasyonu:

```
public class OcrService : IOcrService
{
    public async Task<Dictionary<string, string>>
ExtractReceiptDataAsync(IFormFile image)
    {
        using var stream = image.OpenReadStream();
        var bytes = new byte[stream.Length];
        await stream.ReadAsync(bytes, 0, (int)stream.Length);

        using var imageStream = new MemoryStream(bytes);
        var result = await client.ReadInStreamAsync(imageStream);

        // Metin çıkarma ve parsing işlemleri...
    }
}
```

5.1.5. JWT Authentication

```
builder.Services.AddAuthentication("Bearer")
    .AddJwtBearer("Bearer", options =>
    {
        options.TokenValidationParameters = new
TokenValidationParameters
        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(secretKey),
            ValidateLifetime = true,
            ClockSkew = TimeSpan.Zero
        };
    });
```

5.2. Mobil Uygulama (Flutter)

5.2.1. Proje Yapısı

```
lib/
├── main.dart
├── models/
│   ├── receipt.dart
│   └── user.dart
├── services/
│   └── api_service.dart
└── pages/
    ├── login.dart
    ├── register.dart
    ├── main_screen.dart
    ├── home.dart
    ├── search.dart
    ├── receipts.dart
    ├── profile.dart
    ├── scan_receipt.dart
    └── add_receipt.dart
```

5.2.2. API Servis Katmanı

Singleton pattern ile merkezi API yönetimi:

```
class ApiService {
  static final ApiService _instance = ApiService._internal();
  factory ApiService() => _instance;

  late Dio _dio;
  String? _token;

  ApiService._internal() {
    _dio = Dio(BaseOptions(
      baseUrl: baseUrl,
      connectTimeout: const Duration(seconds: 30),
    ));
```

5.2.3. Token Yönetimi

```
Future<void> loadToken() async {
  final prefs = await SharedPreferences.getInstance();
  _token = prefs.getString('jwt_token');
  _userId = prefs.getInt('user_id');
  _username = prefs.getString('username');
}

Future<void> _saveToken(String token, int userId, String username)
async {
  final prefs = await SharedPreferences.getInstance();
  await prefs.setString('jwt_token', token);
  await prefs.setInt('user_id', userId);
  await prefs.setString('username', username);
  _token = token;
  _userId = userId;
  _username = username;
}
```

5.2.4. OCR Entegrasyonu

```
Future<ApiResponse<Map<String, dynamic>>> scanReceipt(File
imageFile) async {
  String fileName = imageFile.path.split('/').last;
  FormData formData = FormData.fromMap({
    'image': await MultipartFile.fromFile(
      imageFile.path,
      filename: fileName,
    ),
  });

  final response = await _dio.post(
    '/receipts/scan',
    data: formData,
    options: Options(
      headers: {'Content-Type': 'multipart/form-data'},
    ),
  );

  return ApiResponse<Map<String, dynamic>>.fromJson(
    response.data,
    (json) => json as Map<String, dynamic>,
  );
}
```

5.3. Web Uygulaması (ASP.NET Core MVC)

5.3.1. Proje Yapısı

```
smart-receipt-web/
├── Controllers/
│   ├── AuthController.cs
│   ├── HomeController.cs
│   └── ReceiptController.cs
├── Views/
│   ├── Auth/
│   ├── Home/
│   ├── Receipt/
│   └── Shared/
├── Services/
│   ├── IReceiptApiService.cs / ReceiptApiService.cs
│   └── IAuthService.cs / AuthService.cs
├── Models/
│   └── ViewModels.cs
├── Middleware/
│   └── ErrorHandlingMiddleware.cs
└── Program.cs
```

5.3.2. Cookie Authentication

```
builder.Services.AddAuthentication(CookieAuthenticationDefaults.Auth
enticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Auth/Login";
        options.LogoutPath = "/Auth/Logout";
        options.ExpireTimeSpan = TimeSpan.FromMinutes(60);
        options.SlidingExpiration = true;
        options.Cookie.HttpOnly = true;
        options.Cookie.SecurePolicy =
CookieSecurePolicy.SameAsRequest;
    });
```

5.3.3. API Servis Katmanı

```
public class ReceiptApiService : IReceiptApiService
{
    private readonly HttpClient _httpClient;
    private readonly IHttpContextAccessor _httpContextAccessor;

    public async Task<List<ReceiptDto>> GetReceiptsAsync()
    {
        // JWT token'ı cookie'den al ve header'a ekle
        var token = GetTokenFromCookie();
        _httpClient.DefaultRequestHeaders.Authorization =
            new AuthenticationHeaderValue("Bearer", token);

        var response = await _httpClient.GetAsync("receipts");
        response.EnsureSuccessStatusCode();

        var json = await response.Content.ReadAsStringAsync();
```

```
        var apiResponse =
JsonSerializer.Deserialize<ApiResponse<List<ReceiptDto>>>(json);
        return apiResponse?.Data ?? new List<ReceiptDto>();
    }
}
```

5.3.7. Responsive Tasarım

Bootstrap 5 grid sistemi ile responsive layout:

```
<div class="container-fluid">
    <div class="row">
        <!-- Sidebar - Desktop'ta görünür, mobile'da gizli -->
        <div class="col-md-2 d-none d-md-block bg-light sidebar">
            <!-- Navigation -->
        </div>

        <!-- Main Content -->
        <main class="col-md-10 ms-sm-auto px-md-4 py-4">
            <!-- Dashboard content -->
        </main>
    </div>
</div>
```

5.4. Kullanılan Teknolojiler ve Kütüphaneler

5.4.1. Backend

- **.NET 8.0**: Framework versiyonu
- **Entity Framework Core**: ORM
- **SQL Server**: Veritabanı
- **Azure Computer Vision**: OCR servisi
- **JWT Bearer**: Token authentication

5.4.2. Mobil Uygulama

- **Flutter**: ^3.9.2
- **shared_preferences**: ^2.2.2 - Local storage
- **fl_chart**: ^0.66.2 - Grafikler
- **image_picker**: ^1.0.7 - Kamera/Galeri
- **camera**: ^0.11.0 - Kamera kontrolü
- **intl**: ^0.19.0 - Tarih formatlama

5.4.3. Web Uygulaması

- **ASP.NET Core MVC**: .NET 10.0
- **Bootstrap 5**: CSS framework
- **Bootstrap Icons**: Icon kütüphanesi
- **Chart.js**: Grafik çizimi

6. TEST VE SONUÇLAR

6.1. Fonksiyonel Testler

6.1.1. Backend API Testleri

- Kullanıcı kayıt ve giriş işlemleri
- JWT token oluşturma ve doğrulama
- Fiş CRUD işlemleri
- OCR servisi entegrasyonu

6.1.2. Mobil Uygulama Testleri

- Kullanıcı kayıt ve giriş
- Dashboard veri gösterimi
- Fiş listeleme, ekleme, düzenleme, silme
- OCR ile fiş tarama
- Arama ve filtreleme
- Grafik gösterimi

6.1.3. Web Uygulaması Testleri

- Kullanıcı kayıt ve giriş
- Dashboard veri gösterimi
- Fiş CRUD işlemleri
- OCR ile fiş tarama
- Grafik gösterimi (Chart.js)

6.2. Güvenlik Testleri

- JWT token güvenli saklama (mobil)
- Cookie HttpOnly ve Secure flag'leri (web)
- HTTPS iletişim
- Kullanıcı bazlı veri izolasyonu
- Input validasyonu ve sanitization
- CSRF token koruması (web)
- SQL Injection koruması (parameterized queries)

6.3. Kullanılabilirlik Testleri

6.3.1. Mobil Uygulama

- **Platform Desteği**: iOS ve Android
- **Erişilebilirlik**: Ekran okuyucu desteği
- **Hata Yönetimi**: Kullanıcı dostu hata mesajları

6.3.2. Web Uygulaması

- **Tarayıcı Uyumluluğu**: Chrome, Firefox, Edge, Safari
- **Responsive Tasarım**: Mobile, tablet, desktop

7. KARŞILAŞILAN SORUNLAR VE ÇÖZÜMLERİ

7.1. SSL Sertifika Sorunu

Sorun: Development ortamında self-signed SSL sertifikası kullanımı nedeniyle hem mobil hem web uygulamalarında bağlantı hatası.

Çözüm:

Mobil: Dio interceptor ile SSL doğrulaması bypass edildi

Web: HttpClient handler'da SSL doğrulaması bypass edildi

7.2. Platform Özel API URL'leri

Sorun: Android emulator ve iOS simulator farklı localhost adresleri kullanır.

Çözüm: Platform kontrolü ile dinamik URL belirleme:

```
static String get baseUrl {
  if (Platform.isAndroid) {
    return "https://10.0.2.2:7018/api";
  }
  return "https://localhost:7018/api";
}
```

7.3. CORS Politikası

Sorun: Web uygulaması farklı port'tan API'ye erişirken CORS hatası.

Çözüm: API tarafında CORS policy eklendi:

```
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowWebApp", builder =>
    {
        builder.WithOrigins("http://localhost:5007",
"https://localhost:7001")
               .AllowAnyMethod()
               .AllowAnyHeader()
               .AllowCredentials();
    });
});
```

7.4. OCR Metin Parsing

Sorun: Azure OCR'dan gelen ham metin, farklı fiş formatları nedeniyle tutarsız yapıda.

Çözüm:

Backend: Regex pattern matching ile mağaza adı, tarih ve toplam tutar çıkarımı

Web: Client-side JavaScript ile ek parsing ve form doldurma

7.5. Token Yönetimi (Web)

Sorun: Web uygulaması Cookie authentication kullanırken API JWT token gerektiriyor.

Çözüm: Login sonrası JWT token'ı cookie'ye kaydetme ve API çağrılarında kullanma:

```
var token = GetTokenFromCookie();
_httpClient.DefaultRequestHeaders.Authorization =
    new AuthenticationHeaderValue("Bearer", token);
```

7.6. State Management (Mobil)

Sorun: Flutter'da state management için merkezi bir yapı yoktu.

Çözüm: Singleton pattern ile ApiService ve her sayfada setState kullanımı. (Gelecekte Provider veya Riverpod eklenebilir)

8. SONUÇ VE ÖNERİLER

8.1. Proje Sonuçları

SmartReceipt sistemi başarıyla geliştirilmiş ve aşağıdaki özellikler sunulmuştur:

8.1.1. Backend API
- RESTful API servisleri
- JWT tabanlı kimlik doğrulama
- Azure Computer Vision OCR entegrasyonu
- Repository Pattern ile veri erişimi
- Kullanıcı bazlı veri izolasyonu

8.1.2. Mobil Uygulama
- Cross-platform (iOS/Android) desteği
- Güvenli kayıt ve giriş sistemi
- OCR ile otomatik fiş veri çıkarma
- Kapsamlı dashboard ve istatistikler
- Fiş yönetimi (CRUD)
- Arama ve filtreleme
- İnteraktif grafikler

8.1.3. Web Uygulaması
- Responsive tasarım
- Cookie-based güvenli kayıt ve giriş
- OCR ile otomatik fiş veri çıkarma
- Kapsamlı dashboard ve interaktif grafikler
- Fiş yönetimi (CRUD)
- Detaylı raporlama

8.2. Gelecek Geliştirmeler

8.2.1. Kısa Vadeli Öneriler

Backend:

- Response caching ile performans artırımı
- Rate limiting eklenmesi
- Logging sistemi (Serilog)
- Unit test coverage artırılması

Mobil Uygulama:

- Offline mode desteği
- Cloud backup (fiş fotoğrafları)
- Export özelliği (PDF/Excel)
- Dark mode desteği

Web Uygulaması:

- Export özelliği (PDF/Excel)
- Gelişmiş filtreleme (tarih aralığı, mağaza)
- Pagination (büyük veri setleri)

8.2.2. Uzun Vadeli Öneriler

Sistem Geneli:

- AI kategorizasyon (makine öğrenmesi)
- Email bildirimleri
- Push notification (mobil)
- Widget desteği (mobil)

Teknik İyileştirmeler:

- Microservices mimarisi
- Docker containerization
- CI/CD pipeline
- Automated testing (E2E)

8.3. Teknik İyileştirmeler

8.3.1. Mobil Uygulama

- **State Management**: Provider veya Riverpod entegrasyonu
- **Caching**: Hive veya SQLite ile local cache
- **Unit Tests**: Test coverage artırılması

8.3.2. Web Uygulaması

- **Caching**: Response caching ile performans
- **Unit Tests**: Test coverage artırılması

8.4. Güvenlik İyileştirmeleri

- **Rate Limiting**: API çağrı limitleri
- **Input Validation**: Daha kapsamlı sanitization
- **HTTPS**: Production'da zorunlu
- **Token Refresh**: Refresh token mekanizması