Elinizdeki Malzemelere Göre Akıllı Yemek Tarifi Öneri Uygulaması - FOODLYZE

Proje Özeti

FOODLYZE, kullanıcıların evde bulunan malzemelere göre yemek tarifleri bulmalarını kolaylaştıran bir mobil uygulamadır. Kullanıcılar, ürünlerin barkodlarını okutarak ya da manuel olarak ürün bilgisi girerek, bu ürünlerle yapılabilecek tarif önerilerini görüntüleyebilmektedir.

Gıda israfını azaltmayı, yemek planlamasını kolaylaştırmayı ve kullanıcıya özel tarif önerileri sunmayı amaçlayan bu uygulama; kullanıcıların kişisel ihtiyaçlarını dikkate alan dinamik ve yapay zeka destekli öneri sistemi ile fark yaratmayı hedeflemektedir.

Amaç ve Hedefler

- Kullanıcının sahip olduğu malzemelere göre pratik ve lezzetli tarifler önermek.

- Zaman kazandıran, kullanıcı dostu bir arayüz sunmak.

- Gıda israfını azaltmak ve ev ekonomisine katkıda bulunmak.

- Kullanıcı deneyimlerine dayalı sürekli gelişen bir öneri sistemi oluşturmak.

- Kullanıcıların tariflere yorum yapabileceği bir geri bildirim sistemi sağlamak.

Proje Ekibi ve Paydaşlar

Ekip Üyeleri:

- Mert Demirkol (230260200)

- Ahmet Dağıstanlı (230260198)

- Ali Nebi Er (230260152)

Paydaşlar ve Hedef Kitle:

- Son Kullanıcılar: Öğrenciler, çalışan bireyler, aileler

- Yemek İçerik Sağlayıcıları: Yemek blogları, şefler, tarif platformları

- Uygulama Geliştiricileri: Yazılım geliştiriciler, tasarımcılar, veri bilimciler

- Yatırımcılar: Projenin ticari potansiyelinden faydalanmak isteyen girişimciler

Yazılım Mimarisi

1. Katmanlı Mimari:

- Sunum Katmanı: Android kullanıcı arayüzü; barkod okuma, ürün listesi, tarif öneri ekranları

- İş Mantığı Katmanı: Barkoddan gelen veriyi işleyip tarif önerilerini üretir

- Veri Katmanı: OpenFoodFacts, CollectAPI gibi servislerle Retrofit üzerinden iletişim sağlar

- Katmanlar arası sorumluluklar net şekilde ayrılmış, Repository sınıfı oluşturulmuştur.

2. MVVM (Model-View-ViewModel):

- Model: Ürün verileri, tarifler, API veri yapıları

- View: XML tabanlı arayüzler, LiveData gözlemciliği ile ViewModel'e bağlı

- ViewModel: UI ile iş mantığını birbirine bağlar

- MVVM mimarisi Android Jetpack bileşenleri ile uyumlu şekilde yapılandırılmıştır.

Tasarım Desenleri

- Repository Pattern: API'den alınan verileri ViewModel'e iletmek için

- Strategy Pattern : Farklı öneri stratejilerinin dinamik seçimi

- Observer Pattern: LiveData ile veri değişimlerinin izlenmesi

- Dependency Injection: Hilt veya Dagger entegrasyonu ile bağımlılık kontrolü

Yenilikçi Yönler

- Yapay zeka ile kişisel tarif önerisi

- Barkod okuma ile malzeme tanıma

- Gıda israfını azaltma hedefi

- Kullanıcıya özel, gerçek zamanlı öneriler

Kullanılan Teknolojiler

- Mobil Geliştirme: Android Studio, React Native

- API Entegrasyonu: Retrofit, OpenFoodFacts, CollectAPI

- Veri Yönetimi: LiveData, ViewModel, Room

- Kaynaklar: Firebase, Udemy, YouTube eğitim içerikleri

Sonuç ve Gelecek Planları

- Katmanlı mimari ve MVVM başarıyla kurulmuştur

- Repository yapısı uygulanmıştır

- Geliştirme sürecinde Strategy ve DI desenleri entegre edilecektir

- Geri bildirim sisteminin ve AI tabanlı öneri motorunun eklenmesi planlanmaktadır

- Ticari potansiyel için MVP sürüm oluşturulacaktır

Uygulamaya Ait Geliştirilen Kodlar

BARKOD SAYFASI KODU :

```
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.bumptech.glide.Glide;
import com.bumptech.glide.load.resource.bitmap.RoundedCorners;
import com.bumptech.glide.request.RequestOptions;
import com.demirkol.foodlyze.databinding.ActivityBarkodsayfasiBinding;
import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class Barkodsayfasi extends AppCompatActivity {

    private static final int CAMERA_PERMISSION_REQUEST_CODE = 100;
    private TextView resultTextView;
    private ImageView nutritionImageView;
```

```
    private ActivityBarkodsayfasiBinding binding;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding =
ActivityBarkodsayfasiBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        initializeViews();
        checkCameraPermission();
        setupClickListeners();
    }

    private void initializeViews() {
        resultTextView = findViewById(R.id.resultTextView);
        nutritionImageView = findViewById(R.id.urunresmi);

        // Hide debug text by default
        resultTextView.setVisibility(View.GONE);
    }

    private void setupClickListeners() {
        findViewById(R.id.scanButton).setOnClickListener(view -> {
            IntentIntegrator integrator = new
IntentIntegrator(Barkodsayfasi.this);

integrator.setDesiredBarcodeFormats(IntentIntegrator.ALL_CODE_TYPES);
            integrator.setPrompt("Barkodu kameraya yaklaştırın");
            integrator.setCameraId(0);
            integrator.setBeepEnabled(true);
            integrator.setBarcodeImageEnabled(false);
            integrator.setOrientationLocked(false);
            integrator.initiateScan();
        });
    }

    private void checkCameraPermission() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.CAMERA}, CAMERA_PERMISSION_REQUEST_CODE);
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

        if (requestCode == CAMERA_PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
```

```
                showToast("Kamera izni verildi.", false);
            } else {
                showToast("Kamera izni gerekli!", true);
            }
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        IntentResult result =
IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
        if (result != null) {
            String barcode = result.getContents();
            if (barcode != null) {
                showToast("Barkod tarandı: " + barcode, false);
                fetchProductDetails(barcode);
            } else {
                showToast("Barkod tarama iptal edildi.", false);
            }
        }
    }

    private void fetchProductDetails(String barcode) {
        showLoadingState(true);

        OpenFoodFactsAPI apiService =
RetrofitClient.getClient().create(OpenFoodFactsAPI.class);
        Call<ProductResponse> call =
apiService.getProductDetails(barcode);

        call.enqueue(new Callback<ProductResponse>() {
            @Override
            public void onResponse(Call<ProductResponse> call,
Response<ProductResponse> response) {
                showLoadingState(false);

                if (response.isSuccessful() && response.body() != null)
{
                    ProductResponse.Product product =
response.body().getProduct();
                    if (product != null) {
                        populateProductData(product);
                        showToast("Ürün bilgileri yüklendi!", false);
                    } else {
                        showToast("Ürün bulunamadı!", true);
                        clearProductData();
                    }
                } else {
                    showToast("API hatası: " + response.code(), true);
                    clearProductData();
                }
            }

            @Override
```

```
            public void onFailure(Call<ProductResponse> call, Throwable
t) {
                showLoadingState(false);
                showToast("Bağlantı hatası: " + t.getMessage(), true);
                clearProductData();
            }
        });
    }

    private void populateProductData(ProductResponse.Product product) {
        // Ürün adı
        TextView urunAdi = findViewById(R.id.urunadi);
        urunAdi.setText(product.getProductName() != null ?
product.getProductName() : "Bilinmeyen Ürün");

        // Marka
        TextView marka = findViewById(R.id.marka);
        marka.setText(product.getBrands() != null ? product.getBrands()
: "Bilinmeyen Marka");

        // Miktar
        TextView miktar = findViewById(R.id.miktar);
        miktar.setText(product.getQuantity() != null ?
product.getQuantity() : "Bilinmeyen Miktar");

        // Ürün resmi
        ImageView urunResmi = findViewById(R.id.urunresmi);
        if (product.getProductImage() != null &&
!product.getProductImage().isEmpty()) {
            Glide.with(this)
                    .load(product.getProductImage())
                    .apply(new RequestOptions()
                            .transform(new RoundedCorners(16))
                            .placeholder(R.drawable.photoicon)
                            .error(R.drawable.photoicon))
                    .into(urunResmi);
        }

        // İçindekiler resmi
        ImageView icindekilerGorsel =
findViewById(R.id.urunicindekiler_gorsel);
        if (product.getIngredientsImage() != null &&
!product.getIngredientsImage().isEmpty()) {
            Glide.with(this)
                    .load(product.getIngredientsImage())
                    .apply(new RequestOptions()
                            .placeholder(R.drawable.photoicon)
                            .error(R.drawable.photoicon))
                    .into(icindekilerGorsel);
        }

        // İçindekiler metni
        TextView malzemelerMetni = findViewById(R.id.malzemelerMetni);
        String ingredients = product.getIngredientsText() != null ?
product.getIngredientsText() :
                (product.getIngredientsTextEn() != null ?
product.getIngredientsTextEn() : "İçerik bilgisi bulunamadı");
```

```
        malzemelerMetni.setText(ingredients);

        // Etiketler
        TextView malzemeEtiket = findViewById(R.id.malzeme_etiket);
        malzemeEtiket.setText(product.getLabels() != null ?
product.getLabels() : "Etiket bilgisi yok");

        // Nutriscore
        ImageView nutriscoreDurumu =
findViewById(R.id.nutriscoredurumu);
        setNutriscoreImage(nutriscoreDurumu,
product.getNutriscoreGrade());

        // Helal/Haram durumu
        ImageView helalHaramDurum = findViewById(R.id.helalharamdurum);
        setHalalStatus(helalHaramDurum,
product.getIngredientsAnalysisTags());

        // Besin değerleri
        if (product.getNutriments() != null) {
            populateNutritionData(product.getNutriments());
        } else {
            clearNutritionData();
        }

        // Alerjen bilgisi
        TextView alerjen = findViewById(R.id.alerjen);
        String allergenInfo = product.getAllergens() != null ?
product.getAllergens() : "Alerjen bilgisi bulunamadı";
        alerjen.setText(allergenInfo);
    }

    private void populateNutritionData(Nutriments nutriments) {
        // Enerji
        TextView energy = findViewById(R.id.energy);
        energy.setText(formatNutritionValue(nutriments.getEnergy(),
"kcal"));

        // Protein
        TextView proteins = findViewById(R.id.proteins);
        proteins.setText(formatNutritionValue(nutriments.getProtein(),
"g"));

        // Karbonhidrat
        TextView carbohydrates = findViewById(R.id.carbohydrates);

carbohydrates.setText(formatNutritionValue(nutriments.getCarbohydrates(
), "g"));

        // Şeker
        TextView sugars = findViewById(R.id.sugars);
        sugars.setText(formatNutritionValue(nutriments.getSugars(),
"g"));

        // Yağ
        TextView fat = findViewById(R.id.fat);
        fat.setText(formatNutritionValue(nutriments.getFat(), "g"));
```

```
        // Doymuş yağ
        TextView doymusYag = findViewById(R.id.doymusyag);
        doymusYag.setText(formatNutritionValue(nutriments.getFat(),
"g"));

        // Tuz
        TextView tuz = findViewById(R.id.tuz);
        tuz.setText(formatNutritionValue(nutriments.getSalt(), "g"));

        // Lif
        TextView lif = findViewById(R.id.lif);
        lif.setText(formatNutritionValue(nutriments.getSalt(), "g"));

        // Kalsiyum
        TextView calcium = findViewById(R.id.calcium);

calcium.setText(formatNutritionValue(nutriments.getCarbohydrates(),
"mg"));
    }

    private void clearNutritionData() {
        ((TextView) findViewById(R.id.energy)).setText("0 kcal");
        ((TextView) findViewById(R.id.proteins)).setText("0 g");
        ((TextView) findViewById(R.id.carbohydrates)).setText("0 g");
        ((TextView) findViewById(R.id.sugars)).setText("0 g");
        ((TextView) findViewById(R.id.fat)).setText("0 g");
        ((TextView) findViewById(R.id.doymusyag)).setText("0 g");
        ((TextView) findViewById(R.id.tuz)).setText("0 g");
        ((TextView) findViewById(R.id.lif)).setText("0 g");
        ((TextView) findViewById(R.id.calcium)).setText("0 mg");
    }

    private String formatNutritionValue(String value, String unit) {
        if (value != null && !value.isEmpty()) {
            try {
                double numValue = Double.parseDouble(value);
                return String.format("%.1f %s", numValue, unit);
            } catch (NumberFormatException e) {
                return "0 " + unit;
            }
        }
        return "0 " + unit;
    }

    private void setNutriscoreImage(ImageView imageView, String grade)
{
        if (grade != null) {
            switch (grade.toLowerCase()) {
                case "a":
                    imageView.setImageResource(R.drawable.nutiscorea);
                    break;
                case "b":
                    imageView.setImageResource(R.drawable.nuriscoreb);
                    break;
                case "c":
                    imageView.setImageResource(R.drawable.nuriscorec);
```

```
                    break;
                case "d":
                    imageView.setImageResource(R.drawable.nuriscored);
                    break;
                case "e":
                    imageView.setImageResource(R.drawable.nuriscoree);
                    break;
                default:
                    imageView.setImageResource(R.drawable.nutiscorea);
// default
                    break;
            }
        }
    }

    private void setHalalStatus(ImageView imageView,
java.util.List<String> analysisTags) {
        boolean isHalal = false;
        boolean isHaram = false;

        if (analysisTags != null) {
            for (String tag : analysisTags) {
                if (tag.contains("halal")) {
                    isHalal = true;
                } else if (tag.contains("non-halal") ||
tag.contains("haram")) {
                    isHaram = true;
                }
            }
        }

        if (isHalal) {
            imageView.setImageResource(R.drawable.helal);
        } else if (isHaram) {
            imageView.setImageResource(R.drawable.haram);
        } else {
            imageView.setImageResource(R.drawable.cheese); // belirsiz
durumu için
        }
    }

    private void clearProductData() {
        // Tüm alanları temizle
        ((TextView) findViewById(R.id.urunadi)).setText("Ürün Adı");
        ((TextView) findViewById(R.id.marka)).setText("Marka");
        ((TextView) findViewById(R.id.miktar)).setText("Miktar");
        ((TextView)
findViewById(R.id.malzemelerMetni)).setText("İçindekiler");
        ((TextView)
findViewById(R.id.malzeme_etiket)).setText("Etiketler");
        ((TextView) findViewById(R.id.alerjen)).setText("Alerjen
bilgisi bulunamadı");

        // Resimleri sıfırla
        ((ImageView)
findViewById(R.id.urunresmi)).setImageResource(R.drawable.photoicon);
        ((ImageView)
```

```
findViewById(R.id.urunicindekiler_gorsel)).setImageResource(R.drawable.
photoicon);
        ((ImageView)
findViewById(R.id.nutriscoredurumu)).setImageResource(R.drawable.nutisc
orea);
        ((ImageView)
findViewById(R.id.helalharamdurum)).setImageResource(R.drawable.helal);

        // Besin değerlerini sıfırla
        clearNutritionData();
    }

    private void showLoadingState(boolean isLoading) {
        findViewById(R.id.scanButton).setEnabled(!isLoading);
        if (isLoading) {
            resultTextView.setText("Ürün bilgileri yükleniyor...");
            resultTextView.setVisibility(View.VISIBLE);
        } else {
            resultTextView.setVisibility(View.GONE);
        }
    }

    private void showToast(String message, boolean isError) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }
}
```

BARKOD SAYFASI AÇIKLAMA :

📱 Sayfanın Amacı:

Kullanıcı bir ürünün barkodunu taratarak:

- Ürün adı, marka, miktar,
- Ürün görselleri,
- İçindekiler, etiketler,
- Alerjen bilgileri,
- Nutriscore (besin kalitesi puanı),
- Helal/Haram durumu,
- Besin değerleri (enerji, protein, şeker, tuz vb.)

gibi bilgileri uygulama arayüzünde görüntüleyebilir.

🔍 Temel İşlevler:

1. **Barkod Tarama:**
    - `IntentIntegrator` kullanılarak kamera açılır.
    - Tarama sonucu alındığında `onActivityResult` ile alınır.

2. **Kamera İzni:**
   - Android 6.0+ cihazlar için kullanıcıdan kamera izni istenir.
3. **API Kullanımı:**
   - `Retrofit` kullanılarak OpenFoodFacts API'ye istek gönderilir.
   - Barkod ile eşleşen ürün bilgileri JSON olarak alınır.
4. **Veri Gösterimi:**
   - `populateProductData()` fonksiyonu ile API'den gelen bilgiler, ilgili TextView ve ImageView'lara yerleştirilir.
   - Besin değerleri `populateNutritionData()` fonksiyonu ile gösterilir.
5. **Duruma Göre Görsel Ayarlamaları:**
   - `setNutriscoreImage()` ile nutriscore harfi görselleştirilir (A–E).
   - `setHalalStatus()` ile ürünün helal/haram durumu görsel olarak gösterilir.
6. **Veri Temizleme:**
   - Eğer ürün bulunamazsa veya hata alınırsa, `clearProductData()` çağrılır ve tüm bilgiler sıfırlanır.
7. **Yüklenme Durumu:**
   - API yanıtı beklenirken "Ürün bilgileri yükleniyor…" metni gösterilir.

🛠 Kullanılan Teknolojiler:

- **ZXing**: Barkod tarama
- **Retrofit**: API ile veri alışverişi
- **Glide**: Görsel yükleme
- **ViewBinding**: XML bileşenlerine erişim
- **OpenFoodFacts API**: Ürün veri kaynağı

YEMEK TARİFİ SAYFASI KODU :

```
package com.demirkol.foodlyze;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import android.view.View;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.ScrollView;
import android.widget.TextView;
```

```
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.constraintlayout.widget.ConstraintLayout;

import com.demirkol.GeminiApiService;
import com.demirkol.GeminiClient;
import com.demirkol.GeminiRequest;
import com.demirkol.GeminiResponse;
import com.google.android.material.button.MaterialButton;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.textfield.TextInputEditText;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class Yemektarifilayaout extends AppCompatActivity {
    private static final String TAG = "FoodlyzeApp";
    private static final String API_KEY = "APIKEY ANAHTARI";

    // UI Components
    private TextInputEditText editText;
    private TextView resultText;
    private TextView animatedText;
    private MaterialButton submitButton;
    private MaterialButton saveButton;
    private MaterialButton shareButton;
    private FloatingActionButton newRecipeFab;
    private CardView responseCard;
    private CardView inputCard;
    private FrameLayout animatedTextContainer;
    private ProgressBar loadingProgress;
    private ScrollView scrollView;

    private Handler animationHandler;
    private Runnable animationRunnable;
    private boolean isAnimating = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_yemektarifilayaout);

        initializeViews();
        setupClickListeners();
        animationHandler = new Handler(Looper.getMainLooper());
    }

    private void initializeViews() {
        // Input components
        editText = findViewById(R.id.editText);
        submitButton = findViewById(R.id.submitButton);
        inputCard = findViewById(R.id.inputCard);
```

```
        // Response components
        responseCard = findViewById(R.id.responseCard);
        resultText = findViewById(R.id.resultText);
        animatedText = findViewById(R.id.animatedText);
        animatedTextContainer =
findViewById(R.id.animatedTextContainer);
        loadingProgress = findViewById(R.id.loadingProgress);

        // Action buttons
        saveButton = findViewById(R.id.saveButton);
        shareButton = findViewById(R.id.shareButton);
        newRecipeFab = findViewById(R.id.newRecipeFab);
    }

    private void setupClickListeners() {
        // Submit button click
        submitButton.setOnClickListener(v -> {
            String ingredients = editText.getText().toString().trim();

            if (ingredients.isEmpty()) {
                Toast.makeText(this, "Lütfen malzemeleri girin",
Toast.LENGTH_SHORT).show();
                return;
            }

            String prompt = "Elimde şu malzemeler var: " + ingredients
+
                    ". Bu malzemelerle hangi yemekleri yapabilirim?
Kısa tarifler verir misin? " +
                    "Lütfen yanıtını düzenli ve okunabilir şekilde
formatla.";

            startRecipeSearch(prompt);
        });

        // Save button click
        saveButton.setOnClickListener(v -> {
            String recipeContent = resultText.getText().toString();
            if (!recipeContent.isEmpty()) {
                // Tarifi kaydetme işlemi burada yapılabilir
                Toast.makeText(this, "Tarif kaydedildi! 💾",
Toast.LENGTH_SHORT).show();
            }
        });

        // Share button click
        shareButton.setOnClickListener(v -> {
            String recipeContent = resultText.getText().toString();
            if (!recipeContent.isEmpty()) {
                shareRecipe(recipeContent);
            }
        });

        // New recipe FAB click
        newRecipeFab.setOnClickListener(v -> {
            resetForm();
```

```
        });
    }

    private void startRecipeSearch(String prompt) {
        // UI durumunu güncelle
        submitButton.setEnabled(false);
        showResponseCard();
        startLoadingAnimation();

        sendToGemini(prompt);
    }

    private void showResponseCard() {
        // Input card'ı gizle
        inputCard.setVisibility(View.GONE);

        // Response card'ı tam ekran yap
        responseCard.setVisibility(View.VISIBLE);

        // Layout parametrelerini güncelle
        ConstraintLayout.LayoutParams params =
(ConstraintLayout.LayoutParams) responseCard.getLayoutParams();
        params.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;
        params.topToBottom = ConstraintLayout.LayoutParams.UNSET;
        params.topMargin = 0;
        responseCard.setLayoutParams(params);

        resultText.setVisibility(View.GONE);
        findViewById(R.id.actionButtons).setVisibility(View.GONE);
        newRecipeFab.setVisibility(View.GONE);

        // Card animasyonu
        responseCard.setAlpha(0f);
        responseCard.animate()
                .alpha(1f)
                .setDuration(300)
                .start();
    }

    private void startLoadingAnimation() {
        isAnimating = true;
        loadingProgress.setVisibility(View.VISIBLE);
        animatedText.setVisibility(View.VISIBLE);

        // Animasyon metinleri
        String[] loadingMessages = {
                "✨ Düşünüyorum...",
                "🔍 Malzemeleri analiz ediyorum...",
                "👨‍🍳 Tarifleri hazırlıyorum...",
                "🧠 En iyi kombinasyonları buluyorum...",
                "📝 Önerilerinizi yazıyorum..."
        };

        final int[] currentIndex = {0};

        animationRunnable = new Runnable() {
```

```
            @Override
            public void run() {
                if (isAnimating && currentIndex[0] <
loadingMessages.length) {
                    animateText(loadingMessages[currentIndex[0]]);
                    currentIndex[0]++;
                    animationHandler.postDelayed(this, 2000); // 2
saniye aralık
                } else if (isAnimating) {
                    // Mesajlar bittiğinde döngüye al
                    currentIndex[0] = 0;
                    animationHandler.postDelayed(this, 2000);
                }
            }
        };

        animationHandler.post(animationRunnable);
    }

    private void animateText(String text) {
        // Fade out
        ObjectAnimator fadeOut = ObjectAnimator.ofFloat(animatedText,
"alpha", 1f, 0f);
        fadeOut.setDuration(300);

        fadeOut.addListener(new AnimatorListenerAdapter() {
            @Override
            public void onAnimationEnd(Animator animation) {
                animatedText.setText(text);

                // Fade in
                ObjectAnimator fadeIn =
ObjectAnimator.ofFloat(animatedText, "alpha", 0f, 1f);
                fadeIn.setDuration(300);
                fadeIn.start();
            }
        });

        fadeOut.start();
    }

    private void stopLoadingAnimation() {
        isAnimating = false;
        if (animationHandler != null && animationRunnable != null) {
            animationHandler.removeCallbacks(animationRunnable);
        }

        loadingProgress.setVisibility(View.GONE);
        animatedTextContainer.setVisibility(View.GONE);
    }

    private void showResult(String result) {
        stopLoadingAnimation();

        resultText.setText(result);
        resultText.setVisibility(View.VISIBLE);
        findViewById(R.id.actionButtons).setVisibility(View.VISIBLE);
```

```
        newRecipeFab.setVisibility(View.VISIBLE);

        // Result animation
        resultText.setAlpha(0f);
        resultText.animate()
                .alpha(1f)
                .setDuration(500)
                .start();

        // FAB animation
        newRecipeFab.setScaleX(0f);
        newRecipeFab.setScaleY(0f);
        newRecipeFab.animate()
                .scaleX(1f)
                .scaleY(1f)
                .setDuration(300)
                .setStartDelay(200)
                .start();
    }

    private void sendToGemini(String promptText) {
        Log.d(TAG, "Gönderilen prompt: " + promptText);

        GeminiApiService service = GeminiClient.getService();
        GeminiRequest request = new GeminiRequest(promptText);

        Call<GeminiResponse> call = service.generateContent(request,
API_KEY);

        call.enqueue(new Callback<GeminiResponse>() {
            @Override
            public void onResponse(Call<GeminiResponse> call,
Response<GeminiResponse> response) {
                // Buton tekrar aktif et
                submitButton.setEnabled(true);

                Log.d(TAG, "Response code: " + response.code());

                if (response.isSuccessful() && response.body() != null)
{
                    try {
                        GeminiResponse geminiResponse =
response.body();

                        if (geminiResponse.candidates != null &&
                                !geminiResponse.candidates.isEmpty() &&

geminiResponse.candidates.get(0).content != null &&

geminiResponse.candidates.get(0).content.parts != null &&

!geminiResponse.candidates.get(0).content.parts.isEmpty()) {

                            String answer =
geminiResponse.candidates.get(0).content.parts.get(0).text;
                            Log.d(TAG, "Alınan yanıt: " + answer);
```

```
                            runOnUiThread(() -> showResult(answer));

                        } else {
                            Log.w(TAG, "Yanıt boş geldi");
                            runOnUiThread(() -> {
                                stopLoadingAnimation();
                                showError("Yanıt alınamadı. Lütfen
tekrar deneyin.");
                            });
                        }

                    } catch (Exception e) {
                        Log.e(TAG, "Yanıt işlenirken hata: " +
e.getMessage());
                        runOnUiThread(() -> {
                            stopLoadingAnimation();
                            showError("Yanıt işlenirken hata: " +
e.getMessage());
                        });
                    }
                } else {
                    String errorMsg = "Hata oluştu: " +
response.code();
                    if (response.errorBody() != null) {
                        try {
                            errorMsg += " - " +
response.errorBody().string();
                        } catch (Exception e) {
                            errorMsg += " - " + response.message();
                        }
                    }
                    Log.e(TAG, errorMsg);
                    final String finalErrorMsg = errorMsg;
                    runOnUiThread(() -> {
                        stopLoadingAnimation();
                        showError(finalErrorMsg);
                    });
                }
            }

            @Override
            public void onFailure(Call<GeminiResponse> call, Throwable
t) {
                // Buton tekrar aktif et
                submitButton.setEnabled(true);

                Log.e(TAG, "API hatası: " + t.getMessage());
                String errorMsg = "Bağlantı hatası: " + t.getMessage();

                runOnUiThread(() -> {
                    stopLoadingAnimation();
                    showError(errorMsg);
                    Toast.makeText(Yemektarifilayaout.this, errorMsg,
Toast.LENGTH_LONG).show();
                });
            }
        });
```

```
    }

    private void showError(String errorMessage) {
        resultText.setText("❌ " + errorMessage);
        resultText.setVisibility(View.VISIBLE);
        newRecipeFab.setVisibility(View.VISIBLE);
    }

    private void shareRecipe(String recipe) {
        Intent shareIntent = new Intent(Intent.ACTION_SEND);
        shareIntent.setType("text/plain");
        shareIntent.putExtra(Intent.EXTRA_SUBJECT, "🎯 Akıllı Tarif
Önerisi");
        shareIntent.putExtra(Intent.EXTRA_TEXT,
                "🍽️ AI Şef Önerisi:\n\n" + recipe +
                        "\n\n📱 Foodlyze uygulaması ile oluşturuldu.");

        Intent chooser = Intent.createChooser(shareIntent, "Tarifi
Paylaş");
        startActivity(chooser);
    }

    private void resetForm() {
        // Input card'ı tekrar göster
        inputCard.setVisibility(View.VISIBLE);

        // Response card'ı gizle ve eski konumuna getir
        responseCard.setVisibility(View.GONE);
        ConstraintLayout.LayoutParams params =
(ConstraintLayout.LayoutParams) responseCard.getLayoutParams();
        params.topToTop = ConstraintLayout.LayoutParams.UNSET;
        params.topToBottom = R.id.inputCard;
        params.topMargin = (int) (24 *
getResources().getDisplayMetrics().density); // 24dp
        responseCard.setLayoutParams(params);

        editText.setText("");
        resultText.setVisibility(View.GONE);
        findViewById(R.id.actionButtons).setVisibility(View.GONE);
        newRecipeFab.setVisibility(View.GONE);
        submitButton.setEnabled(true);

        // Animasyonu durdur
        stopLoadingAnimation();

        // Input alanına odaklan
        editText.requestFocus();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (animationHandler != null && animationRunnable != null) {
            animationHandler.removeCallbacks(animationRunnable);
        }
```

```
    }
}
```

YEMEK TARİFİ SAYFASI AÇIKLAMA :

Bu sınıf, **Android** için yazılmış bir yemek tarifi öneri ekranıdır. Kullanıcıdan malzemeleri alır, **Gemini API** ile tarif önerisi alır ve sonucu kullanıcıya animasyonlarla gösterir. İşte `Yemektarifilayaout` sınıfının özeti:

📌 Amaç

Kullanıcının girdiği malzemelere göre yapabileceği yemekleri ve kısa tarifleri **Gemini API** üzerinden alıp gösteren bir ekran sunar.

🔧 Bileşenler (UI öğeleri)

- **editText:** Kullanıcının malzemeleri girdiği alan
- **submitButton:** API çağrısı yapan buton
- **responseCard / inputCard:** Yanıt ve giriş görünümleri
- **animatedTextContainer + loadingProgress:** Yükleniyor animasyonu
- **resultText:** AI'dan gelen tarif sonucu
- **saveButton / shareButton / newRecipeFab:** Kaydetme, paylaşma ve sıfırlama butonları

🔁 Ana Akış

1. Kullanıcı malzemeleri girer ve **Gönder** butonuna tıklar.
2. `startRecipeSearch(prompt)` → Gemini API'ye istek gönderilir.
3. `startLoadingAnimation()` → Yükleme mesajları döner.
4. Yanıt geldiğinde `showResult(result)` → Sonuç gösterilir.
5. Kullanıcı tarifi:
    - **Kaydedebilir** (`saveButton`)
    - **Paylaşabilir** (`shareButton`)
    - **Yeni sorguya geçebilir** (`newRecipeFab`)

ANA SAYFA KODU :

```
import android.content.Intent;
import android.os.Binder;
```

```
import android.os.Bundle;
import android.view.View;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.demirkol.foodlyze.databinding.ActivityAnasayfaBinding;
import com.denzcoskun.imageslider.ImageSlider;
import com.denzcoskun.imageslider.constants.ScaleTypes;
import com.denzcoskun.imageslider.models.SlideModel;

import java.util.ArrayList;

public class Anasayfa extends AppCompatActivity {
    private ActivityAnasayfaBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
      binding=ActivityAnasayfaBinding.inflate(getLayoutInflater());
      setContentView(binding.getRoot());
        ArrayList<SlideModel> slideModels= new ArrayList<>();
        slideModels.add(new SlideModel(R.drawable.reklam,
ScaleTypes.FIT));
        slideModels.add(new SlideModel(R.drawable.foodlyzelogo,
ScaleTypes.FIT));
        slideModels.add(new SlideModel(R.drawable.foodlyzlogo2,
ScaleTypes.FIT));

        binding.imageslider.setImageList(slideModels,ScaleTypes.FIT);


       binding.yemektarifiButton.setOnClickListener(new
View.OnClickListener() {
           @Override
           public void onClick(View view) {
               Intent intent= new Intent(Anasayfa.this,
Yemektarifilayaout.class);
               startActivity(intent);
           }
       });


        binding.scanButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(Anasayfa.this,
Barkodsayfasi.class);
                startActivity(intent);
            }
        });
```

```
    }
}
```

ANA SAYFA KODU AÇIKLAMA :

📱 Amaç:

Kullanıcıya bir **görsel slider** ve iki buton sunan ana ekranı oluşturur.

🔧 Yapılanlar:

1. **View Binding** ile layout (`activity_anasayfa.xml`) bağlanır.
2. **ImageSlider** kullanılarak 3 görsel (reklam ve logolar) ekranda döner şekilde gösterilir.
3. **İki buton** tanımlanır:
    - `yemektarifiButton`: Tıklanınca `Yemektarifilayaout` sayfasına geçer.
    - `scanButton`: Tıklanınca `Barkodsayfasi` sayfasına geçer.

🔁 Kullanılan Kütüphaneler:

- `ImageSlider`: Görsel geçişleri için (`denzcoskun` kütüphanesi).
- `ViewBinding`: XML'deki öğelere erişimi kolaylaştırır.

YARDIMCI DİĞER SINIF KODLARI :

```
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitClient {
    private static Retrofit retrofit;
    private static final String BASE_URL =
"https://world.openfoodfacts.org/";

    public static Retrofit getClient() {
        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                    .baseUrl(BASE_URL)
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();
        }
        return retrofit;
```

```
    }
}
```

```
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;
import okhttp3.OkHttpClient;
import okhttp3.logging.HttpLoggingInterceptor;
import java.util.concurrent.TimeUnit;

public class GeminiClient {
    private static final String BASE_URL =
"https://generativelanguage.googleapis.com/";
    private static Retrofit retrofit = null;

    public static GeminiApiService getService() {
        if (retrofit == null) {
            // Logging interceptor ekle (debug için)
            HttpLoggingInterceptor logging = new
HttpLoggingInterceptor();
            logging.setLevel(HttpLoggingInterceptor.Level.BODY);

            // OkHttp client yapılandırması
            OkHttpClient client = new OkHttpClient.Builder()
                    .addInterceptor(logging)
                    .connectTimeout(30, TimeUnit.SECONDS)
                    .readTimeout(30, TimeUnit.SECONDS)
                    .writeTimeout(30, TimeUnit.SECONDS)
                    .build();

            retrofit = new Retrofit.Builder()
                    .baseUrl(BASE_URL)
                    .client(client)
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();
        }
        return retrofit.create(GeminiApiService.class);
    }
}
```

```
package com.demirkol;

import java.util.Collections;
import java.util.List;

public class GeminiRequest {
    public List<Content> contents;

    public static class Content {
        public List<Part> parts;
```

```
        public Content(String text) {
            this.parts = Collections.singletonList(new Part(text));
        }
    }

    public static class Part {
        public String text;

        public Part(String text) {
            this.text = text;
        }
    }

    public GeminiRequest(String promptText) {
        this.contents = Collections.singletonList(new
Content(promptText));
    }
}
```

```
import java.util.List;

public class GeminiResponse {
    public List<Candidate> candidates;

    public static class Candidate {
        public Content content;
    }

    public static class Content {
        public List<Part> parts;
    }

    public static class Part {
        public String text;
    }
}
```

```
package com.demirkol.foodlyze;

import com.google.gson.annotations.SerializedName;
import java.util.List;

public class ProductResponse {
    @SerializedName("product")
    private Product product;

    public Product getProduct() {
        return product;
    }
```

```
    public class Product {

        // Temel bilgiler

        @SerializedName("product_name")
        private String productName; // Ürün adı



        @SerializedName("fiber")
        private String  liforani; //  lif oranı

        @SerializedName("generic_name")
        private String genericName; // Genel ürün ismi

        @SerializedName("quantity")
        private String quantity; // Miktar (örnek: 250g)

        @SerializedName("code")
        private String code; // Barkod

        // İçerik
        @SerializedName("ingredients_text")
        private String ingredientsText; // İçindekiler (kullanıcının
eklediği)

        @SerializedName("ingredients_text_en")
        private String ingredientsTextEn; // İçindekiler (İngilizce)

        @SerializedName("ingredients_analysis_tags")
        private List<String> ingredientsAnalysisTags; // Örn: en:halal,
en:vegetarian

        @SerializedName("allergens")
        private String allergens; // Alerjenler

        @SerializedName("traces")
        private String traces; // Eser miktarda bulunan maddeler

        @SerializedName("warnings")
        private List<String> warnings; // Uyarılar

        // Etiket ve kategori bilgileri
        @SerializedName("labels")
        private String labels; // Etiketler (metin)

        @SerializedName("labels_tags")
        private List<String> labelTags; // Etiketler (tag formatında)

        @SerializedName("categories")
        private String categories; // Kategoriler (metin)

        @SerializedName("categories_tags")
        private List<String> categoriesTags; // Kategori tag'leri

        @SerializedName("brands")
        private String brands; // Marka adı
```

```
        @SerializedName("brands_tags")
        private List<String> brandsTags; // Marka tag'leri

        @SerializedName("origins")
        private String origins; // Köken bilgisi


        @SerializedName("trans-fat")
        private String trans_yag; // Doymuş Yağ

        public String getTrans_yag() {
            return trans_yag;
        }



        // Görseller
        @SerializedName("image_url")
        private String imageUrl; // Genel resim

        @SerializedName("image_small_url")
        private String imageSmallUrl; // Küçük boyutlu resim

        @SerializedName("image_front_url")
        private String productImage; // Ön yüz görseli

        @SerializedName("image_ingredients_url")
        private String ingredientsImage; // İçerik görseli

        @SerializedName("image_nutrition_url")
        private String nutritionImage; // Besin tablosu görseli

        @SerializedName("image_thumb_url")
        private String thumbnailImage; // Küçük önizleme resmi

        // Besin bilgileri
        @SerializedName("nutriscore_grade")
        private String nutriscoreGrade; // Nutriscore harfi (A-E)

        @SerializedName("nutriscore_score")
        private int nutriscoreScore; // Nutriscore puanı

        @SerializedName("nova_group")
        private int novaGroup; // Nova işlenmişlik skoru (1-4)

        @SerializedName("ecoscore_grade")
        private String ecoscoreGrade; // Eco-Score (çevresel etki
puanı)

        @SerializedName("nutriments")
        private Nutriments nutriments; // Besin değerleri

        // Üretici / kaynak
        @SerializedName("manufacturing_places")
        private String manufacturingPlaces; // Üretim yerleri
```

```
        @SerializedName("emb_codes")
        private String embCodes; // Üretici kodları

        @SerializedName("packaging")
        private String packaging; // Ambalaj türü

        @SerializedName("stores")
        private String stores; // Satıldığı mağazalar

        @SerializedName("purchase_places")
        private String purchasePlaces; // Satın alındığı yerler

        @SerializedName("countries")
        private String countries; // Ürünün bulunduğu ülkeler

        @SerializedName("countries_tags")
        private List<String> countriesTags; // Ülke tag'leri

        // Getter'lar (mevcut olanlar ve yeniler)
        public String getProductName() { return productName; }
        public String getGenericName() { return genericName; }
        public String getIngredientsText() { return ingredientsText; }
        public String getIngredientsTextEn() { return
ingredientsTextEn; }
        public String getAllergens() { return allergens; }
        public List<String> getWarnings() { return warnings; }
        public String getLabels() { return labels; }
        public List<String> getLabelTags() { return labelTags; }
        public String getImageUrl() { return imageUrl; }
        public String getImageSmallUrl() { return imageSmallUrl; }
        public String getProductImage() { return productImage; }
        public String getIngredientsImage() { return ingredientsImage;
}
        public String getNutritionImage() { return nutritionImage; }
        public String getThumbnailImage() { return thumbnailImage; }
        public String getCategories() { return categories; }
        public List<String> getCategoriesTags() { return
categoriesTags; }
        public String getBrands() { return brands; }
        public List<String> getBrandsTags() { return brandsTags; }
        public String getOrigins() { return origins; }
        public String getQuantity() { return quantity; }
        public Nutriments getNutriments() { return nutriments; }
        public String getNutriscoreGrade() { return nutriscoreGrade; }
        public int getNutriscoreScore() { return nutriscoreScore; }
        public int getNovaGroup() { return novaGroup; }
        public String getEcoscoreGrade() { return ecoscoreGrade; }
        public String getManufacturingPlaces() { return
manufacturingPlaces; }
        public String getEmbCodes() { return embCodes; }
        public String getPackaging() { return packaging; }
        public String getStores() { return stores; }
        public String getPurchasePlaces() { return purchasePlaces; }
        public String getCountries() { return countries; }
        public List<String> getCountriesTags() { return countriesTags;
}
        public String getCode() { return code; }
```

```
        public List<String> getIngredientsAnalysisTags() { return
ingredientsAnalysisTags; }
        public String getLiforani() {
            return liforani;
        }



    }
}
```

```
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Path;

public interface OpenFoodFactsAPI {

    // Barkod numarasına göre ürün verilerini al
    @GET("api/v0/product/{barcode}.json")
    Call<ProductResponse> getProductDetails(@Path("barcode") String
barcode);
}
```