✅ User Stories

User Story 1: Ortak Para Havuzu Oluşturma

- **Başlık:** Ortak Para Havuzu Oluşturma
- **Kullanıcı Tipi:** Grup Lideri
- **İhtiyaç:** Grup lideri olarak, grup seyahatinde kullanılacak ortak bir para havuzu oluşturmak istiyorum.
- **Fayda:** Böylece, grup üyelerinin tüm seyahat harcamalarını tek bir havuzdan karşılayabilir ve yönetebilirim.
- **Kabul Kriterleri:**

1. Grup lideri, TKpay uygulamasına giriş yapar.
2. "Ortak Para Havuzu Oluştur" seçeneğine tıklar.
3. Havuz adı, başlangıç bakiyesi (her üyenin havuza yapacağı minimum katkı miktarı) ve para birimi gibi bilgileri girer.
4. Grup lideri, ortak para havuzuna ekleyeceği kişilere davet linki gönderir. (Davet linki kullanıcının uygulamaya kaydolurken seçtiği iletişim yolu ile iletilir.)
5. Davet linkleri belirlenen yöntemle grup üyelerine gönderilir ve havuz oluşturulur.
6. Eğer 7 gün içerisinde ortak para havuzuna grup lideri dışında kimse katılmazsa, ortak para havuzu otomatik olarak kapatılır. Havuz oluşturulduktan sonra, grup liderine ve davet edilen üyelere kapanma süresi hakkında bilgi verilir.
7. Davet edilen üyeler, 24 saat içinde daveti kabul eder. Süre dolduğunda hatırlatma bildirimleri gönderilir (x tarafından gönderilen “ortak para havuzu” teklifini kabul etmediniz.”
8. Daveti kabul etmeyen üyeler grup liderine bildirilir. Grup lideri tekrar davet gönderebilir.
9. Havuz oluşturulduktan sonra, tüm grup üyeleri havuzun detayları hakkında bilgilendirilir (adı, başlangıç bakiyesi, para birimi).

User Story 2: Havuz Katılımı ve Para Yükleme

- **Başlık:** Havuz Katılımı ve Para Yükleme
- **Kullanıcı Tipi:** Grup Üyesi
- **İhtiyaç:** Davet edildiğim ortak para havuzuna katılmak ve belirli bir miktar para yüklemek istiyorum.
- **Fayda:** Böylece, grup seyahatinde yapılacak ortak harcamalara katkıda bulunabilir ve takip edebilirim.
- **Kabul Kriterleri:**

1. Grup üyesi, TKpay uygulamasına davet linki (SMS veya e-posta) alır.
2. Davet linkine tıklayarak TKpay uygulamasını açar veya indirir.
3. TKpay uygulamasında daveti kabul eder ve havuza katılır.
4. Ortak para havuzuna katıldıktan sonra "Para Yükle" seçeneğini seçer.
5. Yüklenecek miktarı ve ödeme yöntemini (kredi kartı, banka transferi, vb.) belirler.
6. Para yükleme işlemi başarıyla tamamlanır ve havuz bakiyesi güncellenir.
7. Para yükleme işlemi tamamlandığında, hem grup üyesine hem de grup liderine bildirim gönderilir.

2.1 Kredi Kartı/Banka Kartı ile Para Yükleme

Adımlar:

1. **Ödeme Yöntemi Seçimi:**
   - Kullanıcı, "Para Yükle" ekranında kredi kartı veya banka kartı seçeneğini seçer.
2. **Kart Bilgilerinin Girilmesi:**
   - Kullanıcı, kart numarası, son kullanma tarihi, CVV kodu ve kart sahibinin adı gibi bilgileri girer.
3. **Yükleme Miktarının Belirlenmesi:**

- Kullanıcı, yüklemek istediği miktarı girer.

4. **Ödeme Onayı:**
   - Kullanıcı, "Yükleme İşlemini Onayla" butonuna tıklar.
5. **Ödeme İşlemi:**
   - Sistem, ödeme ağ geçidi (örneğin Stripe, PayPal, yerel bir ödeme sağlayıcısı) üzerinden ödeme işlemini gerçekleştirir.
   - Ödeme ağ geçidi, kart bilgilerini doğrular ve işlemi onaylar.
6. **Bildirim:**
   - Ödeme başarılı olursa, kullanıcıya ve grup liderine bildirim gönderilir.
   - Ödeme başarısız olursa, kullanıcıya işlemin başarısız olduğu ve tekrar denemesi gerektiği bildirilir.

2.2 TKpay Cüzdanından Ortak Havuza Para Yükleme

Adımlar:

1. **Cüzdan Bakiyesinin Kontrolü:**
   - Kullanıcı, TKpay cüzdanındaki bakiyesini kontrol eder.
2. **Yükleme Miktarının Belirlenmesi:**
   - Kullanıcı, TKpay cüzdanından ortak havuza aktarılacak miktarı belirler. Eğer, aktaracağı miktar TKpay cüzdanındaki miktardan fazla ise *“Yeterli bakiyeniz bulunmamaktadır.”* uyarısı verilir ve işlem yapılamaz.
3. **Transfer Onayı:**
   - Kullanıcı, *"Transferi Onayla"* butonuna tıklar.
4. **Transfer İşlemi:**
   - Sistem, TKpay cüzdanındaki bakiyeden belirtilen miktarı ortak havuza aktarır.
5. **Bildirim:**
   - Transfer başarılı olursa, kullanıcıya ve grup liderine bildirim gönderilir.
   - Transfer başarısız olursa, kullanıcıya işlemin başarısız olduğu ve tekrar denemesi gerektiği bildirilir.

Teknik Analiz: TKpay Ödeme Süreci ve QR Kod Entegrasyonu

Genel Yapı ve Süreç

- **Ödeme Talebi Oluşturma**
- **Ödeme Onayı ve Gerçekleştirme**
- **Harcama Takibi ve Kategorilendirme**
- **Bildirimler ve Geri Bildirim**

1. Ödeme Talebi Oluşturma

Adım 1.1: Kullanıcı, TKpay uygulamasına giriş yapar ve ortak para havuzunu seçer.

- **Gereken Bilgiler:** Kullanıcı kimlik doğrulaması, seçilen ortak havuz bilgisi.

Adım 1.2: "Ödeme Yap" seçeneğine tıklar ve ödeme yapmak istediği iş yerinin kategorisini(otel, araç kiralama, uçak bileti vs.) seçer.

- **Gereken Bilgiler:** Kategori adı *(otel, araç kiralama, uçak bileti, restoran, vb.), Kategori ID'si (veritabanında tanımlanmış kategorilere göre)*, İş Yeri Bilgileri: İş yeri adı, İş yeri ID'si *(veritabanında kayıtlı iş yerleri için)*, İş yeri IBAN'ı *(sistem tarafından otomatik olarak atanır, kullanıcıya görünmez)*

Adım 1.3: Ödeme yapmak istediği iş yerini seçer.

- **Gereken Bilgiler:** İş yeri bilgileri (kayıtlı iş yeri ID'si), İş yeri IBAN'ı *(sistem tarafından otomatik olarak atanır, kullanıcıya görünmez)*.

Adım 1.4: Ödeme detaylarını girer.

- **Gereken Bilgiler:** Miktar.

Adım 1.5: Ödeme talebi oluşturulur.

2. Ödeme Onayı ve Gerçekleştirme

Adım 2.2: Ödeme işlemi başlatılır.

- **Gereken Bilgiler:** İşlem başlatma durumu.

Adım 2.3: TKpay'in ödeme ağ geçidi üzerinden ödeme gerçekleştirilir.

Alt Adım 2.3.1: Ödeme Ağ Geçidi İşlemleri

- **Gereken Bilgiler:** Ödeme bilgileri (from, to, tutar).
- **İşlem:** QR kod üzerinden gelen bilgilerin doğrulanması.

```
public class QRPaymentProcessor {
  public static void main(String[] args) {
     String fromAccount = "123456";
      String toAccount = "654321";
      double amount = 100.0;

     if (validateQRData(fromAccount, toAccount, amount)) {
        initiatePayment(fromAccount, toAccount, amount);
     } else {
          sendErrorNotification("Ödeme bilgileri doğrulanamadı.");
     }
  }

  public static boolean validateQRData(String fromAccount, String toAccount, double amount) {
      // QR verisini doğrula
      // Gerçek duruma göre bu fonksiyonu düzenleyin
     return true; // Bu örnekte, doğrulama başarılı olarak varsayılıyor
  }

  public static void initiatePayment(String fromAccount, String toAccount, double amount) {
      // Ödemeyi başlat
     System.out.println("Ödeme başlatıldı: " + amount + " TL, " + fromAccount + " hesabından " + toAccount + " h
  }

  public static void sendErrorNotification(String message) {
      // Hata bildirimi gönder
     System.out.println("Hata bildirimi gönderildi: " + message);
  }
}
```

Alt Adım 2.3.2: Doğrulama Başarılıysa Ödeme İşlemi

- **Gereken Bilgiler:** Onaylanmış ödeme bilgileri.
- **İşlem:** Ödeme işleminin gerçekleştirilmesi.

```
public class PaymentHandler {
  public static void main(String[] args) {
      String fromAccount = "123456";
      String toAccount = "654321";
      double amount = 100.0;
      if (paymentSuccessful()) {
          recordTransaction(fromAccount, toAccount, amount);
          sendSuccessNotification("Ödeme başarılı.");
      } else {
```

```
            reversePayment(fromAccount, toAccount, amount);
            sendErrorNotification("Ödeme başarısız, geri ödeme yapıldı.");
        }
    }
    public static boolean paymentSuccessful() {
        // Ödemenin başarılı olup olmadığını kontrol et
        // Bu örnek kodda, ödeme başarılı olarak varsayılıyor
        return true; // Gerçek duruma göre bu değeri değiştirin
    }
    public static void recordTransaction(String fromAccount, String toAccount, double amount) {
        // İşlemi kaydet
        System.out.println("İşlem kaydedildi: " + amount + " TL, " + fromAccount + " hesabından " + toAccount + "
    }
    public static void sendSuccessNotification(String message) {
        // Başarı bildirimi gönder
        System.out.println("Başarı bildirimi gönderildi: " + message);
    }
    public static void reversePayment(String fromAccount, String toAccount, double amount) {
        // Ödemeyi geri al
        System.out.println("Geri ödeme yapıldı: " + amount + " TL, " + toAccount + " hesabından " + fromAccount +
    }
    public static void sendErrorNotification(String message) {
        // Hata bildirimi gönder
        System.out.println("Hata bildirimi gönderildi: " + message);
    }
}
```

Alt Adım 2.3.3: İşlem Başarılı Bildirimi

- **Gereken Bilgiler:** Başarılı işlem bilgileri.
- **İşlem:** Kullanıcıya bildirim gönderme.

`send_success_notification("Ödeme başarılı.")`

Alt Adım 2.3.4: İşlem Başarısız Bildirimi ve Geri Ödeme

- **Gereken Bilgiler:** Başarısız işlem bilgileri.
- **İşlem:** Kullanıcıya hata bildirimi gönderme ve geri ödeme işlemi.

```
public class PaymentHandler {
    public static void main(String[] args) {
        String fromAccount = "123456";
        String toAccount = "654321";
        double amount = 100.0;
        if (paymentFailed()) {
            reversePayment(fromAccount, toAccount, amount);
            sendErrorNotification("Ödeme başarısız, geri ödeme yapıldı.");
        }
    }
    public static boolean paymentFailed() {
        // Ödemenin başarısız olup olmadığını kontrol et
        // Bu örnek kodda, ödeme başarısız olarak varsayılıyor
        return true; // Gerçek duruma göre bu değeri değiştirin
    }
    public static void reversePayment(String fromAccount, String toAccount, double amount) {
        // Ödemeyi geri al
        System.out.println("Geri ödeme yapıldı: " + amount + " TL, " + toAccount + " hesabından " + fromAccount +
    }
```

```
  public static void sendErrorNotification(String message) {
      // Hata bildirimi gönder
      System.out.println("Hata bildirimi gönderildi: " + message);
  }
}
```

Ek Özellikler ve Gereksinimler

- **QR Kod İle Ödeme:** QR kod taraması ile ödeme işlemi başlatılabilir.
  - **İşlem:** QR kod verilerinin işlenmesi ve ödeme ağ geçidine iletilmesi.

```
public class QRPaymentProcessor {
  public static void main(String[] args) {
      String qrData = "sampleQRData";
     processQRPayment(qrData);
  }
  public static void processQRPayment(String qrData) {
      String[] parsedData = parseQRData(qrData);
      String fromAccount = parsedData[0];
      String toAccount = parsedData[1];
      double amount = Double.parseDouble(parsedData[2]);
      if (validateQRData(fromAccount, toAccount, amount)) {
          initiatePayment(fromAccount, toAccount, amount);
      } else {
          sendErrorNotification("QR kod bilgileri doğrulanamadı.");
      }
  }
  public static String[] parseQRData(String qrData) {
      // QR verisini ayrıştır
      // Bu örnekte, veriyi hardcoded olarak ayrıştırıyorum. Gerçek duruma göre bu fonksiyonu düzenleyin.
      return new String[] {"123456", "654321", "100.0"};
  }
  public static boolean validateQRData(String fromAccount, String toAccount, double amount) {
      // QR verisini doğrula
      return true; // Gerçek duruma göre bu değeri değiştirin
  }
  public static void initiatePayment(String fromAccount, String toAccount, double amount) {
      // Ödemeyi başlat
      System.out.println("Ödeme başlatıldı: " + amount + " TL, " + fromAccount + " hesabından " + toAccount + "
  }
  public static void sendErrorNotification(String message) {
      // Hata bildirimi gönder
      System.out.println("Hata bildirimi gönderildi: " + message);
  }
}
```

- **Sadece Kayıtlı İş Yerlerine Ödeme:** Ödemeler yalnızca sistemde kayıtlı iş yerlerine yapılabilir.
  - **İşlem:** İş yeri bilgileri doğrulaması.

```
public class BusinessPaymentProcessor {
  public static void main(String[] args) {
      String fromAccount = "123456";
      String toAccount = "654321";
      double amount = 100.0;
```

```
      if (isRegisteredBusiness(toAccount)) {
          initiatePayment(fromAccount, toAccount, amount);
      } else {
          sendErrorNotification("Geçersiz iş yeri.");
      }
  }
  public static boolean isRegisteredBusiness(String toAccount) {
      // İş yerinin kayıtlı olup olmadığını kontrol et
      // Bu örnekte, iş yeri kayıtlı olarak varsayılıyor
      return true; // Gerçek duruma göre bu değeri değiştirin
  }
  public static void initiatePayment(String fromAccount, String toAccount, double amount) {
      // Ödemeyi başlat
      System.out.println("Ödeme başlatıldı: " + amount + " TL, " + fromAccount + " hesabından " + toAccount + "
  }
  public static void sendErrorNotification(String message) {
      // Hata bildirimi gönder
      System.out.println("Hata bildirimi gönderildi: " + message);
  }
}
```

- **Başarısız İşlemde Geri Ödeme:** Başarısız işlemlerde para, from hesabına geri döner.
  - **İşlem:** Geri ödeme işlemi ve bildirim gönderme.

```
public class PaymentProcessor {
  public static void main(String[] args) {
     String fromAccount = "123456";
     String toAccount = "654321";
      double amount = 100.0;
     if (paymentFailed()) {
          reversePayment(fromAccount, toAccount, amount);
          sendErrorNotification("Ödeme başarısız, geri ödeme yapıldı.");
      }
  }
  public static boolean paymentFailed() {
     // Ödemenin başarısız olup olmadığını kontrol et
     // Bu örnek kodda, ödeme başarısız olarak varsayılıyor
      return true; // Gerçek duruma göre bu değeri değiştirin
  }
  public static void reversePayment(String fromAccount, String toAccount, double amount) {
      // Ödemeyi geri al
     System.out.println("Geri ödeme yapıldı: " + amount + " TL, " + toAccount + " hesabından " + fromAccount + "
  }
  public static void sendErrorNotification(String message) {
     // Hata bildirimi gönder
     System.out.println("Hata bildirimi gönderildi: " + message);
  }
}
```

User Story 3: Harcama Takibi ve Kategorilendirme

- **Başlık:** Harcama Takibi ve Kategorilendirme
- **Kullanıcı Tipi:** Grup Üyesi

- **İhtiyaç:** Yapılan harcamaları gerçek zamanlı olarak takip etmek ve kategorilere ayırmak istiyorum.
- **Fayda:** Böylece, harcamaların nereye gittiğini görebilir ve harcamaları daha iyi yönetebilirim.
- **Kabul Kriterleri:**

1. Grup üyesi, TKpay uygulamasına giriş yapar.
2. Ortak para havuzunun harcama geçmişini görüntüler.
3. Harcamalar kategorilere ayrılmış şekilde listelenir (otel, araç kiralama, yemek, vb.).
4. Her harcama detaylı bilgilerle (tarih, miktar, kategori, harcama yapan üye) birlikte gösterilir.
5. Grup üyesi, her harcamanın üzerine tıklayarak dekontu görüntüleyebilir.
6. Harcama dekontları, PDF veya resim formatında indirilebilir ve saklanabilir.
7. Her harcama gerçek zamanlı olarak güncellenir ve tüm grup üyeleri tarafından görülebilir.
8. Her harcama yapıldığında veya güncellendiğinde grup üyelerine bildirim gönderilir.
9. Harcama geçmişinde tarih, kategori veya harcama yapan üye gibi kriterlere göre filtreleme ve arama yapılabilir. (optional)

User Story 4: Geri Ödeme ve Denklik

- **Başlık:** Geri Ödeme ve Denklik
- **Kullanıcı Tipi:** Grup Üyesi
- **İhtiyaç:** Seyahatin sonunda, grup üyeleri arasındaki harcama farklarını denkleştirmek istiyorum.
- **Fayda:** Böylece, seyahat sonunda kimse mağdur olmaz ve herkes adil bir şekilde payını öder.
- **Kabul Kriterleri:**

1. Grup üyesi, TKpay uygulamasına giriş yapar.
2. Seyahat sonunda menüden “Katıldığım Havuzlar” sekmesine girer.
3. Geri ödeme alacağı havuzu seçer.
4. "Geri Ödeme ve Denklik" seçeneğine tıklar.
5. Harcama raporları incelenir ve denklik hesaplanır.
6. Kalan miktar, kullanıcıların başlangıçtaki katkı oranlarına göre geri ödenir.
7. Gereken geri ödemeler belirlenir ve işlemler gerçekleştirilir.
8. TKpay uygulaması, otomatik olarak grup üyeleri arasındaki geri ödeme işlemlerini başlatır.
9. Tüm grup üyeleri, geri ödeme durumunu ve denklik süreçlerini takip edebilir.

Senaryo 4.1: Harcamalar Ortak Havuzun Altında Kalırsa

- **Kullanıcı Katkıları ve Harcamalar:**
  - **Kullanıcı A:**
    - Katkı: 600 dolar
    - Harcama: 400 dolar
  - **Kullanıcı B:**
    - Katkı: 300 dolar
    - Harcama: 200 dolar
  - **Kullanıcı C:**
    - Katkı: 100 dolar
    - Harcama: 100 dolar
- **Toplam Katkı:** 600 + 300 + 100 = 1000 dolar
- **Toplam Harcama:** 400 + 200 + 100 = 700 dolar
- **Kalan Miktar:** 1000 - 700 = 300 dolar

Bu durumda, ortak havuzda kalan miktar (300 dolar) katkı oranlarına göre geri ödenir.

- **Kullanıcı A'nın Katkı Oranı:** 600 / 1000 = %60
- **Kullanıcı B'nin Katkı Oranı:** 300 / 1000 = %30
- **Kullanıcı C'nin Katkı Oranı:** 100 / 1000 = %10

Kalan miktarın (300 dolar) katkı oranlarına göre dağıtılması:

- **Kullanıcı A'nın Geri Ödeme Miktarı:** 300 * %60 = 180 dolar
- **Kullanıcı B'nin Geri Ödeme Miktarı:** 300 * %30 = 90 dolar
- **Kullanıcı C'nin Geri Ödeme Miktarı:** 300 * %10 = 30 dolar

User Story 5: Ek Para Yükleme Bildirimi

Başlık: Ek Para Yükleme Bildirimi

Kullanıcı Tipi: Grup Üyesi

İhtiyaç: Harcama yapılmak istendiğinde, ortak havuzdaki miktar yetersiz olduğunda eksik miktarı tamamlamak için bildirim almak istiyorum.

Fayda: Böylece, ortak harcamaların kesintisiz ve sorunsuz bir şekilde yapılmasını sağlayabilir ve eksik kalan miktarı tamamlayarak işlemi gerçekleştirebilirim.

Kabul Kriterleri:

1. Harcama yapılmak istenen işlem, ortak havuzdaki miktarı aşarsa, eksik miktar otomatik olarak hesaplanır.
2. Eksik miktar hakkında tüm grup üyelerine otomatik bir bildirim gönderilir.
3. Bildirimde yapılacak harcamanın miktarı ve eksik kalan miktar belirtilir.
4. Grup üyeleri, eksik miktarı tamamlamak için havuza para yükler.
5. Eksik miktar tamamlandığında, işlem gerçekleştirilir.
6. Eksik miktar tamamlanmazsa, harcama işlemi gerçekleştirilmez ve kullanıcılar bilgilendirilir.

User Story 6: Grup Üyesinin Havuzdan Çıkışı

- **Başlık:** Ortak Para Havuzundan Çıkış
- **Kullanıcı Tipi:** Grup Üyesi
- **İhtiyaç:** Bir grup üyesi olarak, seyahatten vazgeçtiğimde ortak para havuzundan çıkmak ve katkı yaptığım parayı adil bir şekilde geri almak istiyorum.
- **Fayda:** Böylece, kişisel kararlarım doğrultusunda esneklik sağlayabilir ve adil bir şekilde paramı geri alabilirim.
- **Kabul Kriterleri:**

1. Grup üyesi, TKpay uygulamasına giriş yapar.
2. "Katıldığım Havuzlar" menüsüne gider.
3. Çıkmak istediği havuzu seçer.
4. "Havuzdan Çık" seçeneğine tıklar.
5. Uygulama, grup üyesinin çıkışı onaylamasını ister.
6. Grup üyesi çıkışı onaylar.
7. Havuzdan çıkış süreci başlatılır.
8. Grup liderine ve diğer grup üyelerine, üyenin havuzdan çıktığına dair bildirim gönderilir.
9. Grup üyesinin katkı yaptığı para ve harcamalar dikkate alınarak adil bir geri ödeme hesaplanır.
10. Grup üyesine, geri ödeme miktarı ve hesaplama detayları bildirilir.
11. Geri ödeme işlemi gerçekleştirilir ve havuz bakiyesi güncellenir.
12. Yeni havuz durumu tüm grup üyelerine bildirilir.

User Story 7: Seyahat Sonrası veya İptal Durumunda Havuzun Kapanması

- **Başlık:** Seyahat Sonrası veya İptal Durumunda Havuzun Kapanması
- **Kullanıcı Tipi:** Grup Lideri
- **İhtiyaç:** Grup lideri olarak, seyahat tamamlandıktan sonra veya seyahat iptal edildiğinde ortak para havuzunun kapanmasını ve kalan bakiyenin adil bir şekilde dağıtılmasını istiyorum.
- **Fayda:** Böylece, seyahat bittikten veya iptal edildikten sonra gereksiz beklemelerden kaçınarak havuzun kapatılmasını ve paraların iade edilmesini sağlayabilirim.
- **Kabul Kriterleri:**

1. Grup lideri, TKpay uygulamasına giriş yapar.
2. Seyahatin bitiş tarihi geldiğinde veya seyahat iptal edildiğinde havuzun kapanması için seçenek belirler.
3. Seyahat sonunda veya iptal durumunda, grup üyelerine harcamalarını tamamlamaları için hatırlatma bildirimleri gönderilir.
4. Grup lideri, "Havuzu Kapat" seçeneğini seçer.
5. Harcamalar düzenlenir ve kalan bakiye hesaplanır.
6. Kalan bakiye, grup üyelerine adil bir şekilde dağıtılır veya iade edilir.
7. Grup liderine ve grup üyelerine, havuzun kapandığı ve bakiyelerin dağıtıldığına dair bildirim gönderilir.

User Story 8: Havuz Katılımı Olmazsa Kapanma Süreci

- **Başlık:** Havuz Katılımı Olmazsa Kapanma Süreci
- **Kullanıcı Tipi:** Grup Lideri
- **İhtiyaç:** Grup lideri olarak, davet ettiğim grup üyeleri havuza katılmazsa havuzun otomatik olarak kapanmasını ve katkı yaptığım paranın geri iade edilmesini istiyorum.
- **Fayda:** Böylece, havuza kimse katılmadığında gereksiz beklemelerden kaçınarak katkı yaptığım paranın geri iadesini alabilirim.
- **Kabul Kriterleri:**

1. Grup lideri, TKpay uygulamasına giriş yapar.
2. "Ortak Para Havuzu Oluştur" seçeneğine tıklar ve havuz detaylarını girer.
3. Grup lideri, grup üyelerini davet eder.
4. Grup lideri, davet gönderildikten sonra 7 gün boyunca üyelerin katılımını bekler.
5. Belirlenen süre içinde havuza hiçbir üye katılmazsa, uygulama havuzun otomatik olarak kapanacağını bildirir.
6. Süre sonunda havuza kimse katılmamışsa, havuz otomatik olarak kapanır.
7. Grup liderine ve davet edilen üyelere havuzun kapandığına dair bildirim gönderilir.
8. Grup liderinin katkı yaptığı para, otomatik olarak geri iade edilir.
9. Havuz kapanışı ve para iadesi işlemi tamamlanır.