Pi Course — Mini Full-Stack Görevi

Django/DRF + Flutter

Hazırlayan: Pi Course Teknik Ekip
Sürüm: v1.0
Tarih: August 11, 2025

Amaç

Pi Course için öğrencilerin eğitmenleri bulup ders talebi oluşturabildiği ufak bir **MVP** geliştirin. Backend **Django + DRF**, mobil istemci **Flutter** ile yazılacaktır. Amaç; uçtan uca akışı, temiz mimariyi, test kültürünü ve ürün odaklı düşünmeyi gözlemlemek.

Teslim İçeriği (Özet)

- Django + DRF ile çalışan bir API
- Flutter ile çalışan basit bir mobil uygulama (Android veya iOS—biri yeterli)
- Kısa bir **README** (kurulum + kullanım)
- En azından kritik akışları kapsayan **otomatik testler**
- Kısa bir **Mimari Notlar** bölümü (niçin böyle tasarladınız?)

Önerilen süre: 6–8 saat. MVP odaklı kalın; "Artı Puanlar" opsiyoneldir.

1 MVP Kapsamı

1.1 Backend (Django + DRF)

Veri Modeli (Öneri)

- `User` (`AbstractUser'dan`): `role` = `student` | `tutor`
- `Subject`: `name`
- `TutorProfile`: `user (OneToOne)`, `bio`, `hourly_rate`, `rating (0-5)`, `subjects (M2M Subject)`
- `StudentProfile`: `user (OneToOne)`, `grade_level (str | optional)`
- `LessonRequest`: `student (FK User)`, `tutor (FK User)`, `subject (FK Subject)`, `start_time (DateTime, ISO8601 UTC)`, `duration_minutes (int)`, `status: pending | approved | rejected`, `created_at`

Zorunlu Uç Noktalar

- `POST /api/auth/register` — kullanıcı rolü ile kayıt (student/tutor)
- `POST /api/auth/login` — JWT (örn. drf-simplejwt)
- `GET /api/me` — oturum açan kullanıcının profil verisi
- `PATCH /api/me` — profil güncelleme (örn. bio, grade_level)
- `GET /api/subjects` — basit liste
- `GET /api/tutors?subject=<id>&ordering=-rating&search=<q>` — filtre + arama + sıralama

- `GET /api/tutors/{id}` — detay
- `POST /api/lesson-requests` — **yalnızca** öğrenci oluşturabilir
- `GET /api/lesson-requests?role=student|tutor&status=pending|approved|rejected` — oturum sahibinin ilgili talepleri
- `PATCH /api/lesson-requests/{id}` — **yalnızca** ilgili eğitmen `approved/rejected` yapabilir

Kurallar / Güvenlik

- JWT zorunlu (read-only public uçlar hariç).
- Rol bazlı izinler (student yeni talep açar; tutor talebi onaylar/reddeder; diğerleri 403).
- Girdi doğrulama ve anlamlı hata mesajları.
- Sayfalama (DRF `LimitOffsetPagination` yeterli).

Doküman tasyon & Tohum Veri

- OpenAPI şeması (örn. `drf-spectacular`) ve basit Swagger UI (`/api/docs`).
- Basit seed: 3–5 tutor, 2–3 subject, 1–2 student. (Management command veya fixture.)

Testler (Minimum)

- Kayıt + giriş akışı (happy path).
- Rol izinleri (403 senaryoları).
- Derse talep akışı: create (student), approve/reject (tutor).

1.2 Mobil (Flutter)

Zorunlu Ekranlar/Akış

- **Giriş/Kayıt**: Öğrenci/Eğitmen rol seçimiyle kayıt, JWT ile giriş.
- **Eğitmen Listesi**: Filtre (Subject), Arama (Ad/Bio), Sıralama (Rating).
- **Eğitmen Detayı**: Bilgiler + "Ders Talep Et" butonu.
- **Ders Talebi Oluştur**: Subject seçimi, tarih/saat, süre, opsiyonel not.
- **Taleplerim**: Öğrenci görünümü (kendi talepleri); Eğitmen görünümü (gelen talepler + onay/ret).

Teknik Tercihler

- State management: **Riverpod** veya **Bloc** (README'de kısaca gerekçelendiriniz).
- HTTP istemci: `dio` veya `http`.
- Model/JSON: `json_serializable` veya `freezed` + `json_serializable`.
- Hata yönetimi: Ağ ve doğrulama hatalarında kullanıcıya anlaşılır geri bildirim.
- Token saklama: `flutter_secure_storage` (veya eşdeğeri).
- Basit responsive ve temiz UI; piksel mükemmel olmasına gerek yok.

Minimum Kalite Barı

- Servis katmanı ayrımı (`api_client`, `repositories`, `features`).
- Boş/Loading/Hata durumlarına uygun UI.

2 Beklenen API Örnekleri

Kayıt

```
POST /api/auth/register
{
  "email": "ali@picourse.com",
  "password": "Passw0rd!",
  "role": "student"
}
```

Login (JWT)

```
POST /api/auth/login
{
  "email": "ali@picourse.com",
  "password": "Passw0rd!"
}
-- response --
{
  "access": "<jwt>",
  "refresh": "<jwt>"
}
```

Eğitmen Listesi

```
GET /api/tutors?subject=2&ordering=-rating&search=physics
-- response (200) --
{
  "count": 2,
```

```
  "results": [
    {
      "id": 5,
      "name": "Dr. Ay e Demir",
      "subjects": [{"id":2,"name":"Physics"}],
      "hourly_rate": 500,
      "rating": 4.8,
      "bio": "ODT  fizik doktora..."
    }
  ]
}
```

Ders Talebi Oluşturma (student)

```
POST /api/lesson-requests
Authorization: Bearer <access>
{
  "tutor_id": 5,
  "subject_id": 2,
  "start_time": "2025-08-15T10:00:00Z",
  "duration_minutes": 60,
  "note": "Kuantum giri "
}
```

Talebi Onaylama (tutor)

```
PATCH /api/lesson-requests/12
Authorization: Bearer <access>
{ "status": "approved" }
```

3 Değerlendirme Kriterleri

- **Doğru Çalışan Akışlar (backend+mobil) – 30%**
- **Kod Kalitesi & Mimari – 20%**
- **Testler – 15%**
- **API Tasarımı & Güvenlik – 15%**
- **UI/UX ve Hata Yönetimi – 10%**
- **Dokümantasyon & Kurulum Kolaylığı – 10%**

4 Teslimat Kuralları

- **Repo**: Tek repo (monorepo) veya iki repo (backend / mobile) — tercihinizi README'de belirtin.

- **Kurulum**:
  - Backend: Sanal ortam, bağımlılıklar, migrate, seed, çalıştırma.
  - Mobil: Flutter versiyonu, env/base URL ayarı, çalıştırma.
- Test komutları ve beklenen çıktı örnekleri.
- Örnek **demo hesapları** (örn. `student@demo.com` / `tutor@demo.com` + şifre).

5 Artı Puan (Opsiyonel)

- Docker Compose (web + db) ve `make` kısayolları.
- `drf-spectacular` ile düzgün OpenAPI ve Swagger UI.
- Basit rate limiting veya throttling (talep oluşturma).
- Backend'de `select_related` / `prefetch_related` ile N+1 önleme.
- Flutter'da **pull-to-refresh**, **sonsuz kaydırma** (pagination).
- Basit CI (GitHub Actions) ile testlerin otomatik çalışması.

6 Notlar

- "Mükemmel" yerine **çalışan ve temiz** bir MVP hedefleyin.
- Kullandığınız ek kütüphaneleri ve nedenlerini README'de belirtin.
- Zaman yetmezse, README'nin "Kalanlar/Trade-off'lar" bölümünde neyi neden ertelediğinizi yazın.