**Adım 1. Detaylı planlama ve tasarım**

**Adım 1.1 Projenin Tüm Gereksinimlerinin Özeti:**

**1. Erişim ve Kullanıcı Yönetimi:**

* **Kayıt:** Herhangi bir e-posta adresi ile kayıt olunabilecek.
* **Onay:** Yeni üyelikler **Yönetici onayı** ile aktifleşecek.
* **Roller:** Sadece "Veli" (standart kullanıcı) ve "Yönetici" (tam yetkili) rolleri olacak.
* **Zorunlu Profil Bilgileri:** Ad Soyad, E-posta, Telefon Numarası, Öğrenci Adı Soyadı, Okul.
* **Profil Gizliliği:** Zorunlu profil bilgileri **sadece Yöneticiler** tarafından görülebilecek.
* **Forum İçi Görünürlük:** Yazışmalarda kullanıcıların gerçek **Adı ve Soyadı** görünecek.

**2. Forum Temel İşlevleri:**

* **Konu Oluşturma:** Tüm onaylı Veliler yeni konu başlığı oluşturabilecek.
* **Yorum Yazma:** Tüm onaylı Veliler mevcut konu başlıklarına yorum yazabilecek.
* **Yorum Yapısı:** Yorumlar, konu altında **alt alta kronolojik** olarak sıralanacak.
* **Düzenleme/Silme:** Kullanıcılar kendi mesajlarını/konu başlıklarını gönderdikten sonraki **ilk 1 saat içinde** düzenleyebilecek veya silebilecek. Düzenlemeler işaretlenecek.
* **İçerik Formatlama:** Mesaj yazarken **basit bir WYSIWYG editör** ile temel formatlama (kalın, italik, liste, link) yapılabilecek.

**3. Belge Paylaşımı:**

* **İzin Verilen Türler:** PDF, JPG, PNG, DOC, DOCX, XLS, XLSX, PPT, PPTX. (Ses/Video yok).
* **Maksimum Boyut:** Dosya başına **25 MB**.
* **Yükleme Yeri:** Belgeler hem **konu başlıklarına/yorumlara eklenebilecek** hem de **ayrı bir genel "Dosyalar" bölümünde** listelenecek/yüklenebilecek.
* **Önizleme:** Dosya önizlemesi **olmayacak**, sadece dosya adı ve indirme linki gösterilecek.

**4. Organizasyon ve Arama:**

* **Kategoriler:** Konu başlıkları için **kategoriler kullanılacak**. Kategoriler Yöneticiler tarafından **eklenip düzenlenebilecek**.
* **Etiketler:** Kullanıcılar konu başlığı oluştururken serbestçe **etiket (#tag) ekleyebilecek**.
* **Arama Kapsamı:** **Basit anahtar kelime araması** yapılacak. Arama; konu başlıkları, mesaj içerikleri, kullanıcı adları (Ad Soyad) ve dosya adlarını kapsayacak.
* **Gelişmiş Arama:** İlk aşamada **gerekli değil**.

**5. Etkileşim ve Moderasyon:**

* **Bildirimler:** Şu durumlarda **hem arayüzde (ikon/sayaç) hem de e-posta ile** bildirim gönderilecek:
  + Kendi konusuna/yorumuna cevap geldiğinde.
  + Takip ettiği konuya cevap geldiğinde.
  + @Ad Soyad şeklinde bahsedildiğinde.
  + Yöneticiden duyuru yapıldığında.
* **Bildirim Ayarları:** Kullanıcılar profil ayarlarından **bildirim tercihlerini (e-posta dahil) yönetebilecek**.
* **İçerik Raporlama:** Uygunsuz içeriği Yöneticilere bildirmek için "**Rapor Et**" butonu olacak.

**6. Dil ve Tasarım:**

* **Dil:** Arayüz **sadece Türkçe** olacak.
* **Tasarım:** Belirli bir marka kimliği yok, **modern ve temiz bir tasarım** tercih edilecek.

**Adım 1.2: Veritabanı şeması**

Belirlediğimiz gereksinimleri karşılayacak, Django ORM (Object-Relational Mapper) ile uyumlu ve PostgreSQL veritabanında iyi performans gösterecek bir yapı tasarlayalım.

İşte önerilen veritabanı şeması (Django model isimleri ve alanları kullanılarak):

**1. User (Django'nun Hazır Kullanıcı Modeli - django.contrib.auth.models.User)**

Bu model Django tarafından sağlanır ve temel kullanıcı kimlik doğrulama işlevlerini (kullanıcı adı/e-posta, şifre, giriş/çıkış) yönetir. Biz bu modeli kullanacağız ve ek bilgiler için genişleteceğiz.

* id: Otomatik artan Primary Key (PK).
* password: Güvenli şekilde hash'lenmiş şifre (Django yönetir).
* last\_login: Son giriş zamanı (DateTimeField, Django yönetir).
* is\_superuser: Süper kullanıcı durumu (BooleanField).
* username: Kullanıcı adı (CharField, unique). E-posta adresini veya benzersiz bir ID'yi burada tutabiliriz.
* first\_name: Kullanıcının adı (CharField, **Zorunlu** yapacağız).
* last\_name: Kullanıcının soyadı (CharField, **Zorunlu** yapacağız).
* email: Kullanıcının e-posta adresi (EmailField, unique, **Zorunlu**).
* is\_staff: Yönetici paneline erişim yetkisi (BooleanField - **Yöneticiler için True**).
* is\_active: Kullanıcının aktif olup olmadığı (BooleanField - **Yönetici onayından sonra True** olacak).
* date\_joined: Kayıt tarihi (DateTimeField, otomatik).

**2. UserProfile (Kullanıcı Profil Detayları)**

Django'nun User modelini genişletmek için birebir (OneToOne) ilişkili model.

* user: User modeline OneToOneField bağlantısı (PK olarak da kullanılır).
* phone\_number: Kullanıcının telefon numarası (CharField, **Zorunlu**).
* student\_full\_name: Öğrencinin Adı Soyadı (CharField, **Zorunlu**).
* school\_name: Okulun adı (CharField, **Zorunlu**).
* is\_approved: Yöneticinin onayı (BooleanField, default=False). *User.is\_active alanı ile senkronize edilebilir.*
* notification\_preferences: Bildirim tercihleri (JSONField, default={}). Örn: {"konuma\_yanit\_eposta": true, "takip\_eposta": true, "bahsedilme\_eposta": false}

**3. Category (Konu Kategorileri)**

Konu başlıklarını gruplamak için.

* id: Otomatik artan PK.
* name: Kategori adı (CharField, unique, max\_length=100).
* slug: URL uyumlu kategori adı (SlugField, unique).
* description: Kategori açıklaması (TextField, isteğe bağlı).
* created\_at: Oluşturulma tarihi (DateTimeField, auto\_now\_add=True).
* updated\_at: Son güncellenme tarihi (DateTimeField, auto\_now=True).

**4. Tag (Etiketler)**

Konu başlıklarına atanacak etiketler.

* id: Otomatik artan PK.
* name: Etiket adı (CharField, unique, max\_length=50).
* slug: URL uyumlu etiket adı (SlugField, unique).

**5. Topic (Konu Başlıkları)**

Forumdaki ana tartışma başlıkları.

* id: Otomatik artan PK.
* title: Konu başlığı (CharField, max\_length=255).
* author: Konuyu başlatan kullanıcı (User modeline ForeignKey, on\_delete=models.SET\_NULL, null=True). *Kullanıcı silinirse konu kalır, yazarı 'null' olur.*
* category: Konunun ait olduğu kategori (Category modeline ForeignKey, on\_delete=models.SET\_NULL, null=True, blank=True). *Kategori silinirse konu kalır, kategorisi 'null' olur.*
* tags: Konuya ait etiketler (Tag modeline ManyToManyField, blank=True).
* created\_at: Oluşturulma tarihi (DateTimeField, auto\_now\_add=True).
* last\_activity\_at: Konudaki son aktivite zamanı (DateTimeField, auto\_now\_add=True, db\_index=True). *Yeni yorum geldiğinde güncellenir, sıralama için kullanılır.*
* is\_locked: Konunun yorumlara kilitli olup olmadığı (BooleanField, default=False). *Yönetici kilitleyebilir.*

*Not: Konunun ilk içeriği, bu konuyla ilişkili ilk Post kaydı olacaktır.*

**6. Post (Mesajlar/Yorumlar)**

Konu başlıklarına yazılan cevaplar.

* id: Otomatik artan PK.
* topic: Mesajın ait olduğu konu (Topic modeline ForeignKey, on\_delete=models.CASCADE). *Konu silinirse mesajlar da silinir.*
* author: Mesajı yazan kullanıcı (User modeline ForeignKey, on\_delete=models.SET\_NULL, null=True).
* content: Mesaj içeriği (TextField). *WYSIWYG editörden gelen HTML veya Markdown burada saklanır.*
* created\_at: Oluşturulma tarihi (DateTimeField, auto\_now\_add=True).
* updated\_at: Son güncellenme tarihi (DateTimeField, auto\_now=True). *Düzenlemeleri takip etmek için.*
* is\_first\_post: Bu mesajın konunun ilk mesajı olup olmadığını belirtir (BooleanField, default=False). *Kolaylık sağlar.*

**7. Attachment (Dosya Ekleri)**

Yüklenen dosyalar (hem mesajlara eklenenler hem de genel dosyalar).

* id: Otomatik artan PK.
* file: Yüklenen dosya (FileField, upload\_to ile dosya yolu belirlenir). *Django dosya depolama mekanizmasını kullanır.*
* uploader: Dosyayı yükleyen kullanıcı (User modeline ForeignKey, on\_delete=models.SET\_NULL, null=True).
* post: Dosyanın eklendiği mesaj (Post modeline ForeignKey, on\_delete=models.CASCADE, null=True, blank=True). *Eğer dosya genel bölümdeyse bu alan null olur.*
* is\_general: Dosyanın genel "Dosyalar" bölümüne ait olup olmadığını belirtir (BooleanField, default=False). *True ise geneldir, False ise bir posta eklidir.*
* original\_filename: Kullanıcının yüklediği orijinal dosya adı (CharField).
* file\_size: Dosya boyutu (PositiveIntegerField, byte cinsinden).
* mime\_type: Dosya türü (CharField, örn: 'application/pdf', 'image/jpeg').
* uploaded\_at: Yüklenme tarihi (DateTimeField, auto\_now\_add=True).
* description: Dosya açıklaması (CharField, blank=True). *Özellikle genel dosyalar için.*

**8. Notification (Bildirimler)**

Kullanıcılara gönderilecek bildirimler.

* id: Otomatik artan PK.
* recipient: Bildirimi alan kullanıcı (User modeline ForeignKey, on\_delete=models.CASCADE).
* actor: Bildirimi tetikleyen kullanıcı (User modeline ForeignKey, on\_delete=models.SET\_NULL, null=True).
* verb: Bildirim eylemi (CharField, örn: 'yanıtladı', 'bahsetti', 'duyurdu').
* target\_content\_type, target\_object\_id, target: Bildirimin ilgili olduğu nesneye (Topic, Post vb.) işaret eden GenericForeignKey alanı.
* is\_read: Bildirimin okunup okunmadığı (BooleanField, default=False, db\_index=True).
* created\_at: Oluşturulma tarihi (DateTimeField, auto\_now\_add=True, db\_index=True).

**9. ReportedContent (Rapor Edilen İçerikler)**

Kullanıcıların rapor ettiği mesajlar/konular.

* id: Otomatik artan PK.
* reporter: İçeriği rapor eden kullanıcı (User modeline ForeignKey, on\_delete=models.SET\_NULL, null=True).
* content\_content\_type, content\_object\_id, content\_object: Rapor edilen içeriğe (Post veya Topic) işaret eden GenericForeignKey alanı.
* reason: Raporlama sebebi (TextField, isteğe bağlı).
* status: Rapor durumu (CharField, choices=['yeni', 'incelendi', 'reddedildi'], default='yeni').
* reported\_at: Raporlama zamanı (DateTimeField, auto\_now\_add=True).
* resolved\_at: Çözülme zamanı (DateTimeField, null=True, blank=True).
* resolver: Raporu çözen yönetici (User modeline ForeignKey, on\_delete=models.SET\_NULL, null=True, blank=True).

**10. TopicFollow (Konu Takibi)**

Kullanıcıların hangi konuları takip ettiğini belirten ilişki tablosu.

* id: Otomatik artan PK.
* user: Takip eden kullanıcı (User modeline ForeignKey, on\_delete=models.CASCADE).
* topic: Takip edilen konu (Topic modeline ForeignKey, on\_delete=models.CASCADE).
* followed\_at: Takip etmeye başlama zamanı (DateTimeField, auto\_now\_add=True).
* Meta: unique\_together = ('user', 'topic') (Bir kullanıcı bir konuyu sadece bir kez takip edebilir).

Bu veritabanı şeması, belirlediğimiz tüm gereksinimleri karşılamak üzere tasarlanmıştır. Django ORM ile bu yapıyı modellere dönüştürmek oldukça kolay olacaktır.

**Adım 1.3: API endpointlerin tasarımı**

Frontend (React) uygulamamızın Backend (Django) ile nasıl konuşacağını tanımlayan API (Application Programming Interface) endpoint'lerini tasarlayalım. Bu endpoint'ler, web tarayıcısının sunucudan veri istemesini, sunucuya veri göndermesini ve çeşitli işlemleri tetiklemesini sağlayacak yollardır.

Django REST Framework (DRF) kullanarak RESTful prensiplerine uygun bir API tasarlayacağız. Genellikle JSON formatında veri alışverişi yapacağız ve standart HTTP metodlarını (GET, POST, PUT, PATCH, DELETE) kullanacağız.

İşte önerilen API endpoint'leri (Temel URL: /api/v1/ varsayılmıştır):

**1. Kimlik Doğrulama (Authentication - /auth/)**

* **POST /auth/register/**
  + **Amaç:** Yeni kullanıcı (Veli) kaydı oluşturur. User (inactive) ve UserProfile (unapproved) kayıtları yaratılır.
  + **İstek Verisi:** email, password, first\_name, last\_name, phone\_number, student\_full\_name, school\_name
  + **Yanıt:** Başarı mesajı (Örn: { "message": "Kayıt başarılı. Yönetici onayı bekleniyor." })
  + **İzinler:** Herkese açık (AllowAny).
* **POST /auth/login/**
  + **Amaç:** Kullanıcı girişi yapar. Başarılı girişte kimlik doğrulama anahtarı (token) döner.
  + **İstek Verisi:** email, password
  + **Yanıt:** { "token": "...", "user": { "id": ..., "first\_name": ..., "last\_name": ... } } (Token ve temel kullanıcı bilgisi).
  + **İzinler:** Herkese açık (AllowAny).
* **POST /auth/logout/**
  + **Amaç:** Kullanıcı çıkışı yapar (Token'ı geçersiz kılar).
  + **İzinler:** Kimlik doğrulamış kullanıcı (IsAuthenticated).
* **POST /auth/password/reset/**
  + **Amaç:** Şifre sıfırlama sürecini başlatır (kullanıcının e-postasına link gönderir).
  + **İstek Verisi:** email
  + **İzinler:** Herkese açık (AllowAny).
* **POST /auth/password/reset/confirm/**
  + **Amaç:** E-postadaki link ile gelen token ve yeni şifre ile sıfırlamayı tamamlar.
  + **İstek Verisi:** uid, token, new\_password
  + **İzinler:** Herkese açık (AllowAny).

**2. Kullanıcılar ve Profil (/users/)**

* **GET /users/me/**
  + **Amaç:** O an giriş yapmış olan kullanıcının detaylarını (profil bilgileri, bildirim ayarları vb.) getirir.
  + **Yanıt:** User ve UserProfile modelinden ilgili alanlar (Yöneticiye özel bilgiler hariç).
  + **İzinler:** IsAuthenticated.
* **PATCH /users/me/**
  + **Amaç:** Giriş yapmış kullanıcının kendi profil bilgilerini (izin verilen alanlar, örn: bildirim tercihleri) güncellemesini sağlar.
  + **İstek Verisi:** Güncellenecek alanlar (Örn: { "notification\_preferences": {...} }).
  + **Yanıt:** Güncellenmiş kullanıcı/profil bilgileri.
  + **İzinler:** IsAuthenticated.
* *(Yönetici Kullanıcı Yönetimi: Yeni kullanıcı onaylama, kullanıcı engelleme gibi işlemler başlangıçta Django Admin paneli üzerinden veya daha sonra geliştirilecek ayrı admin endpoint'leri ile yapılabilir.)*

**3. Kategoriler ve Etiketler (/categories/, /tags/)** (Kullanıcılar için Genellikle Salt Okunur)

* **GET /categories/**
  + **Amaç:** Tüm forum kategorilerini listeler.
  + **Yanıt:** Kategori nesnelerinin listesi (id, name, slug).
  + **İzinler:** IsAuthenticated.
* **GET /tags/**
  + **Amaç:** Tüm etiketleri listeler (veya ?search= ile otomatik tamamlama için filtrelenmiş liste).
  + **Yanıt:** Etiket nesnelerinin listesi (id, name, slug).
  + **İzinler:** IsAuthenticated.
* *(Kategori/Etiket Ekleme/Düzenleme/Silme: Başlangıçta Django Admin paneli üzerinden Yöneticiler tarafından yapılır.)*

**4. Konu Başlıkları (Topics - /topics/)**

* **GET /topics/**
  + **Amaç:** Konu başlıklarını listeler. Sayfalamayı (pagination), filtrelemeyi (?category=<slug>, ?tag=<slug>, ?author=<user\_id>) ve sıralamayı (?ordering=-last\_activity\_at) destekler.
  + **Yanıt:** Sayfalanmış konu başlığı listesi (yazar adı, kategori, etiketler, yorum sayısı, son aktivite zamanı gibi özet bilgilerle).
  + **İzinler:** IsAuthenticated.
* **POST /topics/**
  + **Amaç:** Yeni bir konu başlığı ve ilk mesajı oluşturur. İsteğe bağlı kategori, etiketler ve ek dosyalar içerebilir.
  + **İstek Verisi:** { "title": "...", "category": <category\_id>, "tags": [<tag\_id>, ...], "post\_content": "İlk mesaj içeriği...", "attachments": [<dosya\_verisi1>, ...] }
  + **Yanıt:** Oluşturulan yeni Topic nesnesinin detayları.
  + **İzinler:** IsAuthenticated.
* **GET /topics/{topic\_id}/**
  + **Amaç:** Belirli bir konunun detaylarını ve içerdiği mesajları (sayfalanmış olarak) getirir.
  + **Yanıt:** Konu detayları + sayfalanmış Post nesneleri listesi (yazar adı, mesaj içeriği, ekler vb.).
  + **İzinler:** IsAuthenticated.
* **DELETE /topics/{topic\_id}/**
  + **Amaç:** Bir konuyu siler. (Sadece Yöneticiler için).
  + **İzinler:** IsAdminUser.
* **POST /topics/{topic\_id}/follow/**
  + **Amaç:** Kullanıcının bir konuyu takibe almasını sağlar.
  + **İzinler:** IsAuthenticated.
* **DELETE /topics/{topic\_id}/follow/**
  + **Amaç:** Kullanıcının bir konuyu takipten çıkarmasını sağlar.
  + **İzinler:** IsAuthenticated.
* *(Konu Kilitleme/Açma gibi Yönetici işlemleri Django Admin veya ayrı admin endpoint'leri ile yapılabilir.)*

**5. Mesajlar/Yorumlar (Posts - /posts/)**

* **POST /topics/{topic\_id}/posts/** (Bir konu altına mesaj eklemek için)
  + **Amaç:** Belirli bir konuya yeni bir mesaj (yorum) ekler. İsteğe bağlı ek dosyalar içerebilir.
  + **İstek Verisi:** { "content": "Yorum içeriği...", "attachments": [<dosya\_verisi1>, ...] }
  + **Yanıt:** Oluşturulan yeni Post nesnesinin detayları.
  + **İzinler:** IsAuthenticated (ve konu kilitli değilse).
* **PATCH /posts/{post\_id}/**
  + **Amaç:** Bir mesajı günceller. Sadece mesajın sahibi (ilk 1 saat içinde) veya Yönetici tarafından yapılabilir.
  + **İstek Verisi:** { "content": "Güncellenmiş içerik..." }
  + **Yanıt:** Güncellenmiş Post nesnesi.
  + **İzinler:** Özel İzin (IsOwnerWithin1Hour or IsAdminUser).
* **DELETE /posts/{post\_id}/**
  + **Amaç:** Bir mesajı siler. Sadece mesajın sahibi (ilk 1 saat içinde) veya Yönetici tarafından yapılabilir.
  + **İzinler:** Özel İzin (IsOwnerWithin1Hour or IsAdminUser).

**6. Dosya Ekleri (Attachments - /attachments/)**

* **POST /attachments/**
  + **Amaç:** Dosya yükler. Yüklenen dosya ya bir post\_id ile ilişkilendirilir ya da is\_general=True olarak işaretlenir. (Multipart/form-data olarak yüklenir).
  + **İstek Verisi:** file (dosyanın kendisi), post (opsiyonel post ID), is\_general (Boolean), description (opsiyonel).
  + **Yanıt:** Oluşturulan Attachment nesnesinin detayları (ID, dosya adı, indirme URL'si vb.).
  + **İzinler:** IsAuthenticated.
* **GET /attachments/general/**
  + **Amaç:** Genel "Dosyalar" bölümündeki dosyaları listeler. Sayfalama destekler.
  + **Yanıt:** Sayfalanmış Attachment nesneleri listesi (is\_general=True olanlar).
  + **İzinler:** IsAuthenticated.
* **GET /attachments/{attachment\_id}/**
  + **Amaç:** Belirli bir dosyanın detaylarını getirir (indirme URL'si dahil). *Dosya sunumu Nginx veya bulut depolama üzerinden yapılır, API sadece URL verir.*
  + **Yanıt:** Attachment detayları ve indirme URL'si.
  + **İzinler:** IsAuthenticated.
* **DELETE /attachments/{attachment\_id}/**
  + **Amaç:** Bir dosyayı siler. Sadece yükleyen kişi veya Yönetici silebilir.
  + **İzinler:** Özel İzin (IsOwner or IsAdminUser).

**7. Arama (/search/)**

* **GET /search/**
  + **Amaç:** Temel anahtar kelime araması yapar.
  + **Sorgu Parametresi:** ?q=<aranan\_kelime>
  + **Yanıt:** Arama sonuçları listesi (Örn: { "topics": [...], "posts": [...] } gibi gruplanmış veya tek bir liste).
  + **İzinler:** IsAuthenticated.

**8. Bildirimler (/notifications/)**

* **GET /notifications/**
  + **Amaç:** Giriş yapmış kullanıcının bildirimlerini listeler. Okunmamışları filtreleme (?read=false) ve sayfalama destekler.
  + **Yanıt:** Sayfalanmış Notification nesneleri listesi.
  + **İzinler:** IsAuthenticated.
* **POST /notifications/mark-all-read/**
  + **Amaç:** Kullanıcının tüm okunmamış bildirimlerini okundu olarak işaretler.
  + **İzinler:** IsAuthenticated.
* **POST /notifications/{notification\_id}/mark-read/**
  + **Amaç:** Belirli bir bildirimi okundu olarak işaretler.
  + **İzinler:** IsAuthenticated.

**9. İçerik Raporlama (/report/)**

* **POST /report/**
  + **Amaç:** Uygunsuz görülen bir içeriği (Post veya Topic) Yöneticilere raporlar.
  + **İstek Verisi:** { "content\_type": "app\_label.model\_name", "object\_id": <içerik\_id>, "reason": "Raporlama sebebi..." } (content\_type: örn 'forum.post')
  + **Yanıt:** Başarı mesajı.
  + **İzinler:** IsAuthenticated.
* *(Raporların incelenmesi/yönetimi: Başlangıçta Django Admin paneli üzerinden.)*

Bu endpoint listesi, frontend uygulamasının ihtiyaç duyacağı temel backend etkileşimlerini kapsamaktadır. Geliştirme sırasında DRF Serializer'ları ile yanıtların detaylı yapısı belirlenecek ve özel izinler (örneğin, 1 saatlik düzenleme süresi) kodlanacaktır.

**Adım 2: Ortam Kurulumu (Environment Setup)**

Geliştirme ortamı tamamen Cloud ortamında olsun. Kendi bilgisayarıma bir şey kurmak istemiyorum.