RandevuX – Requirements

# 1) Proje Amacı

RandevuX, kullanıcıların birbirleriyle çevrim içi randevu planlaması, yönetimi ve isteğe bağlı olarak görüntülü görüşme yapabilmesi için tasarlanmış bir web uygulamasıdır. Amaç, hem frontend (Next.js) hem backend (Nest.js + PostgreSQL) tarafında modern geliştirme yetkinliklerini göstermek, aynı zamanda authentication, CRUD, gerçek zamanlı iletişim ve medya aktarımı gibi teknik becerileri bir araya getirmektir.

# 2) SMART Hedefler

### 2.1 Authentication & Profil

- \*\*Specific:\*\* Kullanıcılar e-posta/parola ile kayıt ve giriş yapabilmeli, profil bilgilerini görebilmeli.  
- \*\*Measurable:\*\* 20 denemenin ≥%95’inde başarılı giriş sağlanmalı.  
- \*\*Achievable:\*\* NestJS JWT, bcrypt ile uygulanabilir.  
- \*\*Relevant:\*\* Diğer tüm işlemlerin ön koşulu.  
- \*\*Time-bound:\*\* 2. gün sonuna kadar tamamlanmalı.

### 2.2 Randevu CRUD + Çakışma Kontrolü

- \*\*Specific:\*\* Randevular eklenmeli, listelenmeli, güncellenmeli ve silinebilmeli. Aynı saat aralıklarında çakışma engellenmeli.  
- \*\*Measurable:\*\* 10 test senaryosunda %100 doğru sonuç alınmalı.  
- \*\*Achievable:\*\* PostgreSQL ve Prisma ile kolayca uygulanabilir.  
- \*\*Relevant:\*\* Uygulamanın ana fonksiyonu.  
- \*\*Time-bound:\*\* 3. gün sonuna kadar tamamlanmalı.

### 2.3 Gerçek Zamanlı Bildirimler

- \*\*Specific:\*\* Randevu oluşturma, iptal ve güncelleme olayları anında iletilmeli.  
- \*\*Measurable:\*\* Eventler 1 saniye içinde UI’da görünmeli.  
- \*\*Achievable:\*\* NestJS WebSocket Gateway + Next.js socket-client.  
- \*\*Relevant:\*\* Modern uygulamalarda kullanıcı deneyimi için kritik.  
- \*\*Time-bound:\*\* 4. gün sonuna kadar tamamlanmalı.

### 2.4 WebRTC Görüşme (Opsiyonel)

- \*\*Specific:\*\* Randevu detay sayfasında 1:1 video görüşme yapılabilmeli.  
- \*\*Measurable:\*\* Aynı ağda yapılan 3 denemenin ≥%80’inde başarı.  
- \*\*Achievable:\*\* WebRTC API + WebSocket signaling.  
- \*\*Relevant:\*\* Realtime & medya aktarımı yetkinliği gösterir.  
- \*\*Time-bound:\*\* 5–6. gün arasında.

### 2.5 Sunum & Kullanılabilirlik

- \*\*Specific:\*\* Kullanıcıya net 3 ekran: Login/Register, Randevu Listesi, Randevu Detay.  
- \*\*Measurable:\*\* Demo akışı 2 dakikanın altında tamamlanmalı.  
- \*\*Achievable:\*\* Next.js App Router + Tailwind ile hızlıca.  
- \*\*Relevant:\*\* Değerlendirici için anlaşılır ve kolay test edilebilir.  
- \*\*Time-bound:\*\* 7. gün.

# 3) Kapsam

Dahil:  
- JWT auth, parola hash  
- Randevu modeli (title, startsAt, endsAt, inviteeEmail, notes, status)  
- CRUD ve çakışma kontrolü (PostgreSQL)  
- Realtime eventler (WebSocket)  
- Opsiyonel: WebRTC 1:1 görüşme  
  
Hariç:  
- Şifre sıfırlama, e-posta doğrulama  
- Rol yönetimi, admin panel  
- Çoklu davetli, takvim entegrasyonu  
- Prod dağıtım, CI/CD, ölçek testleri

# 4) Veritabanı Kullanımı

PostgreSQL tercih edilmiştir. Çünkü prod seviyesinde güvenli, ölçeklenebilir ve güçlü indeks desteği vardır. Randevu çakışma kontrolü için zaman aralıklarının kıyaslanması PostgreSQL’de kolaydır.  
  
• Bağlantı .env:  
 DATABASE\_URL="postgresql://randevux\_user:password@localhost:5432/randevux?schema=public"  
  
• Alanlar: TIMESTAMPTZ (zaman dilimi farklarını minimize eder)  
• İndeks: (creatorId, startsAt), (inviteeId, startsAt)  
• Çakışma kuralı: (new.startsAt < existing.endsAt) AND (new.endsAt > existing.startsAt)

# 5) Kullanıcı Hikâyeleri & Kabul Kriterleri

- US1: Kullanıcı kayıt olur → AC: Email benzersiz, zayıf parola reddedilir.  
- US2: Kullanıcı giriş yapar → AC: Hatalı giriş 401, başarılı giriş token döner.  
- US3: Kullanıcı randevu oluşturur → AC: Creator=current user, invitee var.  
- US4: Kullanıcı randevularını listeler → AC: Creator veya invitee olduğu randevular.  
- US5: Kullanıcı randevusunu günceller/siler → AC: Sadece creator yapar.  
- US6: Çakışma kontrolü yapılır → AC: Overlap varsa 400 döner.  
- US7: Realtime bildirim alır → AC: Event <1sn içinde UI’a düşer.  
- US8 (Opsiyonel): WebRTC görüşme → AC: Oda bazlı, karşı video görünür.

# 6) Veri Modeli

**User**

* id (PK)
* email (UNIQUE, **lowercase** normalizasyonu)
* passwordHash
* name
* createdAt, updatedAt (TIMESTAMPTZ)

**Appointment**

* id (PK)
* title (uzunluk sınırı önerilir)
* startsAt (TIMESTAMPTZ, **CHECK**: startsAt < endsAt)
* endsAt (TIMESTAMPTZ)
* status (**ENUM**: scheduled | canceled | done, default scheduled)
* notes? (isteğe bağlı, makul uzunluk)
* creatorId (FK → User.id, **NOT NULL**)
* inviteeId (FK → User.id, **NOT NULL**)
* createdAt, updatedAt (TIMESTAMPTZ)
* **INDEX’ler:** (creatorId, startsAt), (inviteeId, startsAt)
* **Opsiyonel ileri seviye:** tstzrange + GIST exclusion (ileride)

# 7) Teknik İsterler

Backend: NestJS, Prisma, PostgreSQL  
- API p95 <150ms lokal  
- WebSocket namespace: /ws  
- CORS: http://localhost:3000  
  
Frontend: Next.js (App Router), Tailwind  
- 3 temel ekran  
- Token guard ile yönlendirme  
  
Güvenlik: bcrypt hash, JWT Bearer, class-validator  
  
Dokümantasyon: README (kurulum/çalıştırma), API özeti, demo akışı

# 8) Başarı Metrikleri

- Login testleri ≥%95 başarı  
- CRUD p95 gecikme <150 ms  
- Event teslimi <1 sn  
- Çakışma testi 10/10 doğru sonuç

# 9) Zamanlama (1 Hafta)

• Gün 1: Proje iskeleti, Prisma + Postgres migrate  
• Gün 2: Auth  
• Gün 3: Randevu CRUD + çakışma  
• Gün 4: WebSocket eventleri  
• Gün 5: Opsiyonel WebRTC  
• Gün 6: UX & validasyon  
• Gün 7: README + demo