[Skip to content](https://replit.com/@zaac7tri/R4-new#main-content)

Programdaki takvimi ve takvim 2'ye dair her şeyi yaz nasıl göründüğü birbiriyle nasıl ilişkisi olduğu ders eklemenin büyütmenin küçültmenin yer değiştirmenin nasıl olacağı konu planı nasıl çalıştığı kod yapısı tablo boyutlu tüm detayları ince ayrıntısına kadar yazmalısın çünkü ben bunu yapay zeka vereceğim ve o da aynısını bana yapacak tüm özelliklerin birebir aynı olması için yapay zekaya çok kapsamlı bir komut vermem lazım ve bu komut da programa dair her şeyin olması lazım onun için hem görünüm hem front et hem de back and kısmındaki ve ara yüzü kısmındaki tüm detayları anlatman lazım

41 minutes ago

41 minutes ago

•Free

Saved your changes before starting work

Mükemmel! Programdaki takvim sistemini detaylı bir şekilde analiz ettim. İşte size başka bir yapay zekaya verebileceğiniz kapsamlı açıklama:

TAKVİM SİSTEMİ - DETAYLI AÇIKLAMA

GENEL YAPI

Programda **iki tane takvim sistemi** var ve bunlar birbiriyle entegre çalışıyor:

1. **TAKVİM 1** - Haftalık Ders Çizelgesi (WeeklySchedule)
2. **TAKVİM 2** - Konu Bazlı Plan (TopicPlanner)

TAKVİM 1: HAFTALIK DERS ÇİZELGESİ (WeeklySchedule)

GÖRÜNÜM VE ARAYÜZ

Üst Kısım (Header):

* **Başlık**: "Haftalık Ders Çizelgesi"
* **Alt başlık**: "Takvim 1 — Üstten dersi sürükleyip gün/saat alanına bırakın"

Filtre Butonları:

[Tümü] [LGS] [TYT] [AYT] [YDT]

* Bu butonlarla dersler kategoriye göre filtreleniyor
* Aktif buton koyu renk (default variant), diğerleri açık renk (outline variant)

Ders Badge'leri (Sürüklenebilir):

* Her ders bir badge şeklinde gösteriliyor
* Format: "Ders Adı (Kategori)"
* Örnek: "Matematik (TYT)", "Fizik (AYT)"
* Badge'ler sürüklenebilir (draggable)
* Cursor: grab (tutma işareti)

Takvim Grid Yapısı:

Tablo Yapısı:

┌────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┐

│ Saat │ Pzt │ Salı │ Çrş │ Prş │ Cuma │ Cmt │ Pazar │

├────────┼─────────┼─────────┼─────────┼─────────┼─────────┼─────────┼─────────┤

│ 07:00 │ │ │ │ │ │ │ │

│ 07:30 │ │ │ │ │ │ │ │

│ 08:00 │ │ │ │ │ │ │ │

│ ... │ ... │ ... │ ... │ ... │ ... │ ... │ ... │

│ 23:30 │ │ │ │ │ │ │ │

└────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┘

**Grid Özellikleri**:

* Grid sütun yapısı: 72px repeat(7, minmax(60px, 1fr))
* İlk sütun 72px (saat gösterimi için)
* 7 gün sütunu, her biri minimum 60px, eşit genişlikte
* Satır yüksekliği: 28px (ROW\_H = 28)
* Toplam satır sayısı: 34 (ROWS = 34)

**Zaman Parametreleri**:

* Başlangıç: 07:00 (START\_MIN = 7 \* 60 = 420 dakika)
* Bitiş: 24:00 (END\_MIN = 24 \* 60 = 1440 dakika)
* Adım: 30 dakika (STEP = 30)
* Satır sayısı: (1440 - 420) / 30 = 34 satır

**Saat Gösterimi**:

* Sadece tam saatler gösteriliyor (07:00, 08:00, 09:00 vb.)
* Saat etiketleri: text-[11px], sağ tarafta, dikey ortalanmış

Ders Blokları (Slot):

**Görsel Özellikler**:

* Arka plan: bg-primary/15 (açık renk, yarı şeffaf)
* Kenarlık: border border-primary/30
* Yuvarlatılmış köşeler: rounded
* Blur efekti: backdrop-blur
* İçerik padding: p-2 pr-8
* Sol/sağ boşluk: left-2 right-2

**Ders Bloğu İçeriği**:

┌──────────────────────────────────┐

│ Matematik (TYT) [X] │

│ 09:00 - 10:30 │

└──────────────────────────────────┘

* Üst satır: Ders adı ve kategori (font-medium, truncate)
* Alt satır: Başlangıç - Bitiş saati (text-[11px], text-muted-foreground)
* Sağ üst köşe: Silme butonu (Trash2 ikonu, 7x7px)

**Resize Handle'ları** (Boyutlandırma Tutacakları):

* **Üst kenar**:
  + Yükseklik: 8px (h-2)
  + Cursor: n-resize (yukarı-aşağı ok)
  + Tooltip: "Üstten sürükleyerek kısalt/uzat (30dk)"
* **Alt kenar**:
  + Yükseklik: 8px (h-2)
  + Cursor: s-resize (yukarı-aşağı ok)
  + Tooltip: "Alttan sürükleyerek kısalt/uzat (30dk)"

Alt Bilgi Çubuğu:

Haftalık toplam: 24.5 saat

* Toplam çalışma süresini gösterir
* Uyarılar:
  + 5 saatten az: "Çalışma süren çok düşük!" (amber-600)
  + 10 saatten fazla: "Plan çok yoğun!" (red-600)

İŞLEVSELLİK VE ETKİLEŞİMLER

1. DERS EKLEME (Drag & Drop):

**Adımlar**:

1. Kullanıcı üstteki badge'lerden birini tutar (onDragStart)
2. Grid üzerinde bir hücreye sürükler
3. Hücrenin üzerine geldiğinde hover efekti (hover:bg-muted/30)
4. Bırakır (onDrop)

**Kod Akışı**:

- onDragStart: subjectId kaydedilir (window.\_dragSubjectId)

- dropOn(day, startMin):

- Varsayılan süre: 60 dakika

- Çakışma kontrolü (overlaps fonksiyonu)

- Çakışma varsa: "Bu saatte başka bir ders var!" hatası

- Çakışma yoksa: addWeeklySlot(yeni slot)

**Veri Yapısı** (WeeklySlot):

{

id: string, // UUID

studentId: string, // Öğrenci ID

day: 1|2|3|4|5|6|7, // Gün (1=Pzt, 7=Paz)

start: "HH:MM", // Başlangıç saati

end: "HH:MM", // Bitiş saati

subjectId: string // Ders ID

}

2. DERS TAŞIMA (Move):

**Adımlar**:

1. Kullanıcı mevcut ders bloğunu tutar
2. Farklı bir gün/saate sürükler
3. Bırakır

**Kod Akışı**:

- onDragStart: "move-slot:{id}" kaydedilir

- dropOn:

- window.\_dragSlotMove'dan id ve duration alınır

- Yeni pozisyonda çakışma kontrolü

- updateWeeklySlot(id, yeni day/start/end)

3. BOYUTLANDIRMA (Resize):

**Üstten boyutlandırma**:

* Pointer down → resizingRef set edilir (edge: "top")
* Pointer move → deltaRows hesaplanır (dy / ROW\_H)
* newStart = origStart + (deltaRows \* 30dk)
* updateWeeklySlot anında uygulanır

**Alttan boyutlandırma**:

* Aynı mantık, edge: "bottom"
* newEnd = origEnd + (deltaRows \* 30dk)

**Sınırlamalar**:

* Minimum süre: 30 dakika (STEP)
* Çakışan dersler arasında sınır
* Saat aralığı: 07:00 - 24:00

4. DERS SİLME:

* Sağ üst köşedeki Trash2 ikonuna tıklama
* removeWeeklySlot(id) çağrılır
* Slot listeden silinir, UI güncellenir

VERİ KATMANI (Backend & Storage)

Frontend API Fonksiyonları:

- loadWeeklySlots(): WeeklySlot[]

- getWeeklySlotsByStudent(studentId): WeeklySlot[]

- addWeeklySlot(slot): Promise<void>

- removeWeeklySlot(id): Promise<void>

- updateWeeklySlot(id, patch): Promise<void>

- saveWeeklySlots(slots): Promise<void>

Backend API Endpoints:

GET /api/weekly-slots // Tüm slotları getir

GET /api/weekly-slots/:studentId // Öğrenciye göre

POST /api/weekly-slots // Yeni slot ekle

PUT /api/weekly-slots/:id // Slot güncelle

DELETE /api/weekly-slots/:id // Slot sil

Veritabanı Tablosu:

CREATE TABLE weekly\_slots (

id TEXT PRIMARY KEY,

studentId TEXT NOT NULL,

day INTEGER NOT NULL, -- 1-7

startTime TEXT NOT NULL, -- "HH:MM"

endTime TEXT NOT NULL, -- "HH:MM"

subjectId TEXT NOT NULL

)

Caching & Güncelleme:

* Frontend'de weeklySlotsCache değişkeni
* Her değişiklikte window.dispatchEvent('weeklySlotsUpdated')
* Components bu eventi dinler ve yeniden render olur

TAKVİM 2: KONU BAZLI PLAN (TopicPlanner)

GÖRÜNÜM VE ARAYÜZ

Üst Kısım:

* **Başlık**: "Konu Bazlı Plan"
* **Alt başlık**: "Takvim 2 — Konular, Takvim 1 ders bloklarına sırayla yerleştirilir"

Kontrol Paneli:

[Tarih Seçici (input type="date")] [Planı Uygula]

* Tarih seçici: Hafta başlangıcı (Pazartesi) otomatik hesaplanır
* "Planı Uygula" butonu: Plan uygulandığında progress güncellenir

Legend (Renk Göstergesi):

● TYT ● AYT ● YDT

* TYT: bg-primary (mavi)
* AYT: bg-accent (turuncu/farklı renk)
* YDT: bg-amber-500 (sarı)

GÜNLÜK LİSTE GÖRÜNÜMÜ

Her gün için ayrı bir bölüm:

Pazartesi — 2025-01-06 Toplam: 180 dk

┌──────────────────────────────────────────────────────┐

│ [09:00-10:30] Matematik (TYT) — Limit ve Süreklilik │

│ ████████████████░░░░░░░░ (60%) │

│ Kalan: 36 dk │

├──────────────────────────────────────────────────────┤

│ [10:30-11:30] Fizik (AYT) — Newton Yasaları │

│ ███████████░░░░░░░░░░░░░ (45%) │

│ Kalan: 55 dk │

└──────────────────────────────────────────────────────┘

**Konu Kartı Özellikleri**:

* **Arka plan renkleri**:
  + TYT: bg-primary/10 border-primary/30
  + AYT: bg-accent/10 border-accent/30
  + YDT: bg-amber-500/10 border-amber-500/30
* **İçerik**:
  + Saat aralığı badge: [09:00-10:30]
  + Ders adı ve konu: "Matematik (TYT) — Limit ve Süreklilik"
  + Süre: "90 dk"
  + Progress bar: Yükseklik 1.5, bg-muted + bg-primary
  + Kalan süre: "Kalan: 36 dk"

KONU DURUMLARI BÖLÜMÜ

Altta tüm konuların ilerlemesi:

Konu Durumları

┌──────────────────────────────────────────────┐

│ [Matematik] Limit ve Süreklilik 54/90 dk [Sıfırla] │

│ [Fizik] Newton Yasaları 45/100 dk [Sıfırla] │

│ [Kimya] Asit-Baz 80/80 dk [Sıfırla] │

└──────────────────────────────────────────────┘

İŞLEVSELLİK - PLANLAMA ALGORITMASI

planWeek Fonksiyonu:

Bu fonksiyon **TAKVİM 1'deki haftalık slotları** alır ve **konuları** sırayla yerleştirir.

**Algoritma Adımları**:

1. **Hazırlık**:

- Öğrencinin tüm konularını yükle (loadTopics)

- Her konunun progress bilgisini al (getProgressByStudent)

- Haftalık slotları gün ve saate göre sırala

1. **Konuları derslere göre grupla**:

- topicsBySubject Map'i oluştur

- Her ders için konuları order'a göre sırala

1. **Her slot için**:

for (slot in weeklySlots) {

slotSüresi = slot.end - slot.start (dakika)

while (slotSüresi > 0) {

- Bu ders için sonraki tamamlanmamış konuyu bul (pickNext)

- Eğer konu yoksa, döngüden çık

- allocated = min(slotSüresi, konuKalan)

- Plana ekle: {

date, start, end, subjectId, topicId,

allocated, remainingAfter

}

- Konu progress'ini güncelle (in-memory)

- slotSüresi -= allocated

- Slot içindeki cursor'ı ilerlet (currentStartMin += allocated)

}

}

1. **Örnek Senaryo**:

Slot: Pazartesi 09:00-11:00 (120dk), Matematik (TYT)

Matematik konuları:

1. Limit (kalan: 90dk)

2. Türev (kalan: 150dk)

Plan:

- 09:00-10:30: Limit (90dk) → Limit tamamlandı

- 10:30-11:00: Türev (30dk) → Türev'de 120dk kaldı

Veri Yapısı (PlannedEntry):

{

date: "2025-01-06", // YYYY-MM-DD

start: "09:00", // HH:MM

end: "10:30", // HH:MM

subjectId: "math-tyt-123",

topicId: "limit-456",

allocated: 90, // dakika

remainingAfter: 0 // dakika

}

Planı Uygula (applyPlan):

async function applyPlan() {

for (entry of plan) {

await updateProgress(

studentId,

entry.topicId,

entry.allocated

)

}

// Progress updated, UI refreshes

}

VERİ KATMANI

Subjects (Dersler):

{

id: string,

name: string, // "Matematik"

category: "TYT"|"AYT"|"YDT"|"LGS",

code?: string,

description?: string,

color?: string

}

Topics (Konular):

{

id: string,

subjectId: string, // Hangi derse ait

name: string, // "Limit ve Süreklilik"

avgMinutes: number, // Ortalama çalışma süresi (dk)

estimatedHours?: number,

difficulty?: string,

order?: number, // Sıralama

description?: string

}

TopicProgress (İlerleme):

{

id: string,

studentId: string,

topicId: string,

completed: number, // Tamamlanan dakika

remaining: number, // Kalan dakika

completedFlag?: boolean, // Manuel tamamlanma

lastStudied?: string,

notes?: string

}

API Endpoints:

GET /api/subjects // Tüm dersleri getir

POST /api/subjects // Dersleri kaydet

GET /api/topics // Tüm konuları getir

POST /api/topics // Konuları kaydet

GET /api/progress // Tüm progress'leri getir

POST /api/progress // Progress kaydet

TAKVİMLER ARASI İLİŞKİ

Çalışma Akışı:

1. **Kullanıcı TAKVİM 1'de haftalık programa dersler ekler**:
   * Örnek: Pazartesi 09:00-11:00 Matematik (TYT)
   * Örnek: Salı 14:00-15:30 Fizik (AYT)
2. **Konular sistemde tanımlı**:
   * Matematik altında: Limit (90dk), Türev (150dk), İntegral (120dk)
   * Fizik altında: Newton (100dk), Enerji (80dk)
3. **TAKVİM 2 otomatik olarak planı oluşturur**:
   * Haftalık slotları alır
   * Her slot için o dersin konularını sırayla yerleştirir
   * Konu süreleri ve progress'e göre dağıtım yapar
4. **Kullanıcı "Planı Uygula" derse**:
   * Her konunun progress'i güncellenir
   * Completed artırılır, remaining azalır
   * Konu tamamlandıysa completedFlag = true

Otomatik Güncellemeler:

// Event listeners

window.addEventListener('weeklySlotsUpdated', handleUpdate)

window.addEventListener('subjectsUpdated', handleUpdate)

window.addEventListener('topicsUpdated', handleUpdate)

window.addEventListener('progressUpdated', handleUpdate)

* WeeklySlots değişince → TAKVİM 2 planı yeniden hesaplar
* Topics değişince → Plan yeniden oluşturulur
* Progress güncellenince → Konu durumları ve progress bar'lar güncellenir

TEKNİK DETAYLAR

Grid Hesaplamaları:

// Saat → Dakika

function toMin(t: "HH:MM"): number {

const [h, m] = t.split(":").map(Number)

return h \* 60 + m

}

// Dakika → Saat

function fmt(mins: number): "HH:MM" {

const h = Math.floor(mins / 60)

const mm = mins % 60

return `${h.toString().padStart(2,'0')}:${mm.toString().padStart(2,'0')}`

}

// Pixel → Row index

rowIndex = (minutes - START\_MIN) / STEP

// Row → Pixel position

top = rowIndex \* ROW\_H

height = (endMin - startMin) / STEP \* ROW\_H

Çakışma Kontrolü:

function overlaps(a: WeeklySlot, b: WeeklySlot): boolean {

if (a.day !== b.day) return false

const s1 = toMin(a.start), e1 = toMin(a.end)

const s2 = toMin(b.start), e2 = toMin(b.end)

// Kesişim varsa true

return Math.max(s1, s2) < Math.min(e1, e2)

}

Resize Mantığı:

onPointerMove(event) {

const dy = event.clientY - startY

const deltaRows = Math.round(dy / ROW\_H)

if (edge === "top") {

newStart = origStart + (deltaRows \* STEP)

newStart = clamp(newStart, minStart, origEnd - STEP)

} else {

newEnd = origEnd + (deltaRows \* STEP)

newEnd = clamp(newEnd, origStart + STEP, maxEnd)

}

updateWeeklySlot(id, { start: fmt(newStart), end: fmt(newEnd) })

}

STYLING VE TAILWIND CLASS'LARI

Grid Container:

className="w-full overflow-auto rounded border"

Header Row:

className="p-2 text-xs font-medium border-b bg-muted/40"

Time Cell:

style={{ height: ROW\_H }}

className="border-t border-l last:border-r hover:bg-muted/30 transition-colors"

Slot Card:

className="absolute left-2 right-2 rounded bg-primary/15 border border-primary/30 text-xs p-2 pr-8 flex items-center gap-2 backdrop-blur pointer-events-auto cursor-move"

Progress Bar:

// Container

className="mt-2 h-1.5 w-full rounded bg-muted overflow-hidden"

// Fill

className="h-full bg-primary"

style={{ width: `${percentage}%` }}

ÖNEMLİ NOTLAR

1. **30 dakikalık adımlar**: Tüm ders ekleme/boyutlandırma 30dk'lık bloklarda
2. **Otomatik hizalama**: Resize sırasında snap-to-grid davranışı
3. **Çakışma önleme**: Aynı gün/saatte iki ders olamaz
4. **Sıralı planlama**: Konular order değerine göre sırayla işlenir
5. **Progress takibi**: Her konu için ayrı ilerleme kaydı
6. **Responsive**: Grid mobil cihazlarda scroll edilebilir
7. **Real-time sync**: API değişiklikleri anında UI'da yansır

Bu açıklama ile başka bir yapay zeka sistemin **birebir aynısını** oluşturabilir. Tüm görsel detaylar, algoritma mantığı, veri yapıları ve API endpoint'leri dahil edilmiştir.

4 minutes of work

Sence bu sisitmede ğeliştiilmesi gerek noktalar var. Mantığı nasıl daha iyi hale getiririz

20 minutes ago

Bunu programa yaptırmak için yapay zekaya yaptırmak için çok detaylı anlatım yap

15 minutes ago

Kullanıcı, yukarıda bahsettiğim geliştirmeleri bir başka yapay zekaya yaptırmak için çok detaylı bir anlatım istiyor. Yani bir nevi "prompt" veya "instruction set" hazırlamam gerekiyor.

Chat mode'dayım ve kullanıcı dokümantasyon istemiyor ama çok detaylı bir anlatım istiyor ki başka bir AI'a bu işi yaptırabilsin. Bu durumda, detaylı bir implementation guide yazmalıyım.

Mevcut sistemi zaten detaylı anlattım. Şimdi iyileştirmeleri nasıl implement edeceğini adım adım, çok net bir şekilde anlatmalıyım.

TAKVİM SİSTEMİ GELİŞTİRME - DETAYLI UYGULAMA KILAVUZU

Aşağıdaki her bölüm, yapay zekaya verebileceğiniz **tam komut formatında** yazılmıştır. İstediğiniz özelliği seçip, o bölümü direkt yapay zekaya kopyala-yapıştır yapabilirsiniz.

1. AKILLI ÇAKIŞMA YÖNETİMİ

KOMUT:

TAKVİM 1 (WeeklySchedule) sistemine akıllı çakışma yönetimi ekle:

MEVCUT DURUM:

- Çakışma olunca sadece "Bu saatte başka bir ders var!" hatası gösteriliyor

- Kullanıcı manuel olarak başka bir saat bulmalı

YENİ İSTENEN ÖZELLİKLER:

1. OTOMATIK KAYDIRMA:

- Çakışma tespit edilince, en yakın boş hücreyi otomatik bul

- Kullanıcıya sor: "Bu saatte X dersi var. [En yakın boş saat: 14:30] Otomatik yerleştir?"

- Eğer kabul ederse, otomatik o saate yerleştir

2. SWAP (DEĞİŞTİR) ÖNERİSİ:

- "Bu saatte Matematik var. Değiştirmek ister misin?" dialog göster

- Evet derse: İki dersi yer değiştir

- Hayır derse: İptal et

3. GÖRSEL İYİLEŞTİRME:

- Sürüklerken çakışan hücreleri kırmızı highlight et

- Boş hücreleri yeşil highlight et

- En uygun 3 boş hücreyi parlat (pulse animasyonu)

UYGULAMA:

A) dropOn fonksiyonunu güncelle:

```javascript

const dropOn = (day, startMin) => {

setError("");

const candidate = createSlotCandidate(day, startMin);

// Çakışma kontrolü

const conflicting = slots.find(s => overlaps(s, candidate));

if (conflicting) {

// En yakın boş saati bul

const nearestSlot = findNearestEmptySlot(day, startMin, candidate.duration);

if (nearestSlot) {

// Dialog göster

showConflictDialog({

conflictingSlot: conflicting,

suggestedSlot: nearestSlot,

onAutoPlace: () => dropOn(nearestSlot.day, nearestSlot.startMin),

onSwap: () => swapSlots(candidate, conflicting),

onCancel: () => setError("İşlem iptal edildi")

});

} else {

setError("Bu saatte başka bir ders var ve boş saat bulunamadı!");

}

return;

}

// Çakışma yoksa normal ekle

addWeeklySlot(candidate);

};

B) Yeni fonksiyonlar ekle:

// En yakın boş slot bul

function findNearestEmptySlot(targetDay, targetMin, duration) {

const allSlots = [];

// Aynı gün önce kontrol et

for (let min = START\_MIN; min < END\_MIN - duration; min += STEP) {

if (isSlotEmpty(targetDay, min, duration)) {

allSlots.push({

day: targetDay,

startMin: min,

distance: Math.abs(min - targetMin)

});

}

}

// Diğer günleri kontrol et

for (let day = 1; day <= 7; day++) {

if (day === targetDay) continue;

for (let min = START\_MIN; min < END\_MIN - duration; min += STEP) {

if (isSlotEmpty(day, min, duration)) {

allSlots.push({

day,

startMin: min,

distance: Math.abs(min - targetMin) + (Math.abs(day - targetDay) \* 1440)

});

}

}

}

// En yakını döndür

return allSlots.sort((a, b) => a.distance - b.distance)[0];

}

// Slot boş mu kontrol et

function isSlotEmpty(day, startMin, duration) {

const testSlot = {

day,

start: fmt(startMin),

end: fmt(startMin + duration)

};

return !slots.some(s => overlaps(s, testSlot));

}

// İki slotu yer değiştir

function swapSlots(newSlot, existingSlot) {

const tempDay = existingSlot.day;

const tempStart = existingSlot.start;

const tempEnd = existingSlot.end;

updateWeeklySlot(existingSlot.id, {

day: newSlot.day,

start: newSlot.start,

end: newSlot.end

});

addWeeklySlot({

...newSlot,

day: tempDay,

start: tempStart,

end: tempEnd

});

}

C) UI Dialog Komponenti ekle:

* ConflictDialog.tsx oluştur
* shadcn/ui AlertDialog kullan
* 3 buton: "Otomatik Yerleştir", "Değiştir", "İptal"
* Önerilen saati göster: "14:30-15:30 boş, oraya yerleştirelim mi?"

D) Görsel feedback:

* onDragOver sırasında çakışma kontrolü yap
* Çakışan hücreye: className="bg-red-100 border-red-500"
* Boş hücrelere: className="bg-green-50 border-green-300"
* En yakın 3 hücreye: className="animate-pulse bg-blue-100"

---

## 2. SPACED REPETITION (AKILLI TEKRAR) SİSTEMİ

### KOMUT:

TAKVİM 2 (TopicPlanner) sistemine spaced repetition (aralıklı tekrar) algoritması ekle:

TEORİ:

* İlk öğrenme: Tam öğren
* 1 gün sonra: %20 tekrar
* 3 gün sonra: %15 tekrar
* 7 gün sonra: %10 tekrar
* 14 gün sonra: %10 tekrar

VERİ YAPISI GÜNCELLEMESİ:

1. TopicProgress interface'ine ekle:

export type TopicProgress = {

id: string;

studentId: string;

topicId: string;

completed: number;

remaining: number;

completedFlag?: boolean;

// YENİ ALANLAR:

lastStudied?: string; // Son çalışma tarihi (ISO)

masteryLevel?: number; // 0-100 arası ustalık skoru

reviewSchedule?: { // Tekrar takvimi

nextReviewDate: string; // Bir sonraki tekrar tarihi

reviewCount: number; // Kaç kez tekrar edildi

reviewHistory: { // Tekrar geçmişi

date: string;

duration: number; // dakika

quality: number; // 1-5 arası kalite skoru

}[];

};

}

1. Database tablosunu güncelle:

ALTER TABLE progress ADD COLUMN lastStudied TEXT;

ALTER TABLE progress ADD COLUMN masteryLevel INTEGER DEFAULT 0;

ALTER TABLE progress ADD COLUMN reviewSchedule TEXT; -- JSON

PLANLAMA ALGORİTMASI GÜNCELLEMESİ:

1. planWeek fonksiyonunu güncelle:

export async function planWeekWithReviews(

studentId: string,

weekStartISO: string

): Promise<PlannedEntry[]> {

await ensureProgressForStudent(studentId);

const slots = getWeeklySlotsByStudent(studentId);

const progress = getProgressByStudent(studentId);

const topics = loadTopics();

// Tekrar edilmesi gereken konuları bul

const reviewsNeeded = findReviewsNeeded(progress, weekStartISO);

// Normal plan + review planını birleştir

const normalPlan = generateNormalPlan(slots, topics, progress);

const reviewPlan = generateReviewPlan(slots, reviewsNeeded, weekStartISO);

// Öncelik: Reviewlar önce

return [...reviewPlan, ...normalPlan];

}

// Tekrar edilmesi gereken konuları bul

function findReviewsNeeded(progress: TopicProgress[], weekStartISO: string) {

const weekStart = new Date(weekStartISO);

const weekEnd = new Date(weekStart.getTime() + 7 \* 24 \* 60 \* 60 \* 1000);

return progress.filter(p => {

if (!p.reviewSchedule || !p.completedFlag) return false;

const nextReview = new Date(p.reviewSchedule.nextReviewDate);

return nextReview >= weekStart && nextReview <= weekEnd;

}).map(p => {

const topic = loadTopics().find(t => t.id === p.topicId);

const reviewDuration = calculateReviewDuration(p, topic);

return {

topicId: p.topicId,

subjectId: topic.subjectId,

duration: reviewDuration,

priority: calculateReviewPriority(p),

scheduledDate: p.reviewSchedule.nextReviewDate

};

}).sort((a, b) => b.priority - a.priority);

}

// Tekrar süresi hesapla

function calculateReviewDuration(progress: TopicProgress, topic: Topic) {

const reviewCount = progress.reviewSchedule?.reviewCount || 0;

const baseMinutes = topic.avgMinutes;

// Her tekrarda süre azalır

if (reviewCount === 0) return baseMinutes \* 0.3; // %30

if (reviewCount === 1) return baseMinutes \* 0.2; // %20

if (reviewCount === 2) return baseMinutes \* 0.15; // %15

return baseMinutes \* 0.1; // %10

}

// Review önceliği hesapla (gecikenler önce)

function calculateReviewPriority(progress: TopicProgress) {

const nextReview = new Date(progress.reviewSchedule.nextReviewDate);

const today = new Date();

const daysOverdue = Math.floor((today - nextReview) / (1000 \* 60 \* 60 \* 24));

return daysOverdue \* 10 + (100 - progress.masteryLevel);

}

TEKRAR TAKVIMINI OTOMATIK OLUŞTUR:

// Konu tamamlandığında tekrar takvimi oluştur

export async function completeTopicAndScheduleReviews(

studentId: string,

topicId: string,

quality: number // 1-5 arası

) {

const progress = loadProgress().find(

p => p.studentId === studentId && p.topicId === topicId

);

if (!progress) return;

const today = new Date().toISOString().slice(0, 10);

const reviewSchedule = generateReviewSchedule(today, quality);

progress.completedFlag = true;

progress.lastStudied = today;

progress.masteryLevel = quality \* 20; // 1->20, 5->100

progress.reviewSchedule = reviewSchedule;

await saveProgress([...loadProgress()]);

}

// Fibonacci benzeri aralıklarla review schedule oluştur

function generateReviewSchedule(completionDate: string, quality: number) {

const baseDate = new Date(completionDate);

const intervals = quality >= 4

? [1, 3, 7, 14, 30] // Yüksek kalite: daha uzun aralık

: [1, 2, 4, 7, 14, 21]; // Düşük kalite: daha sık tekrar

const schedule = intervals.map(days => {

const reviewDate = new Date(baseDate);

reviewDate.setDate(reviewDate.getDate() + days);

return reviewDate.toISOString().slice(0, 10);

});

return {

nextReviewDate: schedule[0],

reviewCount: 0,

reviewHistory: []

};

}

UI GÜNCELLEMELERİ:

1. Konu kartlarına review badge ekle:

{progress.reviewSchedule?.nextReviewDate && (

<Badge variant="outline" className="ml-2">

🔄 Tekrar: {progress.reviewSchedule.nextReviewDate}

</Badge>

)}

1. Plan görünümünde review'ları farklı renklendir:

// Normal konu: bg-primary/10

// Review konusu: bg-purple-100 border-purple-400

1. Kalite değerlendirme dialog'u ekle:

* Konu tamamlandığında "Bu konuyu ne kadar iyi anladınız? 1-5" sor
* Seçime göre review schedule oluştur

---

## 3. AKILLI PLANLAMA (ZORLuk & ÖNCELİK BAZLI)

### KOMUT:

TAKVİM 2'ye akıllı planlama algoritması ekle. Konuları sadece sıraya göre değil, zorluk ve önceliğe göre yerleştir.

VERİ YAPISI:

1. Topic interface'ine ekle:

export type StudyTopic = {

id: string;

subjectId: string;

name: string;

avgMinutes: number;

order?: number;

difficulty?: string;

// YENİ ALANLAR:

difficultyScore?: number; // 1-10 arası zorluk

priority?: number; // 1-10 arası öncelik

deadline?: string; // Son tarih (ISO)

prerequisites?: string[]; // Önkoşul konu ID'leri

energyLevel?: 'high' | 'medium' | 'low'; // Gereken enerji

}

1. WeeklySlot'a enerji bilgisi ekle:

export type WeeklySlot = {

// ... mevcut alanlar

energyType?: 'high' | 'medium' | 'low'; // Bu saatteki enerji durumu

}

AKILLI SIRALAMA ALGORİTMASI:

// Konuları akıllı sırala

function smartSortTopics(topics: StudyTopic[], progress: TopicProgress[]) {

const today = new Date();

return topics

.map(topic => {

const prog = progress.find(p => p.topicId === topic.id);

// Skor hesaplama

let score = 0;

// 1. Deadline yakınsa +50 puan

if (topic.deadline) {

const daysUntil = Math.floor(

(new Date(topic.deadline) - today) / (1000 \* 60 \* 60 \* 24)

);

if (daysUntil <= 7) score += 50;

else if (daysUntil <= 14) score += 30;

else if (daysUntil <= 30) score += 10;

}

// 2. Öncelik puanı

score += (topic.priority || 5) \* 10;

// 3. Zorluk puanı (zorlar daha önce)

score += (topic.difficultyScore || 5) \* 5;

// 4. Kalan süre fazlaysa +puan

if (prog && prog.remaining > 120) score += 20;

// 5. Hiç başlanmamışsa +puan

if (prog && prog.completed === 0) score += 15;

return { ...topic, calculatedScore: score };

})

.sort((a, b) => b.calculatedScore - a.calculatedScore);

}

// Slotları enerji seviyesine göre sınıflandır

function categorizeSlotsByEnergy(slots: WeeklySlot[]) {

return slots.map(slot => {

const hour = parseInt(slot.start.split(':')[0]);

let energyType: 'high' | 'medium' | 'low';

if (hour >= 8 && hour <= 11) {

energyType = 'high'; // Sabah: Yüksek enerji

} else if (hour >= 14 && hour <= 17) {

energyType = 'medium'; // Öğleden sonra: Orta

} else {

energyType = 'low'; // Akşam: Düşük

}

return { ...slot, energyType };

});

}

// Konu ile slot'u eşleştir

function matchTopicToSlot(topic: StudyTopic, slot: WeeklySlot) {

const topicEnergy = topic.energyLevel || 'medium';

const slotEnergy = slot.energyType || 'medium';

// Enerji eşleşme skoru

const energyMatch = {

'high-high': 10,

'high-medium': 5,

'high-low': 0,

'medium-high': 7,

'medium-medium': 10,

'medium-low': 7,

'low-high': 3,

'low-medium': 7,

'low-low': 10

};

return energyMatch[`${topicEnergy}-${slotEnergy}`] || 5;

}

PLANLAMA FONKSİYONU GÜNCELLEMESİ:

export async function planWeekSmart(

studentId: string,

weekStartISO: string

): Promise<PlannedEntry[]> {

await ensureProgressForStudent(studentId);

const slots = categorizeSlotsByEnergy(

getWeeklySlotsByStudent(studentId)

.sort((a, b) => a.day - b.day || a.start.localeCompare(b.start))

);

const topics = loadTopics();

const progress = getProgressByStudent(studentId);

// Derslere göre konuları grupla ve akıllı sırala

const topicsBySubject = new Map();

for (const topic of topics) {

const arr = topicsBySubject.get(topic.subjectId) || [];

arr.push(topic);

topicsBySubject.set(topic.subjectId, arr);

}

// Her dersin konularını akıllı sırala

topicsBySubject.forEach((topicList, subjectId) => {

const sorted = smartSortTopics(topicList, progress);

topicsBySubject.set(subjectId, sorted);

});

const out: PlannedEntry[] = [];

// Slotlara konuları yerleştir

for (const slot of slots) {

const subjectTopics = topicsBySubject.get(slot.subjectId) || [];

let remainingTime = minutesBetween(slot.start, slot.end);

let currentMin = toMin(slot.start);

while (remainingTime > 0 && subjectTopics.length > 0) {

// En uygun konuyu bul

const bestTopic = subjectTopics

.filter(t => {

const prog = progress.find(p => p.topicId === t.id);

return prog && !prog.completedFlag && prog.remaining > 0;

})

.map(t => ({

topic: t,

matchScore: matchTopicToSlot(t, slot)

}))

.sort((a, b) => b.matchScore - a.matchScore)[0];

if (!bestTopic) break;

const topic = bestTopic.topic;

const prog = progress.find(p => p.topicId === topic.id)!;

const allocated = Math.min(remainingTime, prog.remaining);

out.push({

date: dateFromWeekStart(weekStartISO, slot.day),

start: fmt(currentMin),

end: fmt(currentMin + allocated),

subjectId: slot.subjectId,

topicId: topic.id,

allocated,

remainingAfter: prog.remaining - allocated,

energyMatch: bestTopic.matchScore // Ekstra bilgi

});

prog.remaining -= allocated;

if (prog.remaining <= 0) prog.completedFlag = true;

remainingTime -= allocated;

currentMin += allocated;

}

}

return out;

}

UI İYİLEŞTİRMELERİ:

1. Konu kartlarına deadline badge:

{topic.deadline && (

<Badge variant={daysUntil <= 7 ? "destructive" : "secondary"}>

📅 {topic.deadline}

</Badge>

)}

1. Enerji göstergesi:

{energyMatch >= 8 ? "⚡ Mükemmel eşleşme" :

energyMatch >= 5 ? "✓ Uygun" :

"⚠️ Az uyumlu"}

1. Zorluk göstergesi:

<div className="flex gap-1">

{Array.from({length: topic.difficultyScore || 5}).map((\_, i) => (

<div key={i} className="w-2 h-4 bg-red-500 rounded" />

))}

</div>

---

## 4. UNDO/REDO SİSTEMİ

### KOMUT:

TAKVİM 1'e undo/redo (geri al/tekrarla) özelliği ekle.

GLOBAL STATE YÖNETİMİ:

1. useUndo custom hook oluştur:

// client/hooks/useUndo.ts

import { useState, useCallback } from 'react';

type HistoryEntry<T> = {

state: T;

timestamp: number;

action: string; // "add", "remove", "update", "resize", "move"

};

export function useUndo<T>(

initialState: T,

maxHistory: number = 20

) {

const [history, setHistory] = useState<HistoryEntry<T>[]>([

{ state: initialState, timestamp: Date.now(), action: 'init' }

]);

const [currentIndex, setCurrentIndex] = useState(0);

const currentState = history[currentIndex]?.state || initialState;

const push = useCallback((newState: T, action: string) => {

setHistory(prev => {

const newHistory = prev.slice(0, currentIndex + 1);

newHistory.push({

state: newState,

timestamp: Date.now(),

action

});

// Max history sınırı

if (newHistory.length > maxHistory) {

newHistory.shift();

return newHistory;

}

return newHistory;

});

setCurrentIndex(prev => prev + 1);

}, [currentIndex, maxHistory]);

const undo = useCallback(() => {

if (currentIndex > 0) {

setCurrentIndex(prev => prev - 1);

return history[currentIndex - 1].state;

}

return currentState;

}, [currentIndex, history, currentState]);

const redo = useCallback(() => {

if (currentIndex < history.length - 1) {

setCurrentIndex(prev => prev + 1);

return history[currentIndex + 1].state;

}

return currentState;

}, [currentIndex, history, currentState]);

const canUndo = currentIndex > 0;

const canRedo = currentIndex < history.length - 1;

return {

state: currentState,

push,

undo,

redo,

canUndo,

canRedo,

history: history.slice(0, currentIndex + 1)

};

}

1. WeeklySchedule'a entegre et:

export default function WeeklySchedule({ sid }: { sid: string }) {

const [subjects, setSubjects] = useState([]);

// Undo/Redo state

const {

state: slots,

push: pushHistory,

undo,

redo,

canUndo,

canRedo,

history

} = useUndo<WeeklySlot[]>([], 30);

// Slots değiştiğinde API'ye kaydet

useEffect(() => {

if (slots.length > 0) {

saveWeeklySlots(slots).catch(console.error);

}

}, [slots]);

// İlk yükleme

useEffect(() => {

const initialSlots = getWeeklySlotsByStudent(sid);

if (initialSlots.length > 0) {

pushHistory(initialSlots, 'load');

}

}, [sid]);

// Ders ekleme

const addSlot = (slot: WeeklySlot) => {

const newSlots = [...slots, slot];

pushHistory(newSlots, 'add');

addWeeklySlot(slot); // API call

};

// Ders silme

const removeSlot = (id: string) => {

const newSlots = slots.filter(s => s.id !== id);

pushHistory(newSlots, 'remove');

removeWeeklySlot(id); // API call

};

// Ders güncelleme

const updateSlot = (id: string, patch: Partial<WeeklySlot>) => {

const newSlots = slots.map(s =>

s.id === id ? { ...s, ...patch } : s

);

pushHistory(newSlots, 'update');

updateWeeklySlot(id, patch); // API call

};

// Keyboard shortcuts

useEffect(() => {

const handleKeyDown = (e: KeyboardEvent) => {

if ((e.ctrlKey || e.metaKey) && e.key === 'z' && !e.shiftKey) {

e.preventDefault();

if (canUndo) {

const prevState = undo();

saveWeeklySlots(prevState);

}

} else if ((e.ctrlKey || e.metaKey) && (e.key === 'y' || (e.key === 'z' && e.shiftKey))) {

e.preventDefault();

if (canRedo) {

const nextState = redo();

saveWeeklySlots(nextState);

}

}

};

window.addEventListener('keydown', handleKeyDown);

return () => window.removeEventListener('keydown', handleKeyDown);

}, [undo, redo, canUndo, canRedo]);

return (

<Card>

<CardHeader>

<div className="flex items-center justify-between">

<div>

<CardTitle>Haftalık Ders Çizelgesi</CardTitle>

<CardDescription>

Takvim 1 — Üstten dersi sürükleyip gün/saat alanına bırakın

</CardDescription>

</div>

{/\* Undo/Redo Buttons \*/}

<div className="flex gap-2">

<Button

variant="outline"

size="sm"

onClick={() => {

const prevState = undo();

saveWeeklySlots(prevState);

}}

disabled={!canUndo}

title="Geri Al (Ctrl+Z)"

>

<Undo2 className="h-4 w-4" />

</Button>

<Button

variant="outline"

size="sm"

onClick={() => {

const nextState = redo();

saveWeeklySlots(nextState);

}}

disabled={!canRedo}

title="Tekrarla (Ctrl+Y)"

>

<Redo2 className="h-4 w-4" />

</Button>

{/\* History Dropdown \*/}

<DropdownMenu>

<DropdownMenuTrigger asChild>

<Button variant="outline" size="sm">

<History className="h-4 w-4" />

</Button>

</DropdownMenuTrigger>

<DropdownMenuContent align="end" className="w-64">

<DropdownMenuLabel>İşlem Geçmişi</DropdownMenuLabel>

<DropdownMenuSeparator />

{history.slice().reverse().map((entry, i) => (

<DropdownMenuItem key={i}>

<span className="text-xs text-muted-foreground">

{formatActionName(entry.action)} - {formatTime(entry.timestamp)}

</span>

</DropdownMenuItem>

))}

</DropdownMenuContent>

</DropdownMenu>

</div>

</div>

</CardHeader>

{/\* ... rest of component \*/}

</Card>

);

}

// Helper fonksiyonlar

function formatActionName(action: string) {

const names = {

add: '➕ Ders eklendi',

remove: '🗑️ Ders silindi',

update: '✏️ Ders güncellendi',

resize: '↔️ Boyut değiştirildi',

move: '↗️ Ders taşındı',

load: '📂 Yüklendi'

};

return names[action] || action;

}

function formatTime(timestamp: number) {

return new Date(timestamp).toLocaleTimeString('tr-TR', {

hour: '2-digit',

minute: '2-digit'

});

}

GEREKLI İKONLAR:

import { Undo2, Redo2, History } from "lucide-react";

TOAST BİLDİRİMLERİ:

// Undo yapıldığında

toast.info("Son işlem geri alındı", {

action: {

label: "Tekrarla",

onClick: () => redo()

}

});

---

## 5. TEMPLATE (ŞABLON) SİSTEMİ

### KOMUT:

Hazır program şablonları sistemi ekle: "LGS Hazırlık", "YKS Son 3 Ay", "TYT Yoğun" gibi.

VERİ YAPISI:

export type ScheduleTemplate = {

id: string;

name: string;

description: string;

category: 'LGS' | 'YKS' | 'TYT' | 'AYT' | 'Custom';

slots: Omit<WeeklySlot, 'id' | 'studentId'>[];

subjects: {

id: string;

name: string;

category: string;

}[];

estimatedWeeklyHours: number;

difficulty: 'Kolay' | 'Orta' | 'Yoğun' | 'Çok Yoğun';

tags: string[];

};

ÖRNEK ŞABLONLAR:

const TEMPLATES: ScheduleTemplate[] = [

{

id: 'lgs-balanced',

name: 'LGS Dengeli Program',

description: 'Hafta içi günde 3 saat, hafta sonu 4 saat çalışma',

category: 'LGS',

estimatedWeeklyHours: 23,

difficulty: 'Orta',

tags: ['lgs', 'dengeli', '8.sınıf'],

subjects: [

{ id: 'mat-lgs', name: 'Matematik', category: 'LGS' },

{ id: 'fen-lgs', name: 'Fen Bilimleri', category: 'LGS' },

{ id: 'tur-lgs', name: 'Türkçe', category: 'LGS' },

{ id: 'sos-lgs', name: 'Sosyal Bilgiler', category: 'LGS' },

{ id: 'ing-lgs', name: 'İngilizce', category: 'LGS' },

{ id: 'din-lgs', name: 'Din Kültürü', category: 'LGS' }

],

slots: [

// Pazartesi

{ day: 1, start: '17:00', end: '18:30', subjectId: 'mat-lgs' },

{ day: 1, start: '19:00', end: '20:00', subjectId: 'tur-lgs' },

// Salı

{ day: 2, start: '17:00', end: '18:30', subjectId: 'fen-lgs' },

{ day: 2, start: '19:00', end: '20:00', subjectId: 'ing-lgs' },

// Çarşamba

{ day: 3, start: '17:00', end: '18:30', subjectId: 'mat-lgs' },

{ day: 3, start: '19:00', end: '20:00', subjectId: 'sos-lgs' },

// Perşembe

{ day: 4, start: '17:00', end: '18:30', subjectId: 'fen-lgs' },

{ day: 4, start: '19:00', end: '20:00', subjectId: 'din-lgs' },

// Cuma

{ day: 5, start: '17:00', end: '18:30', subjectId: 'mat-lgs' },

{ day: 5, start: '19:00', end: '20:00', subjectId: 'tur-lgs' },

// Cumartesi

{ day: 6, start: '10:00', end: '12:00', subjectId: 'mat-lgs' },

{ day: 6, start: '14:00', end: '16:00', subjectId: 'fen-lgs' },

// Pazar

{ day: 7, start: '10:00', end: '12:00', subjectId: 'sos-lgs' },

{ day: 7, start: '14:00', end: '16:00', subjectId: 'deneme-sinavi' }

]

},

{

id: 'yks-intensive',

name: 'YKS Son 3 Ay Yoğun',

description: 'Sınav öncesi yoğun çalışma programı',

category: 'YKS',

estimatedWeeklyHours: 50,

difficulty: 'Çok Yoğun',

tags: ['yks', 'yoğun', '12.sınıf', 'son-3-ay'],

subjects: [

{ id: 'mat-tyt', name: 'Matematik', category: 'TYT' },

{ id: 'fiz-tyt', name: 'Fizik', category: 'TYT' },

{ id: 'kim-tyt', name: 'Kimya', category: 'TYT' },

{ id: 'bio-tyt', name: 'Biyoloji', category: 'TYT' },

{ id: 'mat-ayt', name: 'Matematik', category: 'AYT' },

{ id: 'fiz-ayt', name: 'Fizik', category: 'AYT' },

{ id: 'kim-ayt', name: 'Kimya', category: 'AYT' },

{ id: 'bio-ayt', name: 'Biyoloji', category: 'AYT' }

],

slots: [

// Her gün sabah 06:00-08:00, öğlen 12:00-14:00, akşam 17:00-22:00

...generateIntensiveSlots()

]

}

];

function generateIntensiveSlots() {

const slots = [];

for (let day = 1; day <= 7; day++) {

slots.push(

{ day, start: '06:00', end: '08:00', subjectId: 'mat-tyt' },

{ day, start: '12:00', end: '14:00', subjectId: 'fiz-tyt' },

{ day, start: '17:00', end: '19:00', subjectId: 'mat-ayt' },

{ day, start: '19:30', end: '22:00', subjectId: 'fiz-ayt' }

);

}

return slots;

}

UI KOMPONENTİ:

// TemplateDialog.tsx

export function TemplateDialog({

studentId,

onApply

}: {

studentId: string;

onApply: (template: ScheduleTemplate) => void;

}) {

const [selectedTemplate, setSelectedTemplate] = useState<ScheduleTemplate | null>(null);

return (

<Dialog>

<DialogTrigger asChild>

<Button variant="outline">

<FileTemplate className="mr-2 h-4 w-4" />

Şablon Kullan

</Button>

</DialogTrigger>

<DialogContent className="max-w-4xl max-h-[80vh] overflow-y-auto">

<DialogHeader>

<DialogTitle>Program Şablonları</DialogTitle>

<DialogDescription>

Hazır program şablonlarından birini seçip hızlıca başlayabilirsiniz

</DialogDescription>

</DialogHeader>

<Tabs defaultValue="all">

<TabsList>

<TabsTrigger value="all">Tümü</TabsTrigger>

<TabsTrigger value="LGS">LGS</TabsTrigger>

<TabsTrigger value="YKS">YKS</TabsTrigger>

<TabsTrigger value="TYT">TYT</TabsTrigger>

<TabsTrigger value="AYT">AYT</TabsTrigger>

</TabsList>

<TabsContent value="all" className="space-y-4 mt-4">

{TEMPLATES.map(template => (

<Card

key={template.id}

className={cn(

"cursor-pointer transition-all",

selectedTemplate?.id === template.id && "ring-2 ring-primary"

)}

onClick={() => setSelectedTemplate(template)}

>

<CardHeader>

<div className="flex items-start justify-between">

<div>

<CardTitle>{template.name}</CardTitle>

<CardDescription>{template.description}</CardDescription>

</div>

<Badge variant={

template.difficulty === 'Kolay' ? 'secondary' :

template.difficulty === 'Orta' ? 'default' :

template.difficulty === 'Yoğun' ? 'warning' :

'destructive'

}>

{template.difficulty}

</Badge>

</div>

</CardHeader>

<CardContent>

<div className="space-y-3">

<div className="flex items-center gap-4 text-sm">

<span className="flex items-center gap-1">

<Clock className="h-4 w-4" />

{template.estimatedWeeklyHours} saat/hafta

</span>

<span className="flex items-center gap-1">

<BookOpen className="h-4 w-4" />

{template.subjects.length} ders

</span>

<span className="flex items-center gap-1">

<Calendar className="h-4 w-4" />

{template.slots.length} slot

</span>

</div>

<div className="flex flex-wrap gap-1">

{template.tags.map(tag => (

<Badge key={tag} variant="outline" className="text-xs">

{tag}

</Badge>

))}

</div>

{selectedTemplate?.id === template.id && (

<div className="mt-4 p-3 bg-muted rounded-md">

<h4 className="text-sm font-medium mb-2">Dersler:</h4>

<div className="flex flex-wrap gap-2">

{template.subjects.map(s => (

<Badge key={s.id}>{s.name} ({s.category})</Badge>

))}

</div>

<h4 className="text-sm font-medium mt-3 mb-2">Program Özeti:</h4>

<div className="text-xs text-muted-foreground">

{summarizeSchedule(template.slots)}

</div>

</div>

)}

</div>

</CardContent>

</Card>

))}

</TabsContent>

</Tabs>

<DialogFooter>

<Button

disabled={!selectedTemplate}

onClick={() => {

if (selectedTemplate) {

applyTemplate(selectedTemplate, studentId);

onApply(selectedTemplate);

}

}}

>

Şablonu Uygula

</Button>

</DialogFooter>

</DialogContent>

</Dialog>

);

}

// Şablonu uygula

async function applyTemplate(template: ScheduleTemplate, studentId: string) {

// 1. Dersleri oluştur/kontrol et

const existingSubjects = loadSubjects();

for (const subj of template.subjects) {

const exists = existingSubjects.find(s => s.id === subj.id);

if (!exists) {

await addSubject({

...subj,

code: subj.id,

description: `${template.name} şablonundan eklendi`

});

}

}

// 2. Mevcut slotları temizle (opsiyonel - kullanıcıya sor)

const shouldClear = await confirm('Mevcut programı temizleyip şablonu uygulamak ister misiniz?');

if (shouldClear) {

const currentSlots = getWeeklySlotsByStudent(studentId);

for (const slot of currentSlots) {

await removeWeeklySlot(slot.id);

}

}

// 3. Şablon slotlarını ekle

for (const slot of template.slots) {

await addWeeklySlot({

id: crypto.randomUUID(),

studentId,

...slot

});

}

toast.success(`${template.name} şablonu uygulandı!`);

}

function summarizeSchedule(slots: any[]) {

const byDay = {};

slots.forEach(s => {

if (!byDay[s.day]) byDay[s.day] = 0;

const duration = minutesBetween(s.start, s.end);

byDay[s.day] += duration;

});

const days = ['Pzt', 'Sal', 'Çar', 'Per', 'Cum', 'Cmt', 'Paz'];

return Object.entries(byDay)

.map(([day, mins]) => `${days[day-1]}: ${(mins/60).toFixed(1)}s`)

.join(', ');

}

ŞABLON YÖNETİMİ:

// Kullanıcının kendi şablonunu kaydetmesi

export async function saveAsTemplate(

studentId: string,

name: string,

description: string

) {

const slots = getWeeklySlotsByStudent(studentId);

const subjectIds = [...new Set(slots.map(s => s.subjectId))];

const subjects = loadSubjects().filter(s => subjectIds.includes(s.id));

const template: ScheduleTemplate = {

id: `custom-${Date.now()}`,

name,

description,

category: 'Custom',

subjects: subjects.map(s => ({

id: s.id,

name: s.name,

category: s.category || 'Custom'

})),

slots: slots.map(({ id, studentId, ...rest }) => rest),

estimatedWeeklyHours: weeklyTotalMinutes(studentId) / 60,

difficulty: calculateDifficulty(slots),

tags: ['özel', 'custom']

};

// Local storage'a kaydet

const customTemplates = JSON.parse(

localStorage.getItem('customTemplates') || '[]'

);

customTemplates.push(template);

localStorage.setItem('customTemplates', JSON.stringify(customTemplates));

toast.success('Şablon kaydedildi!');

}

---

Bu komutların her biri, yapay zekaya direkt vererek ilgili özelliği implement ettirebilirsiniz. Her komut:

1. ✅ \*\*Mevcut durumu\*\* açıklıyor

2. ✅ \*\*Veri yapılarını\*\* gösteriyor

3. ✅ \*\*Algoritmaları\*\* detaylı yazıyor

4. ✅ \*\*UI bileşenlerini\*\* kod ile veriyor

5. ✅ \*\*Entegrasyon adımlarını\*\* belirtiyor

Hangi özellikle başlamak istersiniz? Daha detaylı açıklama istediğiniz bir bölüm var mı?

3 minutes of work

Working

Ask questions, plan your work…

BuildPlanEdit

Preview

Agent

Agent can make changes, review its work, and debug itself automatically.