SIM Inventory Management (React + Windsurf)

Profesyonel bir **SIM Card Inventory Management** uygulamasını, Windsurf aracılığıyla sıfırdan (React + TypeScript) geliştirmek için uçtan uca plan ve hazır **prompt paketi**. Aşağıdaki akış, gerçek API (Swagger) ile entegre olup, erişim yoksa MSW ile **mock** modunda da çalışır.

**0) Kapsam & Demo Senaryoları (E2E)**

1. **Onboard SIM Batch (MVNE)**
   * Admin CSV ile fiziksel SIM ICCID veya eSIM profillerini içe aktarır.
   * Sistem benzersizlik/format validasyonlarını yapar, **SIM Resource** olarak kaydeder (status = Available).
2. **Search SIM Resources**
   * Admin; ICCID, IMSI, status, type (eSIM/Physical), batchId, tarih aralıklarına göre arar.
   * Sunucu tarafı pagination/sort/filters.
3. **Create SIM Order (MVNO)**
   * MVNO belirli SIM’leri tahsis etmek için **Order** oluşturur.
   * Order; SIM Resource referanslarını ve hedef lifecycle aksiyonunu içerir.
4. **Update SIM Lifecycle (MVNO)**
   * Activate, Suspend, Terminate, Release, Retire aksiyonları.
   * SIM status güncellenir, Order detail’de yansır.
5. **View Order Status**
   * Admin, Order detayında bağlı SIM kaynaklarının son durumlarını görür (timeline/aktivite log).
6. **Retire SIM Resource (MVNE)**
   * SIM’i Retired yapma veya Available ise **Delete** (guard: sadece Available iken silinebilir).

**Roller**: MVNE\_Admin, MVNO\_User (RBAC ile yetki kısıtları)

**1) Domain Model (TMF Resource Inventory esinli)**

**SIM Resource**

* id: string (UUID)
* iccid: string (unique, required)
* imsi?: string
* type: "Physical" | "eSIM"
* status: "Available" | "Allocated" | "Active" | "Suspended" | "Terminated" | "Retired"
* batchId?: string
* profileType?: "prepaid" | "postpaid" | string
* createdDate: string (ISO)
* lastModifiedDate: string (ISO)

**Order**

* id: string
* requestedAction: "Allocate" | "Activate" | "Suspend" | "Terminate" | "Release" | "Retire"
* resourceIds: string[] (SIM id listesi)
* status: "Pending" | "InProgress" | "Completed" | "Failed"
* createdDate, lastModifiedDate
* events: { at: string; status: string; message?: string }[]

**State Geçişleri (özet)**

* Available -> Allocated -> Active
* Active -> Suspended -> Active | Terminated
* Allocated -> Release -> Available
* Any -> Retired (terminal)
* **Delete**: sadece Available iken.

**Validasyon Kuralları**

* iccid benzersiz + zorunlu, type sadece Physical|eSIM, create’da status=Available default.

**2) Teknoloji & Mimarî**

* **Build**: Create React App (react-scripts) + React 18 + TypeScript
* **UI:** Ant Design v5, @ant-design/icons, kompakt toolbar ve Popover kullanımları
* **Veri yönetimi:** react-query v3 (QueryClient proje genelinde), axios
* **Form/Doğrulama:** antd Form; ek olarak react-hook-form ve yup kullanılabilir durumdadır
* **Kimlik Doğrulama:** Keycloak (`src/contexts/KeycloakContext.tsx`), token otomatik yenileme ve `apiService.setAuthToken`
* **Stil/Temalar**: Ant Design tema değişkenleri (CSS variables) üzerinden özelleştirme; tema paneli için `ThemeContext.tsx`
* **Routing**: React Router
* **Mock**: MSW (Service Worker) — Swagger erişim yoksa
* **Test**: Vitest + Testing Library (unit), Playwright (e2e – opsiyonel)
* **Kalite**: ESLint + Prettier + Husky (pre-commit)
* **CI/CD**: GitHub Actions; Dockerfile + multi-stage
* **Env**: .env içinde VITE\_API\_BASE\_URL, VITE\_USE\_MOCK

**Dizin Yapısı (öneri)**

src/

  App.tsx

  index.tsx

  i18n.ts

  routes/

    AppRoutes.tsx

  contexts/

    KeycloakContext.tsx

    ThemeContext.tsx

  components/

    layout/ (MainLayout, navbar/sidebar)

    common/ (LoadingSpinner, vb.)

  pages/

    dashboard/ (Dashboard.tsx)

    sim-resources/ (list, detail, create, batch import)

    sim-orders/ (list, detail, create)

    reports/ (Reports.tsx)

    auth/, user/

  services/

    api.service.ts

  types/

    sim.types.ts

  styles/ (global stiller)

  utils/

**3) API Entegrasyon Stratejisi**

**Swagger**: <https://dri-api.dnext-pia.com/api/resourceInventoryManagement/v4/swagger-ui/index.html>

Base URLs:

- Resource Inventory (TMF639): <https://dri-api.dnext-pia.com/api/resourceInventoryManagement/v4>

- Resource Ordering (TMF652): <https://dro-api.dnext-pia.com/api/resourceOrderingManagement/v4>

- Party Management (TMF632): https://dri-api.dnext-pia.com/api/partyManagement/v4

1. **- Kimlik doğrulama:** Keycloak token, axios interceptor ile `Authorization: Bearer <token>`
2. **Axios Instance & Interceptor**
   * Authorization bearer token ya da mTLS vs. gerekiyorsa buradan enjekte edilir.
   * Hata kodlarını (400/401/403/404/409/500) kullanıcı-dostu mesajlara map’leyin.
3. **Mock Modu (MSW)**
   * .env ile VITE\_USE\_MOCK=true ise MSW açılır.
   * Handlers: /sim-resources, /orders, /batches uçları için CRUD + lifecycle.

**4) UI Bilgi Mimarisi & Ekranlar**

**Rotalar**

* / Dashboard (istatistikler, hızlı aksiyonlar)
* /sims Liste & filtre & bulk işlemler
* /sims/:id Detay (status timeline, order ilişkileri)
* /orders Liste & oluşturma
* /orders/:id Detay (linked SIMs + event log)
* /batches Liste & onboarding geçmişi
* /onboarding Batch Upload Wizard (CSV)
* /admin Ayarlar, profil, RBAC görünümü

**Temel Bileşenler**

* SimTable, SimFilters, StatusBadge, LifecycleMenu
* BatchUploadDialog (CSV şablon indirme + örnek validasyon)
* OrderForm (SIM seçimi + hedef aksiyon)
* ConfirmDialog, Toast (optimistic updates + undo)

**UX Notları**

* Büyük listeler için **server-side pagination**.
* Filtreler URL query string’e yazılsın (paylaşılabilir link).
* Hata/empty/loading durum kartları.
* Erişilebilirlik (A11y), klavye navigasyonu.

**5) Kurulum Komutları (öneri)**

# Proje iskeleti

npm create vite@latest sim-inventory -- --template react-ts

cd sim-inventory

# Kitler

npm i axios @tanstack/react-query react-router-dom \

zod react-hook-form \

class-variance-authority clsx \

@radix-ui/react-avatar @radix-ui/react-dialog @radix-ui/react-dropdown-menu \

lucide-react

# Tailwind

npm i -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

# shadcn/ui

npm i -D shadcn-ui@latest

npx shadcn-ui@latest init

# component eklerken: npx shadcn-ui@latest add button input table dialog toast badge

# OpenAPI codegen (opsiyonel)

npm i -D openapi-typescript-codegen

# MSW + Testler

npm i -D msw vitest @testing-library/react @testing-library/jest-dom @types/jest \

playwright @playwright/test eslint prettier husky lint-staged

# Git hook

npx husky init

**tailwind.config.cjs**: content: ["./index.html","./src/\*\*/\*.{ts,tsx}"]

**.env** (ör.)

VITE\_API\_BASE\_URL=https://dri-api.dnext-pia.com/api/resourceInventoryManagement/v4

VITE\_USE\_MOCK=false

**6) React Query – Query Key & Cache Stratejisi**

* ['sims', {page, q, status, type, batchId }] - `['simResources', searchCriteria, currentPage, pageSize]` – liste
* ` ['simResource', id] ` – detay
* ` ['simStatistics', token] ` – istatistikler
* ` ['recentSims', token] `, `['recentOrders', token]` – dashboard listeleri (default sıralama uygulanır)
* ` ['simOrders', token, searchParams, currentPage, pageSize]` – sipariş listesi

**Invalidation:**

* - Lifecycle/Allocation/Import sonrası: `simResource`, `simResources`, `simStatistics`
* - Sipariş işlemleri sonrası: `simOrders`

**7) CSV Şablonu (Batch Onboarding)**

Kolonlar: iccid,imsi,type,profileType,batchId

Örnek:

iccid,imsi,type,profileType,batchId

8937201001234567890,286011234567890,Physical,prepaid,2025-09-01-A

8937201001234567891,,eSIM,postpaid,2025-09-01-A

Validasyonlar:

* iccid zorunlu + regex/length kontrolü (genelde 19–20 hane Luhn varyantı)
* type ∈ {Physical,eSIM}
* imsi opsiyonel; varsa numeric/length

**8) Hata Haritalama (Örnek)**

* **400**: Validasyon – kullanıcıya form alanı bazlı göster.
* **401/403**: Oturum/Yetki uyarısı, tekrar giriş.
* **404**: Kayıt bulunamadı.
* **409**: Çakışma (duplicate ICCID) – onboarding sırasında satır işaretleme.
* **500**: Beklenmeyen – retry öner.

**9) Güvenlik & RBAC**

* Context’te role sakla (MVNE\_Admin vs.).
* UI, role’a göre aksiyonları gizler/disable eder (örn. Delete sadece Admin ve status=Available iken).
* Token yenileme (silent refresh) interceptor’da.

**10) Prompt Paketi**

**🏷 Prompt [No] — [Başlık]**

**🎯 Amaç**  
👉 Bu prompt ile hedeflenen işin özeti

**📌 Kapsam**

* Yapılacak iş maddeler halinde

**🛠 Teknik Gereksinimler**

* Dosya yolları
* Kütüphaneler
* Yapı adları

**🔗 Entegrasyon Kuralları**

* API
* Auth & RBAC
* Cache stratejileri

**✔ Validasyon & İş Kuralları**

* Veri doğrulama
* Edge-case yönetimi

**🎨 UI/UX Beklentileri**

* Hata/boş/loading durumları
* Erişilebilirlik

**✅ Test & Kabul Kriterleri**

* Ölçülebilir, test edilebilir koşullar

**⏳ Bağımlılıklar**

* Önce tamamlanması gerekenler

**📦 Çıktı Formatı**

* AI’den beklenen dosya/kod/komut listesi

🏷 Prompt 01 — Proje İskeleti & Temel Kurulum

🎯 Amaç  
Vite tabanlı TypeScript React projesi kur ve temel proje iskeletini oluştur.

📌 Kapsam

* Proje adı: sim-inventory
* Tailwind, shadcn/ui, React Router, React Query, React Hook Form, Zod kurulumu
* Dosya ağacının hazırlanması ve içeriklerinin doldurulması

🛠 Teknik Gereksinimler

* Dosyalar:
  + src/app/providers/QueryProvider.tsx
  + src/app/providers/ThemeProvider.tsx
  + src/lib/axios.ts
  + src/styles/globals.css
  + src/app/router/index.tsx
  + src/components/ui/\*
  + src/pages/Home.tsx
  + src/main.tsx

🔗 Entegrasyon Kuralları

* Axios instance baseURL: import.meta.env.VITE\_API\_BASE\_URL
* QueryProvider içinde React Query devtools opsiyonel

✔ Validasyon & İş Kuralları

* Yok (ilk kurulum)

🎨 UI/UX Beklentileri

* Varsayılan tema ile dummy Home sayfası açılır

✅ Test & Kabul Kriterleri

* npm run dev ile proje açılır
* Ana sayfa ve temel layout görülür

⏳ Bağımlılıklar

* Yok (ilk adım)

📦 Çıktı Formatı

* Tam kod dosyaları

🏷 Prompt 02 — UI Kiti & Tema

🎯 Amaç  
Kurumsal tema ve temel UI bileşenlerini eklemek.

📌 Kapsam

* Navbar, Sidebar, Content layout
* StatusBadge, EmptyState, ErrorState, LoadingSpinner

🛠 Teknik Gereksinimler

* Ant Design v5, shadcn/ui
* Renkler: kurumsal minimal tema

🔗 Entegrasyon Kuralları

* Tema ThemeProvider ile yönetilmeli

✔ Validasyon & İş Kuralları

* Yok

🎨 UI/UX Beklentileri

* / sayfasında örnek status badgeleri
* Boş içerik kartları

✅ Test & Kabul Kriterleri

* Layout bileşenleri görünür
* Badge ve boş kart örnekleri çalışır

⏳ Bağımlılıklar

* Prompt 01 tamamlanmalı

📦 Çıktı Formatı

* UI component dosyaları

🏷 Prompt 03 — OpenAPI Client Entegrasyonu

🎯 Amaç  
Swagger JSON’dan OpenAPI client üretmek.

📌 Kapsam

* Client src/lib/api altında
* apiClient.ts export noktası
* Axios interceptor: Authorization + hata mapping

🛠 Teknik Gereksinimler

* openapi-typescript-codegen
* Axios + token injection

🔗 Entegrasyon Kuralları

* Keycloak token Authorization header’a eklenecek
* Hata kodları user-friendly mesajlara map’lenmeli

✔ Validasyon & İş Kuralları

* Yok

🎨 UI/UX Beklentileri

* Hata mesajları kullanıcı dostu olmalı

✅ Test & Kabul Kriterleri

* src/lib/api içinde modeller oluşur
* health endpoint çağrısı başarılı

⏳ Bağımlılıklar

* Prompt 01

📦 Çıktı Formatı

* Client dosyaları + axios interceptor

🏷 Prompt 04 — SIM Listesi (Search + Pagination)

🎯 Amaç  
SIM kaynaklarını listeleyen ekran oluşturmak.

📌 Kapsam

* /sims rotası
* SimTable, SimFilters, useSimQueries hook
* Filtre: iccid, imsi, status, type, batchId
* Server-side pagination

🛠 Teknik Gereksinimler

* Query key: ['sims', {page,q,status,type,batchId}]
* React Query + Ant Design Table

🔗 Entegrasyon Kuralları

* API: /sim-resources
* URL query string senkronize edilmeli

✔ Validasyon & İş Kuralları

* ICCID format kontrolü

🎨 UI/UX Beklentileri

* Boş/hata/loading state kartları

✅ Test & Kabul Kriterleri

* Mock API ile 100+ kayıt listelenir
* Filtreleme & sayfalama çalışır

⏳ Bağımlılıklar

* Prompt 03

📦 Çıktı Formatı

* SimList.tsx, useSimQueries.ts

🏷 Prompt 05 — SIM Detay & Lifecycle Aksiyonları

🎯 Amaç  
SIM detay ekranı ve lifecycle işlemleri.

📌 Kapsam

* /sims/:id rotası
* SimDetail + timeline
* LifecycleMenu: Activate, Suspend, Terminate, Release, Retire

🛠 Teknik Gereksinimler

* Optimistic update + rollback
* Toast uyarıları

🔗 Entegrasyon Kuralları

* API: /sim-resources/:id
* RBAC ile kontrol

✔ Validasyon & İş Kuralları

* Retire her durumda
* Delete sadece Available

🎨 UI/UX Beklentileri

* Status badge ve timeline görselleştirilmeli

✅ Test & Kabul Kriterleri

* Status değişiklikleri UI’da görünür
* Hata → rollback yapılır

⏳ Bağımlılıklar

* Prompt 04

📦 Çıktı Formatı

* SimDetail.tsx, LifecycleMenu.tsx

🏷 Prompt 06 — Batch Onboarding Wizard (CSV Upload)

🎯 Amaç  
CSV ile toplu SIM onboarding.

📌 Kapsam

* /onboarding wizard

1. CSV yükle
2. Önizleme + validasyon
3. Sunucuya gönder

🛠 Teknik Gereksinimler

* MSW mock desteği
* Validasyon: duplicate ICCID

🔗 Entegrasyon Kuralları

* API: /batches
* 409 duplicate handling

✔ Validasyon & İş Kuralları

* ICCID regex
* Hatalı satır kırmızı işaretlenir

🎨 UI/UX Beklentileri

* Hata durumları kullanıcıya açıkça gösterilir

✅ Test & Kabul Kriterleri

* 2 başarılı, 1 hatalı satır örneği çalışır

⏳ Bağımlılıklar

* Prompt 03

📦 Çıktı Formatı

* BatchWizard.tsx

🏷 Prompt 07 — Order Oluşturma (MVNO)

🎯 Amaç  
Order oluşturma ekranı eklemek.

📌 Kapsam

* /orders sayfası
* SIM çoklu seçim
* RequestedAction seçimi
* Order detaya yönlendirme

🛠 Teknik Gereksinimler

* Mock event stream

🔗 Entegrasyon Kuralları

* API: /orders
* Status değişimi → refetch

✔ Validasyon & İş Kuralları

* Order sadece geçerli SIM’lerle

🎨 UI/UX Beklentileri

* Order status akışı (Pending → InProgress → Completed/Failed)

✅ Test & Kabul Kriterleri

* Akış simüle edilir

⏳ Bağımlılıklar

* Prompt 04–06

📦 Çıktı Formatı

* OrderForm.tsx

🏷 Prompt 08 — Retire & Delete Guard’ları

🎯 Amaç  
Güvenli Retire/Delete işlemleri.

📌 Kapsam

* Retire: her SIM
* Delete: sadece Available + MVNE\_Admin

🛠 Teknik Gereksinimler

* Confirm dialog
* Role kontrolü

🔗 Entegrasyon Kuralları

* API hata mapping
* RBAC enforced

✔ Validasyon & İş Kuralları

* Status kontrolü
* Role kontrolü

🎨 UI/UX Beklentileri

* Guarded butonlar

✅ Test & Kabul Kriterleri

* RBAC kuralları doğrulanır

⏳ Bağımlılıklar

* Prompt 05

📦 Çıktı Formatı

* DeleteGuard.tsx

🏷 Prompt 09 — Order Detayı & Event Timeline

🎯 Amaç  
Order detay ekranı ve timeline.

📌 Kapsam

* /orders/:id
* Event timeline + SIM listesi

🛠 Teknik Gereksinimler

* refetchInterval=5s

🔗 Entegrasyon Kuralları

* API: /orders/:id

✔ Validasyon & İş Kuralları

* Status ilerleyişi mock event ile

🎨 UI/UX Beklentileri

* Completed → success alert
* Failed → retry butonu

✅ Test & Kabul Kriterleri

* Event timeline çalışır

⏳ Bağımlılıklar

* Prompt 07

📦 Çıktı Formatı

* OrderDetail.tsx

🏷 Prompt 10 — Auth Mock & RBAC

🎯 Amaç  
Rol bazlı mock auth eklemek.

📌 Kapsam

* Login ekranı: kullanıcı + rol
* Roller: MVNE\_Admin, MVNO\_User

🛠 Teknik Gereksinimler

* Context ile role saklama

🔗 Entegrasyon Kuralları

* Role göre aksiyon gizleme

✔ Validasyon & İş Kuralları

* Delete sadece Admin görür

🎨 UI/UX Beklentileri

* Basit login ekranı

✅ Test & Kabul Kriterleri

* MVNO\_User Delete butonunu göremez

⏳ Bağımlılıklar

* Prompt 01

📦 Çıktı Formatı

* AuthContext.tsx, Login.tsx

🏷 Prompt 11 — Testler & MSW

🎯 Amaç  
Unit, component ve e2e testler eklemek.

📌 Kapsam

* Unit: Zod validasyon, hooks query keys
* Component: SimTable + filtre
* E2E: Playwright ile liste, filtre, lifecycle

🛠 Teknik Gereksinimler

* Vitest + Testing Library + Playwright

🔗 Entegrasyon Kuralları

* MSW mock kullanılacak

✔ Validasyon & İş Kuralları

* Test senaryoları UI ile uyumlu

🎨 UI/UX Beklentileri

* Yok

✅ Test & Kabul Kriterleri

* npm test başarılı
* En az 1 e2e geçer

⏳ Bağımlılıklar

* Prompt 04–10

📦 Çıktı Formatı

* Test dosyaları

🏷 Prompt 12 — CI/CD + Docker

🎯 Amaç  
CI/CD ve Docker entegrasyonu.

📌 Kapsam

* Dockerfile (multi-stage)
* GitHub Actions (build + test)
* Preview deploy

🛠 Teknik Gereksinimler

* Docker
* GitHub Actions

🔗 Entegrasyon Kuralları

* Pipeline testleri çalıştırmalı

✔ Validasyon & İş Kuralları

* Yok

🎨 UI/UX Beklentileri

* Yok

✅ Test & Kabul Kriterleri

* docker build . başarılı
* CI pipeline geçer

⏳ Bağımlılıklar

* Prompt 01–11

📦 Çıktı Formatı

* Dockerfile
* .github/workflows/\*.yml

**11) Kabul Kriterleri (Özet)**

* **Performans**: Liste sayfası 10k kayıt mock’ta akıcı scroll (sanallaştırma opsiyonel: @tanstack/react-virtual).
* **Erişilebilirlik**: Form alanları label’lı; kontrast uygun; klavye ile erişilebilir.
* **Hata Yönetimi**: API hataları kullanıcı anlayacağı dille; tekrar dene imkanı.
* **Güvenlik**: Yetkisiz aksiyonlar UI’da mümkün değil; token var ise header’a eklenir.

**12) Riskler & Alternatifler**

* Swagger erişimi kısıtlı olabilir → **MSW** ile offline geliştirme.
* OpenAPI şeması tam değil → zod ile UI-level ek validasyon.
* Lifecycle akışları farklı iş kuralları gerektirebilir → Hook’larda guard fonksiyonları.

**13) Sonraki Adımlar**

1. Prompt 01’i çalıştır, repo’yu oluştur.
2. Prompt 03 ile OpenAPI client’i üret (Swagger JSON URL hazırla).
3. Prompt 04–06 ile temel işlevleri tamamla; 07–09 ile sipariş/lifecycle akışını bitir.
4. 10–12 ile test/CI ve dağıtım.

**14) Ekstra Komutlar**

🏷 Prompt 01 — Profesyonel i18n Otomatik Çeviri Yapısı  
🎯 Amaç  
👉 React projesinde çoklu dil desteğini profesyonel şekilde kurmak ve tr.json’u temel dil olarak diğer dillere otomatik çevirmek.

📌 Kapsam  
• react-i18next yapılandırması  
• OpenAI SDK ile otomatik çeviri script’i (translate.js)  
• src/locales altındaki JSON dil dosyalarının yönetimi  
• GitHub Actions ile otomatik çeviri pipeline’ı

🛠 Teknik Gereksinimler  
• Dosya yolları:

* src/locales/tr.json
* src/locales/en.json
* src/locales/de.json
* src/locales/fr.json
* src/translate.js
* src/i18n.js
* .github/workflows/translate.yml  
  • Kütüphaneler: react-i18next, i18next, openai, dotenv  
  • Yapı adları: i18n, translate, workflow

🔗 Entegrasyon Kuralları  
• API: OpenAI resmi SDK, .env içindeki OPENAI\_API\_KEY  
• Auth & RBAC: Proje kullanıcı auth gerekmiyor  
• Cache stratejileri: JSON dosyaları local dosya sistemi

✔ Validasyon & İş Kuralları  
• Dil dosyaları boş veya eksik anahtarları kontrol et  
• Çeviri tamamlanamazsa hata logla

🎨 UI/UX Beklentileri  
• React tarafında dil seçimi dropdown veya buton ile değiştirilebilmeli  
• Hata/boş/loading durumları console veya toast ile bildirilir

✅ Test & Kabul Kriterleri  
• tr.json güncellenince diğer dil dosyaları otomatik oluşturulmalı  
• Workflow tamamlandığında commit edilmiş dosyalar GitHub’da gözükmeli  
• React sayfasında dil değişimi sorunsuz çalışmalı

⏳ Bağımlılıklar  
• .env dosyasındaki OPENAI\_API\_KEY hazır olmalı  
• tr.json minimum içerik ile dolu olmalı

📦 Çıktı Formatı  
• translate.js  
• i18n.js  
• Örnek tr.json ve en.json  
• GitHub Actions translate.yml

🏷 Prompt 02 — Tema Ayarları Sidebar & Dinamik Renkler  
🎯 Amaç  
👉 Kullanıcının uygulama temasını özelleştirebileceği bir sidebar ve renk seçim arayüzü oluşturmak.

📌 Kapsam  
• Navbar, Sidebar, Buton, Sayfa ve Yazı renklerini değiştirme  
• Color picker veya hazır renk dropdown kullanımı  
• Seçilen renklerin localStorage’da saklanması  
• Varsayılana döndür butonu

🛠 Teknik Gereksinimler  
• Dosya yolları: src/components/ThemeSidebar.jsx veya .tsx  
• Kütüphaneler: React, React Hooks, TailwindCSS veya plain CSS  
• Yapı adları: ThemeSidebar, useTheme, CSS variables

🔗 Entegrasyon Kuralları  
• Dinamik CSS değişkenleri ile tema uygulanacak  
• UI sidebar mevcut layout ile uyumlu olacak

✔ Validasyon & İş Kuralları  
• Color picker boş geçilemez, geçersiz renk engellenir  
• localStorage okunamazsa varsayılan renkler uygulanır

🎨 UI/UX Beklentileri  
• Sidebar sade ve kullanıcı dostu  
• Renk seçimi anında sayfaya yansır  
• Hata/boş/loading durumları minimal feedback ile gösterilir  
• Erişilebilirlik: kontrast ve focus durumları sağlanır

✅ Test & Kabul Kriterleri  
• Renk değişiklikleri sayfa yenilense bile korunmalı  
• Varsayılana döndür butonu doğru çalışmalı  
• Her component CSS değişkenlerini doğru okuyup uygular

⏳ Bağımlılıklar  
• React projesi ve functional component desteği  
• CSS variable desteği

📦 Çıktı Formatı  
• ThemeSidebar.jsx  
• useTheme.js (hook)  
• Örnek CSS değişkenleri ve default renk seti