Dinero Case

Merhabalar, case çalışmamı tamamladım. React (Vite), Typescript, TailwindCSS, shadcn-ui kullanarak geliştirdim.

Projeyi Başlatmak:

1 - npm install: Proje bağımlılıklarını yüklüyoruz

2 - npm run dev: Projeyi başlatıyoruz

Projenin yapısında api-store kalsörü içerisinde index.ts adında bir dosya bulunuyor. Bu dosya istekleri handle ettiğimiz kısım. Backend api ye güvenli ve tutarlı bir istek göndermemizi sağlıyor. Tüm olası senaryolarıda tüm istekler için buradan yönetebiliriz. Aynı dizinde bulunan global kalsörü içerisinde ise gerekli endpointler barınıyor.

Dark/Light mode özelliğini ekledim. hooks klasörü içerisine yazdığım basit hook ve provider içerisindeki theme provider ile bunu yönetebiliyoum. Dark/Light mode özelliğinin daha kolay çalışması içinde index.css dosyası içerisinde light ve dark classı için aynı değişkene ait renk kodunu özelleştirdim.

Form validasyonu için zod kullandım, react-hook-form ile birlikte. Bu sayede useState kullanımını azalttım ve tek bir yerden tüm custom input elementlerime prop vererek formu yönettim. Components > custom > form-elements klasörü içeriisnde form için custom elementler var. input, phone-number, upload-file, salary-expectation gibi. Bu sayede reusable component kullamınıza önem verdim ve bu elementleri başka bir form içerisinde tekrar daha az kod yazarak yönetebileceğiz.

Shadcn-ui kullanma amacımda her web sitesinde oluşabilecek temel component ihtiyaçlarını karşılayabilmekti. Shadcn içerisindeki componentleri projeminizin

gereksinimlerine göre özelleştirerek daha temiz ve az kod yazarak bu ihtiyacı karşılamış oldum.

Form submit edilirken mutation kullandım. Bunun da sebebi success, error ve isPending durumunu tek bir yerden yönetebilmek.

Lottie animasyonlarını form pending durumunda loading animasyonunu butonun içerisinde olacak şekilde konumlandırdım. Başarılı durumda ise form alanı yerinde çıkacak olan success animation'ı ekledim. Altında ise tekrar başvuru formuna ulaşabileceği bir buton bulunmakta.

Error handling'i çoğu senaryo için uyguladım.

CV havuzu oluşturdum. Eklenen cv'ler ana dizinde uploads/cv içerisine ekleniyor. Ancak bunun için ayrı bir server.cjs dosyası oluşturdum ve localhost:3001 de çalışıyor. React projesi ile beraber çalıştırılması lazım. Local'de node server.cjs diye çalıştırıldıktan sonra test edilebilir. Vite projesinin kendi içerisinde bunu gerçekleştiremediğim için bu şekilde bir çözüm buldum. server.cjs 3001 portunda çalışıyor.

Mobil, tablet ve desktop olmak üzere responsive tasarım olacak şekilde geliştirdim.

Not: Projede modules yapısı kullandım ve bu yapı özellikle kapsamlı projelerde tercih ettiğim bir yapıdır. Özellikle NextJs projelerimde app klasörü içerisindeki page.tsx dosyasının **server-side rendering** yapısını bozmamak adına modules altında ilgili sayfa ile bir kalsör açıp ilgili tüm componentleri oraya yazmaya dikkat ederim. Örneğin auth sayfalarımız olsun. auth/sign-in, auth/sign-up ve auth/verify. Burada şu yolu izlerim;

modules > auth > ui > views > signin-view.tsx - signup-view.tsx - verify-view.tsx

modules > auth > ui > components > ilgili componentler

modules > auth > ui > sections > ilgili sections

modules > auth > server/procedures.ts ⇒ authProcedures için tRPC isteklerimiz.

Bu views leri page.tsx içerisine import edip kullanırım ve metaData gibi SEO açısından daha dinamik ve yönetilebilir bir yapı kurmuş oluyorum. Ayrıca, bir isteği prefect etmek ya da URL parametrelerinden gelen değeri promise ile hazırlamak gibi durumlarda, page.tsx dosyasının yapısı bozulmadığı için bir sorun yaşamıyorum.

NOT: File uploads için server.cjs 3001 protunda çalıştığı için vercelde dosya yükleme çalışmaz. Ayrı bir sunucuda (render.com vs) bunu deploy edebilirim ancak genelde çok yavaş çalışıyorlar ve onun için ayrı bir backend projesi ayağa kaldırmanın pek mantıklı olmayacağını düşündüm. Ancak localde test edilebilir.

Github: https://github.com/gorkemyagci/dinero-case

Vercel: https://dinero-case.vercel.app