



SOFTITO YAZILIM BİLİŞİM AKADEMİSİ

4. DÖNEM 1. GRUP

FRONTEND BİTİRME PROJESİ RAPORU

CodeChallenge

İnteraktif Kodlama ve Eğitim Platformu

HAZIRLAYAN

Hale Alacan

DEĞERLENDİRME KURULU

softlto Frontend Eğitim Kadrosu

Temmuz, 2026

softlto Yazılım Bilişim Akademisi, İstanbul

İÇİNDEKİLER

1. GİRİŞ	1
1.1 Projenin Amacı	1
1.2 Proje Kısıtları	1
2. MATERYAL VE YÖNTEM	2
2.1 Kullanılan Teknolojiler ve Kütüphaneler	2
2.2 Mimari Yapı ve Durum Yönetimi (State Management)	2
2.3 Rol Tabanlı Yetkilendirme ve Sistem Rollerini	3
3. UYGULAMA VE MODÜLLER	4
3.1 Projenin Kurulumu ve Geliştirme Ortamı	4
3.2 Arayüz Tasarımları ve Sayfa Modülleri	4
4. SONUÇ	7
5. KAYNAKÇA	8

1. GİRİŞ

Yazılım geliştirme eğitimi günümüzde teorik anlatımlardan ziyade pratik uygulamaya, yani yazarak öğrenmeye (learn-by-doing) kaymaktadır. Öğrencilerin öğrendikleri teorik bilgileri tarayıcı üzerinde anında test edebilmeleri, kod yazıp gerçek zamanlı geri bildirim almaları öğrenme kalıcılığını büyük ölçüde artırmaktadır.

Bu bitirme projesi kapsamında geliştirilen **CodeChallenge** platformu, özellikle Frontend (Ön Yüz) teknolojilerini öğrenmek isteyen öğrencilere modern, akıcı, interaktif ve oyunlaştırılmış bir öğrenme deneyimi sunmak amacıyla tasarlanmıştır.

1.1 Projenin Amacı

CodeChallenge, kullanıcıların HTML, CSS ve JavaScript teknolojilerini tarayıcılarından çıkmadan, herhangi bir kod editörü veya derleyici kurmalarına gerek kalmadan öğrenebilecekleri ve kendilerini test edebilecekleri tek sayfalı bir web uygulamasıdır (SPA).

Projenin temel hedefleri şunlardır:

- **Pratik Odaklı Eğitim:** Kullanıcılara teorik ders içeriklerinin hemen ardından o dersle ilgili canlı bir kodlama görevi sunarak bilgiyi pekiştirmek.
- **Akıllı Doğrulama Sistemi:** Öğrencilerin yazdığı kodları dinamik kurallar ve fonksiyon testleri ile tarayıcı üzerinde anlık olarak analiz edip doğru/yanlış değerlendirmesi yapmak.
- **Kapsamlı Rol Yapısı:** Sistemde sadece öğrenciler değil, öğrencilerin takibini yapıp ödev/görev atayan Mentorlar ve tüm sistemi denetleyen Adminler için de özel arayüzler sunmak.
- **Oyunlaştırma ve Motivasyon:** Grafikselleştirilmiş göstergeler, tamamlanma yüzdeleri, kazanılan başarı sertifikaları ve puanlama sistemleriyle öğrenme sürecini teşvik etmek.

1.2 Proje Kısıtları

Projenin geliştirilme sürecinde karşılaşılan ve kabul edilen teknik sınırlar şunlardır:

- İstemci Tarafı Simülasyonu:** Uygulama, backend (arka yüz) veri tabanı bağlantılarını simüle etmek amacıyla verileri tarayıcının `localStorage` alanında saklamakta ve durum yönetimini (state management) istemci tarafında gerçekleştirmektedir.
- Kod Değerlendirme Limitleri:** JavaScript derleme ve test süreçleri tarayıcı JavaScript motoru üzerinde dinamik fonksiyon oluşturma yöntemiyle gerçekleştirilmektedir. Bu nedenle gelişmiş güvenlik kısıtlamaları (sandbox) tarayıcı sınırlarıyla kısıtlıdır.
- Responsive Tasarım Kısıtı:** Kod editörü ve canlı önizleme gibi karmaşık bileşenler geniş ekran alanı gerektirdiğinden, mobil görünümde bazı düzenlemeler yapılmış olsa da ana kodlama deneyimi masaüstü ve tablet cihazlar için optimize edilmiştir.

2. MATERYAL VE YÖNTEM

2.1 Kullanılan Teknolojiler ve Kütüphaneler

Projede güncel ve modern web teknolojileri tercih edilmiştir:

- **React (v19.0):** Kullanıcı arayüzü bileşen tabanlı, performanslı ve reaktif bir yapıyla kurgulanmıştır.
- **Vite (v6.2):** Hızlı geliştirme (HMR) ve optimize edilmiş üretim derlemeleri için yeni nesil araç zinciri olarak kullanılmıştır.
- **Redux Toolkit (v2.12) & React-Redux (v9.3):** **** Uygulama genelinde kullanıcı oturumu, ders ilerlemeleri, sohbet geçmişleri, sınav sonuçları ve mentor atamaları gibi karmaşık durumların tek bir merkezden yönetilmesini sağlamıştır.**
- **Tailwind CSS (v4.1):** Projenin modern, şık ve responsive arayüz tasarımlarında hızlı ve esnek stil yönetimi sağlamak amacıyla Tailwind CSS tercih edilmiştir.
- **Motion (v12.23):** Sayfa geçişleri, modal pencereler ve ders tamamlama animasyonlarının pürüzsüz ve premium hissettirmesi amacıyla entegre edilmiştir.
- **Lucide React (v0.546):** Kullanıcı dostu ve sade ikon tasarımları için kullanılmıştır.
- **Express & Node.js:** Sunucu taraflı prototipleme ve yerel dağıtım testleri için altyapı olarak entegre edilmiştir.

2.2 Mimari Yapı ve Durum Yönetimi (State Management)

Uygulamanın kalbini oluşturan durum yönetimi, Redux Toolkit mimarisiyle kurulmuştur. `src/store/index.js` dosyası içerisinde tanımlanan ana "slice" şu temel veri yapılarını yönetir:

```
// store/index.js durum şablonu
const initialState = {
  user: {
    name: 'Alex Dev',
    email: 'dev@codechallenge.io',
    role: 'user', // user | mentor | admin
    registeredCourses: {}, // 'html-temelleri': 85 vb.
    completedTests: {}
  },
  courses: [], // Tüm kurslar, müfredatlar ve tamamlanma durumları
  tests: [], // Sistemdeki sınavlar ve soru sayıları
  testQuestions: {}, // Sınav sorularının detayları
  platformUsers: [], // Admin/Mentorların yönettiği kullanıcı listesi
  chatMessages: {}, // Kullanıcılar arası sohbet kayıtları (email bazlı)
  mentorTasks: [] // Mentorların öğrencilere atadığı ödevler
};
```

Uygulama, çoklu sekme (multi-tab) senkronizasyonunu desteklemek için tarayıcının `window.addEventListener('storage', ...)` olayını dinler. Böylece bir sekmede tamamlanan bir kodlama görevi veya gönderilen bir mesaj, açık olan diğer tüm sekmelerde anında güncellenir.

2.3 Rol Tabanlı Yetkilendirme ve Sistem Roller

Uygulamada 3 farklı kullanıcı rolü tanımlanmıştır. Kullanıcının giriş yaptığı e-posta adresine ve hesabına tanımlı `role` değerine göre arayüz dinamik olarak şekillenir:

- **Öğrenci (User):** Kursları inceler, satın alır, dersleri çalışır, kodlama egzersizlerini çözer, sınavlara girer, sertifika kazanır ve mentoruyla sohbet eder.
- **Mentor (Eğitmen):** Kendisine atanan öğrencilerin listesini görür. Öğrencilerin hangi kursta ne kadar ilerlediğini, sınavlardan kaç puan

aldığını grafiklerle analiz eder. Öğrencilere yeni ödevler/görevler atayabilir ve onlarla canlı sohbet edebilir.

- **Yönetici (Admin):** Sistemdeki toplam öğrenci, mentor, aktif ders ve ciro gibi metrikleri izler. Yeni ders ekleme, kullanıcı rolleri değiştirme, aktif/pasif durumlarını yönetme yetkisine sahiptir.

3. UYGULAMA VE MODÜLLER

3.1 Projenin Kurulumu ve Geliştirme Ortamı

Projenin yerel bilgisayarda çalıştırılması oldukça basittir. Node.js yüklü bir sistemde aşağıdaki adımlar izlenir:

3.1.1 Bağımlılıkların Yüklenmesi

Terminal veya PowerShell üzerinden proje kök dizinine gidilerek bağımlılıklar yüklenir:

```
npm install
```

3.1.2 Geliştirme Sunucusunun Başlatılması

Vite geliştirme sunucusu varsayılan olarak 3000 portunda çalışacak şekilde yapılandırılmıştır. Projeyi ayağa kaldırmak için:

```
npm run dev
```

Sunucu başarıyla başladığında tarayıcıdan `http://localhost:3000` adresine gidilerek uygulamaya erişilir.

3.2 Arayüz Tasarımları ve Sayfa Modülleri

3.2.1 Kimlik Doğrulama ve Kayıt (Login & Register)

`Login.jsx` kullanıcının sisteme e-posta ve şifresiyle giriş yapmasını sağlar. Giriş anında girilen e-postaya göre rol tespiti yapılır. `Register.jsx` ise yeni

kullanıcıların kaydını yaparak varsayılan öğrenci yetkisiyle sisteme dahil olmasını sağlar.

3.2.2 Öğrenci Paneli (Dashboard)

Giriş yapan öğrencileri karşılayan ilk ekrandır. Bu panelde; genel ilerleme yüzdeleri, tamamlanan ders ve test oranları, aktif kayıtlı kurs kartları ve ilerleme çubukları, kazanılan sertifikalar ve mentor tarafından atanmış son teslim tarihi bulunan aktif ödevler gösterilir.

3.2.3 Kurs Keşfi ve Detayları (Courses & CourseDetail)

`Courses.jsx` platformdaki tüm eğitimleri listeler ve isim aramasına göre anlık süzme yapar. `CourseDetail.jsx` ise ders müfredatını listeler; satın alınmamış eğitimler için ödeme sayfasına yönlendirme sağlar.

3.2.4 İnteraktif Kodlama Playground'u (InteractivePlayground)

Ekran sol panel (talimatlar ve görev), orta panel (kod editörü ve simüle edilmiş çıktı konsolu) ve sağ panel (yazılan HTML/CSS kodlarının canlı `iframe` önizlemesi) olmak üzere üç bölümden oluşur.

Sistem, HTML/CSS görevlerinde Regex ile kod yapılarını kontrol ederken; JavaScript görevlerinde `new Function` oluşturarak öğrencinin yazdığı fonksiyonu belirli test parametreleriyle çalıştırır. Örn:

```
validate: (code) => {
  try {
    const fn = new Function(code + '; return faktoriyel;');
    const faktoriyel = fn();
    return faktoriyel(5) === 120 && faktoriyel(3) === 6;
  } catch (e) { return false; }
}
```

3.2.5 Test ve Sınav Sistemi (Tests)

Çoktan seçmeli testler, kalan süre takibi, şıkların seçilmesi ve sınav bitiminde anlık puan hesaplama özellikleri sunar. Sonuçlar doğrudan profil durumuna işlenerek veri katmanına kaydedilir.

3.2.6 Satın Alma ve Ödeme Akışı (Checkout & PaymentResult)

`Checkout.jsx` kart ve fatura bilgisi alanlarını içerirken, `PaymentResult.jsx` ödemenin başarıyla tamamlandığını bildirerek satın alınan dersi anında öğrencinin hesabına tanımlar.

3.2.7 Eğitimci (Mentor) Kontrol Paneli (MentorDashboard)

Eğitmenlerin kendilerine bağlı öğrencilerin ders ilerlemelerini grafiksel olarak analiz etmesini ve son teslim tarihi olan yeni ödevler atamasını sağlar.

3.2.8 Yönetici (Admin) Kontrol Paneli (AdminDashboard)

Toplam kullanıcı, aktif ders ve ciro gibi metrikleri izleme; kullanıcılara yeni mentor rolleri tanımlama ve aktiflik durumlarını yönetme yetkilerini barındırır.

3.2.9 Öğrenci - Mentor İletişim Portalı (Chat)

Öğrencilerin takıldıkları yerlerde mentorlarıyla anlık yazışabilmelerini sağlayan mesajlaşma alanıdır. Sohbet geçmişi Redux state'i üzerinden yönetilir ve LocalStorage'da kalıcı hale getirilir.

4. SONUÇ

Bu bitirme projesinde, modern bir interaktif eğitim platformunun ihtiyaç duyduğu tüm ön yüz bileşenleri başarıyla hayata geçirilmiştir. **CodeChallenge** projesi, React 19'un sunduğu bileşen yapısı, Redux Toolkit'in güçlü durum yönetimi ve Tailwind CSS v4'ün hızlı stil geliştirme yetenekleri birleştirilerek yüksek performanslı ve kullanıcı dostu bir SPA (Single Page Application) olarak inşa edilmiştir.

Özellikle tarayıcı tabanlı çalışan interaktif kod editörü ve dinamik doğrulama motoru sayesinde, öğrencilere kodlarını yazdıktan hemen sonra anlık geri bildirim sunan işlevsel bir altyapı oluşturulmuştur. Rol tabanlı yönlendirmeler (Öğrenci, Mentor, Admin) ve aralarındaki entegrasyon (sohbet, ödev atama, analiz ekranları) platformun gerçekçi bir ürün senaryosuna sahip olduğunu kanıtlamaktadır.

Sonuç olarak proje, modern ön yüz geliştirme pratiklerini, temiz kod prensiplerini ve esnek arayüz tasarımlarını başarıyla sergileyen kapsamlı bir mezuniyet çalışması haline getirilmiştir.

5. KAYNAKÇA

[1] React Resmi Dokümantasyonu (React 19) - <https://react.dev/>

[2] Redux Toolkit Resmi Dokümantasyonu - <https://redux-toolkit.js.org/>

[3] Tailwind CSS v4.0 Geliştirici Kılavuzu - <https://tailwindcss.com/>

[4] Vite.js Proje Yapılandırma Rehberi - <https://vite.dev/>

[5] Motion Animasyon Kütüphanesi Dokümantasyonu - <https://motion.dev/>