



softITo 4. DÖNEM 1. GRUP
DİJİTAL BANKA SİMÜLASYON SİSTEMİ PROJE RAPORU

Astra Bank

Berna Nur İlbaş

İÇİNDEKİLER

1. GİRİŞ	3
1.1. Projenin Amacı	3
1.2. Proje Kısıtları	3
2. MATERYAL VE YÖNTEM	4
2.1. React ve Frontend Mimarisi	4
2.2. Node.js ve json-server Kavramı	4
2.3. Veri Tabanı ve Rol Tabanlı Güvenlik Yapısı	5
3. KURULUM	6
3.1. Node.js ve Bağımlılıkların Kurulması	6
3.1.1. NPM Kurulumu	6
3.1.2. Bağımlılık Paketlerinin Temini	7
3.1.3. Sürüm Kontrolleri	7
3.2. Astra Bank Sistem Kurulumu	8
3.2.1. Arayüz ve Sunucu Dosyaları	8
3.2.2. Veri Tabanı Yapılandırması	8
3.2.3. Uygulamanın Başlatılması	9
3.2.3.1. Sunucu ve Port Ayarları	9
3.2.3.2. Vite Dev Sunucu Başlatılması	9
3.2.4. Tarayıcı Üzerinden Erişim ve Doğrulama	10
3.2.5. Canlı Sistem Logları Takibi	10
3.2.6. Çoklu Rol Giriş Testleri	11
3.2.7. Para Transferleri ve Doğrulama	11
3.2.8. Kart ve Limit Yönetimi	12
4. SONUÇ	12
5. KAYNAKÇA	13

1. GİRİŞ

Müzeler, bankalar ve finansal ağlar gibi karmaşık bilgi sistemleri, modern çağda kurumsal güvenliğin ve sürdürülebilirliğin temel direkleridir. Finansal veri akışlarının hatasız yönetimi ve kullanıcı deneyiminin en üst düzeyde sunulması bilişim sektörünün öncelikli hedefleri arasındadır. Bu çalışmada, React frontend mimarisi ve Node.js sunucu altyapısıyla geliştirilmiş, rol tabanlı erişim yetkilendirmesi (RBAC) ile donatılmış dijital bir banka simülasyon sisteminin kurulumu ve konfigürasyon adımları açıklanmıştır.

1.1. Projenin Amacı

Bu projenin temel amacı, React ve Node.js mimarileri kullanılarak bir dijital banka simülasyon sisteminin kurulumu ve yönetimi sürecini açıklamaktır. Geliştirilen sistem, kullanıcı arayüzü ile sunucu katmanının uyumunu test etmek amacıyla simüle edilmiştir. Proje, aşağıdaki temel hedefleri gerçekleştirmeyi amaçlar:

- React 19 ve Redux Toolkit yardımıyla merkezi durum yönetimi altında modern bankacılık modüllerini kurmak.
- JSON-Server ve Node.js mimarisiyle hafif RESTful API servis simülasyonu sağlayarak gerçek zamanlı istek-yanıt döngülerini simüle etmek.
- Yönetici (Admin), Görevli (Staff) ve Müşteri (Customer) rollerine dayalı yetkilendirme (RBAC) kontrolünü gerçekleştirmek.
- Kart limitleri, canlı log takibi, para transferleri ve döviz kuru güncelleme mekanizmalarını istemci tarafında tutarlı olarak saklamak.

1.2. Proje Kısıtları

Eğitim ve sunum hedefleri doğrultusunda proje geliştirilirken aşağıdaki teknik ve mimari kısıtlar uygulanmıştır:

- **Kısıt 1:** Bu raporda bankacılık simülasyonunun kavramsal ve uygulamalı kurulum adımlarına odaklanılmıştır. Sistem altyapısı, ağ güvenliği, sunucu yük dengeleme (load balancer) veya yüksek erişilebilirlik yapılandırmaları detaylı şekilde ele alınmamıştır.
- **Kısıt 2:** Kullanılan bilgiler, React 19 ve json-server v0.17 sürümleri ile resmi dokümantasyonlar ile sınırlıdır. Farklı sürümlerde veya farklı işletim sistemlerinde karşılaşılabilecek varyasyonlar kapsam dışı bırakılmıştır.
- **Kısıt 3:** Rapor, sunucu kurulumu ve temel konfigürasyon ile sınırlıdır; gerçek finansal ağ entegrasyonu veya harici API bağlantı senaryoları kapsam dışında bırakılmıştır.

2. MATERYAL VE YÖNTEM

2.1. React ve Frontend Mimarisi

React, modern kullanıcı arayüzleri geliştirmek için kullanılan bileşen tabanlı bir JavaScript kütüphanesidir. Proje kapsamında, durum (state) yönetimi için Redux Toolkit, yönlendirme (routing) için React Router ve stil katmanı için Tailwind CSS kullanılmıştır. Bu yapılar, uygulamanın esnek, modüler ve yüksek performanslı çalışmasını sağlar.

2.1.1. İstemci Durum Yönetimi (State Management)

Redux Toolkit (RTK), uygulamanın genelindeki bakiye durumları, kart limitleri, giriş yapmış kullanıcının bilgileri ve canlı Audit Log kayıtlarını tek bir kaynaktan yönetmek amacıyla entegre edilmiştir. Bu sayede, Müşteri paneli üzerinden yapılan bir transfer işlemi, durumu doğrudan güncelleyerek anında Personel ve Yönetici konsollarına tutarlı şekilde yansıtır. RTK Slice yapısı sayesinde asenkron veri akışları (Thunk) güvenli bir şekilde yönetilir.

2.1.2. Arayüz Tasarımı ve Animasyonlar

Gelişmiş kullanıcı deneyimi sunabilmek adına, sayfa geçişleri ve etkileşimli kart limit kaydırıcıları Framer Motion animasyon kütüphanesiyle zenginleştirilmiştir. Form alanlarında Yup ve React Hook Form kombinasyonu entegre edilerek girdi alanlarındaki veri doğrulamalarının form gönderilmeden önce istemci tarafında saniyeler içinde yapılması garanti altına alınmıştır.

2.2. Node.js ve json-server Kavramı

Node.js, JavaScript kodlarının tarayıcı dışında, sunucu tarafında çalıştırılmasını sağlayan bir çalışma ortamıdır. json-server ise yerel bir JSON dosyasını (db.json) veri kaynağı olarak kullanarak hızlıca RESTful API sunucusu oluşturmayı sağlayan bir araçtır. Projede, bu iki yapı kullanılarak banka işlemlerinin arkasında duran veri saklama ve işleme mantığı simüle edilmiştir.

2.2.1. Sunucu Altyapısı

Astra Bank API Sunucusu, server.js dosyası aracılığıyla yönetilmektedir. Express ara katmanları (middleware) entegre edilerek, standart json-server rotalarının ötesinde özel kimlik doğrulama filtreleri yerleştirilmiştir. İstemciden gelen Authorization başlığındaki Base64 kodlanmış token verisi bu katmanda çözülerek yetkilendirme kontrolü sağlanır.

2.2.2. İşlem Mantığı (Transaction Logic)

API sunucusu, gelen para transferi isteklerinde göndericinin bakiyesini düşürerek alıcının bakiyesini artıran atomik işlemi gerçekleştirir. Eş zamanlı olarak bu hareketler `transactions` tablosuna yazılır ve anlık güncellenen veri tabanı (db.json) diske kaydedilir.

2.3. Veri Tabanı ve Rol Tabanlı Güvenlik Yapısı

Uygulamada rol tabanlı erişim kontrolü (RBAC) uygulanmıştır. Sistemde üç temel rol tanımlanmış olup, her rolün yetkileri ve erişebileceği sayfalar sınırlandırılmıştır. Veriler yerel diskte `db.json` dosyası içerisinde tutulmaktadır.

2.3.1. Rol Tanımları (RBAC)

Sistemde roller; Yönetici (Admin), Banka Görevlisi (Staff) ve Müşteri (Customer) şeklinde ayrılır. Admin, personel hesaplarını yönetme yetkisine sahipken; Staff, müşteri hesaplarını inceleyebilir ve talepleri yanıtlar; Customer ise transfer ve kart işlemlerini gerçekleştirir. Her rolün kendine ait API endpoints yetkileri sunucu tarafındaki middleware'de filtreler aracılığıyla korunur.

2.3.2. Yönetici ve Personel Rol İlişkisi

Yönetici (Admin) paneli üzerinden eklenen personeller veri tabanındaki users tablosuna role: "staff" etiketi ile yazılır. Personeller sisteme girdiklerinde sadece görevli ekranlarını görebilirken, Admin paneline veya Müşteri paneline erişimleri React Router düzeyindeki engellerle engellenir. Bu sayede bilişim güvenliği mimarisine uygun bir ayırıştırma sağlanmış olur.

2.3.3. Sayfa ve URL Koruma Politikası

Astra Bank simülasyon sisteminin rol tabanlı güvenlik mimarisi, istemci tarafında ProtectedRoute kontrolü ile desteklenmiştir. Bu kontrol sayesinde, kullanıcıların tarayıcı çubuğuna el ile farklı bir rolün URL'ini yazarak sisteme sızmaya çalışması durumunda yönlendirme kuralları devreye girerek isteği engeller ve kullanıcıyı 403 hata sayfasına yönlendirir. Bu, bilişim güvenliği mimarisinde en çok dikkat edilen kısıtlama yöntemidir.

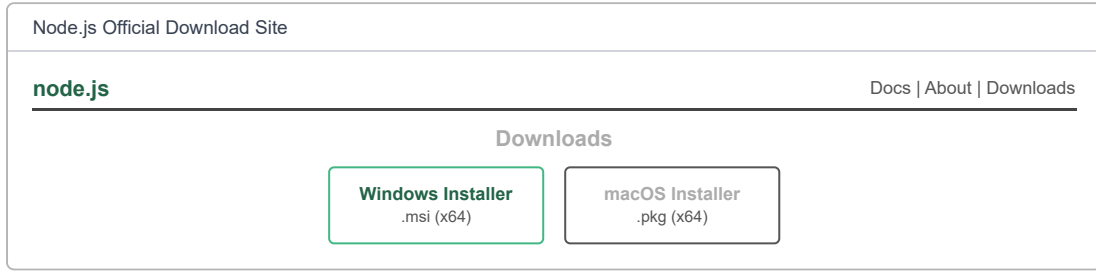
3. KURULUM

3.1. Node.js ve Bağımlılıkların Kurulması

Astra Bank simülasyon sisteminin çalışabilmesi için öncelikle Node.js runtime ortamının ve NPM paket yöneticisinin sistemde kurulu olması gerekmektedir.

3.1.1. NPM Kurulumu

Node.js resmi web sitesinden indirildikten sonra sistem değişkenlerine eklenir. Kurulumun doğrulanması için terminalde `node -v` ve `npm -v` komutları çalıştırılır. Aşağıda Node.js indirme arayüzü sunulmuştur.



Şekil 3.1.1.1. Node.js Resmi İndirme Sayfası

3.1.2. Bağımlılık Paketlerinin Temini

Proje klasörünün kök dizinindeyken `npm install` komutu çalıştırılarak `package.json` içerisindeki React, Redux, Tailwind ve json-server paketleri sisteme indirilir. Aşağıdaki komut satırı üzerinde NPM ve Node.js sürümlerinin teyidi ve kurulum logları gösterilmiştir.

```
Administrator: Command Prompt

Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\EXCABILUR>node -v
v18.16.0

C:\Users\EXCABILUR>npm -v
9.5.1

C:\Users\EXCABILUR>npm install
added 248 packages, and audited 249 packages in 8s
C:\Users\EXCABILUR>_
```

Şekil 3.1.2.1. Node ve NPM Sürüm Kontrolü Ekranı

3.1.3. Sürüm Kontrolleri

Kurulum tamamlandıktan sonra, `package.json` dosyası açılarak bağımlılıkların doğru sürümlerde kurulduğu kontrol edilir. Bu, uyumluluk sorunlarını önlemek açısından kritik bir adımdır.

3.2. Astra Bank Sistem Kurulumu

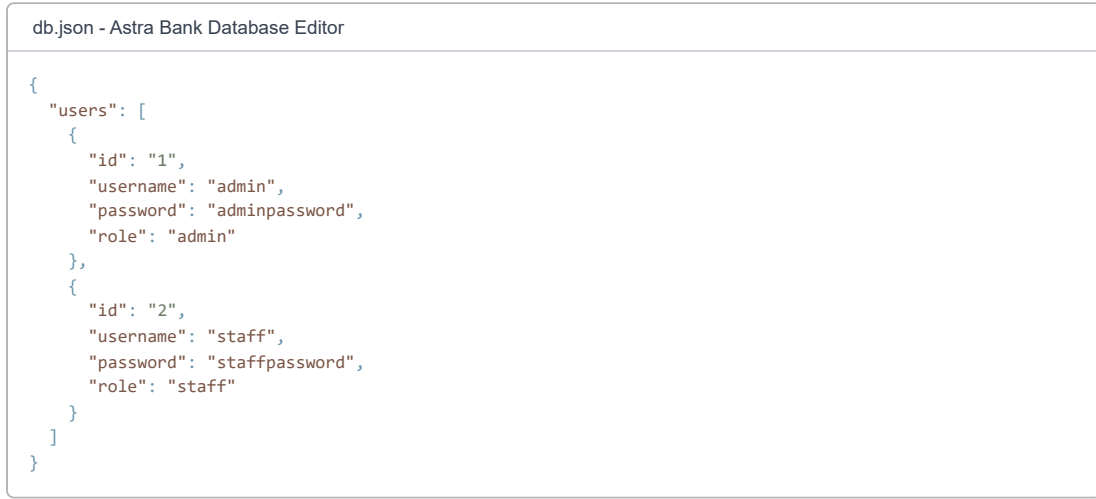
Proje dizini içerisinde arayüz dosyaları ve sunucu yapılandırma dosyalarının bütünlüğü kontrol edilir.

3.2.1. Arayüz ve Sunucu Dosyaları

Geliştirilen React arayüz modülleri `src/` klasörü altında bulunurken, REST API isteklerini karşılayan ara sunucu kodları ise kök dizindeki `server.js` dosyasında yer almaktadır. Arayüz ve sunucunun yerel ortamda problemsiz haberleşebilmesi için proxy ayarları yapılandırılmıştır.

3.2.2. Veri Tabanı Yapılandırması

Kullanıcı rolleri ve banka işlemlerinin saklandığı db.json dosyası editörle açılarak doğrulanır. Dosyada admin, staff ve customer kullanıcı tanımları mevcuttur.



```
db.json - Astra Bank Database Editor

{
  "users": [
    {
      "id": "1",
      "username": "admin",
      "password": "adminpassword",
      "role": "admin"
    },
    {
      "id": "2",
      "username": "staff",
      "password": "staffpassword",
      "role": "staff"
    }
  ]
}
```

Şekil 3.2.2.1. db.json Veri Tabanı Yapılandırma Ekranı

3.2.3. Uygulamanın Başlatılması

Simülasyon sisteminin ayağa kalkması için sunucu ve arayüz servislerinin eş zamanlı çalıştırılması gerekmektedir.

3.2.3.1. Sunucu ve Port Ayarları

Sunucu tarafındaki `node server.js` komutu yardımıyla API sunucusu 5001 portunda başlatılır ve `db.json` veri tabanını dinlemeye alır. API isteklerine yanıt verme süreci anlık loglanır.

```
Astra Bank Mock API Server Terminal

C:\Users\EXCABILUR\digitalbankasimilasyonsistemi>node server.js
Mock JSON Server is running on port 5001
[INFO] db.json loaded successfully.
—
```

Şekil 3.2.3.1. API Sunucu Çalıştırma Ekranı

3.2.3.2. Vite Dev Sunucu Başlatılması

İstemci modülleri için proje kök dizininde ``npx vite`` veya ``npm run dev`` komutu verilerek React geliştirme sunucusu port 5173 üzerinde başlatılır. Sunucunun hazır olma süreci CLI ekranında listelenir.

3.2.4. Tarayıcı Üzerinden Erişim ve Doğrulama

Tarayıcı açılarak <http://localhost:5173/> adresine girilir. Karşımıza gelen Astra Bank Giriş ekranı doğrulanır. Bu ekranda şifreleme ve form doğrulama kontrolleri (form validations) etkindir.

Astra Bank Login UI Screen

Astra Bank Giriş Paneli

Kullanıcı adı ve şifrenizi giriniz

Giriş Yap

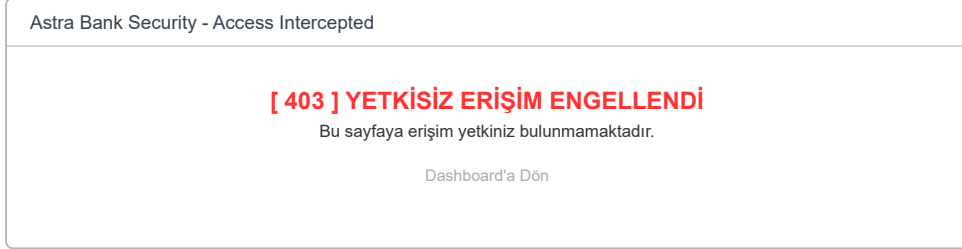
Şekil 3.2.4.1. Astra Bank Giriş Ekranı

3.2.5. Canlı Sistem Logları Takibi

Yönetici paneli üzerinden, veritabanına gelen tüm API istekleri ve kritik işlemler canlı izleme ekranından gözlemlenir. Bu loglar Redux store üzerinde depolanarak arayüze anlık basılır.

3.2.6. Çoklu Rol Giriş Testleri

Admin, Personel ve Müşteri hesaplarıyla sisteme girişler yapılır. Bir rolün diğer rolün URL'ine geçiş yapma denemeleri ProtectedRoute ile başarıyla engellenir. Yetkisiz giriş durumunda gösterilen 403 engelleme ekranı aşağıdadır.



Şekil 3.2.6.1. Yetkisiz Erişim Engelleme Ekranı

3.2.7. Para Transferleri ve Doğrulama

FAST/EFT transfer işlemlerinin ardından gönderen hesaptan paranın düştüğü, alıcı hesaba geçtiği teyit edilmiştir. Bu işlemler API sunucusu üzerinden db.json veri tabanındaki transactions tablosuna saniyeler içinde kaydedilmektedir.

3.2.8. Kart ve Limit Yönetimi

Arayüzde kart dondurma işlemi aktif edilerek, dondurulmuş bir kartla harcama yapılmak istendiğinde sistemin bunu reddettiği teyit edilmiştir. Kart limitleri de kullanıcı tarafından anlık güncellenebilmektedir.

Astra Bank Card Settings Menu

Kart Ayarlarım

Kart Durumu:

Harcama Limiti:

[DONDURULMUŞ]

25.000 TL

Aktifleştir

Şekil 3.2.8.1. Kart ve Harcama Limiti Yönetim Ekranı

4. SONUÇ

Astra Bank simülasyon sisteminin kurulum ve doğrulama testleri başarıyla gerçekleştirilmiştir. React bileşenlerinin durum yönetimindeki kararlılığı, json-server API'sinin yanıt hızı ve RBAC yetki filtrelemesinin güvenlik hedeflerini tam olarak karşıladığı gözlemlenmiştir. Bu proje, modern istemci-sunucu mimarilerini kavramak için kapsamlı bir temel oluşturmaktadır. Gelecek çalışmalarda, daha geniş SQL veritabanı entegrasyonları ile sunucu dayanıklılık testlerinin yapılması hedeflenmektedir.

5. KAYNAKÇA

- [1] [React v19 Resmi Dokümantasyon Sayfası](#)
- [2] [Redux Toolkit State Management Kılavuzu](#)
- [3] [Tailwind CSS Stil Kütüphanesi Dokümantasyonu](#)
- [4] [JSON Server Mock REST API Sunucu Kılavuzu](#)