ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM 4062 PROJE RAPORU

MetricsMon: System Monitoring Toolkit

Damla Kara

19290251

Doç. Dr. Recep Eryiğit

6/2023

ÖZET

Bu projede, bir sistem izleme uygulaması yapmanın bir yolu araştırıldı. Proje 3 ana kısımdan oluşur; kullanıcı tarafı arkaplan yazılımı, sunucu taraflı arkaplan yazılımı ve kullanıcının sonuçları görüntüleyebileceği web arayüzü. Arayüz, tüm işlemleri gerçekleştirmek için kendi kullanıcı dostu web GUI'sine sahiptir.

İÇİNDEKİLER

1. GİRİŞ

MetricsMon: System Monitoring Toolkit, bir izleme yazılımı aracıdır. Proje, ağ kullanımı, CPU yükü ve disk alanı tüketimi gibi izleme ölçümleri sağlar.

2. MOTİVASYON

Bu projenin seçimindeki ana motivasyon hem sunucu taraflı hem de ön yüz taraflı yazılım geliştirimi ve etkileşimini uygulamalı bir şekilde derinlemesine öğrenmektir. Ek olarak, bu proje, yapı itibari ile; veritabanları, sunucu-kullanıcı etkileşimleri, yetkilendirmeler gibi birçok gerçek hayatta yüksek seviyede ihtiyaç duyulan konuların tek bir projede öğrenilmesi için güzel bir örnektir. Bu projenin seçimi ve tasarımı esnasında yanda belirtilen açık kaynak projeler incelenmiştir; Zabbix, PandoraFMS.

3. MATERYAL

Bu bölümde yazılımda genel olarak tercih edilen; işletim sistemi, teknolojiler, kütüphaneler ve kodlama dillerinin açıklamaları bulunmaktadır.

3.1 İşletim Sistemi Seçme

Bu projenin hem kullanıcı taraflı hem de sunucu taraflı yazılımları GNU/Linux tabanlı sunucular için tasarlanmıştır.

3.2 Teknolojiler

Aşağıdaki tanımlar, yukarıda belirtilen teknolojilerin neler olduğunu ve projede nasıl kullanılacağını özetlemektedir:

3.3 Python Nedir?

Python, yorumlanmış bir üst düzey programlama dilidir. Dinamik olarak yazılır ve nesnelerin dolaşılmasına izin verir. Ayrıca geniş bir standart kitaplık sağlar. Python, hızlı uygulama geliştirme (Rapid Application Development) için kullanılır ve genellikle uygulamalar arasında bir yapıştırma dili olarak kullanılır. Python ayrıca otomasyon ve yapılandırma yönetimi gibi sistem yönetimi görevleri için yaygın olarak kullanılır. Python, GNU Genel Kamu Lisansı sürüm 2 altında yayınlanan açık kaynaklı bir yazılımdır.

3.4 MySQL Nedir?

En yaygın kullanılan Açık Kaynak SQL veritabanı yönetim sistemi MySQL'dir. Hızlı, güvenilir ve kullanımı basittir. İstemci/sunucu uygulamaları oluşturmak için MySQL, çok çeşitli programlama dilleriyle birlikte kullanılabilir. Ayrıca, veritabanlarını bir grafik kullanıcı arabirimi kullanarak yönetmenize izin veren, MySQL Workbench adlı

kullanıcı dostu bir grafik kullanıcı arabirimi (GUI) sağlar. [8] Bu projede, sunucu tarafı ve istemci tarafı uygulamalardan toplanan verileri depolamak için MYSQL hizmeti kullanılacaktır. UNIX ve Windows dahil çok sayıda işletim sistemi MySQL'i destekler. MYSQL servisini kurmak için resmi web sayfasındaki yönergeler takip edilebilir.

3.5 Flask Nedir?

Flask, geliştiricilerin web uygulamalarını hızlı ve basit bir şekilde oluşturup ölçeklendirmelerini sağlamak için tasarlanmış, geliştiricilere yönelik bir mikroframework olarak adlandılır. Önceden var olan üçüncü taraf kitaplıkların zaten bir veritabanı soyutlama katmanı, form doğrulama veya diğer bileşenler gibi ortak işlevler sağladığı bileşenlerden yoksundur. Ancak Flask, uygulama özelliklerini Flask'ın kendisinde uygulanmış gibi ekleyebilen uzantıları destekler. Nesne-ilişkisel eşleyiciler, form doğrulama, yükleme işleme, birkaç açık kimlik doğrulama protokolü ve popüler frameworklerle ilişkili bir dizi yardımcı program için uzantılar vardır.

3.6 Psutil Kütüphanesi Nedir ve Nasıl Kullanılır?

Psutil (Python System and Process Utilities), Python programcılarının aktif süreçler ve sistem kullanımı (CPU, bellek, depolama) hakkında bilgi almasına olanak tanıyan bir çapraz platform (cross-platform) kitaplıktır. Bu nedenle, bu kitaplık sistem izleme, profil oluşturma, işlem kaynaklarını sınırlama ve çalışan işlemlerin yönetimi için kullanılır.

3.7 OS Kütüphanesi Nedir ve Nasıl Kullanılır?

Python OS modülü, kullanıcı ile işletim sistemi ile etkileşim oluşturma olanağı sağlar. İşletim sistemi tabanlı görevleri gerçekleştirmek, işletim sistemi ile ilgili verileri elde etmek için kullanılabilecek çeşitli pratik işletim sistemi özellikleri sağlar.

3.8 shlex Kütüphanesi Nedir ve Nasıl Kullanılır ?

Shlex modülü, basit kabuk benzeri sözdizimlerini ayrıştırmak için bir sınıf uygular. Kendi etki alanına özgü dilinizi yazmak veya alıntılanan dizeleri ayrıştırmak için kullanılabilir.

3.9 requests Kütüphanesi Nedir ve Nasıl Kullanılır ?

requests modülü, python'da GET, POST gibi çeşitli HTTP istekleri atabilmemizi sağlar. Bu modül ile attığımız istekler bütün yanıt verilerini (response data) içeren bir Response Object döner.

3.10 threading Kütüphanesi Nedir ?

Python threading modülü, Python'da yeni yürütme dizileri (threads) oluşturmanıza ve yönetmenize olanak tanır.

3.11 socket Kütüphanesi Nedir ?

soket modülü, istemci ve sunucu programları dahil olmak üzere kapsamlı ağ uygulamaları oluşturmak için soket modülü çeşitli nesneler, sabitler, işlevler ve ilgili istisnalar sunar.

3.12 flask_mysqldb Kütüphanesi Nedir ?

Flask-MySQLdb, Flask için MySQL bağlantısı kurmaya yardımcı olan bir kütüphanedir.

3.13 Visual Studio Nedir?

Visual Studio, Microsoft tarafından sunulan bütünleşmiş bir geliştirme ortamı sunar. Programcıların komut satırı, GUI tabanlı ve web tabanlı programlar dahil olmak üzere çeşitli programlar tasarlamasına olanak tanır. Akıllı sekme tamamlama, sürükle ve bırak grafik öğeleri vb. gibi inanılmaz derecede pratik özellikler sunar. Visual Studio, bu projedeki tüm programlar için birincil geliştirme ortamı olarak hizmet edecektir.

3.14 React Nedir?

React, kullanıcı arabirimleri(user interfaces) veya UI bileşenleri oluşturmak için bildirime dayalı, verimli ve esnek bir JavaScript kitaplığıdır. "Bileşenler" adı verilen küçük ve yalıtılmış kod parçalarından karmaşık kullanıcı arayüzleri oluşturmanıza olanak tanır.

3.15 Redux Nedir?

Redux, uygulama durumunu yönetmek ve merkezileştirmek için açık kaynaklı bir JavaScript kitaplığıdır.

3.16 Redux Toolkit Nedir?

Redux Toolkit, Redux mantığı için bir yaklaşımdır. Mağaza(store)kurulumu, azaltıcılar (reducers) oluşturma, değiştirilemez güncelleme mantığı ve daha fazlası gibi yaygın kullanım durumlarını basitleştirmeye yönelik yardımcı programları içerir.

3.17 flask_cors Kütüphanesi Nedir?

Kaynaklar Arası Kaynak Paylaşımını (CORS) işlemek için bir Flask uzantısı, kaynaklar arası AJAX'ı mümkün kılar.

3.18 jwt Kütüphanesi Nedir?

JSON Web Tokenlarını kodlamayı (encode) ve kodu çözmeyi (decode) sağlar.

3.18 json Kütüphanesi Nedir?

JSON verileriyle çalışmak için kullanılabilen Python'da yerleşik (built-in) bir pakettir.

3.19 subprocess Modülü Nedir?

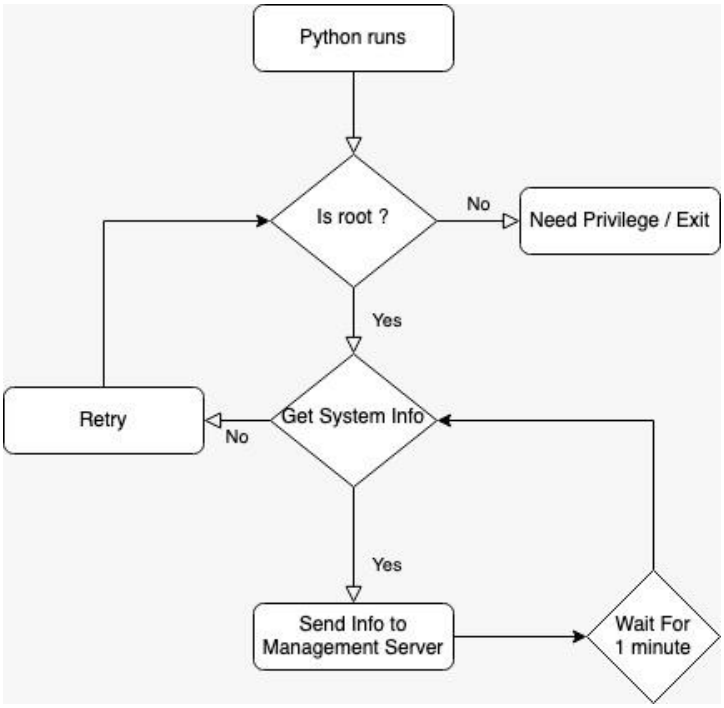
Alt işlem modülü, yeni işlemler oluşturmanıza, bunların giriş/çıkış/hata borularına bağlanmanıza ve dönüş kodlarını almanıza olanak tanır.

4. DİZAYN

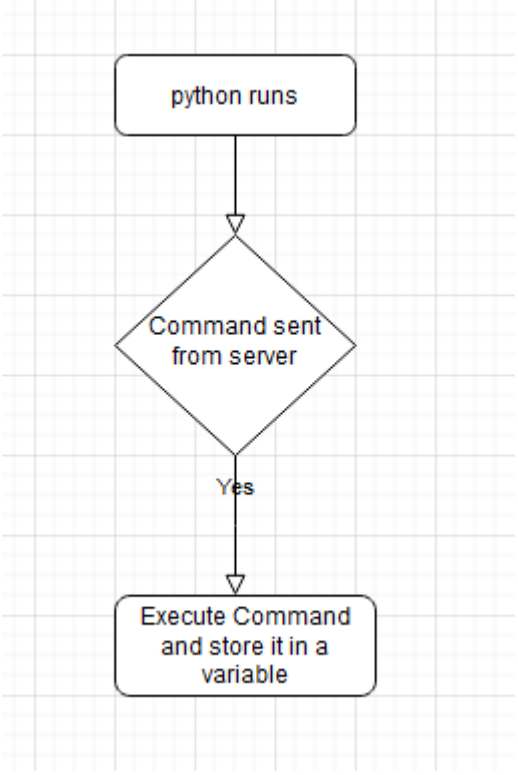
Bu bölüm projenin teknik dizaynı ve şemalarını içermektedir.

4.1 Kullanıcı Taraflı Arkaplan Yazılım Tasarımı

Kullanıcı taraflı uygulama şu teknik yeteneklere sahip olmalıdır: kullanıcının yetkisini kontrol etme, kullanıcı tarafından girilen komutların komut listesinde olup olmadığını kontrol etme, komutları alıp sunucuya gönderme, her x saniyede sistemin CPU kullanımı, RAM kullanımı, depolama kullanımı gibi bilgileri alıp sunucuya gönderme, mesaj almak, yanıt göndermek gibi iletişim gereksinimleri için HTTP isteklerini kullanma.



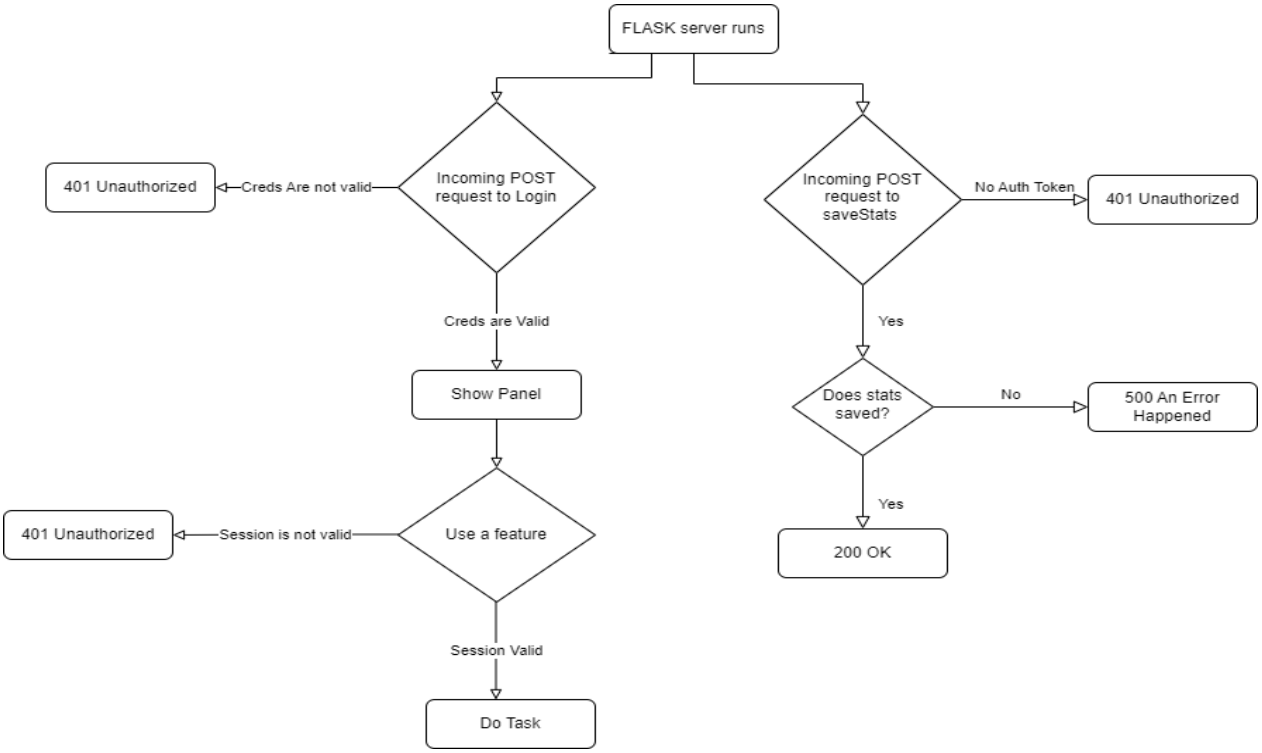
Şekil 4.1 İstemci taraflı uygulama genel şema



Şekil 4.2. İstemci taraflı uygulama etkileşim şeması

4.2 Yönetim Paneli Taraflı Web Uygulaması Tasarımı

Yönetim paneli web uygulaması, yönetim panelinin ana uygulamasıdır. Uygulama; kullanıcı taraflı yazılımdan gelen verileri kaydetme, kullanıcı tarafı yazılımlara komut gönderme ve ilgili komutların cevaplarını alma, aktif verileri panel kullanıcılarına sunma ve panel üzerinde yetkilendirme işlemlerinin yapılmasından sorumludur.



Şekil 4.3 Yönetim Paneli Genel Şema

5. UYGULAMA

5.1 Ön-Uç Tarafı Uygulamanın İmplementasyonu

Uygulamanın ön-uç (front-end) tarafı React, Redux, Redux Toolkit kütüphaneleri ile kodlanmıştır. Temelde Dashboard, Hosts ve Profile page olmak üzere kullanıcıya 3 ana fonksiyonel sayfa sunar.

5.1.1 Yeni bir React Projesi Oluşturma

Create React App ile sıfırdan yeni bir proje oluşturulur. Bu yöntem yeni bir tek sayfalık uygulama (single-page application) oluşturmanın en iyi yoludur. En son JavaScript özelliklerini kullanabilmemiz için geliştirme ortamını hazırlar. Aşağıdaki kod npm ve create-react-app ile yeni bir proje oluşturma işlemini gösterir.

5.1.2 Kütüphaneleri İndirme

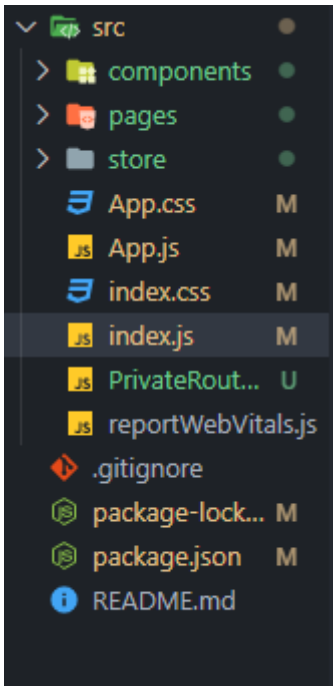
Gerekli olan kütüphaneler npm kullanılarak makinemize indirilir. Ön üç tarafında gerekli olan kütüphaneler; istemci tarafı yönlendirme için react-roter-dom, http istekleri oluşturma ve yanıtları kullanabilmek için axios, raporun materyal kısmında açıklanan sebepler için react-redux, @reduxjs/toolkit indirilir. Bootstrap CDN linkleri olarak index.html sayfasına eklenerek kullanılır.

Aşağıdaki kod bu kütüphanelerin indirme işlemini gösterir.

```
npm install react-redux @reduxjs/toolkit react-router-dom axios
```

5.1.3 Kaynak Kodları Organize Etme

React temelinde bileşenlerden (component) oluştuğundan ötürü projelerde dosya sayısı oldukça artar ve başa çıkmak zorlaşır. Bu yüzden özellikle büyük React projelerinde dosyaları yapılandırmak oldukça önemlidir. Dosya yapılandırmak için birden farklı yöntem olsa da bu projede ele alınan yöntem kısaca redux store kod parçaları için store, bileşenleri içeren component ve uygulamadaki her sayfa için pages dosyasından oluşur. Dosya yapısı aşağıda gösterilmiştir:



5.1.4 index.js Yapısı

react-redux kütüphanesinin sağladığı <Provider> bileşeni, Redux deposunu (store), Redux deposuna erişmesi gereken tüm iç içe bileşenler için kullanılabilir hale getirir. react-router-dom kütüphanesinin sağladığı <BrowserRouter> bileşeni, kütüphanenin sağladığı rota özelliklerinin kullanılabilmesi için en dış kapsamda kullanılmalıdır. Bütün iç içe bileşenlerin erişebilmesi için en dıştaki App.js bileşenine sağlanır. Sonuç olarak indeks.js yapısı aşağıdaki gibidir:

```
import ReactDOM from 'react-dom/client';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { Provider } from 'react-redux';
import { BrowserRouter } from 'react-router-dom';
import { store } from './store/store';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
   <Provider  store={store} >
    <BrowserRouter>

      <App />
    </BrowserRouter>
   </Provider>
  </React.StrictMode>
);
```

5.1.5 App.js Yapısı

Bütün sayfa rotası işlemleri diğer bileşenleri içerdiğinden dolayı bu bileşen içinde yapılmış olup, redux store kullanılarak uygulama ilk yüklendiğinde gerekli veriler öncelikle burada alınmıştır. İlk yüklenmeden sonra verilerin güncel tutulması açısından her 30 saniyede bir yeni veriler alınır. Oluşturulmuş özel bileşende token kontrolü ile sayfalar ve veriler korunur. App.js bileşeninin son hali aşağıdadır:

```
-------some imports-------
function App() {
  const theme = useSelector(state  => state.theme.darkMode)
  const dispatch = useDispatch()
  useEffect(() => {
    dispatch(getData());
    let interval = setInterval(() => {
      dispatch(getData());
    }, 30000);
    return () => {
      clearInterval(interval);
    };
  }, []);

  return (
    <div className={`App ${theme && "dark-mode"}`}>

      <Routes>
        <Route path="/" element={<Login />} />
        <Route path="/dashboard" element={<PrivateRoute component={Dashboard}
/>} />
        <Route  path="/hosts/details" element={<PrivateRoute
component={HostDetails} />} />
        <Route path="/hosts" element={<PrivateRoute component={Hosts} />} />
-        <Route path="/profilePage" element={<PrivateRoute
component={ProfilePage} />} />
      </Routes>
    </div>
  );
}
const PrivateRoute = ({component : Component}) => {
  const { token } = useSelector((state) => state.auth);

  return token ? <Component /> : <Navigate to="/" replace />;
};


export default App;
```

5.1.6 Tema Seçeneğinin Oluşturulması

Kullanıcıya “dark” ve “light” olmak üzere 2 farklı tema seçeneğinin oluşturulmasını sağlayabilmek için `"@reduxjs/toolkit"` kütüphanesi ile bir “slice” oluşturulmuştur.

```
import { createSlice } from "@reduxjs/toolkit";
const themeSlice = createSlice({
    name: "theme",
    initialState: {darkMode: false},
    reducers: {
        toggleDarkMode (state) {
            state.darkMode = !state.darkMode
        }
    }
})

export const { toggleDarkMode } = themeSlice.actions;

export default themeSlice.reducer
```

Oluşturulan bu kod parçasını kullanarak kullanıcının bir buton ile uygulamada tema değişimi yapabilmesini sağlayan özel ThemeProvider bileşeni aşağıdaki gibi oluşturulmuştur. Bu bileşende, her sayfa değişiminde veya uygulamanın yeniden yüklenmesi gibi durumlarda seçilen temanın kaybolmaması için bir anahtar(key) oluşturulup localStorage’da depolanmıştır.

```
import React, { useEffect } from 'react'
import { useDispatch, useSelector } from 'react-redux';
import { toggleDarkMode } from '../store/themeSlice';

const THEME_KEY = 'my-app-theme';

const ThemeProvider = ({children}) => {
  const dispatch = useDispatch();
  const darkMode = useSelector((state) => state.theme.darkMode);

  useEffect(() => {
    const savedTheme = localStorage.getItem(THEME_KEY);
    if (savedTheme === 'dark') {
      dispatch(toggleDarkMode());
    }
  } , [dispatch])

  useEffect(() => {
    const body = document.querySelector('body');
    if (darkMode) {
      body.classList.add('dark-mode');
```

```
    } else {
      body.classList.remove('dark-mode');
    }

    localStorage.setItem(THEME_KEY, darkMode ? 'dark' : 'light');
  }, [darkMode]);

  const handleToggleDarkMode = () => {
    dispatch(toggleDarkMode());
  };

  return (
    <div className=' px-2 py-1 icon dark'>
    <button onClick={handleToggleDarkMode} className='border-0 bg-transparent
icon dark'>
      {darkMode ? <i className="bi bi-brightness-high"></i> : <i className="bi
bi-moon"></i>}
    </button>
    {children}
  </div>
  )
}

export default ThemeProvider
```

5.1.7 Kullanıcı Yetkilendirmesinin Yapılması

Kullanıcı kimlik doğrulaması için @reduxjs/toolkit kütüphanesinin createSlice method ile ayrı bir slice oluşturulur. Yine aynı kütüphanenin createAsyncThunk

methodu kullanılarak kullanıcıdan kullanıcı adı ve şifre alıp axios ile post isteği atan, arka uçtan gelen cevaba göre kullanıcıyı yönlendiren bir giriş yapma fonksiyonu oluşturulur. Arka uçtan dönen cevaba göre token alınıp yerel depolamaya (local storage) kaydedilir ve kullanıcı uygulamaya yönlendilir. Hata var ise slice'ın state'inde bulunan değişken güncellenir ve kullanıcıya ilgili mesaj gösterilir. Kullanıcının çıkış yapmasını sağlayan fonksiyon da bu kısımda yer alır, state ve yerel depolamadaki (local storage) ilgili değerleri boşaltarak kullanıcının çıkış yapması sağlanır. Kimlik doğrulama kodu aşağıdaki gibidir:

```
import { createAsyncThunk, createSlice } from "@reduxjs/toolkit";
import axios from 'axios';


const initialState = {
    token: localStorage.getItem('token') || null,
```

```
    user: null,
    isLoading: false,
    error: null,
  };

  export const login = createAsyncThunk(
    'auth/login',
    async ({ username, password }, { rejectWithValue }) => {
      try {
        const response = await axios.post(
          "http://192.168.32.131:5000/login",
          {
            username,
            password
          },
          {
            headers: {
              'Content-Type': 'application/json'
            }
          }
        );
        return response.data;
      } catch (error) {
          // Access specific properties of the error response
          return rejectWithValue(error.response);

      }
    }
  );

const authSlice = createSlice({
    name: "auth",
    initialState,
    reducers:{
        logout (state){
            state.token = null
            state.user=null

            localStorage.removeItem("token")
        }
    },
    extraReducers: (builder)=> {
        builder
            .addCase(login.pending, (state) => {
                state.isLoading = true;
                state.error = null;
              })
            .addCase(login.fulfilled, (state, action) => {
                state.isLoading = false;
```

```
                state.token = action.payload;
                state.user = action.payload.user;

                localStorage.setItem("token", action.payload)
            })
            .addCase(login.rejected, (state, action) => {
                state.isLoading = false;
                state.error = action.payload.data.error;
            })
    }
})


export const {logout} = authSlice.actions

export default authSlice.reducer
```

5.1.8 Arka Uçtan Veriyi Alma

Uygulamada gösterilecek ve kullanılacak verilerin arka uçtan alınabilmesi için ayrı bir redux slice oluşturulur. Bu kısımdaki kod parçasında axios ve createAsyncThunk ile, ilgili endpointe bir get isteği atılır. Bu isteğin başlık (header) kısmına arka uçta kimlik doğrulama yapılabilmesi, data güvenliği için kullanıcı adına tanımlanan ve yerel depolamaya (local storage) kaydedilmiş token eklenir. Arka uçtan dönen yanıt bu slice'ın state kısmında tutulur.

```
import { createAsyncThunk, createSlice } from "@reduxjs/toolkit";
import axios from "axios";


const initialState = {
    data : []
}

export const getData = createAsyncThunk('data/getData',
async (_,{ rejectWithValue }) => {

    try {
      const response = await
axios.get('http://192.168.32.131:5000/showStatistics',{
        headers:  {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${localStorage.getItem("token")}`
          }
    });
```

```
      return response.data;
    } catch (error) {
      return rejectWithValue(error.response.data);
    }
  })


const dataSlice = createSlice({
    name : "data",
    initialState,
    reducers: {},
    extraReducers: (builder) => {
        builder
            .addCase(getData.fulfilled, (state, action)=> {

                state.data = action.payload
            })
            .addCase(getData.rejected, (state, action) => {

            })
    }
})


export default dataSlice.reducer
```

5.1.9 Şifre Değiştirme

Kullanıcının hesabının şifresi değiştirebilmesi için diğer slice'larda olduğu gibi createSlice kullanılarak profil slice'I oluşturulur. createAsyncThunk ve axios ile token eklenerek ilgili endpointe post isteği atılır. Kullanıcıdan kullanıcı adı, eski şifre ve yeni şifre alınarak arka uçtan dönen cevapi le kullanıcıya mesaj gösterilir. İlgili kod parçası aşağıdadır:

```
import { createAsyncThunk, createSlice } from "@reduxjs/toolkit";
import axios from 'axios';
const initialState = {
    message : ""
  };
  export const changePassword = createAsyncThunk(
    'profile/changePassword',
    async (data, { rejectWithValue }) => {
      try {
        const response = await axios.post(
          "http://192.168.32.131:5000/reset_password",
          data,
```

```
          {
            headers: {
              'Content-Type': 'application/json',
              'Authorization': `Bearer ${localStorage.getItem("token")}`
            }
          }
        );
        return response.data;
      } catch (error) {
          // Access specific properties of the error response
          return rejectWithValue(error.response);

      }
    }
  );

const profileSlice = createSlice({
    name: "profile",
    initialState,
    reducers:{},
extraReducers: (builder) => {
      builder.addCase(changePassword.fulfilled, (state, action) => {
        state.message = action.payload
      });
      builder.addCase(changePassword.rejected, (state, action) => {
        console.log('Rejected:', action.payload);
      });
    },
})
```

5.1.10 Kullanıcıdan Komut Alma ve Arka Uca Gönderme

Kullanıcının komut göndermesini sağlayan slice'ta kullanıcıdan alınan hostname, command ve token ile ilgili endpointe axios ve createAsyncThunk ile post isteği atılır. Bu komutun cevabı daha sonra diğer verilerin çekilmesiyle kullanıcıya sunulur. Komut gönderme kod parçası aşağıdadır:

```
import { createAsyncThunk, createSlice } from "@reduxjs/toolkit";
import axios from "axios";

const initialState = {
  commands: []
}

export const sendCommand = createAsyncThunk(
    'command/sendCommand',
    async ({hostname, command}, { rejectWithValue }) => {
      try {
```

```
        const response = await axios.post(
          "http://192.168.32.131:5000/insertCommand",
          {
            hostname,
            command
          },
          {
            headers: {
              'Content-Type': 'application/json',
              'Authorization': `Bearer ${localStorage.getItem("token")}`
            }
          }
        );
        return response;
      } catch (error) {
          // Access specific properties of the error response
          return rejectWithValue(error.response);

      }
    }
  );
const commandSlice = createSlice({
    name: "command",
    initialState,
    reducers: {
    },
    extraReducers: (builder) => {
        builder
        .addCase(sendCommand.fulfilled, (state, action)=> {
        })
        .addCase(sendCommand.rejected, (state, action) => {

        })
    }
})
```

5.1.11 Redux Store Oluşturulması

Farklı fonksiyonaliteye sahip slice'lar oluşturulduktan sonra uygulamada bütün durum (state) ağaçlarına ulaşabilmek ve değiştirebilmek için bir redux store oluşturulur ve slice'lar oluşturulurken export edilen reducer'ları buraya aktarılır.

```
-----some imports------
export const store = configureStore({
    reducer:{
        auth: authReducer,
        data: dataReducer,
        theme: themeReducer,
```

```
        command: commandReducer,
        profile: profileReducer
    },

})
```

5.1.12 Gösterge Paneli

Kullanıcının genel bir bakış sağlayabilmesi ve sistem performansının kolayca anlaşılmasını kolaylaştırmak ve için netliği ve basitliği vurgulayarak görsel olarak çekici ve sezgisel bir gösterge paneli (dashboard) hazırlanmıştır.

Bootstrap sınıf isimleri (class), özel CSS sınıf isimleri ve diğer componentlere ile Gösterge Panelinin genel yapısı aşağıdaki gibidir:

```
-------import kısmı kırpılmıştır-------
  return (
    <div>
        <Header />
        <div className='container'>
        <div className='row'>
               <div className='col-6'>
                   <TotalHosts />
               </div>
               <div className='col-6'>
                  <TotalPerformance />
               </div>
               <div className='col-4'>
                  <CalendarComponent ></CalendarComponent>
               </div>
                <div className='col-8 '>
                   <Threshold />
                    <ServerPerformance />
                </div>
               <div className='col-12 '>
                  <ChartComponent />
                </div>
            </div>
            <div className='row mb-3'>
                <div className='col-7'>
                   <SummaryTable />
                </div>
                <div className='col-5'>
                  <PieChart />
                </div>
            </div>
        </div>
    </div> ) }
```

5.1.13 Header Bileşeni

Kullanıcının uygulamada farklı sayfalara geçiş yapabilmesi, çıkış yapabilmesi, tema özelliğini kullanabilmesi için bütün sayfaların üstünde yer alan bir Header bileşeni oluşturulur. Kullanıcıyı diğer sayfalara yönlendirmek için react-router-dom kütüphanesinin NavLink bileşeni ile sağlanır. Header bileşeni aşağıdaki gibidir:

```
--------import kısmı kırpılmıştır-----
const Header = () => {
  const {token} =  useSelector(state => state.auth)
  const theme = useSelector(state => state.theme.darkMode)
  const dispatch = useDispatch()

  const logoutHandler = () => {
    dispatch(logout())
  }
  return (
    <div className={`d-flex container-fluid header py-1 px-5 justify-content-
end mb-5 ${theme && "dark-mode"}`}>
    <p className={`me-auto header-name fw-bold ${theme && "dark-mode"}`}>
MetricsMon: System Monitoring Toolkit</p>
    {
        token &&
        <>
            <div className='header-item px-2 py-1 me-2'>
                <NavLink  to="/dashboard">Dashboard</NavLink>
            </div>
            <div className='header-item px-2 py-1 me-2'>
                <NavLink  to="/hosts">Hosts</NavLink>
            </div>
            <div className='icon purple px-2 py-1 me-2'>
                <NavLink to="/profilePage" ><i className="bi bi
person"></i></NavLink>
            </div>
                <ThemeProvider />
            <div className='icon primary ms-2 px-2 py-1'
onClick={logoutHandler}>
                <i className="bi bi-box-arrow-right"></i>
            </div>
        </>
    }
    </div>
  )
}
```

5.1.14 Bileşen Sarıcısı (Wrapper)

Bütün bileşenlerin bazı CSS özellikleri aynı olduğundan bu özellikleri iç bileşenlere sağlayan bir bileşen oluşturulur. Seçilen temaya göre farklı CSS özellikleri eklenir. Bu bileşen aşağıdaki gibidir:

```
-----import kısmı kırpılmıştır------
const Card = ({children}) => {
  const theme = useSelector(state  => state.theme.darkMode)

  return (
    <div className={`card mb-3 ${theme && "dark-mode"}`}>
        {children}
    </div>
  )
}

export default Card
```

5.1.15 Total Host ve Total Performance Bileşeni

Kullanıcının genel bir bilgi alabilmesi için gösterge panelinin başında host sayısının toplamını ve hostların toplam performansının gösterildiği bileşenler oluşturulur. Bunlar belirli renklerle ve ilgili ikonlarla kullanıcıya sunulur.

```
----- import kısmı kırpılmıştır -----
const TotalHosts = () => {
    const data= useSelector(state => state.data.data)
  return (
    <Card>
        <div className='d-flex px-2 py-4 justify-content-center'>
                <div className={`me-4 px-4 text-center d-flex align-items-
center icon primary`} > <i className="bi bi-pc-display-horizontal"></i></div>
            <div className='text-start align-items-center'>
                <p className='fw-bold'>Total Hosts</p>
                <div className=' big-font'> {data?.length} </div>
            </div>
        </div>
    </Card>
  )
}
```

Toplam performansın hesaplandığı fonksiyonda her bir sunucu için ağırlık değerleri belirlenir. Bu değerler, her birinin (CPU kullanımı, RAM kullanımı ve depolama kullanımı) önemini temsil eder. Her bir sunucunun CPU kullanımı,

RAM kullanımı ve depolama kullanımı yüzdelik olarak 0 ile 1 arasında bir ölçekte ifade edilir. Bu değerler üzerinden her bir sunucunun performansı hesaplanır. Bu, her metriğin normalleştirilmiş değerini ilgili ağırlıkla çarparak yapılır. Ardından, bu performans değerleri toplanarak sunucunun toplam performansı elde edilir. Her bir sunucunun performansını toplam performansa ekleyerek, toplam performans elde edilir. Tüm sunucular için ortalama performansı hesaplamak için toplam performans sunucu sayısına bölünür. Son olarak, 0-100 arasında bir ölçekte ifade etmek için ortalama performansı 100 ile çarparak sunucu performansı elde edilir. İlgili kod aşağıdadır:

```
------ import kısmı kırpılmıştır----
const TotalPerformance = () => {
    const data = useSelector(state => state.data.data)
    const [serverPerformances, setServerPerformances] = useState([]);

    const calculateServerPerformance = (hosts) => {
        const cpuWeight = 0.5;
        const ramWeight = 0.3;
        const storageWeight = 0.2;

        let totalPerformance = 0;

        hosts.forEach((host) => {
          const normalizedCpu = host.cpu_usage / 100;
          const normalizedRam = host.ram_usage / 100;
          const normalizedStorage = host.storage_usage / 100;

          const performance = (normalizedCpu * cpuWeight) + (normalizedRam *
ramWeight) + (normalizedStorage * storageWeight);

          totalPerformance += performance;
        });
        const averagePerformance = totalPerformance / hosts.length;


        return averagePerformance * 100; // Convert to a scale of 0-100
      };

      useEffect(() => {
        const calculatedPerformances = calculateServerPerformance(data);
        setServerPerformances(calculatedPerformances.toFixed(3));
      }, [data]);

  return (
    <Card>
        <div className='d-flex px-2 justify-content-center py-4'>
```

```
            <div className={`me-4 px-4 text-center d-flex align-items-center
${serverPerformances >50 ? "icon sucess" : "icon danger"}`} >
{serverPerformances > 50 ? <i className="bi bi-database-check"></i> : <i
className="bi bi-database-exclamation"></i>} </div>
            <div className='align-items-center text-start'>
                <p className='fw-bold'>Total Performance</p>
                <div className=' big-font'> {serverPerformances}% </div>
            </div>
        </div>
    </Card>
  )
}

export default TotalPerformance
```

5.1.16 Verilerin Takvim İle Gösterilmesi

Kullanıcıya sunucuların son yeniden başlatma verilerinin mevcut durumu görsel olarak sunulur. Bu kullanıcının verileri hızlı bir şekilde tanımlamasına olanak tanır. React uygulamasında takvim oluşturabilmek için moment, react-big-calendar kütüphaneleri kullanılmıştır.

```
----bazı import kısımları kırpılmıştır----
import { Calendar, momentLocalizer } from 'react-big-calendar';
import 'react-big-calendar/lib/css/react-big-calendar.css';
import moment from 'moment';

const CalendarComponent = () => {
    const data= useSelector(state => state.data.data)
    const localizer = momentLocalizer(moment);
    const events = data?.map(data =>
        (
            {
                start: new Date(data.last_reboot_time),
                end: new Date(data.last_reboot_time),
                title : data.hostname
            }
        )
    )


  return (
    <Card>
        <div className='p-4'>
            <h5 className='text-start'>Last Reboot Times</h5>
            <Calendar
                localizer={localizer}
                events={events}
```

```
                startAccessor="start"
                endAccessor="end"
                style={{ height: 500 }}
                />
            </div>
    </Card>
  )
}
```

5.1.17 Eşik İhlali Hesaplanması ve Gösterilmesi

Kullanıcının kritik verilere anında dikkat çekilmesini saplamak adına eşik ihlali hesaplaması ile alarm bileşeni oluşturulur. Veriler belirli bir eşiği aştığında (%90) kullanıcıya belirli ve renk ikonlarla gösterilir.

```
const Threshold = () => {
    const data = useSelector(state => state.data.data)
    const [thresholdBreaches, setThresholdBreaches] = useState([]);


const checkThresholdBreaches = () => {
    const newBreachedMetrics = data.reduce((breaches, entry) => {
      const { hostname, cpu_usage, ram_usage } = entry;
      const breachedMetricsEntry = { hostname };

      if (parseFloat(cpu_usage) > 90) {
        breachedMetricsEntry.cpu_usage = parseFloat(cpu_usage);
      }

      if (parseFloat(ram_usage) > 90) {
        breachedMetricsEntry.ram_usage = parseFloat(ram_usage);
      }

      if (Object.keys(breachedMetricsEntry).length > 1) {
        breaches.push(breachedMetricsEntry);
      }

      return breaches;
    }, []);
    setThresholdBreaches(newBreachedMetrics);
};

  useEffect(() => {
        checkThresholdBreaches();

    }, []);
```

```
  return (
    <Card>
        <div className='p-4'>
            <h5 className='text-start'>Threshold Breaches</h5>
            {thresholdBreaches.length > 0 &&
thresholdBreaches.map(thresholdBreach => (
                    <div className='d-flex mb-3 align-items-center'>
                        <p className='me-auto'>Threshold breaches detected
for:  {thresholdBreach.hostname}</p>
                        <div className='d-flex flex-column'>
                        {'cpu_usage' in thresholdBreach && <p className='mx-1
d-flex'> CPU { thresholdBreach?.cpu_usage}% <div className='icon purple ms-2
px-1 mb-1'><i class="bi bi-cpu"></i> </div></p> }
                        {'ram_usage' in thresholdBreach && <p className='mx-1
d-flex'> RAM { thresholdBreach?.ram_usage}% <div className='icon purple ms-2
px-1 mb-1'> <i class="bi bi-memory"></i> </div></p> }
                        {'storage_usage' in thresholdBreach && <p> Storage {
thresholdBreach?.storage_usage}% <div className='icon purple ms-2 px-1'> <i
class="bi bi-sd-card"></i></div></p> }
                        </div>
                        <div className='icon danger px-1  ms-1'><i
className="bi bi-exclamation-triangle"></i></div>
                    </div>
            ))
            }
            {
                thresholdBreaches.length === 0 && (
                    <div className='d-flex flex-column align-items-center
justify-content-center' >
                        <div className='icon succes py-3 w-25 my-3'>
                            <i className=" bi bi-check-circle-fill"></i>
                        </div>
                        <p>No threshold breaches were detected.</p>
                    </div>
                )
            }
        </div>
    </Card>
  )
}

export default Threshold
```

5.1.18 Sunucu Performansları

Toplam sunucu performansının dışında kullanıcıya her bir sunucunun kendi performansı da gösterilir. Toplam performans fonksiyonuna benzer olarak

hesaplama yapılır ancak ortalama hesaplama yerine her sunucu için sunucu adı ve performansını içeren bir nesne döndürür.

```
const ServerPerformance = () => {
    const data = useSelector(state=> state.data.data)
    const [serverPerformances, setServerPerformances] = useState();
    const [showMore, setShowMore] = useState(false)
    const calculateServerPerformance = (hosts) => {
        // Define weights for each metric (adjust as per your requirements)
        const cpuWeight = 0.5;
        const ramWeight = 0.3;
        const storageWeight = 0.2;

        const serverPerformances = hosts.map((host) => {
          // Normalize values to a scale of 0-100 (assuming the metrics are in
percentage)
          const normalizedCpu = host.cpu_usage / 100;
          const normalizedRam = host.ram_usage / 100;
          const normalizedStorage = host.storage_usage / 100;

          // Calculate the weighted average for each host
          const performance = (normalizedCpu * cpuWeight) + (normalizedRam *
ramWeight) + (normalizedStorage * storageWeight);

          // Return an object with host information and performance score
          return {
            hostName: host.hostname,
            performance: performance * 100, // Convert to a scale of 0-100
          };
        });

        // Return an array of server performance scores for each host
        return serverPerformances;
      };
      useEffect(() => {
        const calculatedPerformances = calculateServerPerformance(data);
        setServerPerformances(calculatedPerformances);
      }, [data]);

 const handleShowMore = () => {
  setShowMore(current => !current)
 }
      return (
       <Card>
         <div className='d-flex justify-content-between align-items-center
tech-row mb-1 p-3'>
```

```
            <div className='d-flex w-100 flex-column justify-content-
around  '>
            <h4 className='text-start mb-4'>Performance Scor</h4>
                <div className='d-flex w-100 align-items-center justify-
content-between '>
                    <h6 className='fw-bold'>Host Name </h6>
                    <h6 className='fw-bold'>Performance</h6>
                </div>
                {
                  !showMore ?
                    serverPerformances&&
serverPerformances.map((performance,index) => (
                       index <5 ? <div key={index} className='d-flex w-100 mb-
2 align-items-center justify-content-between '>
                            <p>  {performance.hostName} </p>
                            <div className='d-flex align-items-center'>
                                <div className='tech-bar me-1'
style={{width:  performance.performance}}></div>
                                <div className='pb-0'>
{performance.performance.toFixed(3)}%  </div>
                            </div>
                        </div> : ""
                    ))
                    :
                    serverPerformances&&
serverPerformances.map((performance,index) => (
                      <div key={index} className='d-flex w-100 mb-2 align-
items-center justify-content-between '>
                          <p>  {performance.hostName} </p>
                          <div className='d-flex align-items-center'>
                              <div className='tech-bar me-1'
style={{width:  performance.performance}}></div>
                              <div className='pb-0'>
{performance.performance.toFixed(3)}%  </div>
                          </div>
                      </div>
                  ))
                }

            <button className='border-0 mt-3 bg-transparent text-primary'
onClick={handleShowMore}> {showMore ? <p>Hide <i class="bi bi-arrow-up"></i>
</p>: <p>Show More<i class="bi bi-arrow-down"></i></p>}  </button>
            </div>
        </div>
       </Card>
      )
}

export default ServerPerformance
```

5.1.19 Verilerin Çizgi Grafik ile Gösterimi

Verilerin kullanıcıya kolay ayırt edilebilir ve karşılaştırılabilir sunumu için çizgi garfik oluşturulur. react-chartjs-2, chart.js ve randomcolor kütüphaneleri indirilip bileşene import edilir. Kütüphaneleri indirme aşağıdaki gibidir:

```
npm install React-chartjs-2 chart.js randomcolor
```

Arka uçtan gelen veriler etiket(label) ve veriden oluşan grafiğe uygun bir veri kümesi(dataset) oluşturulur. randomColor() method ile her bir çizgi için farklı bir renk atanır. Çizgi grafiği kodu aşağıdaki gibidir:

```
----import kısmı kırpılmıştır-----

const ChartComponent = () => {
  const data = useSelector(state => state.data.data)

  const dataset = data?.map((obj) => ({
    label: obj.hostname,
    data: [obj.cpu_usage, obj.ram_usage, obj.storage_usage],
  }));
  const labels = ["CPU Usage" , "Ram Usage" , "Storage Usage"];

  const chartData = dataset && {
    labels,
    datasets: dataset.map(data =>(
      {
        label: data?.label,
        data: data?.data,
        borderColor: randomColor(),
        backgroundColor: randomColor(),
      }
    )

      )
  };

  const chartOptions = {
    responsive: true,
    plugins: {
      legend: {
```

```
        position: 'top' ,
      },
      title: {
        display: true,
        text: 'Chart.js Line Chart',
      },
    },
    scales: {
      x: {
        grid: {
          borderColor: '#ccc', // Change the color of the X axis grid line
        },
      },
      y: {
        grid:{
          borderColor: '#ccc', // Change the color of the X axis grid line
        },
      },
    },
  };



  return (
      <Card>
        <div className='d-flex align-items-center justify-content-center '
style={{height : "350px"}}>
         <Line data={chartData} options={chartOptions} />
        </div>
      </Card>
  );
};
```

5.1.20 Özet Tablosu Oluşturumu

Daha sonra Hosts sayfasında detaylı gösterilecek verilerin ilk 10 kısmı tablo şeklinde gösterilir. JavaScript'e ait slice() method ile verilerin ilk 10'u elde edilir. Özet tablosuna ait kod aşağıdaki gibidir:

```
-----import kısmı kırpılmıştır----
const SummaryTable = () => {
    const data = useSelector(state => state.data.data)
    const firstTenItems = data.slice(0, 7);

  return (
    <Card>
        <div className='p-3'>
            <h4 className='text-start mb-3'>First 10</h4>
```

```
            <div className=' d-flex align-items-center justify-content-
between'>
                <h6>Host Name</h6>
                <h6>CPU Usage</h6>
                <h6>RAM Usage</h6>
                <h6>Storage Usage</h6>
                <h6>Kernel Version</h6>
                <h6>Last Reboot Time</h6>
            </div>
                {
                    firstTenItems?.map((item,index )=> (
                        <div key={index} className='row text-start px-3 '>
                            <p className='col px-0'> {item.hostname} </p>
                            <p className='col'> {item.cpu_usage} </p>
                            <p className='col'> {item.ram_usage} </p>
                            <p className='col'> {item.storage_usage} </p>
                            <p className='col'> {item.kernel_version} </p>
                            <p className='col'> {item.last_reboot_time} </p>
                        </div>
                    ))
                }
        </div>
                <NavLink to="/hosts" className="text-decoration-none mb-2 mt-
2" >Show More <i class="bi bi-arrow-right"></i></NavLink>
    </Card>

  )
}
```

5.1.21 Yuvarlak Diyagram (Pie Chart) Oluşturumu

Kernel version verisini görsel biçimde özetlemek için bir diyagram oluşturulur. Çizgi grafik için kullanılan kütüphaneler kullanılır. Verilerin diyagramda gösterilebilmesi için öncellikle hangi itemden kaç tane olduğu bulunur. Daha sonrasında item ve itemin toplamı şeklinde kümeli gösterebilmek için bir obje oluşturulur. Yuvarlak diyagrama ait kod aşağıdaki gibidir:

```
----import kısmı kırpılmıştır----
const PieChart = () => {
  const data = useSelector(state=> state.data.data)
  const kernel = data?.map(data => data.kernel_version)
  const pieChartLabels = []
  const pieChartData = []

const countElements = (arr) => {
    const counts = {};
```

```
  arr.forEach((element) => {
    if (counts[element]) {
      counts[element]++;
    } else {
      counts[element] = 1;
    }
  });
  const transformedCounts = [];

  for (const key in counts) {
    const transformedObject = {
      x: counts[key],
      y: key,
    };
    pieChartLabels.push(key)
    pieChartData.push(counts[key])
    transformedCounts.push(transformedObject);
  }

  return transformedCounts;
};

const result = countElements(kernel);
const options = {
  responsive  : true
}


 const Piedata = {
  labels:pieChartLabels,
  datasets: [
    {
      label: '',
      data: pieChartData,
      backgroundColor: [
        'rgba(255, 99, 132, 0.2)',
        'rgba(54, 162, 235, 0.2)',
        'rgba(153, 102, 255, 0.2)',
        'rgba(255, 159, 64, 0.2)',
        'rgba(255, 206, 86, 0.2)',
      ],
      borderColor: [
        'rgba(255, 99, 132, 1)',
        'rgba(54, 162, 235, 1)',
        'rgba(153, 102, 255, 1)',
        'rgba(255, 159, 64, 1)',
        'rgba(255, 206, 86, 0.2)',
      ],
```

```
        borderWidth: 1,
      },
    ],
  };

  return (
      -------bazı JSX kodları kırpılmıştır------
           <Pie data={Piedata} options={options} className=""/>
        -------bazı JSX kodları kırpılmıştır------  )
}
```

5.1.22 Hosts Sayfası

Arka uçtan alınan, system bilgilerini gösteren veriler ayrı bir sayfada tablo şeklinde kullanıcıya sunulur. Bileşen oluşturulduğu anda sortData fonksiyonu çağrılır. Bu fonksiyon ile varsayılan sütuna göre sıralama yapılır. Kullanıcının sıralamayı değiştirmesi ile tıklanan sütün azalan veya artan sıraya göre sıralanır. Kullanıcının her bir hostu inceleyebilmesi için buton bulunur, bu buton ile hostun id'si ile yeni bir url oluşturulur. useLocation, URLSearchParams methodları ile bu değişiklik control edilir ve değişiklik gerçekleştiği takdirde kullanıcı Host Details sayfasına yönlendirilir. Host sayfasına ait kod aşağıdadır:

```
------import kısımları kırpılmıştır----
const Hosts = () => {
  const {data}= useSelector(state => state.data)
  const location = useLocation();
  const searchParams = new URLSearchParams(location.search);
  const details = searchParams.get('details');
  const [sortOrder, setSortOrder] = useState('asc');
  const [selectedColumn, setSelectedColumn] = useState(null);



  useEffect(() => {
    sortData(selectedColumn, sortOrder);
  }, []);

  const sortData = (column, order) => {
    const sorted = [...data];
    sorted.sort((a, b) => {
      if (column === 'cpu_usage') {
        return order === 'asc' ? a.cpu_usage - b.cpu_usage : b.cpu_usage -
a.cpu_usage;
      } else if (column === 'ram_usage') {
        return order === 'asc' ? a.ram_usage - b.ram_usage : b.ram_usage -
a.ram_usage;
```

```
      } else if (column === 'storage_usage') {
        return order === 'asc' ? a.storage_usage - b.storage_usage :
b.storage_usage - a.storage_usage;
      }
    });

    return sorted;
  };

  const handleSort = (column) => {
    if (column === selectedColumn) {
      setSortOrder(sortOrder === 'asc' ? 'desc' : 'asc');
    } else {

      setSelectedColumn(column);
      setSortOrder('asc');
    }
  };

  const sortedData = sortData(selectedColumn, sortOrder);

  if(details){
    return <HostDetails />
  }

  return (
    <div>
        <Header />
        <div className='container'>
          <Card>
          <div className='row text-start fw-bold mb-3 p-4 '>
              <div className='col'>Host Name</div>
              <div className='col' onClick={() => handleSort("cpu_usage")}
>CPU Usage
              {selectedColumn === 'cpu_usage' ? (sortOrder === 'asc' ? ' ▼' :
' ▲') : ' ▼'}
            </div>
              <div className='col' onClick={() => handleSort("ram_usage")}>RAM
Usage
          <span>{selectedColumn === 'ram_usage' ? (sortOrder === 'asc' ? ' ▼'
: ' ▲') : ' ▼'}</span></div>
              <div className='col' onClick={() =>
handleSort("storage_usage")}>Storage Usage
        <span>{selectedColumn === 'storage_usage' ? (sortOrder === 'asc' ? '
▼' : ' ▲') : ' ▼'}</span></div>
              <div className='col'>Kernel Version</div>
              <div className='col'>Last Reboot Time</div>
              <div className='col'>Other</div>
          </div>
```

```
          {
            sortedData?.map(item => (
              <div className='row text-start mb-2 px-4 pb-2 mx-1 border-
bottom'>
              <div className='col'> {item.hostname} </div>
              <div className='col'>{item.cpu_usage}</div>
              <div className='col'>{item.ram_usage}</div>
              <div className='col'> {item.storage_usage}  </div>
              <div className='col'> {item.kernel_version}  </div>
              <div className='col'> {item.last_reboot_time}  </div>
              <div className='col'> <NavLink to={`?details=${item.id}`}
className='border-0 details-button px-2 py-1'>Details</NavLink> </div>
          </div>
            ))
          }

          </Card>
        </div>
    </div>
  )
}
```

5.1.23 Host Detayları Sayfası

Hosts sayfasında gösterilmeyen bazı veriler burada gösterilir. Kullanıcının belirli bir host hakkında detaylı bilgi alabilmesini sağlar. Verilerin içinden JavaScript filter method ile istenen host alınır. Ayrıca bu sayfada komut gönderme, gelen cevabı görüntüleme kısmı bulunur. handleCommandInsert fonksiyonunda kullanıcıdan host adı ve komut alınarak redux store'da oluşturulmuş sendCommand fonksiyonuna gönderilir. Komuttan dönen cevap kullanıcıya gösterilir. İlgili kod aşağıdadır:

```
-----import kısımları kırpılmıştır------
const HostDetails = () => {
    const data = useSelector(state => state.data.data)
    const location = useLocation();
    const searchParams = new URLSearchParams(location.search);
    const details = searchParams.get('details');
    const filteredData = data?.filter(item => item.id == details)
    const host = filteredData[0]
    const commandInputRef = useRef()
    const dispatch = useDispatch()
    const [currentCommand, setCommand] = useState()
    const [currentCommandResult , setCurrentCommandResult] = useState()

    useEffect(() => {
        setCurrentCommandResult(prevState => {
          if(host?.command_result != prevState) {
```

```
                return host?.command_result
            }
        })
    } , [host])


    const handleCommandInsert = () => {
     setCommand(commandInputRef.current.value)
          dispatch(
               sendCommand({
                    hostname: host.hostname,
                    command: commandInputRef.current.value
               })
          )
          dispatch(getData())
          commandInputRef.current.value = ''
    }
  return (
            -------bazı JSX kodları kırpılmıştır----
  )
}
```

5.1.24 Profil Sayfası

Kullanıcıdan, kullanıcı adını, eski şifresini ve yeni şifresini alınıp redux store'da oluşturulan changePassword method ile şifre değiştirilir. İlgili kod aşağıdadır:

```
-----import kısımları kırpılmıştır-----
const ProfilePage = () => {
    const message = useSelector(state => state.profile.message )
    const usernameRef = useRef(null)
    const oldPasswordRef= useRef(null)
    const newPasswordRef = useRef(null)
    const dispatch=useDispatch()

    const handleSubmit = (e) => {
        e.preventDefault()
        dispatch(
            changePassword({
                username: usernameRef.current?.value,
                old_password: oldPasswordRef.current?.value,
                new_password : newPasswordRef.current?.value
            })
        )
    }

  return (
    -------bazı JSX kodları kırpılmıştır-----
```

```
      <form onSubmit={handleSubmit} className='d-flex flex-column col-5 align-
self-center change-password_form align-items-start'>
      <Label  htmlFor="username" id="username"    placeholder="username"
label="Username" propsRef={usernameRef} />
      <Label  htmlFor="oldPassword" id="oldPassword"    placeholder="Old
Password" label="Old Password" propsRef={oldPasswordRef} />
       <Label  htmlFor="newPassword" id="newPassword"    placeholder="New
Password" label="New Password" propsRef={newPasswordRef} />

            -------bazı JSX kodları kırpılmıştır-----
```

5.2 Kullanıcı Tarafı Uygulamanın İmplementasyonu

Uygulamanın kullanıcı tarafı Python dilinde kodlanmış olup şu teknik özelliklere sahiptir: sunucu taraflı uygulamaya bağlanma, sistem hakkındaki bilgileri toplama, toplanan bilgileri yönetim sunucusuna gönderme, sunucudan aldığı sistem komutlarını çalıştırma ve cevaplarını sunucuya dönme.

5.2.1 Kütüphaneleri İmport Etme

Kütüphaneleri kullanabilmemiz için öncelikle sistemimize kurmamız gerekir.

Installation Steps in Linux Ubuntu/Debian

```
  sudo pip install <kütüphaneAdı>
```

Kütüphanelerin kodda kullanılabilmesi için öncelikle import edilmelidir. Aşağıdaki kod parçası kütüphanelerin import işlemini içerir.

```
import os
import psutil
import time
import requests
import json
import subprocess
import socket
import threading
import platform
```

5.2.2 HostName Bilgisinin Eldesi

Import edilen socket kütüphanesinin gethostname() fonksiyonu kullanılarak hostname bilgisi elde edilir.

```
def getHostName(self):
        self.hostName = socket.gethostname()
        return str(self.hostName)
```

5.2.3 CPU Kullanım Bilgisi Eldesi

Import edilen psutil kütüphanesinin cpu_percent() fonksiyonu kullanılarak CPU kullanım bilgisi elde edilir.

```
def cpu_usage(self):
return psutil.cpu_percent(interval=1)
```

5.2.4 RAM Kullanım Bilgisi Eldesi

Import edilen psutil kütüphanesinin virtual_memory() fonksiyonu kullanılarak RAM kullanım bilgisi elde edilir.

```
def ram_usage(self):
svmem = psutil.virtual_memory()
return svmem.percent
```

5.2.5 Kernel Version Bilgisi Eldesi

Import edilen platform kütüphanesinin uname() fonksiyonu kullanılarak kernel versiyon bilgisi elde edilir.

```
def kernel_version(self):
kernel_version = platform.uname().release
return kernel_version
```

5.2.6 Alan Kullanım Bilgisi Eldesi

Import edilen psutil kütüphanesinin disk_usage() fonksiyonu kullanılarak kullanılan alan bilgisi elde edilir.

```
def storage_usage(self):
        storageUsage = psutil.disk_usage("/")
        return str(storageUsage[3])
```

5.2.7 Çalışan Servisler Listesi Bilgisi

Import edilen subprocess kütüphanesi aracılığı ile sistem üzerinde yanda belirtilen parametreler ile systemctl komut çalıştırılarak çalışan servislerin listesi elde edilir '"systemctl --no-pager list-units --type=service --state=running --plain –no-legend".

```
def get_running_services(self):
cmd = ['systemctl', '--no-pager', 'list-units', '--type=service', '--
state=running', '--plain', '--no-legend']
output = subprocess.check_output(cmd, universal_newlines=True)
running_services = [line.split()[0] for line in output.splitlines()]
services_string = ", ".join(running_services)
return services_string
```

5.2.8 Son Yeniden Başlatma Zamanı Bilgisi

Debian sistemler üzerinde bulunan /proc/uptime dosyası okunarak son yeniden başlatma zamanı elde edilir.

```
def get_last_reboot_time(self):
        with open('/proc/uptime', 'r') as uptime_file:
            uptime_seconds = float(uptime_file.readline().split()[0])
            last_reboot_timestamp = time.time() - uptime_seconds
            last_reboot_time = time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(last_reboot_timestamp))
            return last_reboot_time
```

5.2.9 Komut Çalıştırma

Import edilen subprocess kütüphanesinin run() fonksiyonu kullanılarak sistem üzerinde istenilen komutlar çalıştırılır.

```
def execute_command(self, command):
        result = subprocess.run(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
        return result.stdout.decode('utf-8'), result.stderr.decode('utf-8')
```

5.2.10 Sunucudan Komut Alma

Sunucu uygulama üzerinden komut almak için, her 30 saniyede bir ilgili uç noktaya istek atılır ve komut olması durumunda komut çalıştırılır. Bir güvenlik önlemi olarak, komut alma isteğinin istemci taraflı uygulamadan geldiğini doğrulamak adına x-access-key isimli bir HTTP başlığı kod üzerinde tanımlanmış bir erişim kodu ile birlikte isteğe eklenir. İlgili işlem aşağıdaki gibi uygulanmıştır:

```
command_output, command_error = "",""
response = requests.get(command_url, headers={'Content-Type':
'application/json', 'x-access-key':'second_public_key'})
if response.status_code == 200:
    commands = response.json()
    cmd = commands['command']
  #  for command in commands:
    command_output, command_error = info.execute_command(cmd)
else:
    commands = []
```

5.2.11 Sunucuya Bilgileri Gönderme

Daha önce fonksiyonlar aracılığı ile toplanan bilgiler JSON formatına dönüştürülür ve belirlenen API uç noktası üzerine gönderilir. Ilgili işlem aşağıdaki gibi uygulanmıştır:

```
#get kernel version
    kernel_version = info.kernel_version()
    #get running services
    running_services = info.get_running_services()
    #get last reboot time
    last_reboot_time = info.get_last_reboot_time()
    #get hostname
    hostname = info.getHostName()
    # Get the CPU usage
    cpu_usage = info.cpu_usage()
    # Get the RAM usage
    ram_usage = info.ram_usage()
    # Get the storage usage
    storage_usage = info.storage_usage()
    #for partition in storage_usage:
     #   print("Partition {} usage: {}% ({} GB)".format(partition[0],
partition[1], partition[2]))
```

```
    # Create a dictionary with the collected data and command output
    data = {
        "host_name":hostname,
        "cpu_usage": cpu_usage,
        "ram_usage": ram_usage,
        "storage_usage": storage_usage,
        "kernel_version": kernel_version,
        "running_services": running_services,
        "last_reboot_time":last_reboot_time,
        "command_output": (command_output+command_error)
    }
    # Encode the data as JSON
    json_data = json.dumps(data)
    # Send the data to the endpoint
    response = requests.post(data_url, data=json_data, headers={'Content-
Type': 'application/json', 'x-access-key':'second_public_key'})
```

5.3 Sunucu Taraflı Uygulamanın İmplementasyonu

Sunucu taraflı uygulama üzerinde Flask framework'u kullanılmış olup şu teknik özelliklere sahiptir: istemci taraflı uygulamadan gelen verileri işleme ve MySQL veritabanı üzerinde tutma, istemci taraflı uygulama üzerine komutlar gönderme, JWT token yapılandırması ile yetkilendirme işlemleri, kullanıcı parola değiştirme.

5.3.1 Flask Kütüphanelerinin İmport Edilmesi

Kütüphaneleri kullanabilmemiz için öncelikle sistemimize kurmamız gerekir.

Installation Steps in Linux Ubuntu/Debian

```
sudo pip install <kütüphaneAdı>
```

Kütüphanelerin uygulama üzerinde kullanılması için import edilmesi gerekmektedir. Aşağıda bulunan kod parçacığı ilgili import işlemini gösterir:

```
from flask import Flask, jsonify
from flask import request
from flask_mysqldb import MySQL
```

```
from flask_cors import CORS
import jwt
import datetime
import json
```

5.3.2 Veritabanı Bağlantısı Sağlanması

Veritabanında kullanılacak değişkenlerin tanımlanması aşağıdaki gibi yapılmıştır:

```
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'sammy'
app.config['MYSQL_PASSWORD'] = 'password'
app.config['MYSQL_DB'] = 'monitorProject'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'
```

5.3.3 JWT ve İstemci Taraflı Gizli Anahtarların Tanımlanması

Web uygulamasında kullanılacak olan JWT tokenlerinin güvenliğinden sorumlu gizli anahtar ve sunucu üzerine gelen isteklerin, istemci tarafından gönderildiği doğrulama adımında kullanılacak gizli anahtar için değişkenler oluşturulması gerekmektedir. İlgili işlem aşağıdaki gibi yapılmıştır:

```
jwt_secret="mysupersecretsecretkey"
client_access_key = "second_public_key"
```

5.3.4 Istemci Üzerinden Gönderilen Anahtarın Doğrulanması

İstemci üzerinden gelen anahtarın doğrulanması ve client_protected adlı bir fonksiyon oluşturulmuş ve ilgili fonksiyon içerisinde IF yapısı ile kontrol sağlanmıştır.

```
def client_protected(accessKey):

    if accessKey == client_access_key:
        return True
```

```
    else:
        return False
```

5.3.5 JWT Anahtarının Doğrulanması

JWT anahtarının doğrulanması için panel_protected adında bir fonksiyon oluşturulmuş ve içerisinde JWT kütüphanesi üzerinde bulunan decode() fonksiyonu kullanılmıştır.

```
def panel_protected(token):
    # Get the JWT token from the authorization header
    try:
        # Decode the JWT token using the secret key
        payload = jwt.decode(token, jwt_secret, algorithms=['HS256'])
        # If the token is valid, return a success response
        return True
    except jwt.exceptions.DecodeError:
        # If the token is invalid, return an error response
        return False
```

5.3.6 İstemci Hostname Bilgisi Kontrolü

İstemciler üzerinde kullanılan anahtar bilgi hostname bilgisi olarak tanımlanmıştır. Bu nedenle, istemci üzerinden gelen bir istek değerlendirilirken, ilgili istemcinin daha önce veritabanına eklenmiş bir cihaz olup olmadığını kontrol etmek adına, Flask üzerinde check_hostname() adında bir fonksiyon oluşturulmuştur. İlgili fonksiyon MySQL sorguları kullanılarak ilgili hostname bilgisinin varlığını kontrol eder.

```
def check_hostname(hostname):
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT EXISTS(SELECT 1 FROM statistics WHERE hostname =
%s)", [hostname])
    result = cursor.fetchone()
    cursor.close()
    if len(result) >= 1:
        return True
    else:
        return False
```

5.3.7 İstemciden Gelen Bilgilerin Veritabanına Aktarılması

Istemciden gelen bilgilerin veritabanına aktarılması için Flask üzerinde /insertStatistics API uç noktası oluşturulmuştur. Ilgili uç noktaya gelen JSON veriler requests kütüphanesi üzerinde bulunan get_json() fonksiyonu ile okunmuş ve mysql kütüphanesi üzerinde bulunan execute() fonksiyonu ile veritabanı üzerine eklenmiştir. Ek olarak, bilgilerin istemci üzerinden geldiğini doğrulamak adına, veritabanı işlemlerinden önce x-access-key bilgisi daha önce tanımlanan client_protected fonksiyonu ile kontrol edilmiştir.

```
@app.route("/insertStatistics", methods=['POST'])
def register_client():
    accessKey = request.headers['x-access-key']
    if client_protected(accessKey):
        if request.method == 'POST':
            data = request.get_json()
            hostname = data['host_name']
            if check_hostname(hostname) == False:
                cpu_usage = data['cpu_usage']
                storage_usage = data['storage_usage']
                ram_usage = data['ram_usage']
                command_output = data['command_output']
                kernel_version = data['kernel_version']
                running_services = data['running_services']
                last_reboot_time = data['last_reboot_time']
                cur = mysql.connection.cursor()
                cur.execute("INSERT INTO statistics (hostname, cpu_usage,
storage_usage, ram_usage,kernel_version, running_services, last_reboot_time,
command_result) VALUES (%s, %s, %s, %s, %s, %s, %s,%s)", (hostname, cpu_usage,
storage_usage, ram_usage, kernel_version, running_services,last_reboot_time,
command_output))
                mysql.connection.commit()
                cur.close()
            else:
                cpu_usage = data['cpu_usage']
                storage_usage = data['storage_usage']
                ram_usage = data['ram_usage']
                command_output = data['command_output']
                kernel_version = data['kernel_version']
                running_services = data['running_services']
                last_reboot_time = data['last_reboot_time']
                cur = mysql.connection.cursor()
                cur.execute("UPDATE statistics set cpu_usage=%s,
storage_usage=%s, ram_usage=%s, kernel_version=%s,
```

```
running_services=%s,last_reboot_time=%s, command_result=%s where hostname=%s",
( cpu_usage, storage_usage, ram_usage, kernel_version, running_services,
last_reboot_time, command_output, hostname))
                mysql.connection.commit()
                cur.close()
            return "OK",200
    else:
        return "Unauthorized",401
```

5.3.8 Istemci Üzerine Gönderilecek Komutların Veritabanına Eklenmesi

Istemciden üzerine gönderilecek komutların veritabanına eklenmesi için Flask üzerinde /insertCommand API uç noktası oluşturulmuştur. Ilgili uç noktaya gelen JSON veriler requests kütüphanesi üzerinde bulunan get_json() fonksiyonu ile okunmuş ve mysql kütüphanesi üzerinde bulunan execute() fonksiyonu ile veritabanı üzerine eklenmiştir. Ek olarak, bilgilerin yetkili bir kullanıcı üzerinden geldiğini doğrulamak adına, veritabanı işlemlerinden önce JWT token bilgisi daha önce tanımlanan panel_protected fonksiyonu ile kontrol edilmiştir.

```
@app.route("/insertCommand", methods=['POST'])
def set_command():
    auth_header = request.headers.get('Authorization')
    token = auth_header.split(' ')[1]
    if panel_protected(token):
        if request.method == 'POST':
            data = request.get_json()
            hostname = data['hostname']
            command = data['command']
            if check_hostname(hostname):
                cur = mysql.connection.cursor()
                cur.execute("UPDATE statistics set command=%s,
command_result='' where hostname=%s", ( command, hostname))
                mysql.connection.commit()
                cur.close()
                return "OK",200
            else:
                return "Host does not exists",404
        else:
            return "Method not allowed", 401
    else:
        return "Unauthorized",401
```

5.3.9 İstemci Uygulamaya Komut Gönderilmesi

Veritabanı üzerine komutların eklenmesinin ardından, ilgili komutların istemci uygulamaya döndürülmesi için Flask üzerinde /get/command/<name> API uç noktası oluşturulmuştur. İlgili uç nokta üzerine gelen istekler ilk olarak x-access-key kontrolüne tabi tutulmuş ve ilgili anahtarın geçerli olması durumunda komut gösterilmiştir.

```
@app.route("/get/command/<name>", methods=['GET'])
def getCommand(name):
    accessKey = request.headers['x-access-key']

    if client_protected(accessKey):
        if check_hostname(name):
            cursor = mysql.connection.cursor()
            cursor.execute("SELECT command FROM statistics where hostname=%s",
[name])
            row = cursor.fetchone()
            cursor.close()
            return row
        else:
            return "Unauthorized", 401
         # Convert the data to a list of dictionaries
 return "Unauthorized", 401
```

5.3.10 Giriş İşlemleri

Web uygulamasını kullanılacak olan kullanıcılar için Flask üzerinde /login API uç noktası tanımlanmıştır. Ilgili uç noktaya gelen kullanıcıadı ve parola bilgisi veritabanı üzerinde doğrulanmış, ve ilgili verilerin geçerli olması durumunda kullanıcılara JWT anahtarları gönderilmiştir.

```
@app.route('/login', methods=['POST'])
def login():
    # Get the username and password from the request body
    username = request.json.get('username')
    password = request.json.get('password')
```

```
    # Check if the username and password are correct
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM users WHERE username = %s AND password = %s",
(username, password))
    user = cur.fetchone()

    if user:

        # Generate a JWT token with the user's username
        payload = {'username': username}
        token = jwt.encode(payload, jwt_secret, algorithm='HS256')


        return token
    else:
        return jsonify({'error': 'Invalid username or password'}), 401
```

5.3.11 Web Uygulama Parola Güncelleme

Web uygulamasını kullanan kullanıcıların, parolalarını değiştirebilmeleri için Flask üzerinde /reset_password uç noktası oluşturulmuştur. Ilgili uç noktaya gelen istekler ilk olarak JWT anahtar kontrolüne tabi tutulmuş ardından veriler json kütüphanesinde bulunan get_json() fonksiyonu ile okunmuş ve eski parolanın doğru olarak verilmesi durumunda parola güncelleme işlemleri gerçekleştirilmiştir.

```
@app.route('/reset_password', methods=['POST'])
def reset_password():
    auth_header = request.headers.get('Authorization')
    token = auth_header.split(' ')[1]
    if panel_protected(token):
        if request.method == 'POST':
            username = request.json.get('username')
            old_password = request.json.get('old_password')
            new_password = request.json.get('new_password')
            cur = mysql.connection.cursor()
            cur.execute("SELECT password FROM users WHERE username = %s",
(username,))
            user = cur.fetchone()
            if user:
                if user['password'] == old_password:
                    # Reset the password
```

```
                    cur.execute("UPDATE users SET password = %s WHERE username
= %s",
                                (new_password, username))
                    mysql.connection.commit()
                    cur.close()
                    return "Password reset successfully!"
                else:
                    return "Invalid old password."
            else:
                return "Invalid username."
    else:
        return "Unauthorized",401
```

KAYNAKÇA

1. Flask, Welcome to Flask, https://flask.palletsprojects.com/en/2.2.x/
1. Flask, Flask-MySQL, https://flask-mysql.readthedocs.io/en/stable/
2. Tedboy, flask.response, https://tedboy.github.io/flask/generated/generated/flask.Response.html
3. PalletsProjects, The Request Context, https://flask.palletsprojects.com/en/1.1.x/reqcontext/
4. ReadTheDocs, Flask-JSON, https://flask-json.readthedocs.io/en/latest/
5. Python, Secrets, https://docs.python.org/3/library/secrets.html
6. Python, Hashlib, https://docs.python.org/3/library/hashlib.html
7. ReadTheDocs, psutil documentation, https://docs.python.org/3/library/hashlib.html
8. Python, Miscellaneous operating system interfaces, https://docs.python.org/3/library/os.html
9. Python, Simple Lexical Analysis, https://docs.python.org/3/library/shlex.html
10. Python, Thread-Based parellelism, https://docs.python.org/3/library/threading.html
11. Python, Low-level networking interface, https://docs.python.org/3/library/socket.html
12. React, https://react.dev/
13. React, Home v6.12.0 React Router https://reactrouter.com/en/main
14. React, React Charts https://react-chartjs-2.js.org/
15. Redux, Redux Toolkit, https://redux-toolkit.js.org/