

React JS Integration In Communication And Information Office Asset Management Interface

Penerapan React JS Pada Antarmuka Manajemen Aset Dinas Komunikasi Dan Informatika

Brigide Tirenia Loresta¹, Ade Eviyanti²

[{221080200020@mhs.umsida.ac.id}](mailto:221080200020@mhs.umsida.ac.id) , [{adeeviyanti@umsida.ac.id}](mailto:adeeviyanti@umsida.ac.id)

^{1,2}UMSIDA, Fakultas Sains dan Teknologi, Universitas Muhammadiyah Sidoarjo,
JL. Mojopahit No. 666B, Sidoarjo, Jawa Timur, Indonesia

Abstract. *This study aims to implement ReactJS in the development of the Internal Asset Management System application interface in the Aptika Field of the Malang City Communication and Information Service. The development of this application aims to improve efficiency in managing asset data which was previously done manually, by accelerating the process of searching and managing data. The Agile Development approach is used in application development to provide flexibility in adjusting user needs during the development process. The main components of the application include a login system, dashboard, asset management, and user management features. The technologies used in the development of this application include ReactJS, JavaScript ES6+, HTML5, CSS3, and API integration using Axios. The results of the implementation show that the application is able to facilitate asset management faster, more accurately, and more easily accessible to admins and operators. In addition, the use of modular components in ReactJS facilitates maintenance and development of features in the future. Evaluation of application performance using Google Lighthouse and React Developer Tools shows a significant increase in page loading speed and data search.*

Keywords - ReactJS, Agile Development, Asset Management System, Communication and Information Service, web-based applications.

Abstrak. Penelitian ini bertujuan untuk mengimplementasikan ReactJS dalam pengembangan antarmuka aplikasi Sistem Manajemen Aset Internal di Bidang Aptika Diskominfo Kota Malang. Pengembangan aplikasi ini bertujuan untuk meningkatkan efisiensi dalam pengelolaan data aset yang sebelumnya dilakukan secara manual, dengan mempercepat proses pencarian dan pengelolaan data. Pendekatan Agile Development digunakan dalam pengembangan aplikasi untuk memberikan fleksibilitas dalam penyesuaian kebutuhan pengguna selama proses pengembangan. Komponen utama aplikasi meliputi sistem login, dashboard, manajemen aset, dan fitur manajemen pengguna. Teknologi yang digunakan dalam pengembangan aplikasi ini meliputi ReactJS, JavaScript ES6+, HTML5, CSS3, dan integrasi API menggunakan Axios. Hasil dari implementasi menunjukkan bahwa aplikasi mampu memfasilitasi pengelolaan aset dengan lebih cepat, akurat, dan mudah diakses oleh admin dan operator. Selain itu, penggunaan komponen modular dalam ReactJS memudahkan pemeliharaan dan pengembangan fitur di masa depan. Evaluasi kinerja aplikasi menggunakan Google Lighthouse dan React Developer Tools menunjukkan peningkatan signifikan dalam kecepatan pemuatan halaman dan pencarian data.

Kata Kunci - ReactJS, Agile Development, Sistem Manajemen Aset, Diskominfo, aplikasi berbasis web.

I. PENDAHULUAN

Dinas Komunikasi dan Informatika (Diskominfo) Kota Malang merupakan instansi pemerintah yang bertanggung jawab dalam pengelolaan informasi dan teknologi di lingkungan Pemerintah Kota Malang. Diskominfo berperan penting dalam menyediakan beragam layanan berbasis digital kepada masyarakat, mulai dari sistem informasi pemerintahan, layanan publik, hingga aplikasi digital yang mendukung keterbukaan informasi. [1] Di era transformasi digital yang berkembang pesat, kebutuhan akan sistem dan aplikasi yang responsif, mudah digunakan, dan inovatif semakin meningkat.[2] Untuk memenuhi kebutuhan ini, pengembangan aplikasi berbasis web dengan antarmuka pengguna (UI) yang menarik dan fungsional menjadi semakin penting.

Sub-Substansi Prasarana dan Sarana Teknologi Informasi (Aptika) didirikan oleh Direktur Komunikasi dan Pelayanan Informasi yang bertanggung jawab atas pengelolaan, pelaksanaan dan pengembangan portal web kota.[3] Sub-substansi sarana dan prasarana ini belum memiliki penerapan yang sesuai untuk Penyimpanan dan pengelolaan data. Hal ini membuat Sub-Substansi Prasarana dan Sarana Teknologi Informasi (Aptika) sulit untuk menemukan data lama. Ini berarti Akan menghabiskan lebih banyak waktu mencari data secara manual, sehingga mengurangi efisiensi. Dalam pengembangan aplikasi sistem manajemen aset internal Bidang Aptika Diskominfo Kota Malang, tantangan yang dihadapi adalah bagaimana menciptakan antarmuka pengguna yang responsif, efisien, dan mudah digunakan oleh staf dan pegawai dalam pengelolaan aset. Dengan aplikasi berbasis web ini, diharapkan proses penyimpanan dan pencarian data aset menjadi lebih cepat dan akurat dibandingkan metode manual sebelumnya. Selain itu, integrasi teknologi terkini seperti React JS diharapkan dapat meningkatkan kualitas pelayanan, baik dari segi kemudahan akses maupun kecepatan kinerja sistem.

Front End Developer membuat antarmuka pengguna grafis untuk situs web dengan menggunakan HTML, CSS, dan JavaScript. Ini memungkinkan pengguna melihat dan berinteraksi dengan situs web. Salah satu tanggung jawab front end developer ini adalah mengawasi konten yang dilihat pengguna di browser mereka. Mereka merancang aplikasi, menganalisis kode, dan men-debug sisi klien. Ini berarti mereka bertanggung jawab atas tampilan, nuansa, dan desain situs web atau aplikasi web. Seorang front end developer harus memastikan desainer memiliki tampilan dan fitur website yang diinginkan. Sederhanya, front end developer adalah pekerjaan pemrograman yang mengelola dan mengembangkan tampilan untuk aplikasi atau website[4].

Menurut kumar & sigh (2016) yang dikutip dalam Telkom university (2023) ReactJS adalah library JavaScript yang populer yang dikembangkan oleh Facebook yang digunakan dalam proses pengembangan aplikasi mobile dan web untuk membantu membuat komponen antarmuka yang interaktif, stateful, dan mudah digunakan ulang. ReactJS sangat cocok untuk rendering antarmuka yang rumit dan cepat[5].

Penerapan React JS dalam aplikasi ini menjadi penting karena sifatnya yang modular, di mana setiap elemen antarmuka dapat dipisahkan menjadi komponen yang berdiri sendiri.[6] Dengan pendekatan ini, proses pengembangan dan pemeliharaan sistem menjadi lebih mudah dan terstruktur. Penggunaan komponen yang dapat digunakan kembali (*reusable components*) juga mempercepat pengembangan antarmuka pengguna yang kompleks.[7]

Tujuan dari penelitian ini adalah untuk mengimplementasikan React JS dalam pengembangan front-end aplikasi sistem manajemen aset internal Bidang Aptika Diskominfo Kota Malang. Diharapkan dengan penerapan teknologi ini, dapat dihasilkan antarmuka yang dinamis dan ramah pengguna, sehingga mampu meningkatkan efisiensi kerja dan pengelolaan data aset di bidang Aptika.

II. METODE

1. Pendekatan Pengembangan Sistem

Pendekatan yang digunakan dalam pengembangan sistem ini adalah metode *Agile Development*. Agile dipilih karena fleksibilitasnya dalam penyesuaian terhadap kebutuhan pengguna selama proses pengembangan. Metode ini memungkinkan perubahan dan pembaruan komponen antarmuka secara berkelanjutan sesuai dengan feedback dari pengguna akhir. Pengembangan dilakukan dalam beberapa sprint, dengan setiap sprint berfokus pada pengembangan fitur dan perbaikan tertentu. [8]

2. Teknologi yang Digunakan

- React JS: Digunakan untuk pengembangan antarmuka pengguna. React JS dipilih karena kemampuannya untuk merender komponen secara dinamis dan mendukung pengelolaan state yang kompleks.[9] Fitur *Virtual DOM* dari React JS memungkinkan pembaruan antarmuka terjadi lebih cepat tanpa perlu memuat ulang seluruh halaman.
- JavaScript (ES6+): Digunakan sebagai bahasa pemrograman utama dalam pengembangan logika dan fungsionalitas aplikasi.[10]
- HTML5 dan CSS3: Untuk merancang struktur dasar dan styling dari antarmuka aplikasi. CSS3 digunakan bersamaan dengan *CSS-in-JS* atau *styled-components* untuk mendukung modularitas styling dalam komponen React.
- Node.js dan NPM (Node Package Manager): Digunakan untuk mengelola paket dan modul JavaScript yang dibutuhkan dalam proyek, termasuk React dan pustaka pendukung lainnya.
- Axios: Digunakan untuk melakukan *HTTP request* ke server dalam proses pengambilan dan pengelolaan data dari backend. [11]

3. Proses Pengembangan

- Analisis Kebutuhan Analisis kebutuhan dilakukan dengan melibatkan tim Aptika Diskominfo Kota Malang untuk memahami kebutuhan utama terkait pengelolaan aset. Kebutuhan tersebut meliputi kemudahan akses

- data, pencarian aset yang cepat, dan tampilan antarmuka yang intuitif dan mudah dipahami oleh pengguna. [12]
- b. Desain Antarmuka Pengguna Desain antarmuka pengguna dilakukan dengan menggunakan pendekatan *User-Centered Design (UCD)*, di mana pengguna akhir menjadi fokus utama dalam perancangan. Wireframe dan prototipe awal dibuat menggunakan *Figma*, yang kemudian divalidasi oleh pengguna sebelum masuk ke tahap pengembangan.
 - c. Implementasi React JS Implementasi React JS dilakukan dengan membuat komponen-komponen modular untuk setiap bagian antarmuka, seperti *dashboard*, *form input*, tabel data aset, dan sistem pencarian. Komponen-komponen ini dibangun agar dapat digunakan kembali, sehingga dapat mempercepat pengembangan pada fitur-fitur lain yang serupa.
 - d. Integrasi API Axios digunakan untuk mengintegrasikan frontend dengan backend. Data aset disimpan dan diambil dari server melalui API yang telah disediakan oleh tim backend. Setiap perubahan yang terjadi pada data aset akan segera diperbarui pada antarmuka tanpa perlu melakukan reload halaman, berkat penggunaan React JS dan konsep *state management*.
 - e. Pengujian dan Uji Coba Pengujian dilakukan secara berkala selama proses pengembangan menggunakan metode *unit testing* untuk menguji setiap komponen secara individu. Selain itu, dilakukan pula pengujian *end-to-end* untuk memastikan bahwa seluruh sistem berjalan sesuai harapan setelah semua komponen terintegrasi.
4. Evaluasi Kinerja Kinerja aplikasi dievaluasi berdasarkan waktu muat halaman, kecepatan dalam melakukan pencarian data aset, serta feedback dari pengguna terkait pengalaman mereka menggunakan aplikasi. Tools seperti *Google Lighthouse* dan *React Developer Tools* digunakan untuk memantau performa dan optimasi komponen yang mempengaruhi kecepatan aplikasi.[13]

III. HASIL DAN PEMBAHASAN

Di bab ini akan menjelaskan langkah yang sudah di jelaskan pada bab sebelumnya secara rinci.

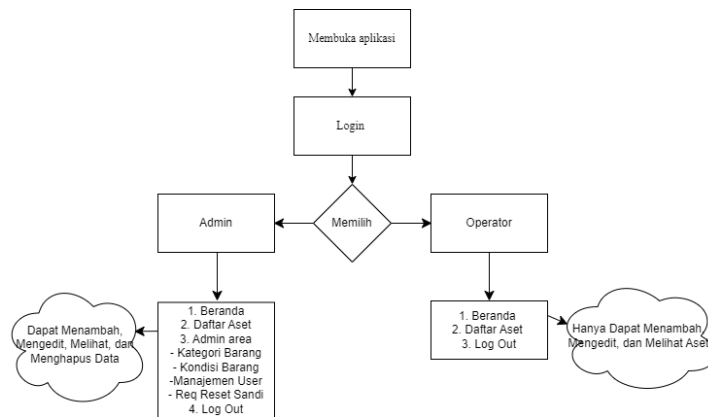
A. Analisis

Backlog adalah daftar yang berisi semua pekerjaan yang perlu dilakukan dalam sebuah proyek. Di dalam backlog berisi Daftar semua fitur, perbaikan, dan kebutuhan yang diinginkan untuk produk akhir. Dikelola dan diprioritaskan oleh Product Owner.

Fitur yang ada dapat dilihat pada tabel berikut ini :

Tabel 1. Fitur Backlog

No	Nama Backlog	Keterangan
1.	Login	- Form login, dengan dua kolom input: satu untuk <i>username</i> dan satu untuk <i>password</i> . - Terdapat pilihan "Lupa Password?" untuk pemulihan kata sandi.
2.	Beranda	- Di bagian bawah, terdapat tombol Login untuk masuk ke web - Di tampilan beranda terdapat bar chart dan pie cart. - Bar chart untuk melihat siapa saja user yang online. - Pie chart untuk mengetahui keseruhan data.
3.	Daftar Aset	- Di bagian bawah terdapat tabel jumlah aset - Ditampilkan daftar aset terdapat kolom berita acara, tambah daftar aset, dan tabel data aset.
4.	(Admin Area) Kategori barang	- Terdapat juga tombol upload, edit, view, delete - Di tampilan kategori barang terdapat kolom tambah kategori barang dan tabel kategori barang.
5.	(Admin Area) Kondisi Barang	- Terdapat juga tombol edit dan delete. - Di tampilan kondisi barang terdapat tombol daftar kondisi barang dan tabel
6.	(Admin Area) Manajemen user	- Terdapat juga tombol hapus - Di tampilan ini terapat siapa saja user yang menggunakan web ini.
7.	(Admin Area) Req reset sandi	- Terdapat tombol tambah untuk menambah user baru. - Terdapat juga tombol edit dan hapus.
8.	Log out	- Menampilkan tabel permintaan reset sandi - Terdapat tomboldelete di dalam tabel. - Tombol untuk keluar dari web.



Gambar 1. Bagan Alur aplikasi

Di dalam web ini terdapat dua user yang dapat mengaksesnya. Pada laman admin dan operator terdapat perbedaan. Di laman admin menampilkan spesial menu admin yaitu admin area yang di dalamnya terdapat kategori barang, kondisi barang, manajemen user dan req reset sandi. Sedangkan di laman operator hanya terdapat dashboard, daftar aset, dan log out.

B. Implementasi

Pada tahap implementasi Aplikasi Sistem Manajemen Aset Internal Bidang Aptika Diskominfo Kota Malang ini berhasil mewujudkan sesuai rencana dengan menampilkan fitur fitur yang memudahkan substansi sarana dan prasarana untuk mengelola data aset secara mudah dan efisien.

Tampilan Web

1. Login



Gambar 1. Laman login

```

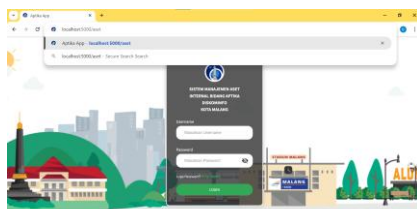
const PrivateRoute = ({ children, requiredLevel }) => {
  const { isLoggedIn } = useAuth();
  const user = JSON.parse(localStorage.getItem('currentUser'));

  if (!isLoggedIn || !user) {
    return <Navigate to="/login" />;
  }

  if (requiredLevel && user.level !== requiredLevel) {
    return <Navigate to="/unauthorized" />; // Halaman unauthorized harus disediakan
  }
}
    
```

Gambar 2. Coding login

Pada **Gambar 1**. Merupakan tampilan halaman login admin maupun operator. Pada tampilan ini terdapat dua kolom untuk mengisi username dan password. Tampilan halaman login ini, tidak ada pembeda untuk login operator dan login admin. Namun, pembeda tersebut terletak pada coding yang ada pada **Gambar 2**. If (requiredLevel && user.level !== requiredLevel). Dimana jika user admin yang login maka akan masuk ke beranda admin, jika level user operator maka akan masuk ke halaman berandaoperator. Jika berhasil masuk maka akan diarahkan ke halaman beranda sesuai lever user tersebut.



Gambar 3. Jika endpoint di ganti dari url

```

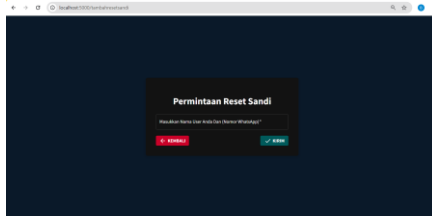
useEffect(() => {
  const savedLoginStatus = localStorage.getItem("isLoggedIn");

  // Jika pengguna tidak login, jangan redirect ke halaman login jika mereka mengakses halaman tertentu
  if (savedLoginStatus !== "true") {
    if (window.location.pathname !== "/tambahresetsandi") {
      navigate("/login");
    }
  } else {
    setIsLoggedIn(true);
  }
}
    
```

Gambar 4. Coding keamanan

Pada **Gambar 3**. Merupakan tampilan login saat user belum mengisi semua kolom dan user secara langsung menembak dari endpoint di url maka user tidak akan bisa masuk dan tetap berada di laman login. karena keamanan web ini menggunakan privat route yang di buktikan pada **Gambar 4**. `const savedLoginStatus = localStorage.getItem("isLoggedIn");` Dimana jika user belum login maka user tidak bisa memasuki halaman halaman yang perlu akses login.

2. Permintaan reset sandi



Gambar 5. Laman req reset sandi

```
try {
  const response = await fetch('http://localhost:4000/requestandi', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(data),
  });
}

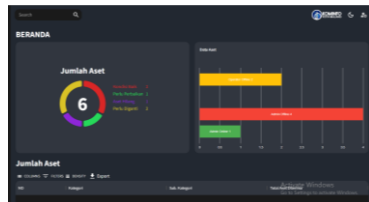
if (response.ok) {
  const result = await response.json();
  console.log('Respon server:', result);
  alert('Data berhasil ditambahkan!');
  setRequestSandi('');
}
```

Gambar 6. Codingan laman req reset sandi

Pada **Gambar 5**. Menunjukkan laman permintaan reset sandi yang terletak di bawah kolom password. Yang mana ketika di klik akan diarahkan ke tampilan seperti gambar diatas dan di buktikan dari codingan di **Gambar 6**. `SetRequestSandi('')`. Pada gambar tersebut digunakan ketika user lupa sandi kemudian user dapat mengisi kolom reset sandi yang mana permintaan tersebut akan muncul di bagian fitur req reset sandi di admin area.

3. Tampilan web beranda Admin dan Operator

Tampilan ini merupakan Tampilan yang ada akan muncul di admin maupun operator.



Gambar 7. Tampilan laman dashboard

```
// Fetch kondisi barang
useEffect(() => {
  const fetchKondisi = async () => {
    try {
      const response = await fetch("http://localhost:4000/kondisi");
      const data = await response.json();
      setKondisiBarang(data);
    } catch (error) {
      console.error("Error fetching kondisi barang:", error);
    }
  };
});
```

Gambar 8. Codingan pie chart

```
// Fetch data from backend
useEffect(() => {
  fetch('http://localhost:4000/users/status')
    .then(response => response.json())
    .then(data => {
      if (Array.isArray(data)) {
        setData(data);
      } else {
        console.error('Data yang diterima bukan array:', data);
      }
    })
    .catch(error => console.error('Error fetching data:', error));
}, []);
```

Gambar 9. Codingan bar chart

```
<DataGrid
  rows={rows}
  columns={columns}
  components={{ Toolbar: CustomToolbar }}
  rowHeight={75}
  componentsProps={{
    toolbar: { onExport: handleExport }
  }}
/>
```

Gambar 10. Codingan Jumlah tabel aset

Gambar 7. Merupakan tampilan ketika user berhasil login maka akan muncul tampilan beranda seperti gambar diatas yang menampilkan pie chart, bar chart dan tabel jumlah aset dengan default mode gelap. `const response = await fetch("http://localhost:4000/kondisi");` Merupakan codingan dari pie chart yang akan mengambil data dari backend kemudian di tampilkan pada pie chart seperti pada **Gambar 8**. Gunanya untuk menunjukkan kondisi jumlah aset secara keseluruhan. Selanjutnya `useEffect( () => { fetch( 'http://localhost:4000/users/status')` Merupakan codingan untuk mengambil data user online atau offline dari backend yang akan di tampilkan di bar chart seperti **Gambar 9**. Di bagian bawah kedua chart tersebut, terdapat table jumlah aset yang menunjukkan jumlah data aset per sub-kategori yang di buktikan pada codingan `rows={rows}` yang ada di **Gambar 10**. `{rows}` merupakan sebuah fungsi yang sudah dibuat dan di dalam fungsi itu terdapat grub data yang nantinya akan dipanggil di dalam codingan tersebut untuk ditampilkan di tabel. Gunanya menunjukkan kategori subkategori dan jumlah aset.

4. Nav Bar (Navigation Bar)



Gambar 11. Navbar

```
// State untuk menyimpan nilai input pencarian
const [searchQuery, setSearchQuery] = useState("");
const { setGlobalSearchQuery } = useContext(SearchContext); // Ambil setter dari SearchContext

// Fungsi untuk menangani pencarian secara otomatis saat pengguna mengetik
useEffect(() => {
  setGlobalSearchQuery(searchQuery); // Kirim query pencarian ke context global
}, [searchQuery, setGlobalSearchQuery]);

// Fungsi untuk menangani pencarian dengan tombol Enter
const handleKeyDown = (e) => {
  if (e.key === "Enter") {
    setGlobalSearchQuery(searchQuery);
  }
};
```

Gambar 12. Coding untuk search bar

```
/* ICONS + LOGO */
<Box display="flex" alignItems="center" className="logo">
  <Box
    component="img"
    src={theme.palette.mode === "dark" ? "/assets/logo-dark.png" : "/assets/logo-light.png"}
    alt="Logo"
    sx={{ height: "40px", marginRight: "15px" }}
  />
  <div>
    <h1>KOMINFO</h1>
  </div>
</Box>
```

Gambar 13. Coding untuk icon Diskominfo

```
</>
<IconButton onClick={colorMode.toggleColorMode}>
  {theme.palette.mode === "dark" ? (
    <DarkModeOutlinedIcon />
  ) : (
    <LightModeOutlinedIcon />
  )}
</IconButton>
```

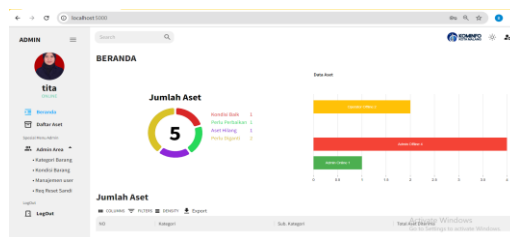
Gambar 14. Coding untuk icon gelap terang

```
<IconButton
  sx={{
    ml: 1,
    backgroundColor: "transparent",
    color: colors.grey(100),
    "&hover": {
      backgroundColor: theme.palette.mode === "dark" ? colors.grey(600) : colors.grey(600),
    },
    borderRadius: "50px",
  }}
  onClick={() => navigate("/SettingAkun")}
  >
  <ManageAccountsIcon />
</IconButton>
```

Gambar 15. Coding untuk icon setting akun

Gambar 11. Merupakan tampilan nav bar yang berfungsi sebagai elemen antarmuka pengguna pada halaman web yang menyediakan navigasi antar halaman atau bagian dalam suatu situs. Nav bar ini terletak dibagian atas halaman web di setiap halamannya. Di dalam navbar ini terdapat kolom search yang berguna untuk mempercepat pencarian data yang di buktikan pada **Gambar 12**. `setGlobalSearchQuery(searchQuery);` merupakan coding untuk melakukan pencarian data. Di sebelah kanan atas terdapat 3 item yaitu logo kominfo yang coding nya di buktikan pada **Gambar 13**. `src={theme.palette.mode === "dark" ? "/assets/logo-dark.png" : "/assets/logo-light.png"}`. ini adalah coding untuk menampilkan gambar logo kominfo. Untuk logo gelap terang di buktikan pada **Gambar 14** `{theme.palette.mode === "dark" ? ( <DarkModeOutlinedIcon /> ) : ( <LightModeOutlinedIcon /> )}` ini merupakan coding untuk mode gelap atau terang. Dan logo pengaturan akun dapat dilihat pada **Gambar 15**. `onClick={() => navigate("/SettingAkun")}` untuk mengarahkan ke halaman setting akun.

5. Mode Terang



Gambar 16. Tampilan mode terang



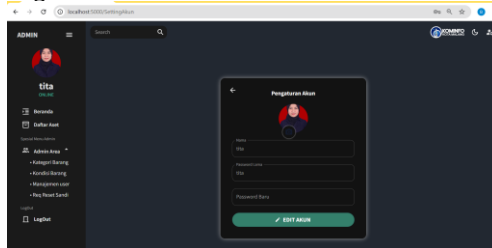
Gambar 17. Tampilan mode gelap

```
// main theme settings
export const themeSettings = (mode) => {
  const colors = Tokens(mode);
  return {
    palette: {
      mode: mode,
      ...(mode === "dark"
        ? // palette values for dark mode
        {
          primary: {
            main: colors.primary[500],
          },
          secondary: {
            main: colors.greenAccent[500],
          },
          neutral: {
            dark: colors.grey[700],
            main: colors.grey[500],
            light: colors.grey[100],
          },
          background: {
            default: colors.grey[1000], // warna background dark
          },
        }
        : // setup light mode
        {
          primary: {
            main: colors.primary[100],
          },
        }
      ),
    },
  };
};
```

Gambar 18. Coding mode terang dan gelap

Pada **Gambar 16**. Menampilkan tampilan mode terang. Sedangkan untuk **Gambar 17** merupakan tampilan mode gelap. Hal ini di buktikan pada **Gambar 18**. Palette: { mode: mode,...(mode === "dark" ? coding ini memiliki fungsi jika user mengklik button mode dark and light maka akan mengganti ke mode dark atau light.

6. Pengaturan akun



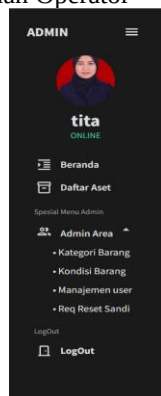
Gambar 19. Laman pengaturan akun

```
fetch('http://localhost:4000/users/${user.id}', {
  method: 'PUT',
  body: formData, // Send FormData
})
.then((response) => response.json())
.then(() => {
  setIsEditing(false);
  alert('Perubahan telah disimpan!');
  navigate(-1); // Kembali ke halaman sebelumnya
})
.catch((error) => console.error('Gagal memperbarui akun:', error));
```

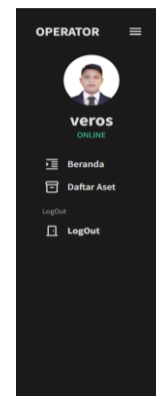
Gambar 20. Coding pengaturan akun

Pada **Gambar 19**. Ketika user menekan icon yang ada di pojok kanan atas, maka akan mengarah ke laman pengaturan akun seperti gambar diatas. Pada halaman ini terdapat kolom nama, kolom password lama, kolom password baru, dan foto. Untuk mengedit hal ini di buktikan pada **Gambar 20**. fetch('http://localhost:4000/users/\${user.id}', {method: 'PUT',body: formData, Coding ini di gunakan untuk mengedit password dan foto.

7. Side bar Admin dan Operator



Gambar 21. Tampilan side bar admin



Gambar 22. Tampilan side bar operator

```

{user.level === 'admin' && ( // Conditional rendering for admin items
  <Typography
    variant="h6"
    color={colors.grey[300]}
    sx={{ m: "15px 0 5px 20px" }}
  />
  <Special Menu Admin
  />
  <Item
    title="Admin Area"
    icon={PeopleOutlinedIcon}
    selected={selected}
    setSelected={setSelected}
    hasDropdown={true}
    dropdownItems={
      [
        { title: "• Kategori Barang", to: "/kategori" },
        { title: "• Kondisi Barang", to: "/kondisi" },
        { title: "• Manajemen user", to: "/users" },
        { title: "• Req Reset Sandi", to: "/resetsandi" },
      ]
    }
    userLevel={user.level}
  />
)}
)
)

```

Gambar 23. Cooding menu sidebar

Pada **Gambar 21**. Merupakan tampilan sidebar admin yang berfungsi sebagai elemen antarmuka pengguna yang menampilkan menu atau informasi tambahan di samping halaman utama situs web. Biasanya digunakan untuk navigasi, menampilkan tautan penting, atau menampilkan informasi tanpa mengganggu konten utama. Terdapat dua tampilan side bar tergantung usernya hal ini dapat di buktikan oleh **Gambar 22**. Pada **Gambar 22**. Merupakan tampilan sidebar operator. Pada side bar operator dan side bar Admin terdapat perbedaan pada bagian admin area hal ini di buktikan pada **Gambar 23**. {user.level === 'admin' && ()}. Cooding ini untuk membedakan tampilan side bar antara user admin dan operator.

8. Daftar Aset Admin dan Operatorr

Gambar 24. Tabel Daftar Aset Admin

Gambar 25. Tabel Daftar Aset Operator

```

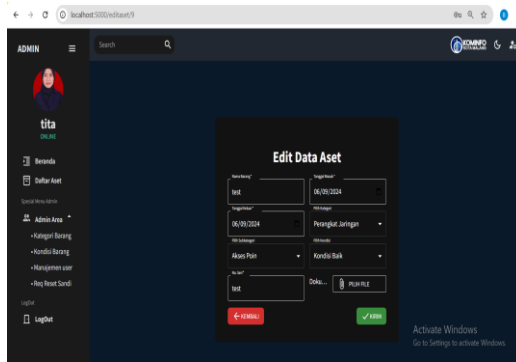
<DataGrid
  rows={filteredAset}
  columns={columns}
  components={{ Toolbar: CustomToolbar }} // Menggunakan CustomToolbar
  getRowId={row => row.id}
  rowHeight={75}
  componentsProps={{
    toolbar: { onExport: handleExport } // Mengirimkan handleExport ke CustomToolbar
  }}
/>
</Box>

```

Gambar 26. Cooding Tabel Aset

Pada **Gambar 24**. Menunjukkan halaman tabel data aset yang telah di inputkan oleh user. Karena user ini terdeteksi berlevel admin, maka tampilan tabel terdapat tombol upload, edit, view, dan delete. Sedangkan pada **Gambar 25**. Merupakan tampilan tabel data aset yang berlevel operator, maka tombol delete tidak ada. Untuk **Gambar 26**. Menunjukkan cooding dari kedua tabel yang ada di halaman daftar aset. rows={filteredAset} merupakan cooding fungsi yang sudah di buat untuk menampilkan dan memfilter data aset sesuai permintaan klien.

9. Tampilan Edit data aset



Gambar 27. Laman edit data aset

```
useEffect(() => {
  const fetchData = async () => {
    try {
      const kategoriResponse = await fetch('http://localhost:4000/kategori');
      const kategoriData = await kategoriResponse.json();
      setKategoriList(kategoriData);

      const subKategoriResponse = await fetch('http://localhost:4000/subkategori');
      const subKategoriData = await subKategoriResponse.json();
      setSubKategoriList(subKategoriData);

      const kondisiResponse = await fetch('http://localhost:4000/kondisi');
      const kondisiData = await kondisiResponse.json();
      setKondisiList(kondisiData);

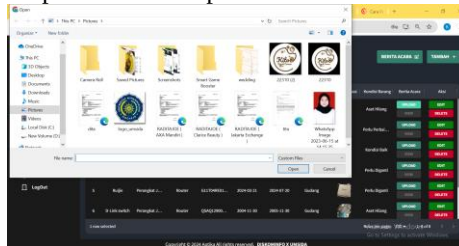
      const asetResponse = await fetch('http://localhost:4000/aset/${id}');
      const asetData = await asetResponse.json();

      if (asetData) {
        setNamaAset(asetData.namaAset);
        setKategori(asetData.kategori);
        setSubKategori(asetData.subkategori);
        setKondisi(asetData.kondisi);
        setTanggalMasuk(asetData.tanggalMasuk);
        setTanggalKeluar(asetData.tanggalKeluar);
        setLocation(asetData.lokasi);
        setKondisiAset(asetData.kondisiAset);
        setDokumentasi(asetData.dokumentasi);
      }
    } catch (error) {
      console.error('Error fetching data:', error);
    }
  }
  fetchData();
}, [id]);
```

Gambar 28. Cooding edit data aset

Pada **Gambar 27**. Merupakan tampilan untuk mengedit data aset. User dapat mengedit semua yang ada di kolom ini. Jika user sudah selesai mengedit maka selanjutnya user bisa menekan tombol kirim untuk menyimpan ke database. Untuk **Gambar 28**. Merupakan coding dari laman edit aset. Semua codingan di atas berfungsi untuk mengambil data aset kemudian data aset tersebut dapat di edit.

10. Tampilan Tombol upload



Gambar 29. Laman Upload

```
const handleUploadBeritaAcara = async (id) => {
  const input = document.createElement("input");
  input.type = "file";
  input.accept = ".jpg, .jpeg, .png"; // Menerima file PDF atau gambar
  input.onChange = async () => {
    const file = input.files[0];
    const formData = new FormData();
    formData.append("beritaacara", file);
  }
}
```

Gambar 30. Cooding Upload

Gambar 29. Merupakan tampilan jika tombol upload di tekan maka akan muncul tampilan file. Pada tombol upload, di gunakan untuk mengupload surat berita acara yang sudah di print berbentuk jpeg hal ini dapat di buktikan pada **Gambar 30**. `const input = document.createElement("input");` coding ini berfungsi untuk memilih gambar.

11. Tampilan Tombol View



Gambar 31. Laman View

```
const handleViewBeritaAcara = (fileUrl) => {
  if (fileUrl.endsWith('.jpg')) {
    // Open the jpg in a new tab
    window.open(fileUrl, '_blank');
  } else {
    // For images, display them in the modal
    setSelectedImage(fileUrl);
    setSelectedBeritaAcara(null);
    setOpen(true);
  }
}
```

Gambar 32. Cooding View

Pada **Gambar 31**. Merupakan tampilan tombol view. Tombol view di gunakan untuk melihat hasil berita acara yang sudah di upload oleh user hal ini dibuktikan pada **Gambar 32**. `setSelectedImage(fileUrl);` untuk menampilkan gambar.

12. Tampilan Tombol Delete



Gambar 33. Tombol delete

```

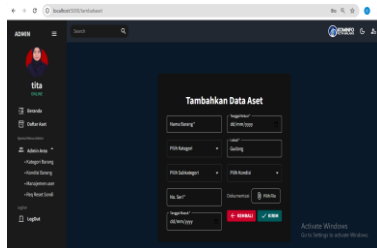
</Button>
{user && user.level === "admin" && ( // Hanya tampilkan tombol delete jika level user adalah admin
<Button
  variant="contained"
  size="small"
  sx={{
    width: "90px",
    fontSize: "14px",
    padding: "2px 11px",
    fontWeight: "bold",
    backgroundColor: theme.palette.mode === 'dark' ? '#ca0128' : '#ff0000',
    '&hover': {
      backgroundColor: theme.palette.mode === 'dark' ? '#ff0000' : '#ca0128',
    },
    color: theme.palette.mode === 'dark' ? '#ffffff' : '#000000',
  }}
  onClick={() => handleDelete(params.row.id)}
>
  Delete
</Button>
)}

```

Gambar 34. Coding delete

Pada **Gambar 33**. Merupakan tombol delete yang ada di tabel aset admin saja. Tombol ini menjadi pembeda tampilan yang ada pada tabel aset admin dan tabel aset operator. Ini dapat dibuktikan pada **Gambar 34**. Apabila lever user tersebut admin tombol tersebut ada, namun jika level user operator maka tombol itu tidak ada. {user && user.level === "admin" && ()} jika level user admin maka tombol delete ini muncul. Jika level operator tombol ini tidak muncul.

13. Tambah Daftar aset



Gambar 35. Laman tambah aset

```

const handleSubmit = async (e) => {
  e.preventDefault();

  if (!namaBarang || !kategori || !subKategori || !noSeri || !tanggalMasuk || !tanggalKeluar || !lokasi || !kondisiBarang || !dokumentasiBarang) {
    alert('Data gagal ditambahkan. Semua field harus diisi!');
    return;
  }

  if (!dokumentasi) {
    alert('Please fill out the documentation field. ');
    return;
  }

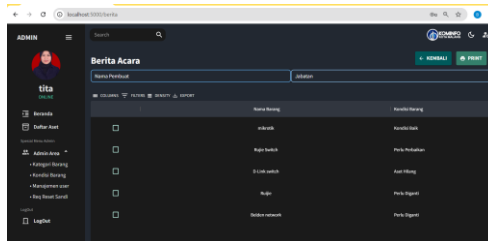
  if (new Date(tanggalKeluar) < new Date(tanggalMasuk)) {
    alert('Tanggal keluar tidak bisa lebih awal dari tanggal masuk');
    return;
  }
}

```

Gambar 36. Coding tambah aset

Pada **Gambar 35**. Menampilkan laman tambah aset. Di dalam laman ini terdapat kolom nama barang, kolom pilih kategori, kolom pilih subkategori, kolom no.seri, kolom tanggal masuk, kolom tanggal keluar, kolom lokasi, kolom pilih kondisi, dan kolom dokumentasi. User harus mengisi semua kolom, jika salah satu kolom tidak terisi maka data tidak bisa ditambahkan. Bisa di lihat pada **Gambar 36**. If (!namaBarang || !kategori || !subKategori || !noSeri || !tanggalMasuk || !tanggalKeluar || !lokasi || !kondisiBarang) alert('Data gagal ditambahkan.Semua field harus diisi!'); return; di gunakan untuk menambah data.

14. Berita Acara



Gambar 37. Laman berita acara

```

const BeritaAcara = () => {
  const theme = useTheme();
  const colors = tokens(theme.palette.mode);
  const [barang, setBarang] = useState(); // State untuk barang
  const [selectedItems, setSelectedItems] = useState(); // State untuk item yang di-checklist
  const navigate = useNavigate();

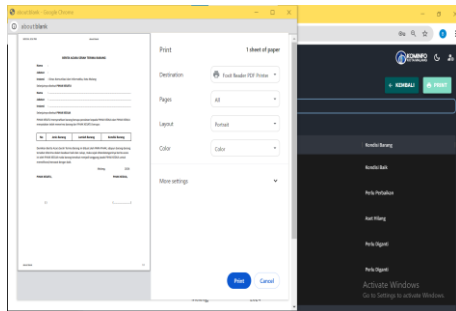
  const [formData, setFormData] = useState({
    namaPembuat: '',
    jabatan: ''
  });
}

```

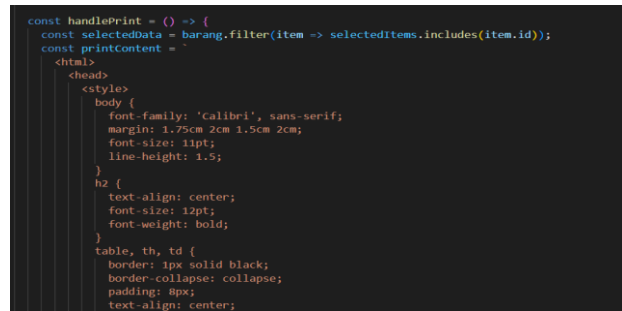
Gambar 38. Coding berita acara

Gambar 37. Menampilkan jika user menekan tombol berita acara yang terdapat pada laman beranda yang mana akan diarahkan ke laman ini. Laman berita acara ini untuk memilih barang apa saja yang ingin di masukkan di berita acara untuk di jadikan laporan dengan menceklis barang yang ingin di print dapat buktikan pada **Gambar 38**. const [selectedItems, setSelectedItems] = useState([]); coding ini di gunakan untuk menceklis barang.

15. Tampilan tombol print



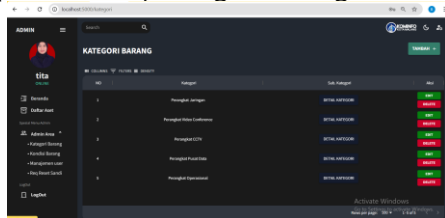
Gambar 39. Tampilan tombol print



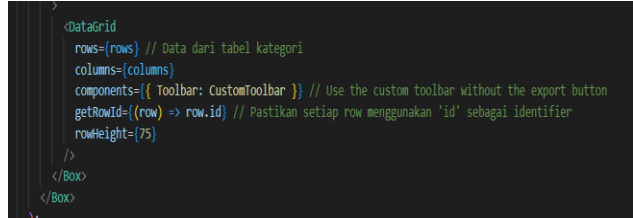
Gambar 40. Coding Fungsi tombol print

Pada **Gambar 39**. Merupakan tampilan untuk print berita acara. Setelah memilih data apa saja yang ingin dimasukkan ke surat, user dapat langsung menekan tombol print dan akan muncul templat berita acara yang secara otomatis data yang sudah dipilih akan tertera di surat. Kemudian user dapat mengeprint berita acara tersebut dapat dilihat pada **Gambar 40**. `const selectedData = barang.filter(item => selectedItems.includes(item.id));` coding ini berfungsi untuk menampilkan doc yang akan di print.

16. (Admin Area) Kategori barang



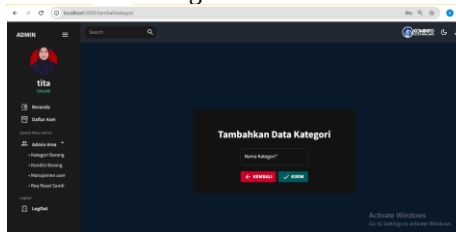
Gambar 41. Laman kategori barang



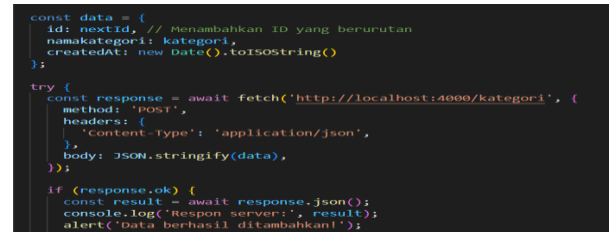
Gambar 42. Coding kategori barang

Pada **Gambar 41**. Menampilkan tabel kategori barang. Pada laman kategori barang menampilkan tabel kategori barang dan sub kategori barang. Di sebelah kanan atas tabel terdapat tombol tambah kategori barang dapat dilihat pada **Gambar 42**. `Rows={rows}`. `{Rows}` adalah sebuah fungsi untuk mengambil data dari tabel kategori dimana nantinya akan di munculkan pada tabel kategori barang.

17. Tambah Data kategori



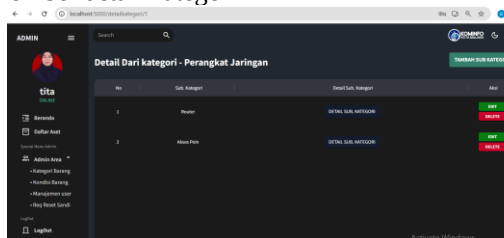
Gambar 43. Laman tambah data kategori



Gambar 44. Coding tambah data kategori

Pada **Gambar 43**. Merupakan tampilan tambah data kategori. Pada laman ini terdapat hanya satu kolom, tombol kembali dan tombol kirim saja. Pada **Gambar 44**. `const response = await fetch('http://localhost:4000/kategori', {method: 'POST',` Merupakan coding untuk menambah kategori.

18. Tombol detail kategori



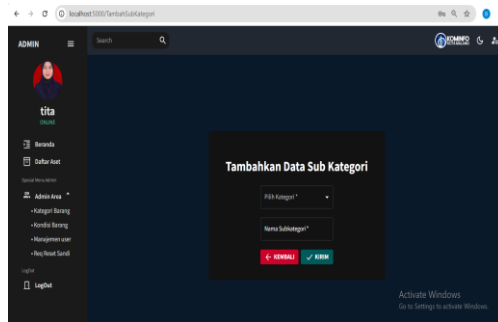
Gambar 45. Tampilan detail kategori barang



Gambar 46. Coding detail kategori barang

Pada **Gambar 45**. Merupakan tampilan jika user menekan tombol detail kategori dari suatu barang yang ada di tabel kategori maka akan muncul tampilan seperti di atas. Pada laman ini di bagian pojok kanan atas terdapat tombol tambah sub kategori. Untuk coding nya dapat dilihat pada **Gambar 46**. `const filteredData = subkategoriData.filter(sub => sub.kategoriId === parseInt(id));` Berfungsi untuk menampilkan sub kategori dari perangkat kaegori yang dipilih.

19. Tambah Data sub kategori



Gambar 47. Laman tambah data sub kategori

```
const handleSubmit = async (e) => {
  e.preventDefault();

  if (!kategoriId || !subkategori) {
    alert('Data gagal ditambahkan, Semua field harus diisi!');
    return;
  }

  const data = {
    namaSubkategori: subkategori,
    kategoriId: parseInt(kategoriId),
    createdAt: new Date().toISOString()
  };

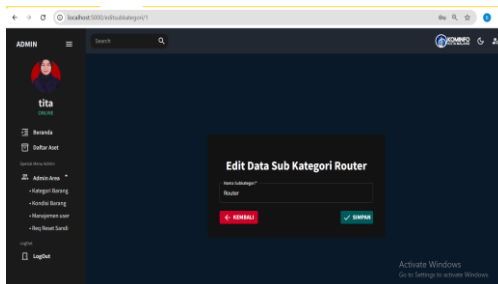
  try {
    const response = await fetch('http://localhost:4000/subkategori', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(data),
    });

    if (response.ok) {
      const result = await response.json();
      console.log('Respon server:', result);
      alert('Data berhasil ditambahkan!');
    }
  }
}
```

Gambar 48. Coding tambah data sub kategori

Pada **Gambar 47**. Merupakan tampilan tambah data sub kategori yang terdapat dua kolom yang harus diisi user, tombol kembali dan tombol kirim. Tambah data sub kategori ini terdapat di atas tabel detail dari kategori barang yang dipilih user. Untuk **Gambar 48**. `const response = await fetch('http://localhost:4000/subkategori', { method: 'POST',` berfungsi untuk menambah data sub kategori.

20. Edit data sub kategori



Gambar 49. Laman edit data sub kategori

```
// Handle submit update subkategori
const handleSubmit = async (e) => {
  e.preventDefault();

  const data = {
    namaSubkategori: subkategori,
    kategoriId: parseInt(kategoriId, 10),
  };

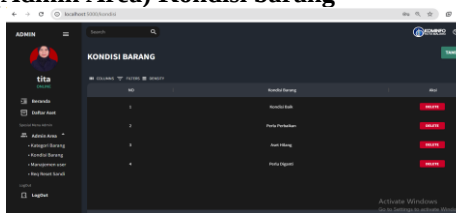
  try {
    const response = await fetch('http://localhost:4000/subkategori/${id}', {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(data),
    });

    if (response.ok) {
      alert('Data berhasil disimpan!');
      navigate('/kategori');
    } else {
      const result = await response.json();
      console.error('Gagal menyimpan data. Respon:', result);
      alert('Data gagal disimpan!');
    }
  }
}
```

Gambar 50. Coding edit data sub kategori

Gambar 49. Ini menampilkan kolom untuk mengedit data sub kategori dari suatu barang yang dipilih. Setelah user mengedit maka selanjutnya user harus menekan tombol simpan untuk menyimpan data di database hal ini dapat dibuktikan pada **Gambar 50**. `const response = await fetch('http://localhost:4000/subkategori/${id}', {method: 'PUT',` yang berfungsi untuk mengedit data sub kategori

21. (Admin Area) Kondisi barang



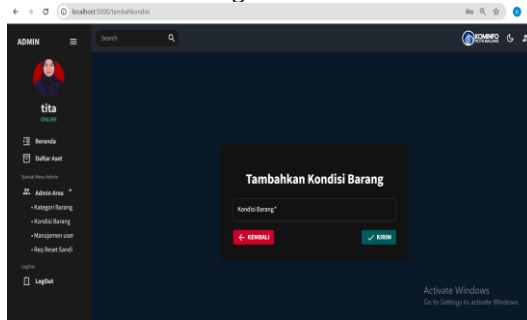
Gambar 51. Laman kondisi barang

```
<DataGrid
  rows={kondisi}
  columns={columns}
  components={{ Toolbar: CustomToolbar }}
  getRowId={row => row.id}
  rowHeight={75}
/>
```

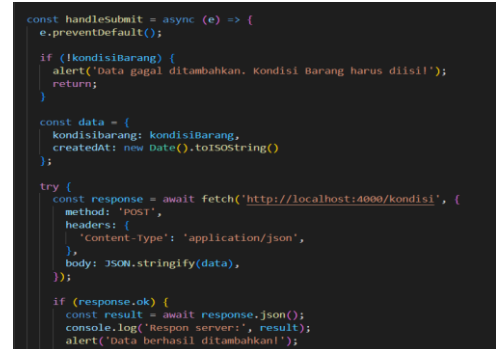
Gambar 52. Coding kondisi barang

Pada **Gambar 51**. Merupakan tampilan laman kondisi barang yang menampilkan tabel kondisi barang dan di atas tabel tersebut terdapat tombol tambah kondisi barang. Untuk **Gambar 52**. Merupakan coding tabel kondisi barang. `rows={kondisi}` berfungsi untuk menampilkan data kondisi barang.

22. Tambah kondisi barang



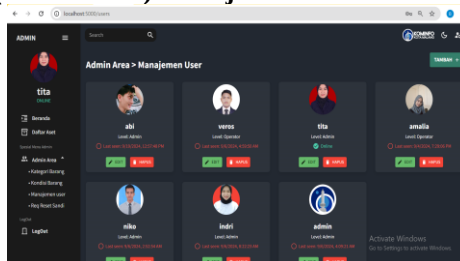
Gambar 53. Laman tambah kondisi barang



Gambar 54. Coding tambah kondisi barang

Pada **Gambar 53**. Merupakan laman untuk menambah kondisi barang. Di laman ini terdapat satu kolom. Setelah user mengisi kolom tersebut maka user harus menekan tombol kirim untuk menyimpannya ke database. Untuk coding nya dapat dilihat pada **Gambar 54**. `const response = await fetch('http://localhost:4000/kondisi', {method: 'POST',` berfungsi untuk menambahkan kondisi barang.

23. (Admin Area) Manajemen User



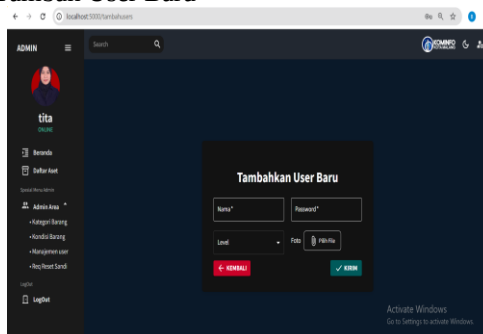
Gambar 55. Laman manajemen user



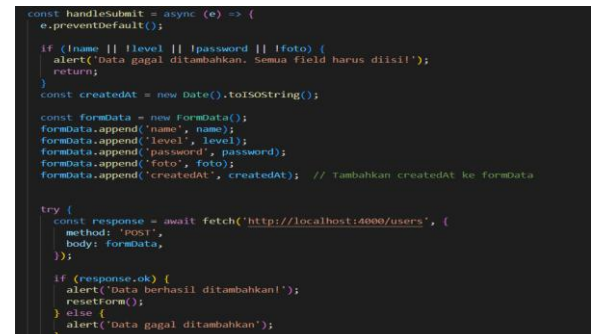
Gambar 56. Coding manajemen user

Pada **Gambar 55**. Menampilkan siapa saja user yang menggunakan web ini beserta foto, level, dan status user. Terdapat juga tombol edit dan hapus. Di sebelah kanan atas juga terdapat tombol tambah user untuk menambahkan user baru. Untuk **Gambar 56**. Merupakan coding untuk laman user. `{users.map( => )}` berfungsi untuk menampilkan semua user.

24. Tambah User Baru



Gambar 57. Laman tambah user baru

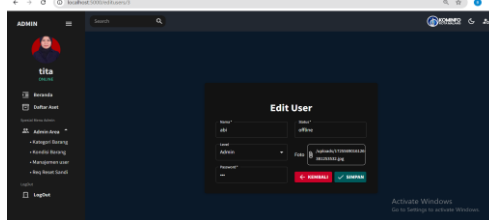


Gambar 58. Coding tambah user baru

Pada **Gambar 56**. Merupakan laman untuk menambahkan user baru. Pada laman ini terdapat empat kolom yaitu kolom nama, kolom level, kolom password, dan kolom untuk mengunggah foto. Ketika

user sudah mengisi semua kolom selanjutnya menekan tombol kirim untuk menyimpan data di database. Untuk coding laman ini dapat dilihat pada **Gambar 57**. `const response = await fetch('http://localhost:4000/users', {method: 'POST',` Berfungsi untuk menambahkan data user baru.

25. Edit User



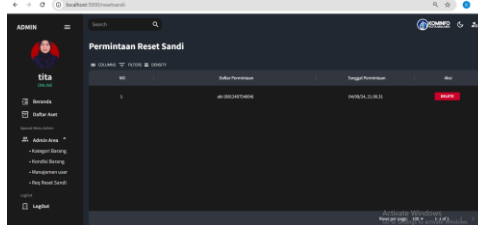
Gambar 58. Laman edit user

```
try {  
  const response = await fetch('http://localhost:4000/users/${id}', {  
    method: 'PUT', // Change to PUT for editing existing data  
    body: formData,  
  });  
}
```

Gambar 59. Coding edit user

Pada **Gambar 58**. Merupakan halaman yang digunakan admin untuk mengubah kata sandi operator yang meminta reset sandi hal ini dapat dibuktikan **Gambar 59**. `const response = await fetch('http://localhost:4000/user/${id}', {method: 'PUT',` Coding ini di gunakan untuk mengedit data user

26. (Admin Area) Req Reset Sandi



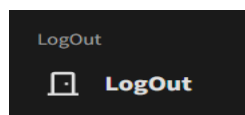
Gambar 60. Laman req reset sandi

```
<DataGrid  
  rows={requestresetsandi}  
  columns={columns}  
  components={{ Toolbar: CustomToolbar }}  
  getRowId={(row) => row.id} // Use 'id' as a unique identifier  
  rowHeight={75}  
/>  
</Box>  
</Box>
```

Gambar 61. Coding req reset sandi

Gambar 60. Merupakan halaman Req reset sandi ini menampilkan pemberitahuan jika ada user yang lupa password maka admin akan menginformasikan melalui nomor whatsapp yang tertera dan admin akan mengonfirmasi apakah user yang lupa sandi ingin mengganti passwordnya. Setelah dikonfirmasi oleh operator selanjutnya admin akan mengedit password di halaman manajemen user atau menghapus akun. Untuk **Gambar 61**. `Rows={requestresetsandi}` merupakan coding untuk tabel req reset sandi.

27. Log Out



Gambar 62. Logout

```
onClick={() => {  
  setSelected("Logout");  
  handleLogout(); // Panggil fungsi logout  
}}  
Logout
```

Gambar 63. Coding logout

Gambar 62. Merupakan halaman logout yang ada pada kedua laman admin maupun operator. Tombol ini bersifat one klik yang dibuktikan pada **Gambar 63**. `onClick={() => {setSelected("logout"); handleLogout();}}` fungsinya untuk keluar dan kemudian diarahkan ke halaman login.

IV. KESIMPULAN

Dengan menggunakan ReactJS dan pendekatan Agile Development, aplikasi Sistem Manajemen Aset Internal Bidang Aptika Diskominfo Kota Malang dapat lebih efisien dalam mengelola data aset. Aplikasi ini memiliki fitur yang memungkinkan pengelolaan aset menjadi lebih cepat, akurat, dan mudah diakses oleh admin dan operator. Penerapan ReactJS dengan komponen modular memungkinkan pengembangan yang lebih cepat dan terstruktur serta kemudahan pemeliharaan aplikasi di masa depan. Aplikasi ini dapat mengawasi perubahan data secara real-time tanpa perlu mengulang halaman karena menggunakan teknologi seperti integrasi API Axios.

Beberapa komponen utama yang dikembangkan termasuk sistem login untuk autentikasi pengguna, dashboard yang menampilkan tabel dan grafik data aset, dan fitur manajemen aset seperti menambah, mengedit, dan melihat data aset melalui antarmuka yang mudah digunakan. Selain itu, ada fitur manajemen user yang memungkinkan Anda mengontrol akun pengguna yang ada di aplikasi. Pengujian kinerja menunjukkan peningkatan dalam pencarian data dan pemuatan halaman menggunakan Google Lighthouse dan React Developer Tools. Selain itu, pengaturan gelap

dan terang meningkatkan pengalaman pengguna. Secara keseluruhan, aplikasi ini mendukung transformasi digital di lingkungan Diskominfo Kota Malang dengan mempermudah Sub-Substansi Prasarana dan Sarana Teknologi Informasi (Aptika) dalam mengelola data aset, mempercepat pencarian, dan mengurangi kesalahan manual.

UCAPAN TERIMA KASIH

Terima kasih yang sebesar-besarnya kepada **DISKOMINFO Kota Malang** atas kesempatan dan dukungan yang telah diberikan selama proses penelitian dan pengembangan aplikasi ini. Terima kasih kepada Ibu **Nurlila Yulvia Chesarani, S.Kom.** sebagai pembimbing lapangan, yang telah memberikan panduan, pengetahuan praktis, dan dukungan selama pelaksanaan penelitian di lapangan.

Terima kasih kepada **Universitas Muhammadiyah Sidoarjo (UMSIDA)**, khususnya kepada Ibu **Ade Eviyanti, S.Kom., M.Kom.** selaku dosen pembimbing, yang telah memberikan bimbingan, arahan, dan dukungan yang sangat berarti sepanjang penyusunan tugas akhir ini. Semua kontribusi dan bantuan yang diberikan sangat berharga dan menjadi motivasi besar dalam menyelesaikan penelitian ini.

REFERENSI

- [1] V. Febrianto, "Pemkot Malang berkomitmen tingkatkan layanan digital," ANTARA. [Online]. Available: <https://jatim.antaranews.com/berita/751638/pemkot-malang-berkomitmen-tingkatkan-layanan-digital>
- [2] D. Widhyaestoeti, K. T. Agnes, J. Jaenudin, F. Rachmawati, F. Fatimah, and B. Wulandari, "Pengembangan Aplikasi Berbasis Web Untuk Meningkatkan Pemesanan Produk UMKM KIMELS HIJAB," *J. Pengabdian Masyarakat*, vol. 9, no. 1, pp. 165–174, 2024, doi: 10.30653/jppm.v9i1.650.
- [3] H. Kusuma and A. Ekatya, "Cara Pemkot Malang Wujudkan Layanan Digital Dengan SPBE," Viva. [Online]. Available: <https://malang.viva.co.id/politik/2038-cara-pemkot-malang-wujudkan-layanan-digital-dengan-spbe?page=1>
- [4] S. Mufti Prasetyo, M. Ivan Prayogi Nugroho, R. Lima Putri, and O. Fauzi, "Pembahasan Mengenai Front-End Web Developer dalam Ruang Lingkup Web Development," *J. Multidisiplin Ilmu*, vol. 1, no. 6, pp. 1015–1020, 2022.
- [5] Admin_dti, "ReactJS : Definisi, Fitur, Manfaat, Cara Kerja, Keunggulan dan Kekurangan," Telkom University Jakarta. [Online]. Available: <https://jakarta.telkomuniversity.ac.id/reactjs-definisi-fitur-manfaat-cara-kerja-dan-keunggulan/>
- [6] L. Iswari and Nasution, "Penerapan React JS Pada Pengembangan FrontEnd," *Automata*, vol. 2, no. 2, pp. 193–200, 2021.
- [7] D. Hindarto, "The Management of Projects is Improved Through Enterprise Architecture on Project Management Application Systems," *Int. J. Softw. Eng. Comput. Sci.*, vol. 3, no. 2, pp. 151–161, 2023, doi: 10.35870/ijsecs.v3i2.1512.
- [8] K. Schwaber and J. Sutherland, "Scrum Guide V7," *Agil. Metrics Agil. Heal. Metrics Predict.*, no. November, pp. 133–152, 2020.
- [9] M. Fariz, S. Lazuardy, and D. Anggraini, "Modern Front End Web Architectures with React.Js and Next.Js," *Int. Res. J. Adv. Eng. Sci.*, vol. 7, no. 1, pp. 132–141, 2022.
- [10] D. Grover and H. Kunduru, *ES6 for Humans*. 2017. doi: 10.1007/978-1-4842-2623-0.
- [11] Axios Documentation, "Getting Started with Axios: HTTP Client for the Browser and Node.js.," Axios. [Online]. Available: <https://axios-http.com/docs/intro>

- [12] A. De Lucia and A. Qusef, "Requirements engineering in agile software development," *J. Emerg. Technol. Web Intell.*, vol. 2, no. 3, pp. 212–220, 2010, doi: 10.4304/jetwi.2.3.212-220.
- [13] G. Developers, "Lighthouse: Performance Auditing for the Web," Google. [Online]. Available: <https://developer.chrome.com/docs/lighthouse/overview>
- [14] H. Ben Kora and M. Manita, "Modern Front-end Web Architecture Using React.Js and Next.Js," *University of Zawia Journal of Engineering Sciences and Technology*, vol. 2, no. 1, pp. 1–13, 2024. DOI: <https://doi.org/10.26629/uzjest.2024.01>.
- [15] Z. Hoy and M. Xu, "Agile Software Requirements Engineering Challenges-Solutions—A Conceptual Framework from Systematic Literature Review," *Information*, vol. 14, no. 6, Art. no. 322, 2023. DOI: <https://doi.org/10.3390/info14060322>.