BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama Penulis melaksanakan kerja magang di PT Inti Karya Indonesia, penulis tergabung sebagai IT yang bertugas untuk membuat *website* yang berada di divisi pergudangan atau *stock*. Berkerja langsung dibawah bimbigan Ibu Sumarni Salim selaku *HRD & GA Manager* dan atas persetujuan dari *General manager* kantor dan direktur PT Inti Karya Indonesia. Berikut merupakan gambaran Struktur spesifik penulis selama melaksanakan kerja magang di PT Inti karya Indonesia:

3.2 Tugas yang Dilakukan

Pada saat pelaksanaan kerja magang , penulis diberi proyek magang untuk membuat *Stock Inventory* berbasis *Website*. Sesuai dengan persetujuan *General Manager* dan arahan *dari HRD & GA Manager* sekaligus merupakan *supervisor* penulis, penulis dipercaya untuk membuat *Stock Inventory Management* berbasis *Website* yang hanya akan digunakan di dalam Perusahaan itu sendiri secara local atau *internal*.

Secara umum, *Project* ini berupa *website* Dimana pengguna memerlukan login sebagai autentikasi sesuai dengan role nya. Disini penulis diberikan *project* untuk membuat 3 *role* yang masing-masing memiliki fungsi tersendiri, kemudian untu masing-masing *role* memiliki halaman nya sendiri. Secara umum, *website stock* ini memiliki fungsi untuk menambah pengguna , *edit , delete,* laporan *stock, print stock,* dan *print* barang.

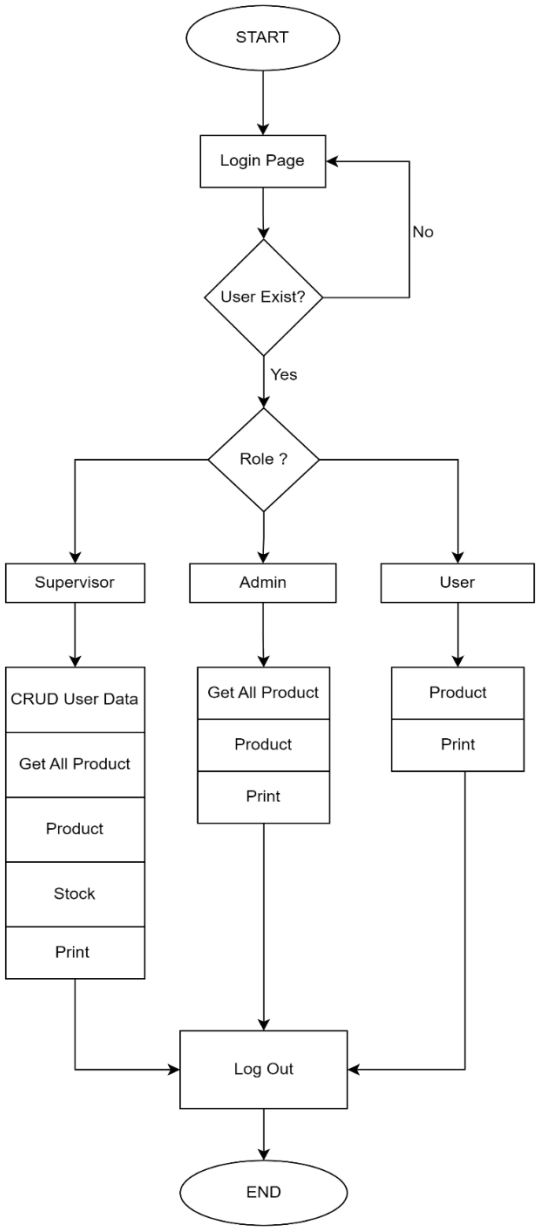
Proses Bisnis *Stock Inventory Website* dimulai dengan langkah masuk ke halaman *login,* di mana pengguna diminta untuk melakukan *login* dengan memberikan *ID* dan *password*. Setelah *login* berhasil, sistem mengidentifikasi *role* pengguna (seperti *supervisor, admin, atau user*) dan mengarahkannya ke halaman yang sesuai dengan hak aksesnya. Sebagai contoh, *supervisor* akan diarahkan ke halaman *supervisor* yang menyediakan akses *CRUD* untuk data pengguna, informasi produk, data stok, dan fitur cetak laporan stok dan produk. Jika login sebagai *admin*, halaman ini akan memberikan akses untuk melihat keseluruhan produk, menambahkan produk baru, dan *print* informasi produk. Pengguna dengan *role* "*User*" diarahkan ke halaman khusus *user*, di mana mereka dapat melihat produk yang telah mereka buat dan mencetak informasi produk tersebut. Setelah pengguna menyelesaikan semua tugasnya di *website* stok, langkah terakhir adalah melakukan *logout* untuk menjaga keamanan sistem dan mencegah potensi penyalahgunaan oleh pengguna atau pekerja lainnya.. berikut dibawah ini merupakan tabel Skema *Authorization* untuk setiap *role* nya:

Tabel 3. 1 Skema Authorization

Role	***Authorization***
Supervisor	• Mendapatkan Akses CRUD untuk halaman *User, Product, Stock*. • Mendapatkan Akses keseluruhan *Product* yang diinput oleh semua *user*. • Mengubah dan mengedit keseluruhan *product* . • Menambah *User* dan *Stock* baru. • Melakukan *CRUD* terhadap semua *User*
Admin	• Mendapatkan Akses CRUD untuk halaman *Product*.

	• Mendapatkan Akses keseluruhan *Product* yang diinput oleh semua *user*.
User	• Mendapatkan Akses CRUD untuk halaman *Product*. • Hanya dapat melihat *product* yang diinput oleh *user* itu sendiri.

Secara umum, Cara kerja *website* ini dapat dilihat di susunan *flowchart* dibawah ini:



Gambar 3. 1 Flowchart Cara Kerja Website

3.2 Uraian Kerja Magang

Berikut dibawah ini merupakan table linimasa selama penulis melaksanakan magang pembuatan *website stock inventory* :

Tabel 3.2 Uraian Kerja Magang

Minggu	Rincian Tugas yang dikerjakan
1	• Briefing *jobdesk* dengan pembimbing lapangan di PT Inti Karya Indonesia. • Diskusi *Project stock website*. • Perencanaan fitur-fitur apa saja yang akan ditampilkan di *website*.
2	• Laporan *progress* mingguan setiap hari senin dalam bentuk *file pdf*. • Mempelajari berbagai fitur yang dibutuhkan oleh Perusahaan. • Merencanakan dan mendiskusikan fitur dan komponen apa saja yang diperlukan untuk pembuatan *website*.
3	• Menentukan pembuatan *project* dengan menggunakan *MERN stack (MySql, Express, React, NodeJs).* • Mempelajari dokumentasi *Express* sebagai *backend server*. • Mencari referensi gambaran struktur *project* • Laporan *progress* mingguan setiap hari senin dalam bentuk *file pdf.*
4-5	• Laporan *progress* mingguan setiap hari senin dalam bentuk *file pdf.*

	• Membuat *Database* yang disertai dengan *entity Relation* • Pembuatan *table user, stock, dan product* pada *database*. Kemudian menghubungkan *backend express pada database mysql* dengan menggunakan *Sequelize.* • Menginstal beberapa *dependencies* pada *backend (argon2, cors, mysql2, express dan sequelize).*
6	• Laporan *progress* mingguan setiap hari senin dalam bentuk *file pdf* • Mengembangkan dan memperbaiki berbagai fitur pada *backend express.* • Membuat *role supervisor, admin dan user guest*. • Membuat fungsi *CRUD* pengguna untuk *role Supervisor* menggunakan Request.rest dan postman untuk pengujian. • *Testing function CRUD* agar terhubung ke *database mysq*l untuk *role supervisor*, serta membuat implementasi fitur pada *stock, dan product.* Untuk table produk hanya bisa diakses oleh *Supervisor dan Admin*. • Membuat fungsi *get all product* untuk *role Supervisor dan admin*. Kemudian membuat fitur *edit dan delete data product* yang hanya bisa diakses oleh *role Supervisor*. • Membuat *Alert "Access Denied"* jika *admin* ingin mengedit dan *delete data* produk dikarenakan hanya *role supervisor* saja yang dapat melakukan *edit dan delete data*.

7	• Laporan *progress* mingguan setiap hari senin dalam bentuk pdf • Mempelajari dokumentasi *React* debagai sisi *Frontend* nya. • Menginstall beberapa *dependencies* penting/ utama agar dapat berkomunikasi dengan *backend* nya seperti*(axios, react-dom, react-redux).* • Menginstall *bulma css* dan *react icons* pada frontend sebagai penunjang pembuatan *website* agar tampilan lebih menarik. • Pembuatan halaman untuk masing-masing role
8-9	• Laporan *progress* Mingguan setiap hari senin dalam bentuk *pdf* • Melakukan *Debugging fitur.*
10	• Mendiskusikan untuk perencanaan penggunaan *website* secara *local.* • *Debugging fitur* yang masih *error*.
11-12	• Melakukan *demo project* tentang penggunaan *website* di PT Inti Karya Indonesia. • Presentasi *Project website* terhadap *supervisor.*

3.2.1 Backend

Dalam pembuatan *Inventory Stock Website*, penulis menggunakan *framework NodeJs yaitu Express* sebagai sisi server nya. Di dalam *framework Express*, penulis membuat *controllers*

untuk masing-masing komponen *controller* yaitu *auth* , *product, user, dan stock.* Selain *controller*, juga terdapat *models* dan *route* pada *backend*

A. Auth

Bagian *auth* terdapat 3 fitur yaitu terdapat *Login, Me dan Logout.* Fungsi dari fitur login adalah untuk mengecek apakah user tersebut sudah masuk ke dalam *database*, sehingga Ketika sudah *exist* maka *user* tersebut bisa *login (user credentials exist).* Di bagian ini juga terdapat password yang sudah di *hashing* dengan menggunakan *argon*. Untuk lebih jelas akan penulis tampilkan potongan code dibawa ini:

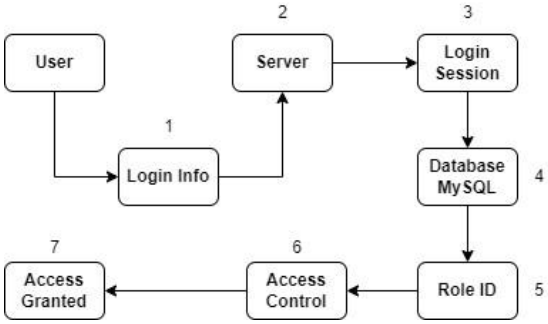
Gambar 3. 2 Snippet code Login dengan role ID

```
4   export const Login = async (req, res) => {
5     // Mendapatkan pengguna berdasarkan alamat email dari permintaan
6     const user = await User.findOne({
7       where: {
8         email: req.body.email,
9       },
10    });
11    // Memeriksa apakah pengguna ditemukan
12    if (!user) return res.status(404).json({ msg: "User Not Found" });
13    // Memverifikasi kata sandi menggunakan argon2
14    const match = await argon2.verify(user.password, req.body.password);
15    // Memeriksa apakah kata sandi sesuai
16    if (!match) return res.status(400).json({ msg: "Wrong Password" });
17    // Menyimpan ID pengguna dalam sesi
18    req.session.userId = user.uuid;
19    // Mengembalikan data pengguna terpilih
20    const { uuid, name, email, role } = user;
21    res.status(200).json({ uuid, name, email, role });
22  };
23
```

Selain itu juga terdapat *Me* dan *Logout.* Dalam implementasi website inventory, ada tiga jenis user dengan perbedaan diatur melalui Role ID. Role ID digunakan untuk menentukan jenis user dan fitur yang dapat diakses. Selain itu, Role ID juga diterapkan pada session untuk memastikan hanya user yang telah login dengan privilege sesuai yang dapat mengakses halaman dan fitur tertentu. Di bagian me ini user yang login akan di cek terlebih dahulu apakah user tersebut sudah login dengan menggunakan *login session sequelize*. Session tersebut akan

masuk ke dalam *database MySQL*. Untuk lebih jelasnya akan penulis lampirkan skema diagram dan potongan code nya

Skema Diagram :



Proses ini memastikan bahwa pengguna yang telah *login* memiliki sesi yang *valid* dan hak akses yang sesuai sebelum diizinkan untuk mengakses halaman dan fitur tertentu dalam *website inventory*. Penjelasan langkah-langkah dalam diagram akan penulis sampaikan di bawah ini:

1. ***User Login Info:***

   Pengguna memasukkan informasi *login* seperti *username* dan *password*.

2. ***Server:***

   Server akan memberikan respon jika *user* belum terdaftar. Jika sudah terdaftar di *database*, maka akan diarahkan ke *login session*.

3. ***Login Session:***

   Server menggunakan *Sequelize* untuk memvalidasi informasi *login session* pengguna.

4. ***Database MySQL:***

   Setelah validasi berhasil, *server* memvalidasi apakah sesi *login* tersebut masih berlaku. Jika *session* sudah *expired*, maka pengguna tersebut akan diarahkan Kembali ke menu *login* awal.

5. ***Role ID:***

   Jika sesi *login* masih berlaku, *server* melakukan verifikasi terhadap *Role ID* pengguna. Jika pengguna login dan masuk sebagai supervisor maka sistem akan diarahkan pada halaman *supervisor*. Begitupun untuk *role Admin dan User* lainnya.

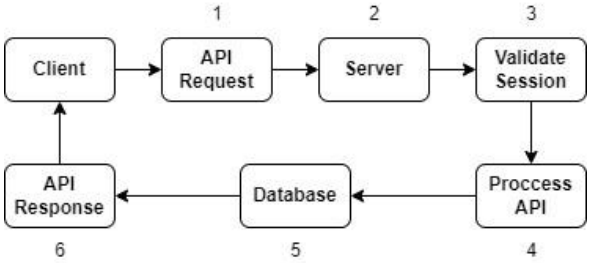
6. ***Access Control:***

   Berdasarkan Role ID, *server* menentukan hak akses pengguna terhadap halaman dan fitur tertentu.

7. ***Access Granted:***

   Jika semua langkah berhasil, *server* memberikan respons bahwa akses diberikan.

API Flow Request Diagram :



- ***Client Request:***

  Setiap kali client melakukan permintaan API, client menyertakan token akses dalam header atau sebagai parameter.

- ***Session Validation:***

  Server memvalidasi session untuk memastikan keasliannya dan mengambil informasi user dan rolenya. Jika session valid, server melanjutkan pemrosesan API.

- ***API Proccesing:***

  Server memproses permintaan *API* sesuai dengan hak akses *user* yang terkandung dalam *token*. Jika *user* memiliki hak akses yang cukup, *server* merespons dengan apa yang diminta.

Berikut dibawah ini merupakan penjelasan dalam bentuk code :

Gambar 3. 3 Snippet code Get me dan Log Out

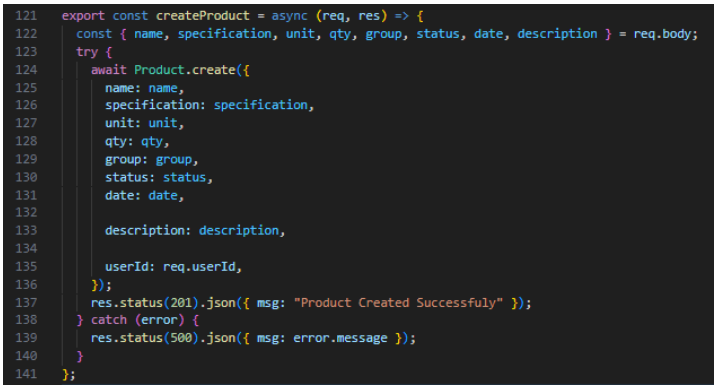
```
23
24  export const Me = async (req, res) => {
25    // Memeriksa apakah pengguna sudah masuk
26    if (!req.session.userId) {
27      return res.status(401).json({ msg: "Please Login Your Account" });
28    }
29    // Mencari pengguna berdasarkan ID sesi
30    const user = await User.findOne({
31      attributes: ["uuid", "name", "email", "role"],
32      where: {
33        uuid: req.session.userId,
34      },
35    });
36    // Memeriksa apakah pe( any ) ditemukan
37    if (!user) return res.status(404).json({ msg: "User Not Found" });
38    // Mengembalikan data pengguna terpilih
39    res.status(200).json(user);
40  };
41
42  export const LogOut = (req, res) => {
43    // Menghapus sesi pengguna
44    req.session.destroy((err) => {
45      if (err) return res.status(400).json({ msg: "Can't Logout" });
46      res.status(200).json({ msg: "You Logged Out" });
47    });
48  };
49
```

Kemudian akan penulis hubungan pada *auth route* yang terdapat pada *route* penulis. Selain itu penulis juga membuat auth *middleware* yang berfungsi untuk mencegah *injection website* yang telah dibuat.

B. CRUD

Fungsi *Create, Read, Update, dan Delete* atau yang biasanya disebut *CRUD* adalah sebuah fungsi utama *dalam website stock inventory* ini. Fungsi *CRUD* ini terdapat di setiap *controller yakni Users, Products, dan Stock. Logic CRUD* yang digunakan oleh penulis mengambil data yang dimasukkan oleh pengguna dari halaman *website* dengan form yang disediakan. Kemudian data-data tersebut yang sudah diisi akan masuk ke dalam *database(Create, Update, dan Delete),* sehingga jika user dengan *role* yang memiliki *previllege* untuk melakukan *update atau delete* akan tercatat di dalam *database MySQL*. Berikut akan penulis jabarkan dalam bentuk gambar untuk potongan setiap *code* yang mengambil fungsi c*reate, update dan delete* pada *product* di bawah ini dari Gambar 3.4 sampai 3.6

Gambar 3. 4 Snippet Code Create Product



```
export const createProduct = async (req, res) => {
  const { name, specification, unit, qty, group, status, date, description } = req.body;
  try {
    await Product.create({
      name: name,
      specification: specification,
      unit: unit,
      qty: qty,
      group: group,
      status: status,
      date: date,

      description: description,

      userId: req.userId,
    });
    res.status(201).json({ msg: "Product Created Successfuly" });
  } catch (error) {
    res.status(500).json({ msg: error.message });
  }
};
```

Gambar 3. 5 Snippet Code Edit Product

```
export const updateProduct = async (req, res) => {
  try {
    const product = await Product.findOne({
      where: {
        uuid: req.params.id,
      },
    });

    if (!product) return res.status(404).json({ msg: "Data Not Found" });

    const { name, specification, unit, qty, group, status, date , description} = req.body;

    if (req.role                                    === product.userId) {
      await Prod (property) specification: any
        { name, specification, unit, qty, group, status, date, description},
        {
          where: {
            id: product.id,
          },
        }
      );

      res.status(200).json({ msg: "Product Updated Successfully" });
    } else {
      res.status(403).json({ msg: "Access Denied" });
    }
  } catch (error) {
    res.status(500).json({ msg: error.message });
  }
};
```

Gambar 3. 6 Snippet Code Delete Product

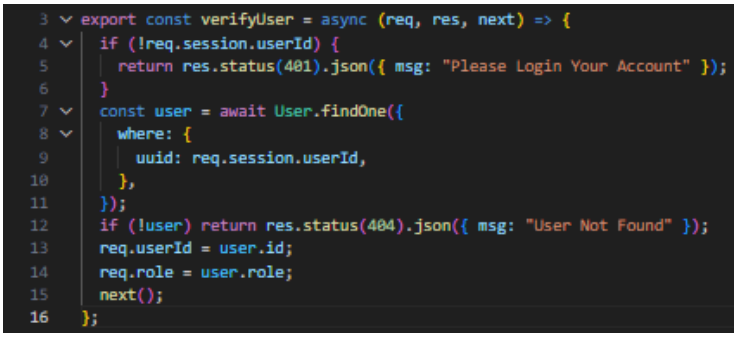
```
export const deleteProduct = async (req, res) => {
  try {
    const product = await Product.findOne({
      where: {
        uuid: req.params.id,
      },
    });

    if (!product) return res.status(404).json({ msg: "Data Not Found" });

    if (req.role === "supervisor" || req.userId === product.userId) {
      await Product.destroy({
        where: {
          id: product.id,
        },
      });

      res.status(200).json({ msg: "Product Deleted Successfully" });
    } else {
      res.status(403).json({ msg: "Access Denied" });
    }
  } catch (error) {
    res.status(500).json({ msg: error.message });
  }
```

C. Middleware

Kemudian penulis juga membuat *middleware* yang berguna untuk *verify user* apakah user tersebut sudah masuk ke dalam *database* atau belum. Fungsi *verify user* ini akan memeriksa apakah terdapat ID pengguna yang didimpan dalam sesi. Jika pengguna tersebut tidak ditemukan, maka sistem akan memberi *respon status 404* dengan pesan *"User Not Found"*. Selain itu, juga ada *middleware supervisor only* Dimana bagian ini,

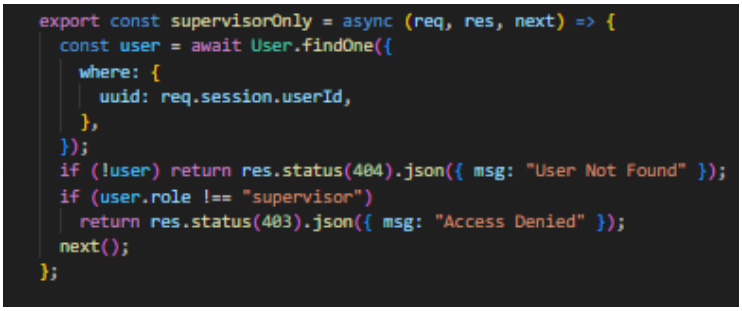
middleware akan mengecek apakah *user login* sebagai *supervisor*. Sebagai contoh, Jika *login* sebagai *admin* akan mendapatkan *respon status 403* dengan pesan *Access Denied*. Jika *login* sebagai *Supervisor* maka akan lanjut ke halaman khusus *supervisor* sendiri saja yang dapat mengaksesnya. Berikut akan penulis tampilkan dalam bentuk snippet:

Gambar 3. 7 Potongan Code Middleware verify user



```
export const verifyUser = async (req, res, next) => {
  if (!req.session.userId) {
    return res.status(401).json({ msg: "Please Login Your Account" });
  }
  const user = await User.findOne({
    where: {
      uuid: req.session.userId,
    },
  });
  if (!user) return res.status(404).json({ msg: "User Not Found" });
  req.userId = user.id;
  req.role = user.role;
  next();
};
```

Gambar 3. 8 Potongan Code Middleware Supervisor Only



```
export const supervisorOnly = async (req, res, next) => {
  const user = await User.findOne({
    where: {
      uuid: req.session.userId,
    },
  });
  if (!user) return res.status(404).json({ msg: "User Not Found" });
  if (user.role !== "supervisor")
    return res.status(403).json({ msg: "Access Denied" });
  next();
};
```

3.2.2 Frontend

Untuk sisi client atau *Frontend*, penulis disarankan menggunakan *react* dalam pembuatan *website inventory stock* tersebut. Pada sisi *frontend React*, penulis memerlukan *dependencies* utama dalam hal ini seperti menginstall *axios* agar dapat berkomunikasi dengan *backend* atau *server* nya.

A. Bulma CSS

Bulma CSS merupakan sebuah kerangka *css* yang penulis gunakan dalam pembuatan *website inventory* ini. Cara kerja nya yaitu penulis harus memamggil *Bulma* dengan menginstal *dependencies* nya terlebih dahulu, kemudian memanggil fungsi bulma menggunakan *Import* . berikut ini merupakan potongan code pada pembuatan page menggunakan bulma css:

Gambar 3. 9 Import Code Bulma CSS

```
1   import React from "react";
2   import { createRoot } from "react-dom/client";
3   import { Provider } from "react-redux";
4   import { store } from "./app/store";
5   import App from "./App";
6   import "bulma/css/bulma.css";
7   import axios from "axios";
8
9   axios.defaults.withCredentials = true;
10
```

B. React Icon

Penulis juga menggunakan icon react untuk menampilkan hasil yang lebih bagus deperti pada *icon view detail, edit,* dan *delete.* untuk lebih jelas akan penulis tampilkan pada table dibawah ini:

Tabel 3.3 Penggunaan React Icon

Fungsi	***Icon***
View- Detail	*AiOutlineEye*
Edit	*FaEdit*
Delete	*FaTrashAlt*
LogOut	*IoLogout*

C. React Alert

Fungsi *React Alert* ini adalah untuk menyakinkan pengguna untuk melakukan *delete data*. *Alert* ini akan muncul

Ketika pengguna ingin *mengdelete data* dan *edit data*. Sebelum bisa menjalankan fungsi tersebut, penulis menginstal *dependencies react alert* terlebih dahulu. Berikut akan penulis jabarkan dalam bentuk gambar dibawah ini tentang *react alert*:

Gambar 3. 10 Snippet Code React Alert

```
34  const confirmDelete = (id) => {
35    confirmAlert({
36      title: "Delete User ",
37      message: "Are you sure you want to delete this user? ",
38      buttons: [
39        {
40          label: "Delete ",
41          onClick: () => deleteUser(id), // Memanggil fungsi delete
42        },
43        {
44          label: "Cancel ",
45        },
46      ],
47    });
48  };
```

D. React Print

Kegunaan *react print* yaitu untuk mengeprint laporan *stock* dan *product website inventory* yang penulis buat. Laporan yang di *print* akan berbentuk dalam *file pdf*. Untuk menjalankan fungsi tersebut, penulis harus menginstall *dependencies react to print* terlebih dahulu. Berikut dibawah ini akan penulis jabarkan code dalam bentuk *snippet:*

Gambar 3. 11 Snippet code React Print

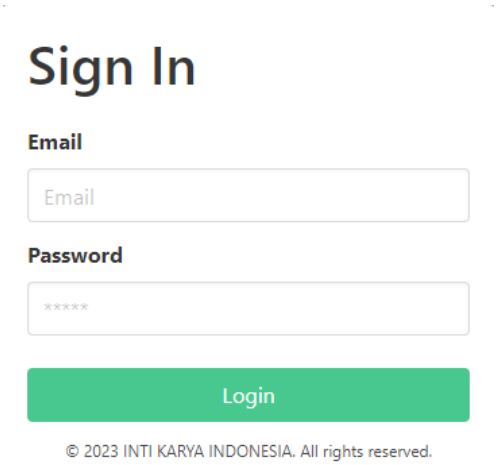
```
56  const handlePrint = useReactToPrint({
57    content: () => componentRef.current,
58  });
59  return (
```

3.3 Hasil Website

3.3.1 Halaman Login

Halaman *Login* adalah page awal yang akan muncul saat *website* tersebut dibuka. Pada halaman ini, pengguna akan disuruh

login dengan mengisi *email* dan *password* agar bisa masuk dan mengakses keseluruhan *website* sesuai dengan *role* yang telah dibuat. Pada halaman *login* ini juga ada sistem *user exist* Dimana *web* tersebut akan mendeteksi apakah pengguna ini sudah terdapat dalam *database mysql* atau belum seperti yang telah di jelaskan pada gambar *flowchart* 3.1. berikut dibawah ini merupakan ilustrasi tampilan halaman *login* pada gambar 3.12:



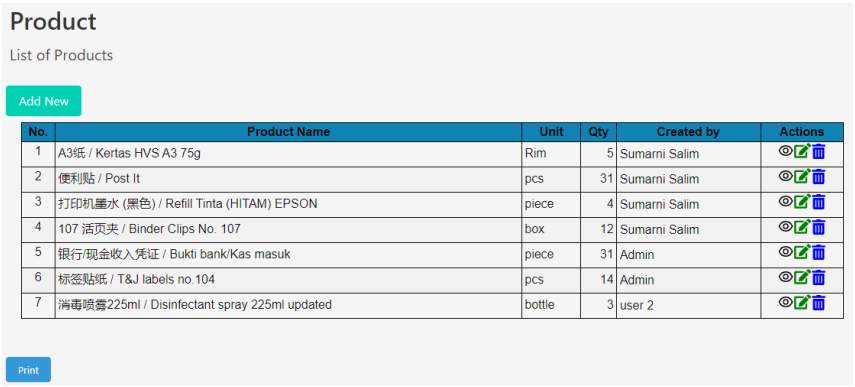
Gambar 3. 12 Halaman Login pada Stock Inventory Web

3.3.2 Halaman Supervisor

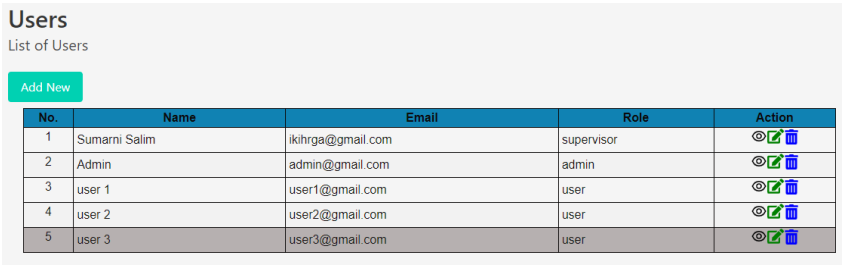
Halaman *supervisor* ini terdapat tampilan *page* yang paling lengkap dari semua *role*. Pada halaman *Supervisor* ini , akan ada *menu* di pojok kiri atas yang berisikan fitur-fitur *dashboard, Products, Users,* dan *Stock*. Halaman *Supervisor* dapat dilihat pada gambar 3.13

Gambar 3. 13 Halaman Supervisor pada Stock Inventory Website

Selanjutnya ada halaman produk, Dimana jika *login* sebagai *supervisor* akan mendapatkan akses untuk melihat keseluruhan produk yang diinput oleh *Admin/User/Supervisor* itu sendiri. Selain itu juga terdapat fitur *Stock* dan *User* yang dapat diakses oleh user dengan *role supervisor*. *Role Supervisor* bisa menambah user sesuai keinginannya dan dapat membuat menjadi *admin* juga. Selain itu juga bisa melakukan *update* data *User/ Admin* jika memiliki kesalahan data pengguna. *Role Supervisor* ini juga dapat melakukan *delete data* pengguna. Secara keseluruhan, halaman *Supervisor* akan dapat dilihat pada gambar 3.14 sampai 3.16.



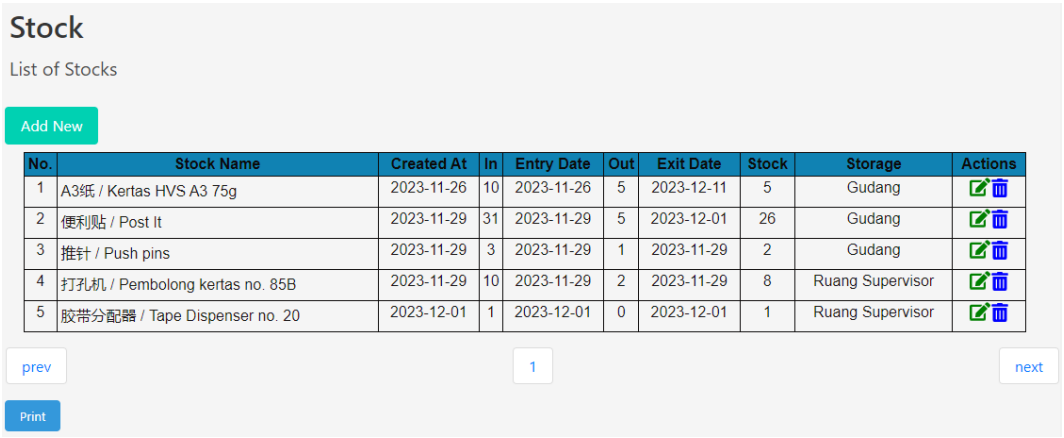
Gambar 3. 14 Tampilan Product pada Halaman Supervisor



Gambar 3. 15 Tampilan Users pada Halaman Supervisor

Untuk *fitur stock* ini juga hanya bisa diakses oleh *role supervisor*, Dimana *role* tersebut bisa menambahkan *stock* yang terdiri dari nama *stock* itu sendiri, produk *created a*t, barang masuk,

tanggal barang masuk, barang keluar, tanggal barang keluar, sisa *stock* yang ada, dan keterangan siapa yang mengambilnya.

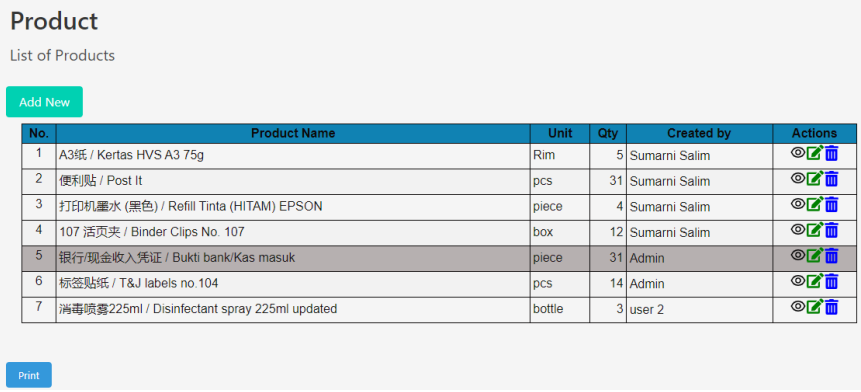


Gambar 3. 16 Tampilan Stock pada Halaman Supervisor

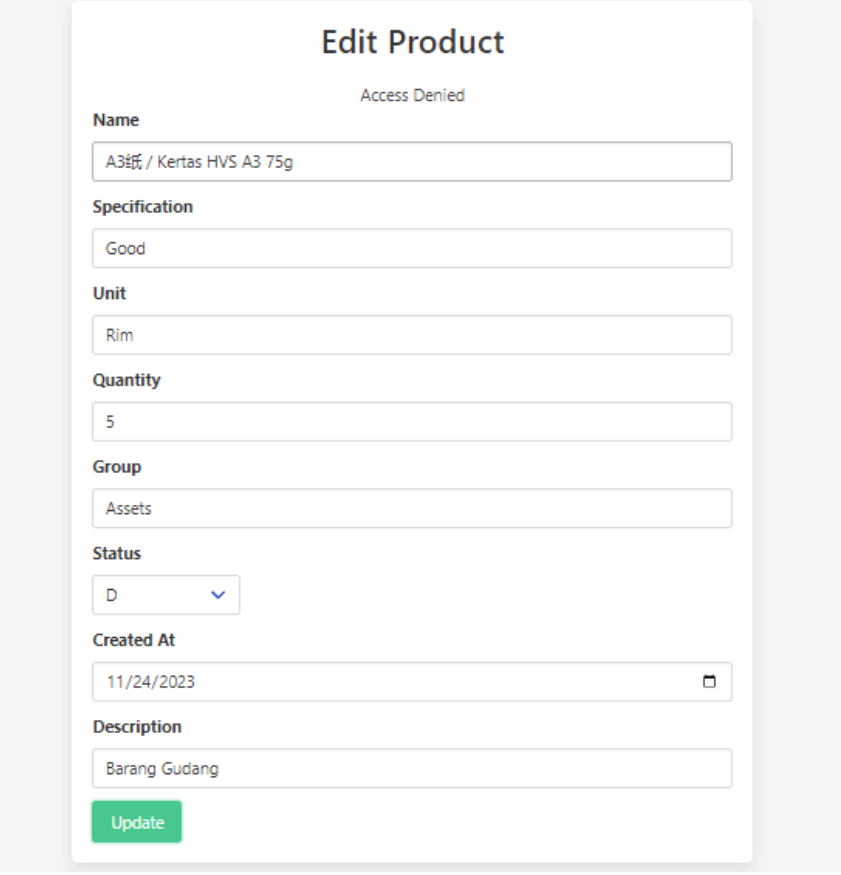
3.3.3 Halaman Admin

Pada halaman awal *Admin* ini, user dengan *role admin* memiliki beberapa fitur yaitu menambah *product* , mendapatkan akses keseluruhan produk yang diinput oleh *Admin/ Supervisor/ User* lainnya. *Halaman Admin* ini dibuat untuk mengecek apakah terdapat kesalahan penulisan produk atau barang yang diinput oleh user, akan tetapi *role Admin* tidak bisa mengedit data dan *delete product data* yang bukan *Admin input*. *Admin* hanya bisa melakukan pengecekan melalui *view detail* yang kemudian jika terdapat kesalahan, *Admin* memberitahu *user* untuk segera *update* hasilnya. Jika *Admin* mencoba *update data, maka Alert Access Denied* akan muncul di halaman *Admin*. Secara keseluruhan, halaman *Admin* akan dapat dilihat pada gambar 3.17 sampai 3.19.

Gambar 3. 17 Halaman Admin pada Stock Inventory Website



Gambar 3. 18 Tampilan Product pada Halaman Admin



Gambar 3. 19 Tampilan Access Denied pada Halaman Admin

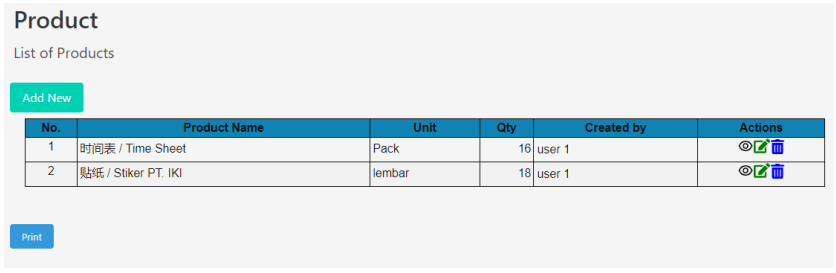
Role Admin tidak bisa melakukan *update data* dan *delete data*, dikarenakan sesuai dengan *request* dari kantor bahwa hanya

role supervisor saja yang dapat melakukan *update data* dan *delete data*. Untuk *role Admin* ini hanya diberikan akses keseluruhan produk dan sudah diatur di *backend* nya. *Role admin* ini hanya bisa mengubah dan mengdelete produk yang dia buat.

3.3.4 Halaman User

Pada Halaman *User* ini, kurang lebih hampir sama dengan halaman *admin*. Akan tetapi *User* hanya bisa melihat produk yang dia buat, mengedit produk yang dia buat dan menghapus data produk. Selain itu *User* juga dapat menggunakan *fitur print* produk yang telah ia buat yang kemudian akan dilaporkan pada atasan dalam bentuk *file pdf*. Secara keseluruhan, halaman *User* dapat dilihat pada gambar 3.20 sampai 3.21.

Gambar 3. 20 Halaman User pada Stock Inventory Website



Gambar 3. 21 Tampilan Produk pada halaman User

3.3.5 Deployment

Setelah presentasi proyek website dilakukan secara langsung di PT Inti Karya Indonesia, terdapat keputusan dari atasan penulis untuk menggunakan website tersebut secara internal tanpa melakukan deployment secara realtime. Berikut adalah penjabaran lebih detail mengenai implikasi dan langkah-langkah yang mungkin perlu diambil:

- **Internal Usage Only**

  Keputusan untuk menggunakan *website* secara *internal* berarti bahwa *website* tersebut tidak akan diakses oleh publik umum. Hanya pengguna yang terdaftar dan memiliki hak akses *internal* yang dapat menggunakan fitur-fitur pada *website.* Informasi dan data yang diakses melalui *website* ini hanya digunakan di dalam perusahaan.

- **Penjelasan mengenai penggunaan *Website***

  Sebagai pengguna Internal di Perusahaan tersebut, perlu diberikan penjelasan kepada pengguna *internal* tentang cara menggunakan *website* tersebut. Ini termasuk penjelasan tentang fitur-fitur yang tersedia, hak akses masing-masing pengguna berdasarkan *Role* ID.

3.4 Kendala yang Ditemukan

Di dalam Praktek Magang pada PT Inti Karya Indonesia, penulis memiliki beberapa kendala dalam *progress* pembuatan *project*. Salah satu nya adalah penulis tidak memiliki mentor yang ahli dalam bidang *project* yang dibuat penulis. Sehingga terdapat kesulitan pada saat pengerjaan *project* tersebut. Selain itu , ada beberapa faktor error yang kurang dimengerti oleh penulis, sehingga perlu penelusuran yang lebih dalam terkait *project website* yang penulis kerjakan. Penulis juga mendapatkan masalah dalam penggunaan *framework react*, yang sebelumnya penulis belum pernah gunakan dan

banyaknya *sensitive case* pada *framework react* yang membuat penulis kebingungan atas *error* yang terjadi.

3.5 Solusi atas Kendala yang Ditemukan

Mencari tahu penyebab error pada project tersebut. Selain itu, penulis juga mencari referensi sebanyak-banyaknya agar dapat memenuhi permintaan pembuatan website tersebut seperti apa. Penulis juga aktif bertanya kepada *forum discussion* seperti komunitas-komunitas framework atau komunitas yang ada kaitannya dengan dunia IT mengenai error yang terjadi selama penulis mengerjakan *project* magang. Selain itu penulis juga sering bertanya kepada teman penulis yang lebih mengerti , sehingga bisa mendapatkan bantuan.