# BAB 3
# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Organisasi

Pada kegiatan magang berada di bawah Departemen *Dev Presales Division* yang dipimpin oleh Kasriandi Guo selaku *Project Manager*. Untuk magang ini bertugas sebagai *Technical Consultant Analyst* dan bertanggung jawab kepada seorang *Tech Lead*, Ivan Wijaya, yang juga menjadi *mentor* atau *supervisor*. Selama kegiatan magang, mempertanggungjawabkan hasil kerja kepada *Tech Lead*, yang juga bertanggung jawab kepada *Project Manager*. Tugas-tugas diberikan oleh *Project Manager* melalui *Tech Lead*. Penugasan yang diberikan berdasarkan pada studi kasus dari *client* yang didapatkan oleh *Project Manager* dan diteruskan ke *Tech Lead* untuk ditugaskan ke tim intern.

## 3.2 Tugas yang Dilakukan

Tugas utama yang dilakukan adalah membangun sebuah aplikasi *inventory system* yang merupakan bagian dari *Enterprise Resource Planning* (ERP) untuk memberikan solusi sistem kepada *client* berdasarkan *use-case* dari *client*. Pada awal magang ditugaskan untuk belajar teknologi mengenai sektor *DevOps* seperti *jenkins*, *docker*, dan *kubernetes*.

Setelah mempelajari teknologi-teknologi tersebut, alu ditugaskan untuk membuat sebuah aplikasi *login* dan menampilkan *detail employee data* dari API menggunakan swiftUI. Proyek ini ditujukan membantu anggota *DevOps Presales* memberikan demonstrasi aplikasi iOS untuk Perusahaan Astra Honda Motor. *Client* memberikan contoh UI, API untuk *login*, dan API untuk melihat *detail employee* yang harus diterapkan dalam aplikasi demo. Pembangunan *prototype* demo aplikasi menggunakan *native* iOS dengan *framework* swiftUI. SwiftUI merupakan *framework* yang disediakan oleh *Apple* untuk menyusun tampilan UI aplikasi. Sintaks dalam swiftUI lebih mudah dibaca dan lebih mudah untuk ditulis. Dengan menggunakan swiftUI, proses penyusunan *layout* akan menjadi sangat sederhana dan cepat.

Setelah menyelesaikan proyek iOS, kemudian ditugaskan untuk mempelajari *fastlane* dan cara penggunaannya untuk meng-*upload* secara otomatis ke *Google Play Store*. Aplikasi yang di *upload* ke *Google Play Store*

merupakan sebuah web *react.js* yang sudah dikonversikan menjadi *android* menggunakan *capacitor*. *Capacitor* merupakan sebuah *open source native runtime* untuk membangun aplikasi *web native*. Penggunaan *capacitor* dapat membantu mempermudah proses konversi aplikasi web menjadi *android* karena hanya tinggal menjalankan serangkaian *command* yang telah disediakan *capacitor* untuk melakukan *build* aplikasinya menjadi *android*. Agar dapat menggunakan *capacitor* harus meng-*install* dependensi bernama *ionic* terlebih dahulu pada proyek yang dikerjakan.

Jenkins digunakan untuk mengotomatisasi *Software Development Life Cycle* (SDLC) aplikasi yang dibuat, termasuk melakukan *build*, *testing*, dan *deploy software*. Jenkins sendiri terdapat sebuah *stage* dimana dalam *stage* dapat dilakukan serangkaian *command-command* yang ingin dijalankan. Singkatnya, Jenkins digunakan untuk melakukan proses *Continous Integration* dan *Continous Deployment*. *Continuous Integration* (CI) merupakan pengintegrasian kode ke dalam repositori kode yang dapat melakukan pengujian secara otomatis dan cepat. CI dapat dilakukan saat terdapat perintah git *push* ke dalam *source code management*. *Continous Delivery* atau *Continuous Deployment* (CD) merupakan tahapan yang dilakukan setelah proses CI selesai dan seluruh kode berhasil terintegrasi, sehingga aplikasi dapat dibangun dan dirilis secara otomatis. Agar dapat menggunakan *jenkins* pada proyek digunakan file bernama ”Jenkinsfile” dimana di dalam Jenkinsfile tersebut berisi *step-step* atau tahapan proses yang akan dilakukan oleh *jenkins*.

*Fastlane* merupakan sebuah *automation tools* yang digunakan untuk melakukan *building* hingga *deployment* langsung ke *Google Play Store*. *Fastlane* bekerja dengan menggunakan file bernama *fastfile*. Dalam *fastfile* berisi sekumpulan *lane* yang didefinisikan oleh pengguna di mana pengguna dapat memanggil *lane* dan melakukan aksi yang didefinisikan dalam *lane* yang dibuat. *Lane* sendiri memiliki cara kerja yang sama dengan *function*, dimana kita mendefinisikan sebuah *function* beserta dengan *action-action* yang dilakukan di dalamnya dan untuk menggunakannya tinggal memanggil nama dari *function* yang telah dibuat. Sebagai contoh, misalkan terdapat sebuah *lane* bernama ”*build*” yang melakukan aksi untuk memperbarui nomor *build* dan melakukan *build bundle* aplikasi. Untuk menjalankan *lane* ”*build*” yang dibuat, menggunakan *command* ”*fastlane build*” di *terminal* untuk menjalankan *lane* tersebut. *Fastlane* berperan sebagai CD dari *pipeline* dimana saat terjadi *push* ke dalam repositori kode, maka dari *jenkins pipeline* secara otomatis akan menjalankan *lane* yang dibuat pada

*fastfile*.

Setelah selesai membuat *jenkins pipeline fastlane* dan berhasil meng-*upload* aplikasinya ke dalam *Google Play Store*, kemudian melakukan demonstrasi *fastlane* kepada tim *DevOps Presales* untuk dipelajari. Setelah menyelesaikan proyek *fastlane jenkins*, kemudian dimasukkan ke dalam proyek membuat sebuah produk ERP *inventory system*. Sebelum proses pembangunan ERP *inventory system* dilaksanakan, bersama dengan tim menentukan berbagai macam teknologi-teknologi yang akan digunakan dalam pembuatan *inventory system*. Tim kemudian menentukan pembangunan ERP *inventory system* dengan teknologi dan *tools* sebagai berikut :

- Front end : React.js *framework* dan Tailwind
- Back end : Go lang *framework*
- Database : Postgre SQL
- CI / CD : Jenkins *Pipeline*, *Docker*, Kubernetes
- UI / UX Design : Figma
- Source Code Management : Bitbucket
- Project Tracking : Jira
- Documentation : Confluence

Modul yang dikerjakan dalam *inventory system* yaitu modul *front end part request* dan modul *front end spare part*. Penggunaan *react.js* dan *tailwind* memungkinkan untuk membuat UI yang interaktif, responsif, dan mudah disesuaikan. Komponen *react.js* yang dapat digunakan kembali serta *library* yang luas dan penggunaan *utility class tailwind* membuat pengembangan aplikasi lebih efisien dan produktif.

ERP *inventory system* yang dibangun memiliki beberapa fitur utama yang ditugaskan yaitu *part request* dan *spare part*. Fitur *part request* bertujuan untuk membuat sebuah *request* atau permintaan akan sebuah barang kepada *warehouse*. *User* dapat membuat sebuah *request* berisi form dan tabel yang berisi berbagai macam barang yang diminta. Stok dari barang yang diminta tidak akan langsung berkurang karena harus disetujui oleh manajer divisi dan manajer *warehouse* sebelum *part request* diterima. Dalam *part request user* dengan *role employee*

hanya melakukan CRUD dari *part request* yang telah dibuat. Manajer dan admin dapat melakukan CRUD dan melakukan *approval* dari *part request* yang dibuat *user*.

Fitur *spare part* digunakan untuk membuat sebuah barang baru ke dalam inventory system. Dalam *spare part user* dengan *role* admin dan manajer dapat melakukan CRUD *spare part*. Sementara itu, *user* dengan role employee hanya dapat melakukan *read* atau melihat *spare part* yang tersedia.

## 3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

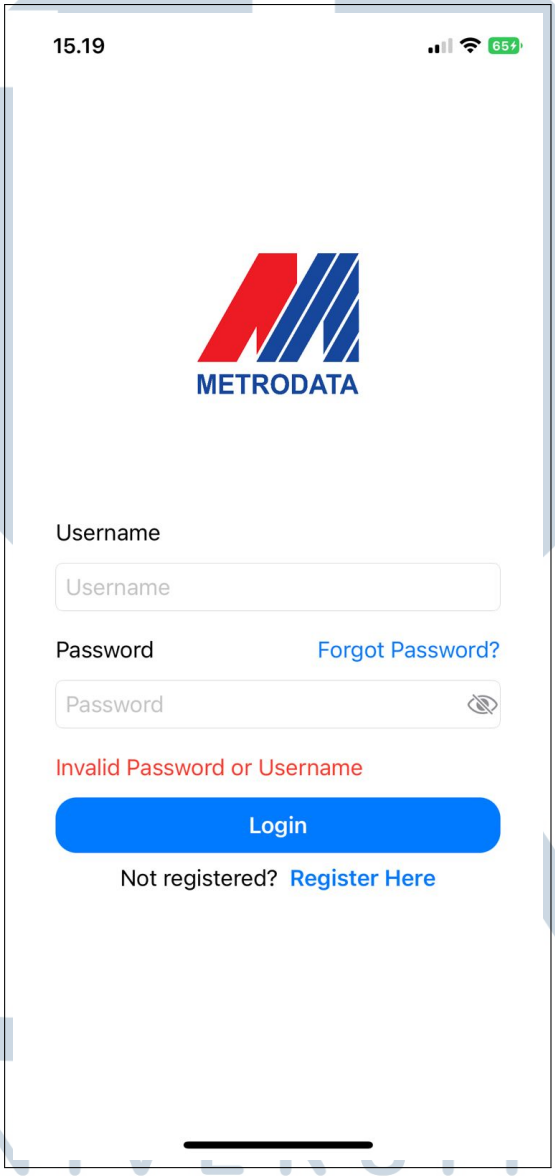
| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 1 | Mempelajari API .NET dan membuat sebuah demo web *login* sederhana menggunakan *react.js* dan mengimplementasikan web *react.js* ke mobile menggunakan webView *android* dan iOS. |
| 2 | Mempelajari mengenai teori DevOps dan melakukan setup instalasi *jenkins* dalam VM Cloud |
| 3 | Mempelejari Jenkins mengenai konfigurasi firewall, job, integrasi git dan belajar mengenai konsep pipeline dalam *jenkins* |
| 4 | Membuat UI *List* Data dan Login page app iOS menggunakan swiftUI. Mengambil data API jwt token ke dalam app. Melakukan deployment app ke *smartphone* iOS. |
| 5 | Membuat logout function di app ios lalu memasukkan aplikasi ke dalam github dan belajar *jenkins* pipeline mengenai environtment, option, parameter, dan trigger. Belajar cara menggunakan bitbucket dan *docker* container |
| 6 | Mempelajari *docker* volume, network, dan belajar membuat sebuah dockerfile |
| 7 | Memodifikasi app ios menggunakan API baru, dan membuat view untuk detail karyawan. Melakukan styling halaman detail karyawan dan melakukan push app ios ke github. Belajar menggunakan *docker* compose dan melakukan instalasi *kubernetes* di pc. Membahas demo app CRUD |
| 8 | Mencari sebuah CRUD demo app menggunakan React.js-Go-Mysql, melakukan push ke bitbucket PT. MII, dan membuat sebuah pipeline di *jenkins* serta integrasi bitbucket dan *jenkins* |
| 9 | Membuat pipeline front-end demo app di *jenkins* dan membuat jenkinsfile untuk melakukan *build* apk. Belajar mengenai *fastlane* |
| 10 | Membuat UI untuk ERP *inventory system* dan melakukan desain ERD bersama dengan tim magang. |
| 11 | Melakukan riset mengenai *tech stack* yang ingin digunakan dalam membuat *inventory system* |
| 12 | Melakukan deployment ke playstore secara manual dan mencoba mengupload app ke playstore menggunakan *jenkins-fastlane* untuk automation-nya. Mempelajari cara penggunaan *fastlane* seperti pembuatan lanenya. |

Tabel 3.2. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 13 | Melakukan demonstrasi cara mendeploy app ke playstore melalui *jenkins fastlane*. Membuat *jenkins* pipeline untuk front-end *user* management, melakukan integrasi bitbucket dengan *jenkins*. |
| 14 | Membuat sebuah unit test sederhana untuk *function login* dan register menggunakan go testify. |
| 15 | Membuat *validation error login* pada *frontend user management*, membuat UI halaman *create part request* dan membuat tabel dalam halaman tersebut. |
| 16 | Membuat *styling* agar halaman *create part request* menjadi responsif, membuat sebuah *button edit* pada tabel saat ditekan akan muncul modal *edit quantity*. Membuat *search bar* untuk mencari *sparepart*, dan *search* berhasil berjalan saat query inputnya menggunakan data statis. Dan membuat function untuk *delete* data pada tabel *part request*. |
| 17 | Membuat modal untuk melakukan penambahan data *spare part* ke dalam tabel *part request*. Melakukan koneksi API untuk *search spare part* dan melakukan *post* ke API *create part request*. |
| 18 | Membuat halaman *edit part request*, melakukan diskusi dengan tim mengenai API *edit part request*. Membantu salah satu anggota tim dalam melakukan koneksi API ke halaman *dashboard part request* |
| 19 | Membuat sebuah repository di nexus untuk menyimpan artifact .aab. Melakukan perubahan terhadap file *fastlane* untuk mengambil artifact dari nexus yang kemudian diupload ke playstore melalui *fastlane*. |
| 20 | Membuat component function modal *delete*. Melakukan koneksi API ke halaman *edit part request*, melakukan validasi dalam form *create* dan *edit part request*, dan menambahkan *button edit* dan *delete* dalam tabel *part request*. dan membuat halaman *create part discrepancy* dan dikoneksikan ke API. |

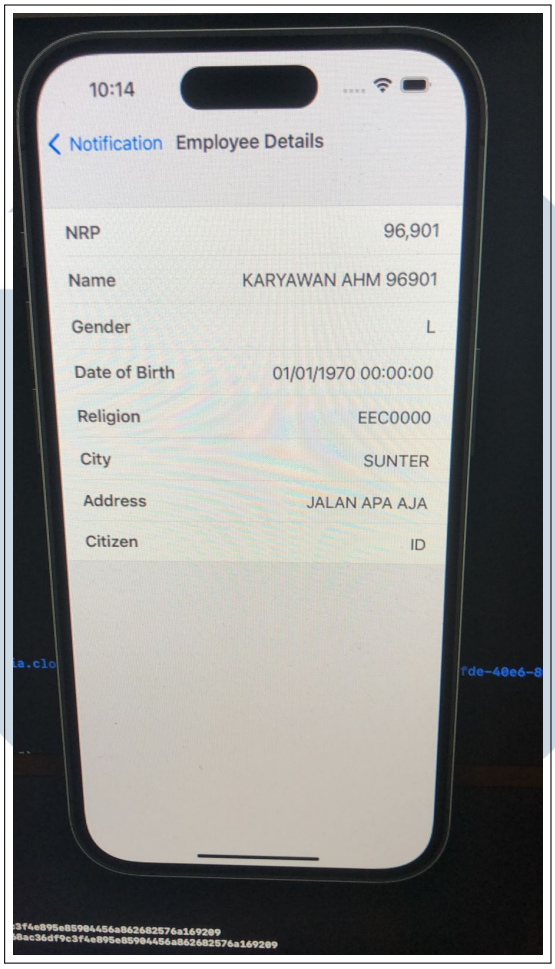
## 3.4 Bukti Hasil Kerja Magang

### 3.4.1 Tampilan Demo Aplikasi iOS




Gambar 3.1. Tampilan halaman *login*

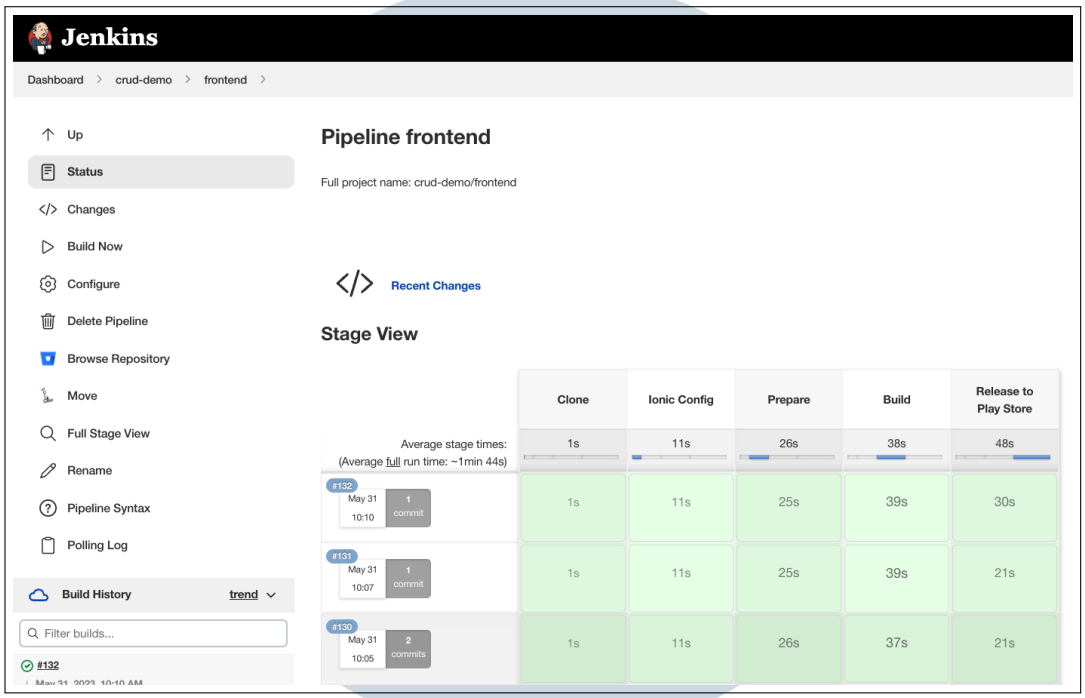
Halaman *login* ini *user* dapat melakukan *login* sesuai dengan *username* dan *password* yang disediakan oleh API *client*. Jika *username* atau *password* tidak sesuai maka akan muncul *error message* berwarna merah seperti pada gambar diatas.




Gambar 3.2. Tampilan halaman *employee detail AHM*

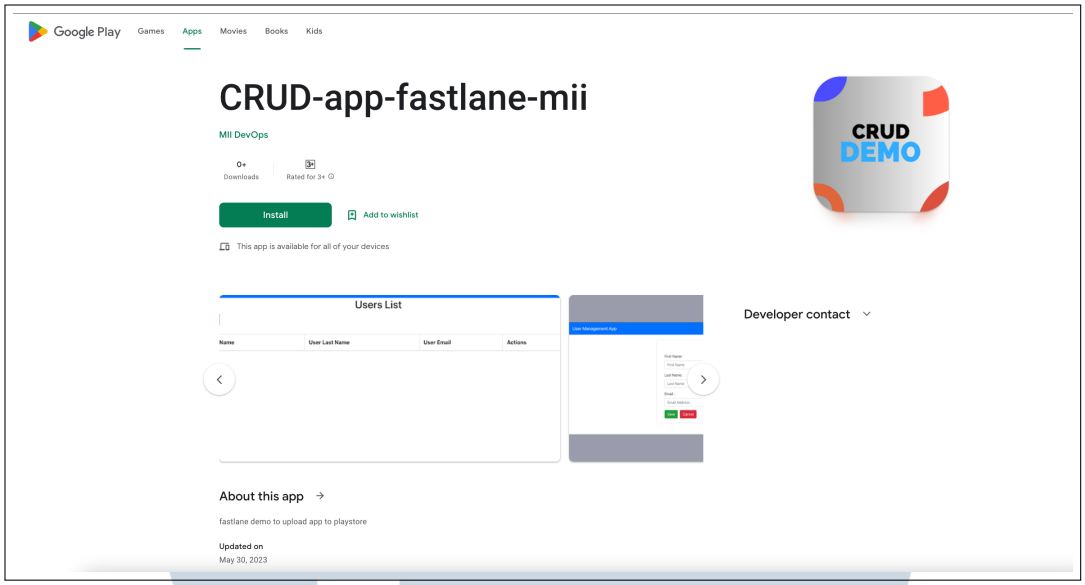
Halaman *employee detail* ini merupakan hasil dari API yang didapatkan setelah *login*. Data-data yang ditampilkan seperti NRP, nama, gender, tanggal kelahiran, agama, kota, alamat, dan kewarganegaraan.

### 3.4.2 Tampilan Jenkins *Pipeline Fastlane*



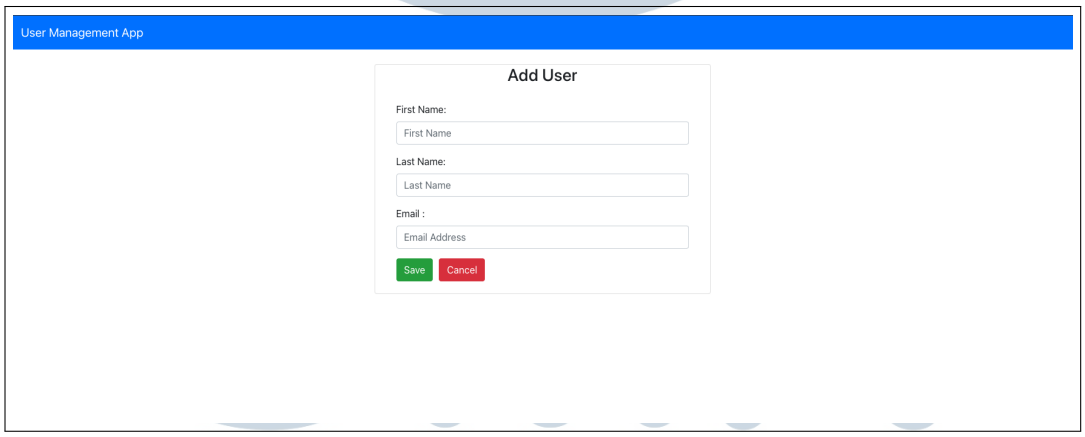

Gambar 3.3. Tampilan halaman *jenkins pipeline cloud*

Halaman *jenkins* ini berada pada server lab perusahaan PT. Mitra Integrasi Infromatika yang dapat diakses menggunakan vpn dan ip yang disediakan. Pada halaman tersebut terdapat status berwarna hijau dari *build* aplikasi yang dijalankan pada server *jenkins*. Terdapat 5 *stage* yang dijalankan oleh *jenkins*. *Stage clone* untuk melakukan *clone* repositori. *Stage ionic config* untuk melakukan konfigurasi *capacitor* dan konversi aplikasi web menjadi *android*. *Stage prepare* untuk melakukan instalasi dependensi yang diperlukan. *Stage Build* untuk melakukan proses *build* aplikasi menjadi bentuk *bundle* atau .aab. Terakhir, *stage release to play store* untuk melakukan rilis aplikasi ke playstore menggunakan *tools* dari *fastlane* yang dijalankan pada server *jenkins*.




Gambar 3.4. Tampilan halaman playstore demo app yang di *upload* ke google *play store*

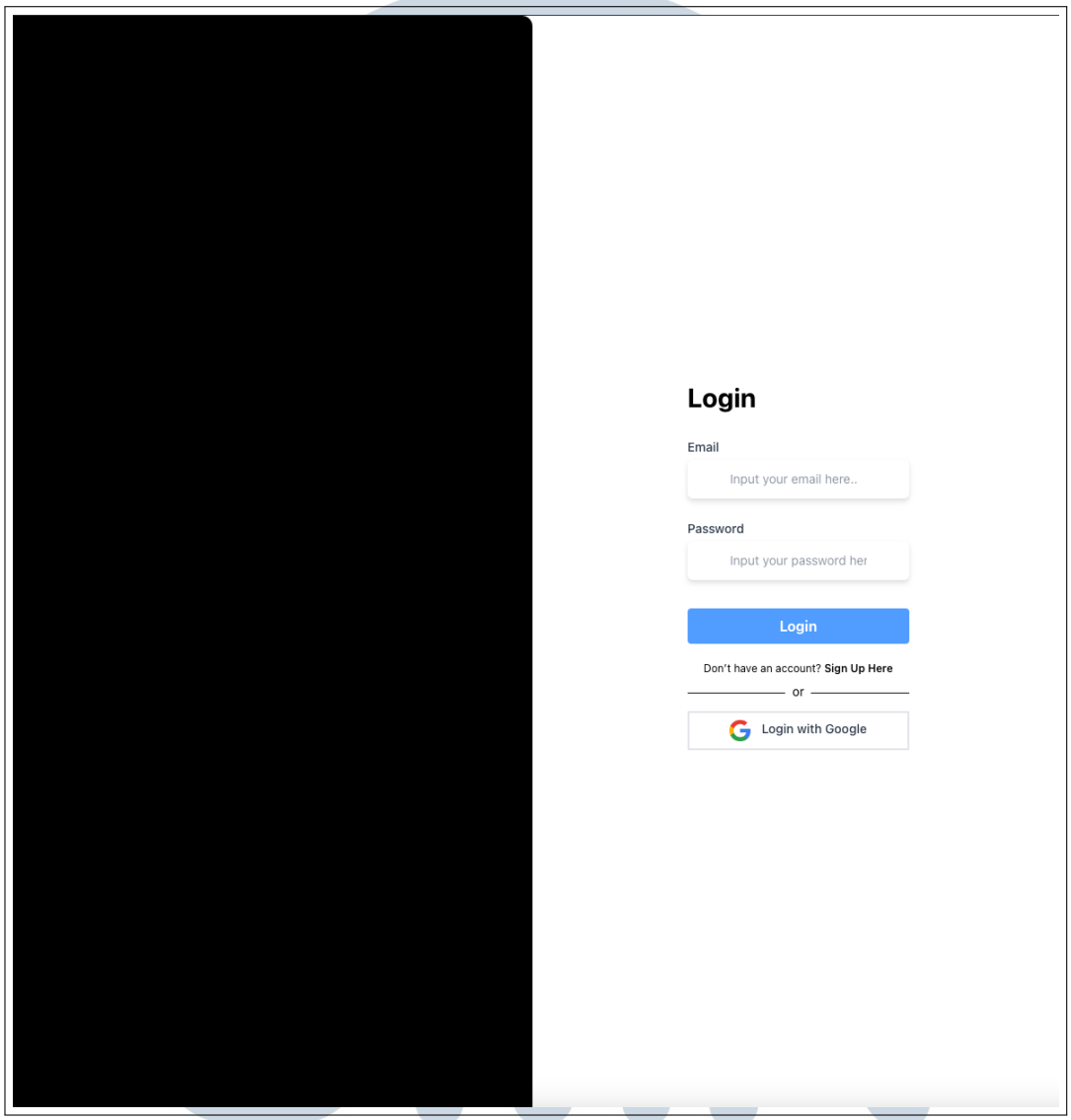
Berikut merupakan aplikasi yang telah diupload ke playstore. Aplikasi tersebut dapat diakses dengan link berikut https://play.google.com/store/apps/details?id=io.ionic.CRUD_App




Gambar 3.5. Tampilan website *react.js* yang digunakan

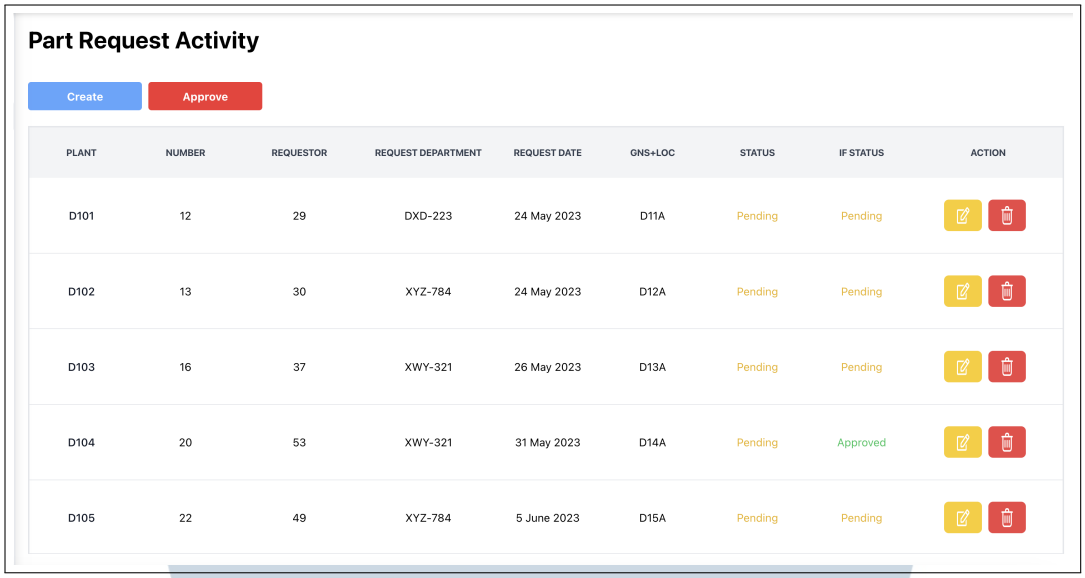
Pada halaman aplikasi, terdapat form untuk add *user* namun hanya berbentuk statis dan tidak ada backendnya. Karena tim DevOps hanya ingin mengetahui bagaimana proses upload aplikasi menggunakan *fastlane* dan *jenkins*.

### 3.4.3 Tampilan Aplikasi *ERP inventory system*



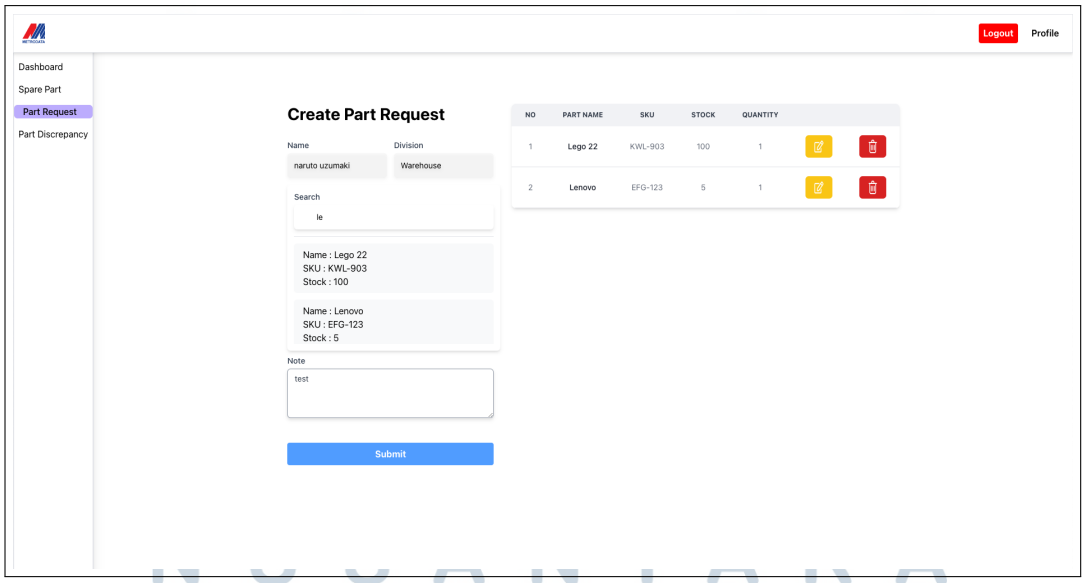

Gambar 3.6. Tampilan halaman *login* dari *inventory system*

Tampilan halaman *login* ini *user* diminta untuk memasukkan *email* dan *password*, apabila *password* dan *username*-nya sudah sesuai maka API akan mengirimkan kembali token akses. Jika *user* tidak memiliki akun maka *user* dapat melakukan *register* pada *button Sign Up Here*.

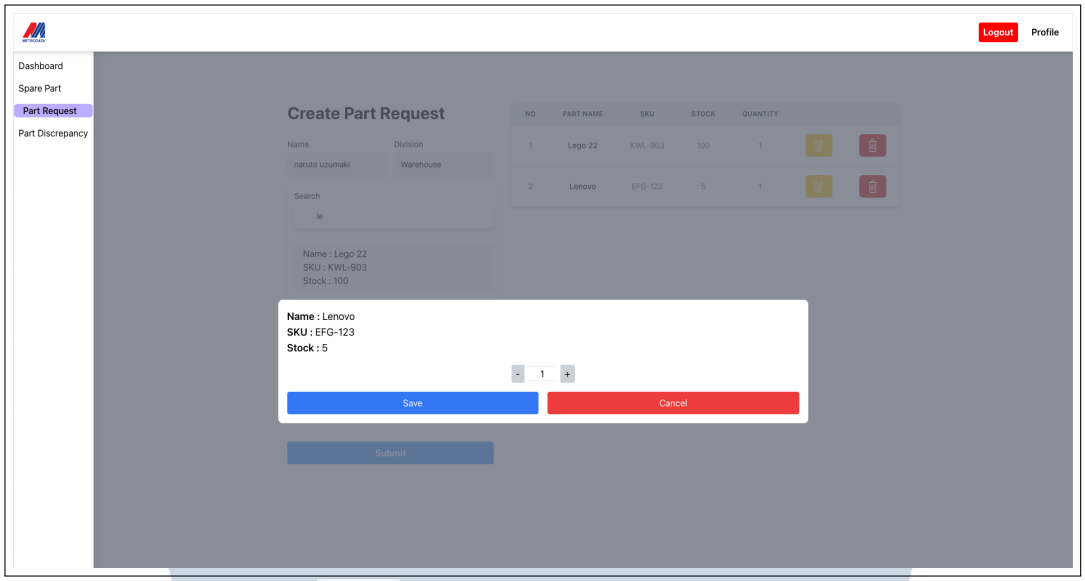



Gambar 3.7. Tampilan Halaman *Dashboard Part Request*

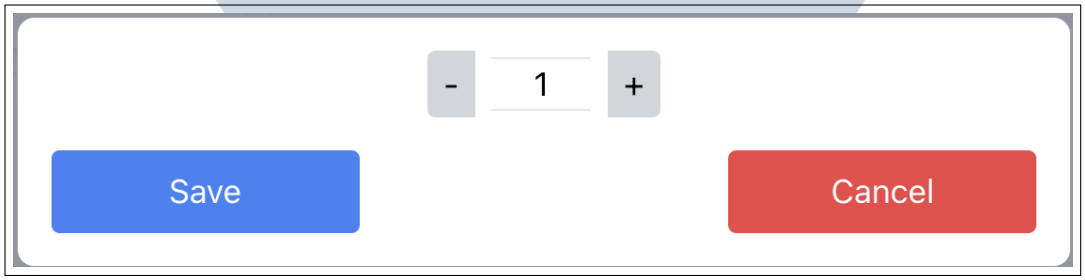
Pada halaman *dashboard part request user* dapat melihat semua data *part request* yang telah dibuat oleh *user* tersebut. *User* dapat membuat sebuah *part request* dengan menggunakan *button Create* seperti pada gambar. *User* dengan *role* admin atau *warehouse manager*, dapat melakukan *approval part request* menggunakan *button Approve*. *User* dapat melakukan *delete* dan *update part request* menggunakan *button* yang ada pada kolom *action*.




Gambar 3.8. Tampilan halaman *Create Part Request*

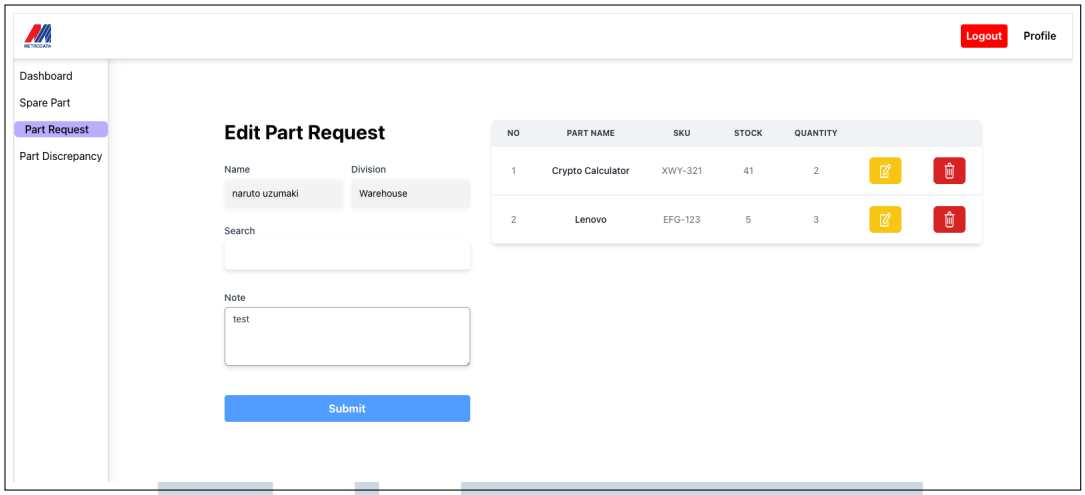



Gambar 3.9. Tampilan Modal *Add Data* ke Tabel *Create Part Request*



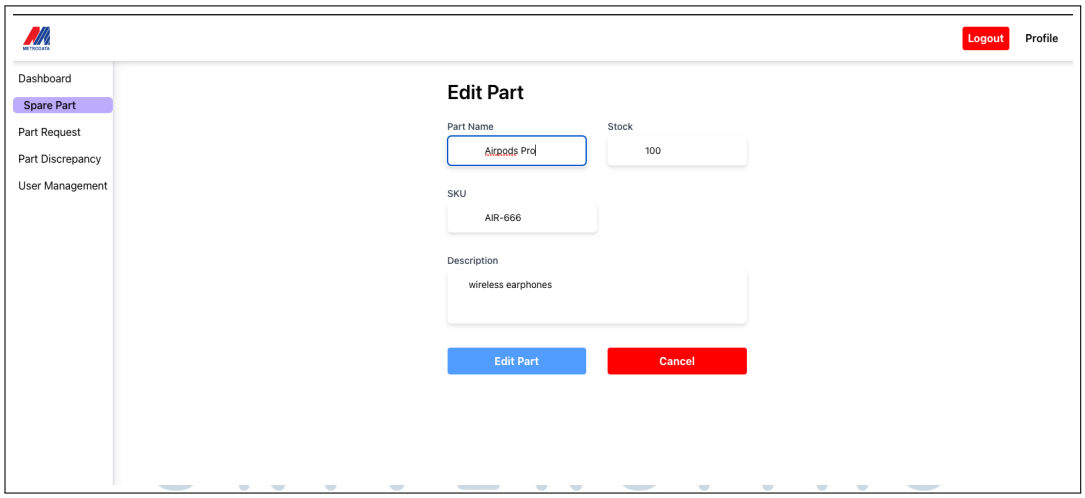

Gambar 3.10. Tampilan Modal *Edit Quantity* pada Tabel *Create Part Request*

Pada halaman *create part request*, *user* dapat melakukan permintaan atas suatu barang / produk. *User* dapat mencari barang yang diinginkan pada *Searchbar*, menuliskan *note*, dan melakukan *edit* maupun *delete* terhadap barang yang diminta. Jika *user* memilih salah satu barang pada *search* maka akan muncul modal seperti pada gambar 3.9 yang menunjukan detail barangnya dengan nama, sku, dan stok. *User* dapat memasukkan jumlah barang yang ingin diminta dengan menggunakan *button*- atau + dan *user* juga bisa melakukan input langsung jumlahnya. Lalu *user* dapat melakukan *save* jika ingin menambahkan barang tersebut ke dalam tabel *create part request* yang ada di samping kanan pada gambar 3.8. Jika *user* memilih *cancel* maka barang tidak akan dimasukkan ke tabel *create part request*. Pada gambar 3.10, *user* dapat melakukan *edit quantity* terhadap barang yang sudah dimasukkan ke dalam tabel dengan menggunakan *button*- atau + dan *user* juga bisa melakukan input langsung jumlahnya. Setelah itu, jika *user* melakukan submit akan muncul animasi sukses ataupun *error* bergantung pada *response* dari API.



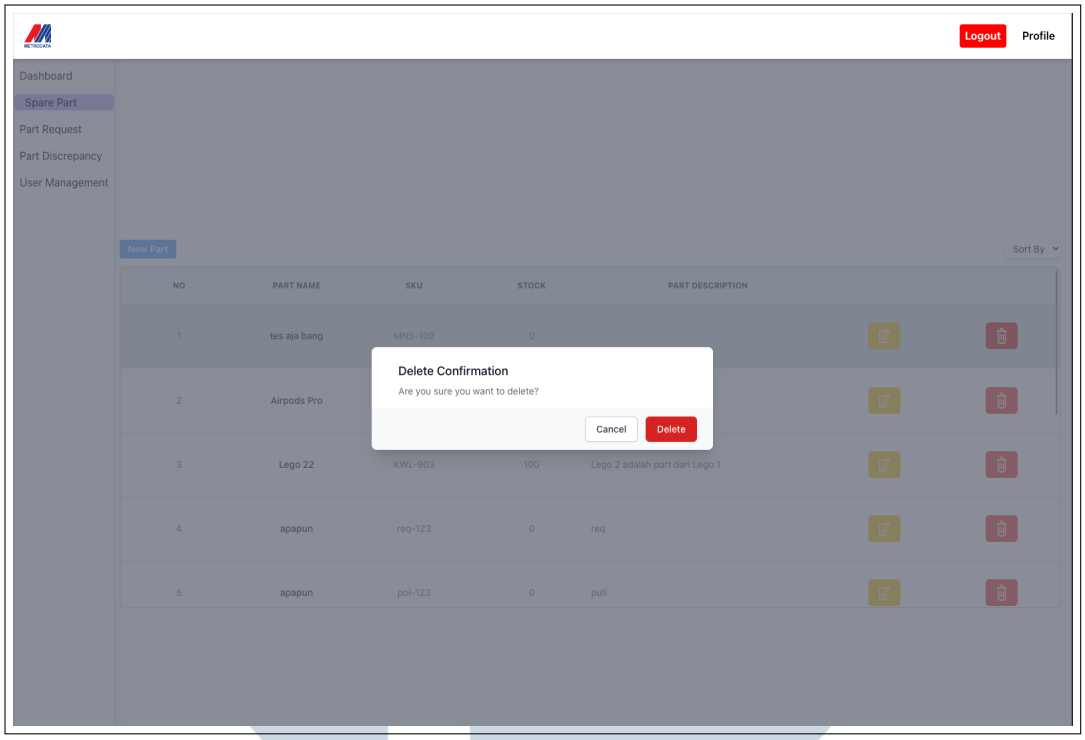

Gambar 3.11. Tampilan Halaman *Edit Part Request*

Pada halaman *edit part request* memanggil API data *part request by id* yang dipilih id-nya oleh *user* pada halaman *dashboard part request* untuk menampilkan data-data yang telah di-*request* sebelumnya dan *user* dapat melakukan *edit* jumlah barang yang diminta, mengubah *note*, menghapus barang yang tidak jadi di *request*, dan juga menambahkan barang lain yang diminta. Jika sudah melakukan *update*, maka *user* dapat melakukan *submit* ke API *edit part request*.




Gambar 3.12. Tampilan Halaman *Edit Spare Part*

Halaman edit *spare part* memanggil API data *spare part by id* yang telah dipilih pada *dashboard spare part*. Lalu dari API tersebut menampilkan part name, stock, sku dan description dari barang yang ingin di edit. Jika *user* sudah melakukan update *spare part*-nya, *user* dapat meng-click *button edit part* untuk melakukan submit ke API edit *spare part*.




Gambar 3.13. Tampilan Modal *Delete Confirmation*

Modal *delete confirmation* diterapkan pada *dashboard part request*, dimana pada *dashboard* tersebut terdapat *list* tabel data *part request* yang ada, saat *buttondelete* dipilih maka akan muncul modal *delete confirmation* seperti pada gambar diatas.

### 3.4.4 Potongan Kode Aplikasi *ERP inventory system*

### A. Kode Halaman Utama *Create Part Request*

```
const CreatePartRequest = () => {
    return (
        <Layout>
        {isLoading && (
            <SubmitLoadingSpinner/>
        )}
        <div className="container md:pl-40 flex lg:flex-row flex-col pt-32 justify-center">
          <div className="lg:basis-1/3 pl-3">
            <form name="create-part-request" onSubmit={(e) => { CreatePartRequestAction(e)}}>
```

```
10                <h1 className="text-3xl font-bold mb-4 text-start
    ">Create Part Request</h1>
11                <div className="sm:grid grid-cols-2">
12                    {/* NAME INPUT */}
13                    {/* ////////// */}
14                    <div className="py-3 pr-3">
15                        <label className="mb-1 block text-sm font-
    medium text-slate-700">Name</label>
16                        <input
17                            className="py-3 px-3 border-0 border-
    gray-200 drop-shadow-md rounded-md text-sm w-full "
18                            value={user.User}
19                            disabled
20                            onChange={(e)=>{onchange(e.target.
    value)}}/>
21                    </div>
22                    {/* DIVISION */}
23                    <div className="py-3 pr-3">
24                    <label className="mb-1 block text-sm font-
    medium text-slate-700">Division</label>
25                        <input
26                            className="py-3 px-3 border-0 border-
    gray-200 drop-shadow-md rounded-md text-sm w-full "
27                            value={user.DivisionName}
28                            disabled
29                            onChange={(e)=>{onchange(e.target.
    value)}}/>
30                    </div>
31                </div>
32                {/* SEARCH */}
33                <div className="py-3 pr-3">
34                    <div className="search bg-white shadow-md
    rounded-md pl-3 pr-3">
35
36                        <label className="mb-1 block text-sm font-
    medium text-slate-700">Search</label>
37                            <input
38                                className="py-3 pl-10 pr-3 border
    -0 border-gray-200 shadow-md rounded-md text-sm w-full"
39                                value={query}
40                                onChange={handleChange}
41                            />
42                        <div className="search-results overflow-y-
```

```
auto max-h-48 mt-3 border-t border-gray-200">
                    {data.map((item, index) => (
                        <div
                            className="text-left border-black
border-ridge border-1 rounded-md my-3 px-4 py-2 w-full bg-gray
-50"
                            key={index}
                            onClick={() =>
handleAddDataToTable(item)}
                        >
                            <p>Name  : {item.part_name}</p>
                            <p>SKU   : {item.sku}</p>
                            <p>Stock : {item.stock}</p>
                        </div>
                    ))}
                    </div>
                </div>
            </div>

            {/* NOTE */}
            <div className="flex items-center flex-row">
                <div className="w-full py-2 pr-3 text-lg
rounded-lg">
                    <label className="mb-1 block text-sm font-
medium text-slate-700">Note</label>
                    <textarea
                        className="border border-gray-400
rounded w-full py-2 px-3 text-sm text-gray-700 leading-tight
drop-shadow-md rounded-md focus:outline-none focus:shadow-
outline resize-none"
                        id="note"
                        rows="4"
                        value={note}
                        onChange={(e) => setNote(e.target.
value)}
                    ></textarea>
                </div>
            </div>
            { error.note &&
                <span className="text-xs text-red-500 italic
">{error.note}</span>
            }

```

```
75                {/* BUTTON SUBMIT */}
76                <div className='flex gap-4 pr-3 md:gap-8'>
77                  <Button text={"Submit"} type={"submit"} disabled
    ={isSubmitting} classname={styleSubmitButton} />
78                  <Button text={"Cancel"} type={'button'}
    classname={styleCancelButton} onclick={()=>{navigate('/
    partdiscrepancy')}}/>
79                </div>
80
81              </form>
82            </div>
83            <div className="lg:basis-2/3 mx-3 my-8">
84                {/* TABLE DATA PART Discrepancy */}
85                <div className="relative overflow-x-auto overflow-
    y-auto max-h-[34rem] shadow-md">
86                <PartTable dataTable={dataTable} handleEditTable={
    handleEditTable} handleDeleteTable={handleDeleteTable} />
87                {modalOpen && (
88                  <ModalComponent
89                      newQuantity={newQuantity}
90                      setNewQuantity={setNewQuantity}
91                      setModalOpen={setModalOpen}
92                      handleMinus={handleMinus}
93                      handlePlus={handlePlus}
94                      handleSave={handleSave}
95                  />
96                  )
97                }
98                {modalOpenPart && (
99                  <ModalComponentPart
100                    selectedItemPart={selectedItemPart}
101                    setNewQuantity={setNewQuantity}
102                    newQuantity={newQuantity}
103                    setModalOpenPart={setModalOpenPart}
104                    handleMinus={handleMinus}
105                    handlePlus={handlePlus}
106                    handleSaveDataToTable={handleSaveDataToTable}
107                  />
108                )}
109
110                </div>
111                { error.tableData &&
112                    <span className="text-xs text-red-500 italic
```

```
    ">{error.tableData}</span>
113                 }
114             </div>
115         </div>
116         <SuccessCheckmark showSuccess={showSuccess} message={
    message}/>
117         <FailureCheckmark showFailure={showFailure} message={
    message}/>
118       </Layout>
119     )
120 }
121
122 export default CreatePartRequest;
```

Kode 3.1: Potongan kode halaman utama *create part request*

Berikut merupakan potongan kode dari gambar 3.7 yang merupakan halaman utama *create part request*. Kode tersebut memanggil 5 komponen utama yaitu komponen *PartTable* yang merupakan komponen untuk menampilkan tabel yang berada pada bagian kanan halaman. Komponen *ModalComponent* untuk modal *edit quantity* atau jumlah barang pada komponen *PartTable*. Komponen *ModalComponentPart* untuk modal *add data* ke tabel dan menampilkan detail dari barang yang dipilih. Komponen *SuccessCheckmark* untuk menampilkan modal animasi sukses saat operasi *create part request* berhasil di *submit* ke API. Dan terakhir, komponen *FailureCheckmark* untuk menampilkan modal animasi *error* saat operasi *create part request* gagal meng-*submit* ke API.

## B. Kode Komponen *PartTable* Pada Halaman *Create Part Request* dan *Edit Part Request*

```
1  import React from 'react';
2
3  const PartTable = ({ dataTable, handleEditTable, handleDeleteTable
       , part }) => {
4    return (
5      <table className="w-full text-sm text-gray-500 text-center">
6        {/* table headers */}
7        <thead className="text-xs text-gray-700 uppercase bg-gray
       -100 sticky top-0">
8          <tr>
9            <th scope="col" className="px-3 py-6 w-20">
10             Part Code
```

```
          </th>
          <th scope="col" className="px-3 py-6 w-20">
            Part Name
          </th>
          <th scope="col" className="px-3 py-6 w-20">
            Req Qty
          </th>
          <th scope="col" className="px-3 py-6 w-20">
            Sys Qty
          </th>
          <th scope="col" className="px-3 py-6 w-20">
            Prod Req (N+1 Shift)
          </th>
          <th scope="col" className="px-3 py-6 w-20">
            Remark
          </th>
          <th scope="col" className="px-3 py-6 w-20">
            Request
          </th>
          <th scope="col" className="px-3 py-6 w-20">
            Action
          </th>
        </tr>
      </thead>
      {/* table body */}
      <tbody>
        {dataTable && dataTable.length > 0 ? (
            dataTable.map((item, index) => (
                <tr key={index} className="bg-white border-b hover
:bg-gray-200">
                    <td className="px-6 py-4">{item.sku}</td>
                    <th scope="row" className="px-6 py-4 font-
medium text-gray-900 whitespace-nowrap">
                        {item.part_name}
                    </th>
                    <td className="px-6 py-4">{item.quantity}</td>
                    <td className="px-6 py-4">
                        {
                            part !== undefined ? (
                                part.find(parts => parts.part_name
 === item.part_name)?.stock ?? 'Not found'
                            ) : (
                                item.stock
```

```
                                )
                            }
                        </td>
                        <td className="px-6 py-4"></td>
                        <td className="px-6 py-4"></td>
                        <td className="px-6 py-4"></td>
                        <td className="px-6 py-4">
                            <div className='flex gap-2 justify-center
    items-center'>
                                <button className="inline-flex items-
    center px-4 py-2 w-12 h-10 bg-yellow-400 hover:bg-yellow-500
    text-white text-sm font-medium rounded-md"
                                    onClick={(e) => handleEditTable(e,
     index)}>
                                    <img src="/images/edit_icon.png"
    className="h-full w-full" alt="Edit" />
                                </button>
                                <button className="inline-flex items-
    center px-4 py-2 w-12 h-10 bg-red-600 hover:bg-red-700 text-
    white text-sm font-medium rounded-md"
                                    onClick={(e) => handleDeleteTable(
    e, index)}>
                                    <img src="/images/delete_icon.png"
     className="h-full w-full" alt="Trash" />
                                </button>
                            </div>
                        </td>
                    </tr>
                ))
            ) : (
                <p></p>
            )}
        </tbody>
    </table>
  );
};

export default PartTable;
```

Kode 3.2: Potongan kode komponen *PartTable*

Kode diatas merupakan kode komponen tabel yang diterapkan pada halaman utama *create part request* dan *edit part request*. Komponen tersebut menerima 4 nilai yang di *passing* dari halaman utama yaitu dataTable yang berisi hasil

dari barang yang dimasukkan ke dalam tabel, handleEditTable untuk melakukan edit jumlah barang yang diminta, handleDeleteTable untuk menghapus data yang diinput pada tabel, dan part untuk menampilkan barang-barang tersedia yang didapatkan dari API.

### C. Kode Komponen Modal *Add* Data ke Tabel *Part Request*

```
import React from 'react';

const ModalComponentPart = ({selectedItemPart, setNewQuantity, newQuantity, setModalOpenPart, handleMinus, handlePlus, handleSaveDataToTable}) => {
 return (
     <div
         className="fixed inset-0 bg-gray-500 bg-opacity-75 flex items-center justify-center"
         onClick={() => setModalOpenPart(false)}
       >
         <div></div>
         <div className="bg-white p-4 rounded-lg md:w-1/3" onClick={(e) => e.stopPropagation()}>
             {
                 selectedItemPart !== null && (
                     <div className="mb-4">
                         <p className="font-semibold text-lg">Name  : <span className="font-normal">{selectedItemPart.part_name}</span></p>
                         <p className="font-semibold text-lg">SKU  : <span className="font-normal">{selectedItemPart.sku}</span></p>
                         <p className="font-semibold text-lg">Stock : <span className="font-normal">{selectedItemPart.stock}</span></p>
                     </div>
                 )
             }
         <div className="flex items-center justify-center mb-4">
             <button
               className="px-2 py-1 bg-gray-300 rounded-l focus:outline-none"
               onClick={handleMinus}
```

```
                disabled={parseInt(newQuantity) <= 1} // Disable
 the button when newQuantity is less than or equal to 1
              >
                -
              </button>
              <input
                className="mx-2 w-12 text-center border-t border
-b"
                type="number"
                //disabled
                min="1"
                max="1000"
                value={newQuantity}
                onChange={(e) => {
                  const newValue = parseInt(e.target.value);
                  if (newValue >= 1) {
                    setNewQuantity(newValue);
                  }
                }}
              />
              <button
                className="px-2 py-1 bg-gray-300 rounded-r focus
:outline-none"
                onClick={handlePlus}
              >
                +
              </button>
            </div>
            <div className="mt-4 flex justify-between space-x-4">
              <button
                className="bg-blue-500 text-white px-4 py-2
rounded md:w-full lg:w-1/3"
                onClick={handleSaveDataToTable}
              >
                Save
              </button>
              <div className="lg:w-1/3"></div>
              <button
                className="bg-red-500 text-white px-4 py-2
rounded md:w-full lg:w-1/3"
                onClick={() => setModalOpenPart(false)}
              >
                Cancel
```

```
                </button>
            </div>
            </div>
        </div>
    );
}

export default ModalComponentPart;
```

Kode 3.3: Potongan kode modal *add* data ke tabel

Kode diatas merupakan kode komponen modal part. Modal ini ditujukan untuk menampilkan detail dari barang yang dipilih pada fitur search pada halaman utama *create part request*. *User* dapat menentukan jumlah barang yang diminta dengan menggunakan tombol + atau - seperti pada gambar 3.9.

### D. Kode Komponen Modal *Edit Quantity* pada Tabel *Create Part Request*

```
import React from 'react';

const ModalComponent = ({newQuantity, setNewQuantity, setModalOpen, handleMinus, handlePlus, handleSave}) => {
    return (
        <div
        className="fixed inset-0 bg-gray-500 bg-opacity-75 flex items-center justify-center"
        onClick={() => setModalOpen(false)}
      >
        <div className="bg-white p-4 rounded-lg md:w-1/3" onClick={(e) => e.stopPropagation()}>
          <div className="flex items-center justify-center mb-4">
            <button
              className="px-2 py-1 bg-gray-300 rounded-l focus:outline-none"
              onClick={handleMinus}
              disabled={parseInt(newQuantity) <= 1} // Disable the button when newQuantity is less than or equal to 1
            >
              -
            </button>
            <input
              className="mx-2 w-12 text-center border-t border-b"
              type="number"
```

```
                value={newQuantity}
                //disabled
                min="1"
                max="1000"
                onChange={(e) => {
                  const newValue = parseInt(e.target.value);
                  if (newValue >= 1) {
                    setNewQuantity(newValue);
                  }
                }}
              />
              <button
                className="px-2 py-1 bg-gray-300 rounded-r focus:outline-none"
                onClick={handlePlus}
              >
                +
              </button>
            </div>
            <div className="mt-4 flex justify-between space-x-4">
                <button
                  className="bg-blue-500 text-white px-4 py-2 rounded md:w-full lg:w-1/3"
                  onClick={handleSave}
                >
                  Save
                </button>
                <div className="lg:w-1/3"></div>
                <button
                  className="bg-red-500 text-white px-4 py-2 rounded md:w-full lg:w-1/3"
                  onClick={() => setModalOpen(false)}
                >
                  Cancel
                </button>
            </div>
          </div>
        </div>
    );
}

export default ModalComponent;
```

Kode 3.4: Potongan kode modal edit quantity

Kode diatas merupakan kode komponen modal *edit quantity*. Modal ini ditujukan untuk melakukan *edit* terhadap *quantity* barang pada tabel *create part request* seperti pada gambar 3.10.

## E. Kode Komponen Modal Animasi *Success Checkmark*

```
import React from 'react';
import '../css/SuccessCheckmark.css'

const SuccessCheckmark = ({ showSuccess, message }) => {
  if (!showSuccess) {
    return null;
  }

  return (
    <div className="max-w-sm rounded overflow-hidden shadow-lg bg-custom-gray fixed top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2 z-50">
        <div className="success-checkmark pt-10">
            <div className="check-icon">
                <span className="icon-line line-tip"></span>
                <span className="icon-line line-long"></span>
                <div className="icon-circle"></div>
                <div className="icon-fix"></div>
            </div>
        </div>
        <div className="px-6 py-4">
            <div class="font-bold text-xl mb-2 pt-5 ml-2 text-green-500">SUCCESS</div>
            <div className="text-md mb-2  ml-2 text-green-500 justify-center item-center">{message}</div>
        </div>
    </div>
  );
};

export default SuccessCheckmark;
```

Kode 3.5: Potongan kode komponen modal *SuccessCheckmark*

```
.success-checkmark {
    width: 80px;
    height: 115px;
```

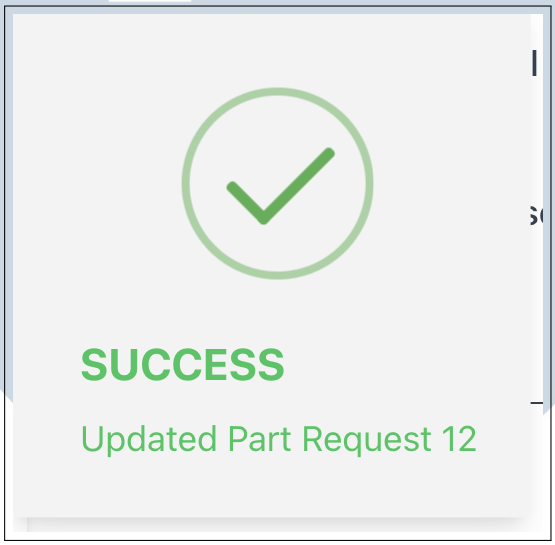
```
    margin: 0 auto;
}

.success-checkmark .check-icon {
    width: 80px;
    height: 80px;
    position: relative;
    border-radius: 50%;
    box-sizing: content-box;
    border: 4px solid #4CAF50;
}

.success-checkmark .check-icon::before,
.success-checkmark .check-icon::after {
    content: '';
    height: 100px;
    position: absolute;
    background: #f3f3f3;
    transform: rotate(-45deg);
}

.success-checkmark .check-icon::before {
    top: 3px;
    left: -2px;
    width: 30px;
    transform-origin: 100% 50%;
    border-radius: 100px 0 0 100px;
}

.success-checkmark .check-icon::after {
    top: 0;
    left: 30px;
    width: 60px;
    transform-origin: 0 50%;
    border-radius: 0 100px 100px 0;
    animation: rotate-circle 4.25s ease-in;
}

.success-checkmark .check-icon .icon-line {
    height: 5px;
    background-color: #4CAF50;
    display: block;
    border-radius: 2px;
```

```
    position: absolute;
    z-index: 10;
}

.success-checkmark .check-icon .icon-line.line-tip {
    top: 46px;
    left: 14px;
    width: 25px;
    transform: rotate(45deg);
    animation: icon-line-tip 0.75s;
}

.success-checkmark .check-icon .icon-line.line-long {
    top: 38px;
    right: 8px;
    width: 47px;
    transform: rotate(-45deg);
    animation: icon-line-long 0.75s;
}

.success-checkmark .check-icon .icon-circle {
    top: -4px;
    left: -4px;
    z-index: 10;
    width: 80px;
    height: 80px;
    border-radius: 50%;
    position: absolute;
    box-sizing: content-box;
    border: 4px solid rgba(76, 175, 80, .5);
}

.success-checkmark .check-icon .icon-fix {
    top: 8px;
    width: 5px;
    left: 26px;
    z-index: 1;
    height: 85px;
    position: absolute;
    transform: rotate(-45deg);
    background-color: #f3f3f3;
}

```

```
@keyframes rotate-circle {
    0%, 5% {
        transform: rotate(-45deg);
    }
    12%, 100% {
        transform: rotate(-405deg);
    }
}

@keyframes icon-line-tip {
    0%, 54% {
        width: 0;
        left: 1px;
        top: 19px;
    }
    70% {
        width: 50px;
        left: -8px;
        top: 37px;
    }
    84% {
        width: 17px;
        left: 21px;
        top: 48px;
    }
    100% {
        width: 25px;
        left: 14px;
        top: 45px;
    }
}

@keyframes icon-line-long {
    0%, 65% {
        width: 0;
        right: 46px;
        top: 54px;
    }
    84% {
        width: 55px;
        right: 0px;
        top: 35px;
    }
```

```
    100% {
        width: 47px;
        right: 8px;
        top: 38px;
    }
}
```

Kode 3.6: Potongan kode styling css animasi komponen *SuccessCheckmark*




Gambar 3.14. Tampilan modal animasi sukses

Potongan kode 3.5 tersebut ditujukan untuk membuat modal animasi sukses dan dapat menerapkan *custom message* saat *user* berhasil melakukan *submit* menggunakan variabel bernama *message*. Potongan kode 3.6 diatas merupakan *styling css* untuk membuat animasi sukses. Css ini diterapkan pada komponen *SuccessCheckmark*.

## F. Kode Komponen Modal Animasi *Failure Checkmark*

```
import React from 'react';
import '../css/FailureCheckmark.css'

const FailureCheckmark = ({ showFailure, message }) => {
  if (!showFailure) {
    return null;
  }

  return (
```

```
    <div className="flex flex-col max-w-sm rounded overflow-hidden shadow-lg bg-custom-gray fixed top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2 z-50 pt-8">
        <div className="flex flex-col justify-center items-center h-full">
            <div className="circle-border-error">
            </div>
            <div className="circle-error">
                <div className="error flex justify-center items-center"></div>
            </div>
        </div>

    <div className="px-6 py-4 flex flex-col justify-center text-center">
        <div className="font-bold text-xl mb-2 pt-5 ml-2 text-red-400 justify-center item-center">OOPS!</div>
        <div className="font-bold text-xl mb-2 ml-2 text-red-400 justify-center item-center">Something Went Wrong</div>
        <div className="text-md mb-2  ml-2 text-red-400 justify-center item-center">{message}</div>
        <div className="text-md mb-2  ml-2 text-red-400 justify-center item-center">Please try again</div>
    </div>
</div>
  );
};

export default FailureCheckmark;
```

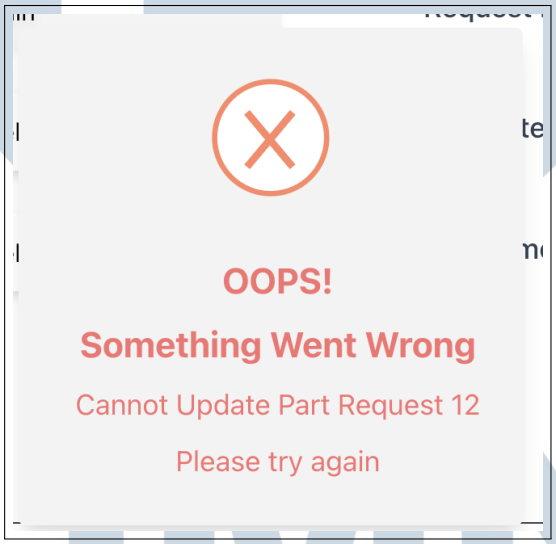
Kode 3.7: Potongan kode modal *Failure Checkmark*

```
.circle-error,
.circle-border-error {
  width: 60px;
  height: 60px;
  border-radius: 50%;
}

.circle-error {
  z-index: 1;
  position: relative;
  background: white;
```

```
    transform: scale(1);
    animation: success-anim 700ms ease;
  }

  .circle-border-error {
    z-index: 0;
    position: absolute;
    transform: scale(1.1);
    animation: circle-anim 400ms ease;
    background: #f86;
  }

  @keyframes success-anim {
    0% {
      transform: scale(0);
    }
    30% {
      transform: scale(0);
    }
    100% {
      transform: scale(1);
    }
  }

  @keyframes circle-anim {
    from {
      transform: scale(0);
    }
    to {
      transform: scale(1.1);
    }
  }

  .error::before,
  .error::after {
    content: "";
    display: block;
    height: 4px;
    background: #f86;
    position: absolute;
  }

  .error::before {
```

```
    width: 40px;
    top: 48%;
    left: 16%;
    transform: rotateZ(50deg);
  }

  .error::after {
    width: 40px;
    top: 48%;
    left: 16%;
    transform: rotateZ(-50deg);
  }
```

Kode 3.8: Potongan kode styling css animasi *Failure Checkmark*




Gambar 3.15. Tampilan modal animasi *error*

Potongan kode 3.7 tersebut ditujukan untuk membuat modal animasi *error* dan dapat menerapkan *custom message* saat *user* gagal dalam proses *submit* menggunakan variabel bernama *message*. Potongan kode 3.8 diatas merupakan *styling css* untuk membuat animasi *error*. Css ini diterapkan pada komponen *FailureCheckmark*.

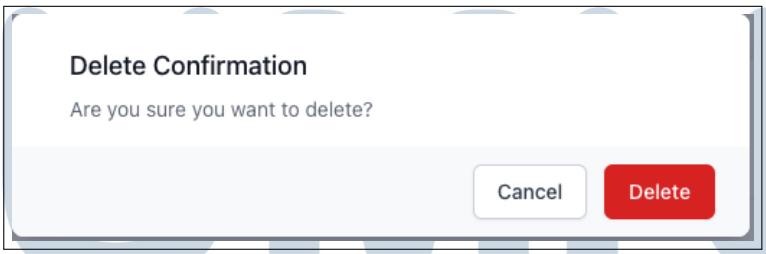
## G. Kode Komponen Modal *Delete Confirmation*

```
import React from 'react'

const DeleteModal = ({isOpen, onConfirm, onCancel}) => {
```

```
4     if (!isOpen) return null;
5
6     return (
7         <div className='fixed z-10 inset-0 overflow-y-auto'>
8         <div className='flex items-end justify-center min-h-screen
      pt-4 px-4 pb-20 text-center sm:block sm:p-0'>
9           <div className='fixed inset-0 transition-opacity' aria-
     hidden='true'>
10            <div className='absolute inset-0 bg-gray-500 opacity
     -75'></div>
11          </div>
12          <span className='hidden sm:inline-block sm:align-middle
     sm:h-screen' aria-hidden='true'>&#8203;</span>
13          <div className='inline-block align-bottom bg-white
     rounded-lg text-left overflow-hidden shadow-xl transform
     transition-all sm:my-8 sm:align-middle sm:max-w-lg sm:w-full'>
14            <div className='bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb
     -4'>
15              <div className='sm:flex sm:items-start'>
16                <div className='mt-3 text-center sm:mt-0 sm:ml-4
     sm:text-left'>
17                  <h3 className='text-lg leading-6 font-medium
     text-gray-900' id='modal-title'>
18                    Delete Confirmation
19                  </h3>
20                  <div className='mt-2'>
21                    <p className='text-sm text-gray-500'>
22                      Are you sure you want to delete?
23                    </p>
24                  </div>
25                </div>
26              </div>
27            </div>
28            <div className='bg-gray-50 px-4 py-3 sm:px-6 sm:flex
     sm:flex-row-reverse'>
29              <button
30                type='button'
31                className='w-full inline-flex justify-center
     rounded-md border border-transparent shadow-sm px-4 py-2 bg-red
     -600 text-base font-medium text-white hover:bg-red-700 focus:
     outline-none focus:ring-2 focus:ring-offset-2 focus:ring-red
     -500 sm:ml-3 sm:w-auto sm:text-sm'
32                onClick={onConfirm}
```

```
                >
                  Delete
                </button>
                <button
                  type='button'
                  className='mt-3 w-full inline-flex justify-center
    rounded-md border border-gray-300 shadow-sm px-4 py-2 bg-white
    text-base font-medium text-gray-700 hover:bg-gray-50 focus:
    outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo
    -500 sm:mt-0 sm:ml-3 sm:w-auto sm:text-sm'
                  onClick={onCancel}
                >
                  Cancel
                </button>
              </div>
            </div>
          </div>
        </div>
      );
    }

export default DeleteModal;
```

Kode 3.9: Potongan komponen modal *delete confirmation*




Gambar 3.16. Tampilan modal *delete confirmation*

Potongan kode diatas merupakan potongan kode untuk modal *confirmation delete* pada *dashboard part request*. Dimana jika *user* memilih *button Delete* maka akan menjalankan *onConfirm* dan menghapus data tersebut dengan memanggil API *delete*. Sedangkan, jika tidak jadi melakukan *delete* maka akan menjalankan onCancel untuk membatalkan aktivitas *delete*.

### 3.4.5 Potongan Kode Jenkins *Pipeline* dan *Fastlane*

#### A. Kode File *fastfile* Pada Direktori *Fastlane*

```
lane :build do |options| # build lane
        update_build_number
        build_bundle
    end

lane :release do |options| # release lane to play store
        upload_to_play_store(
          track: 'internal',
          release_status: "draft",
        )
    end

lane :build_bundle do
        download_dependencies # download android dependencies
        android_set_version_code( # set android version code +1
          version_code: fetch_build_number.to_i
        )
        android_set_version_name(
          version_name: "1.1.0C"
        )

        gradle(task: 'clean')
        gradle(
          task: 'bundle', # bundle
          build_type: 'Release',
          print_command: false,
          properties: {
            "android.injected.signing.store.file" =>
    RELEASE_FILE_PATH,
            "android.injected.signing.store.password" =>
    STORE_PASSWORD,
            "android.injected.signing.key.alias" => KEY_ALIAS, #
    key signing alias
            "android.injected.signing.key.password" =>
    KEY_PASSWORD,
          }
        )
     end

```

```
private_lane :update_build_number do
        buildNumber = fetch_build_number.to_i + 1
        value = sh(command: "echo #{buildNumber} > #{
    BUILD_NUMBER_PATH}")

        # Push Changes to bitbucket server
        push_to_bitbucket
    end

private_lane :fetch_build_number do
        sh(
          command: "cat #{BUILD_NUMBER_PATH}"
        )
    end

private_lane :push_to_bitbucket do
        sh("git add .")
        sh("git commit -m 'Code Version Bump'")
        sh("git push origin master")
    end
```

Kode 3.10: Potongan kode file *fastfile*

*Fastlane* memiliki 2 *lane* yaitu *private lane* dan *lane*. Perbedaan dari *private lane* dengan *lane* yaitu cara pemanggilannya dimana jika *private lane* tidak dapat secara eksplisit digunakan pada *command line* tetapi dapat dipanggil oleh *lane* lainnya. Sedangkan *lane* dapat digunakan pada *command line*. Misalkan sebagai contoh, pada kode diatas terdapat *private_lane* bernama *push_to_bitbucket*, jika kita panggil dengan *command fastlane push_to_bitbucket* maka *fastlane* akan mengagalkan *command* tersebut karena bersifat *private*.

Proses dari *fastlane* ini berkaitan dengan jenkinsfile dimana pada jenkinsfile terdapat *command fastlane build* dan *fastlane release*. *Lane fastlane build* merupakan *lane* untuk melakukan *build* aplikasi menjadi *bundle android*. Pada *lane* tersebut, dilakukan pemanggilan *private lane update_build_number* dan *lane* build_bundle. *Lane update_build_number* digunakan untuk melakukan perubahan pada build number dan melakukan push ke bitbucket untuk meng-*update build number*-nya. Pada *lane build_bundle* dilakukan instalasi dependensi *android*, melakukan *set version code android* terbaru, melakukan *set version name android*, lalu melakukan cleaning *android*, dan terakhir membuat *bundle android* berdasarkan *properties key signing* yang dibuat pada *Android Studio*.

*Lane release* menjalankan fungsi *upload_to_play_store* yang merupakan

fungsi dari *fastlane* untuk meng-*upload* aplikasi ke dalam *Google Play Store*. Agar dapat meng-*upload* ke *Google Play Store* harus membuat *track* dan *release status* dari aplikasi yang di-*upload*.

## B. Kode *Jenkinsfile* pada proyek *react.js*

```
node ('master'){
    stage("Clone") {
        git 'http://mock-ip/scm/demo/crud-frontend.git'
    }
    stage("Ionic Config") {
        sh 'npx ionic cap sync'
        sh 'npx ionic cap build android'
        sh 'npx ionic \emph{capacitor} update' // ionic config to
   run the app.
    }
    stage ("Prepare") {
        echo "Setup"
        dir("android"){
          sh 'fastlane add_plugin versioning_android' //plugin for
    android versioning
          sh 'fastlane add_plugin commit_android_version_bump'
        }
    }
    stage('Build') {
      dir("android") {
        echo "Building"
        sh "fastlane build" // build the app
      }
    }
    stage('Release to Play Store'){
      dir("android") {
        sh "fastlane release" // release to playstore
      }
    }
}
```

Kode 3.11: Potongan kode jenkinsfile pada proyek

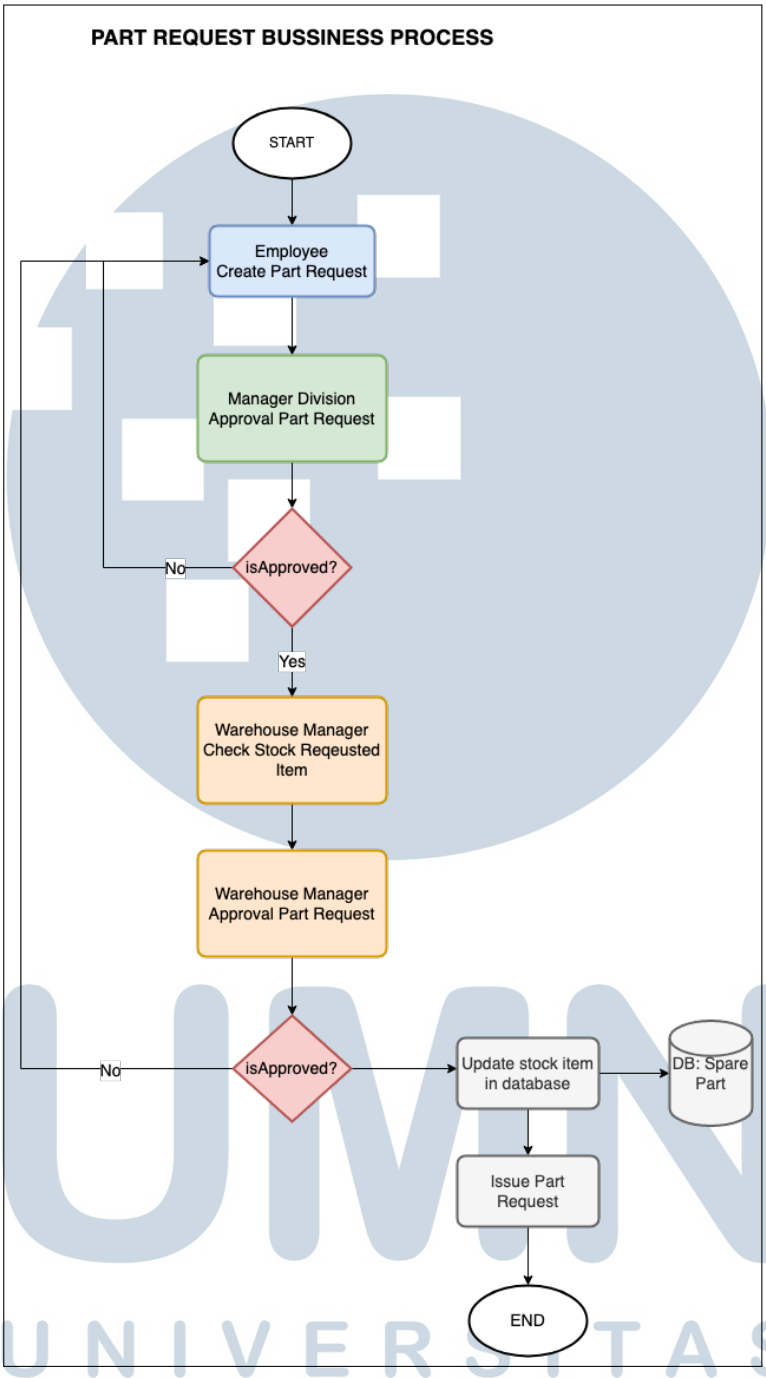


Pada kode tersebut *jenkins* menjalankan 5 proses atau *stage* secara bertahap, stage pertama yaitu *stage Clone* untuk melakukan *clone* dari proyek yang dibuat. *Stage* kedua yaitu *Ionic Config* untuk melakukan *build direktori android* pada proyek *react.js*. Terdapat 3 *command* yang dijalankan pada step tersebut

yaitu *command* ”*npx cap sync*” untuk melakukan sinkronisasi *react.js* dengan *android*, lalu ada ”*npx ionic cap build* android” untuk melakukan *build* aplikasi *android* dengan membuat direktori *android* pada proyek, dan ”*npx ionic* capacitor *update*” untuk melakukan update dependensi. Lalu *stage* ketiga yaitu *Prepare* dimana dilakukan proses *setup fastlane* dengan menambahkan *plugin-plugin* yang digunakan dalam *fastlane* pada direktori atau folder *android*. *Stage* keempat yaitu *Build* untuk melakukan *build* aplikasi menjadi .aab atau *bundle android* menggunakan *command fastlane build*. Terakhir, *stage Release to Play Store* untuk meng-*upload bundle* yang sudah dibuat ke dalam *google play console* yang nantinya akan dirilis ke *google play store* menggunakan *command fastlane release*.

### 3.4.6 Rancangan Flowchart

Berikut merupakan rancangan *flowchart* dari *modul bussiness process part request* yang telah dibuat.




Gambar 3.17. *Flowchart* part request

Pada *flowchart* diatas, proses bisnis dari *part request* yakni *user* dengan *role employee* membuat sebuah *part request*. Lalu, *user* dengan *role manager division* melakukan *approval* kepada *part request* yang dibuat employee. Jika tidak di *approve* oleh *manager division* maka *employee* harus melakukan part request kembali, sedangkan jika di *approve* maka akan dilanjutkan ke pengecekan persediaan barang oleh *warehouse manager* yang kemudian akan melakukan

*approval*. Jika *part request* ditolak oleh *warehouse manager* maka *employee* harus melakukan *create part request* kembali. Sebaliknya, jika *warehouse manager approve part request*, maka sistem akan secara otomatis meng-*update* stok pada *database spare part*, lalu sistem akan meng-*issue part request* yang telah di *approve*.

## 3.5 Kendala dan Solusi yang Ditemukan

Sebagai *Technical Consultant Analyst* selama magang, terdapat beberapa kendala yang dihadapi dalam menjalankan tugas-tugasnya. Salah satu kendala utama adalah kesulitan dalam memahami teknologi-teknologi baru karena masih awam. Solusi dari kendala ini adalah dengan melakukan pembelajaran secara intens dan memahami cara kerja dari tiap teknologi-teknologi yang digunakan.

Kendala lain yang sering dihadapi yaitu kurangnya komunikasi antar tim sehingga sering terjadi miskomunikasi dalam pembuatan ERP *inventory system*. Hal ini dapat menghambat kecepatan atau *velocity* pembuatan sistem. Untuk mengatasi hal ini, tim melakukan *meeting* setiap hari untuk melakukan *sharing* atau hanya sekedar *sync up* untuk mengetahui masing-masing tugas yang dikerjakan.

Kendala lain yang dihadapi saat pembuatan ERP *inventory system* yaitu kesulitan dalam menentukan teknologi yang tepat untuk pengembangan aplikasi. Hal ini dikarenakan terdapat banyak sekali teknologi yang dapat digunakan untuk mengembangkan website. Untuk mengatasi hal ini, dilakukan riset dengan membandingkan kekurangan dan kelebihan dari masing-masing teknologi dan menentukan teknologi yang ingin digunakan dalam pengembangan aplikasi dengan tim.

Kendala lain yang sering dihadapi adalah waktu yang terbatas dan jadwal yang ketat. *Timeline Sprint* yang terlalu ketat dapat menyebabkan penurunan kualitas sistem yang dihasilkan. Untuk mengatasi hal ini, diwajibkan untuk dapat mengelola waktu dan sumber daya secara efektif dan efisien, serta bekerja dengan tim untuk memastikan bahwa tugas-tugas yang diberikan dapat diselesaikan tepat waktu dan menghasilkan sistem yang bagus.

Dalam rangka mengatasi kendala-kendala tersebut, diperlukan keterampilan komunikasi yang baik, kemampuan analisis yang kuat, serta kemampuan manajemen waktu yang baik. Dengan kemampuan-kemampuan tersebut, dapat mengatasi berbagai kendala yang dihadapi dalam menjalankan tugasnya sebagai bagian dari tim.