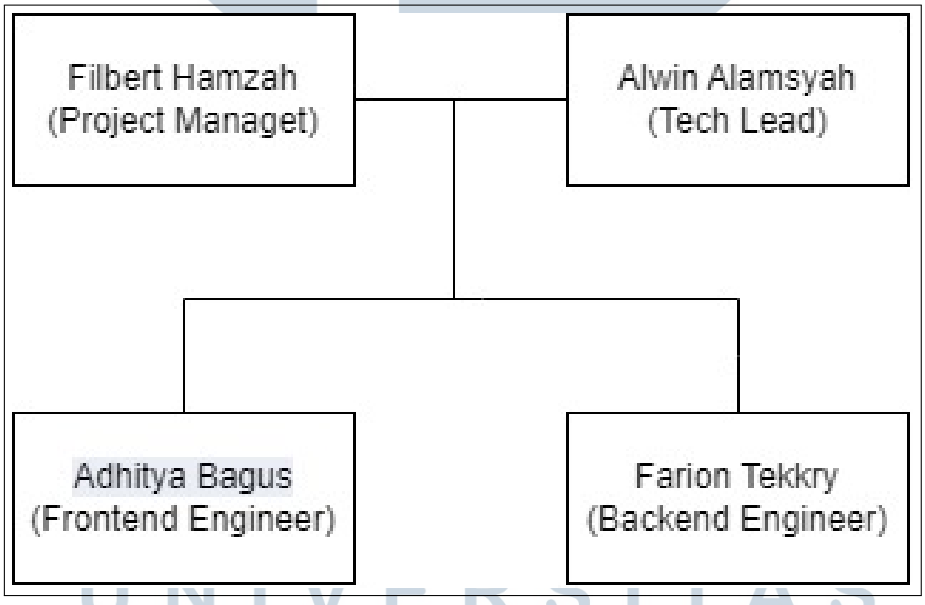
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama proses kerja magang di PT Ganda Visi Jayatama, kedudukan yang ditempati adalah *backend engineer intern* yang termasuk dalam divisi *Software Engineer* yang berada di bawah *Head of Development*. Dalam posisi *backend*, terdapat 1 orang yang berperan sebagai *senior backend engineer*, lalu terdapat 2 orang yang berperan sebagai *junior backend engineer*, dan 1 orang yang berperan sebagai *backend engineer intern* yang semuanya dibimbing langsung oleh Bapak Muhammad Alwin Alamsyah Handoko Putra. Dalam pelaksanaan kerja magang, terdapat tugas-tugas yang diberikan, yaitu membuat API yang dibutuhkan pada modul *project*, melakukan peningkatan terhadap kode yang telah dibuat sebelumnya.



Gambar 3.1. Struktural *Project*

Dalam pengerjaan *project*, terdapat struktural *project* yang dapat dilihat pada 3.1. Tim berisikan 1 orang yang berperan sebagai PM (*Project Manager*), 1 orang yang berperan sebagai *technical lead*, 1 orang yang berperan sebagai *frontend engineer*, dan 1 orang yang berperan sebagai *backend engineer*. Selama proses pengerjaan, komunikasi antar tim dilakukan melalui aplikasi Discord. Tiap 2 minggu sekali pada hari Jumat, akan dilakukan *sprint meeting* yang

di dalamnya akan berisikan *sprint closing* untuk menutup *sprint* sebelumnya dan juga *sprint retrospective* untuk melakukan planning pada *sprint* selanjutnya. Lalu, tiap 2 minggu sekali juga akan dilakukan sesi 1on1 dengan *technical lead* untuk membahas kendala yang dialami dan juga nantinya *technical lead* dapat memberikan *feedback* terhadap apa yang kita hadapi.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang, tugas yang dilakukan diantaranya:

1. Membuat API untuk keperluan *frontend* dalam menampilkan data dan menambahkan fitur pada API yang telah ada.
2. Melakukan *testing* terhadap aplikasi MRO maupun API yang telah dibuat..

3.3 Uraian Pelaksanaan Magang

Selama pelaksanaan magang, terdapat beberapa tugas yang dikerjakan yang dapat dilihat pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Mempelajari *boilerplate backend* perusahaan.
2	Mempelajari *boilerplate backend* perusahaan.
3	Mempelajari *code project* yang telah dibuat sebelumnya.
4	Membuat API CRUD untuk fitur *locator* dan *warehouse*.
5	Melakukan penyesuaian *query locator* pada beberapa modul *inventory*.
6	Melakukan *testing* berdasarkan *test cases* yang telah dibuat.
7	Melakukan *testing* berdasarkan *test cases* yang telah dibuat dan *transfer knowledge* CDC dari *code*.
8	Membuat *pagination* pada *listing item* di semua modul.
9	Membuat *pagination* pada *listing item* di semua modul.
10	Melakukan *testing* berdasarkan *test cases* yang telah dibuat
11	Membuat API CRUD untuk fitur *landing form reason*.
12	Melakukan *refactor* dan *testing* terhadap *code push couch*.
13	Melakukan *refactor* dan *testing* terhadap *code push couch*

Tabel 3.1. Pekerjaan yang dilakukan setiap minggu selama pelaksanaan kerja magang (Lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
14	Menambahkan *filtering* terhadap *listing Vessel Requisition*.
15	Melakukan *bugfixes* pada *filtering listing Vessel Requisition*
16	Melakukan *testing* berdasarkan *test cases* yang telah dibuat.

Selama pelaksanaan kerja magang, tugas yang dilakukan terbagi menjadi 2 tugas, yaitu pembuatan API dan menambahkan fitur baru di API yang telah ada, dan melakukan *testing* pada keseluruhan aplikasi berdasarkan *test cases* yang disediakan.

3.3.1 User Requirements

Sebelum mengerjakan tugas, terdapat beberapa kebutuhan *project* yang dikumpulkan untuk mengembangkan aplikasi MRO, diantaranya adalah:

1. Membuat API CRUD untuk *locators* untuk menentukan titik yang lebih spesifik pada suatu kapal. Data yang dibutuhkan mencakup asset id (kapal tempat *locator* ini berada), id *locator* itu sendiri, id warehouse (apabila *locator* merupakan sebuah *warehouse*), nama dari *locator*, koordinat *locator* (x, y, z), status, keaktifan *locator*, dan kategori *locator*. Pada *endpoint* fitur ini, terdapat 4 metode yang digunakan ketika melakukan *request* HTTP, yaitu GET, POST, PUT, dan DELETE untuk mendapatkan data *locator* yang telah ada, membuat data *locator* baru, memperbarui data *locator* yang telah ada, serta menghapus data *locator* yang ada. Pengguna perlu melakukan autentikasi dengan *login* untuk mendapatkan akses terhadap *endpoint* ini.

2. Membuat API CRUD untuk *landing from reason* untuk menentukan jenis alasan suatu barang diterima. Data yang dibutuhkan pada fitur ini adalah nama dan *code* dari *landing form reason*. Pada *endpoint* fitur ini, terdapat 4 metode yang digunakan ketika melakukan *request* HTTP, yaitu GET, POST, PUT, dan DELETE untuk mendapatkan data *landing form reason* yang telah ada, membuat data *landing form reason* baru, memperbarui data *landing form reason* yang telah ada, serta menghapus data *landing form reason* yang ada. Pengguna perlu melakukan autentikasi dengan *login* untuk mendapatkan akses terhadap *endpoint* ini.

3. Menambahkan *pagination* dan *filtering* pada seluruh modul *inventory*. Pada *pagination*, pengguna dapat memilih untuk menampilkan jumlah data yang diinginkan (10, 25 atau 50 data) dan pengguna juga dapat memilih halaman berapa yang ingin ditampilkan. Pada *filtering*, pengguna dapat menyeleksi data yang ingin ditampilkan, mulai dari data-data dengan rentang waktu tertentu, data-data dengan status tertentu, data-data dengan jenis kapal tertentu, dan sebagainya. *Pagination* dan *filtering* ini dapat ditambahkan pada *query params* ketika melakukan *request HTTP*.

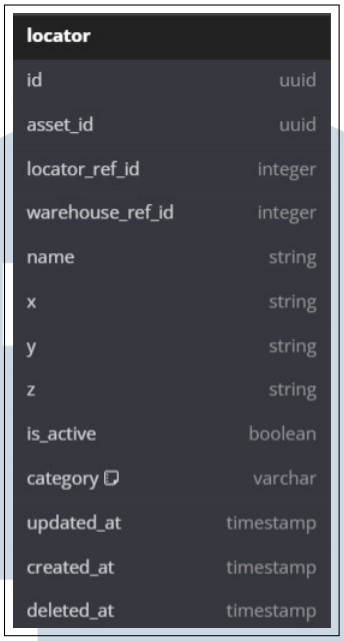
4. Diperlukan juga adanya *testing* untuk memastikan bahwa semua fitur berjalan dengan baik tanpa adanya kendala sebelum digunakan oleh klien. Jenis *testing* yang digunakan adalah *blackbox testing* dengan menjalani seluruh *test cases* yang ada. Isi dari *test cases* akan berupa langkah-langkah melakukan sebuah *case* dan hasil yang diharapkan. Nantinya akan dicatat hasil yang didapat beserta status berhasil atau tidaknya percobaan tersebut.

3.3.2 Membuat Fitur

Selama melaksanakan pekerjaan magang, tugas yang dikerjakan adalah membuat beberapa fitur, diantaranya adalah membuat fitur *locator*, fitur *landing form reason*, dan penambahan *pagination* serta *filtering* pada API yang telah ada sebelumnya. Fitur tersebut dikembangkan dari *database schema* yang telah ada sebelumnya dengan menggunakan PostgreSQL. *Database schema* dari fitur *locator*, *landing form reason* dapat dilihat pada Gambar 3.2 dan Gambar 3.18.

A. Locator CRUD

Locator merepresentasikan suatu lokasi dalam suatu kapal. Dalam pembuatan fitur ini, terdapat skema *database* yang dapat dilihat pada Gambar 3.2.

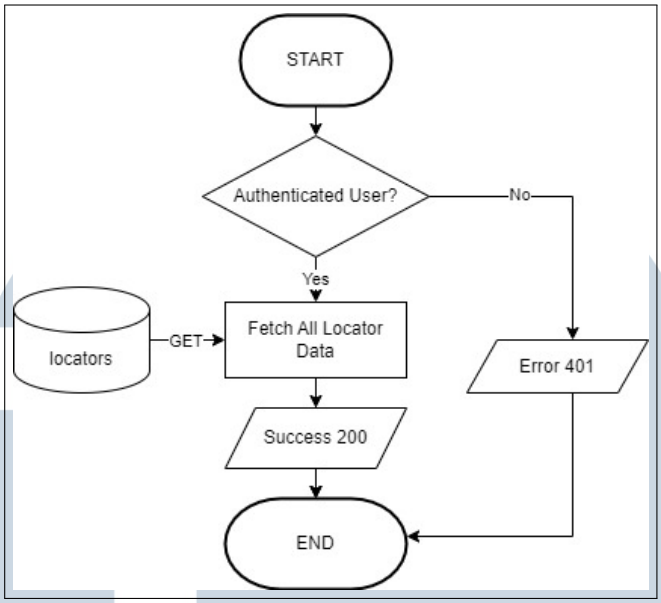


Gambar 3.2. *Database schema locator*

Dalam pembuatan *locator*, terdapat beberapa *endpoint* yang dapat dilihat pada Tabel 3.2

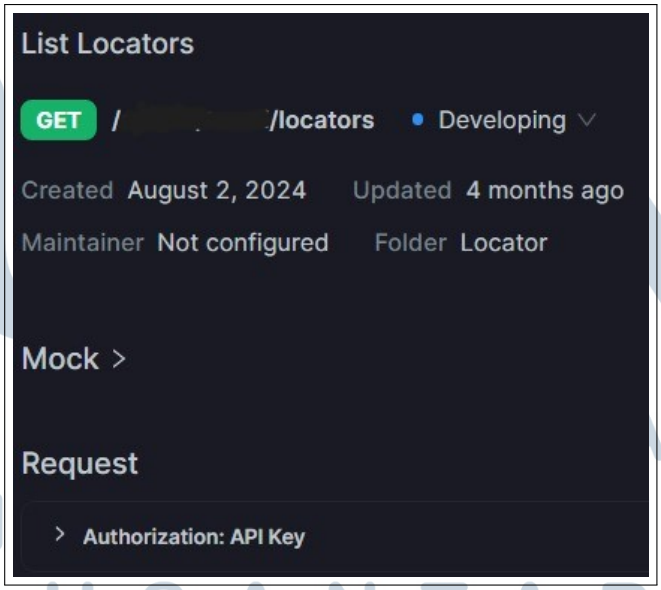
Tabel 3.2. Tabel *endpoint* pada *locator*

Nama API	Endpoint	Metode	Deskripsi
Get All Locators	/locators	GET	API ini berfungsi untuk mendapatkan semua data *locator*.
Get Locator By Id	/locators/:id	GET	API ini berfungsi untuk mendapatkan satu data *locator* yang diinginkan.
Create Locator	/locators/	POST	API ini berfungsi untuk membuat satu *locator* baru.
Update Locator	/locators/:id	PUT	API ini berfungsi untuk memperbarui satu *locator* yang diinginkan.
Delete Locator	/locators/:id	DELETE	API ini berfungsi untuk menghapus satu *locator* yang diinginkan.



Gambar 3.3. Flowchart *get all locators*

Gambar 3.3 merupakan flowchart ketika pengguna ingin mendapatkan semua data locator. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi, apabila pengguna telah terautentikasi, maka semua data locator akan muncul dan begitu pula sebaliknya.



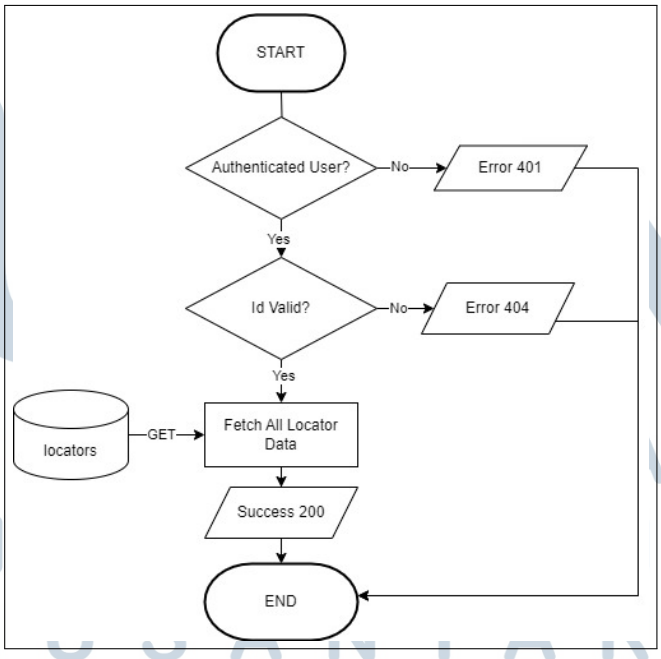
Gambar 3.4. Contoh *request get all locators*

Gambar 3.4 merupakan contoh *request* yang dikirimkan ketika ingin mendapatkan semua data *locator*.

```
{
    "status": true,
    "code": 200,
    "message": "Successfully Fetch All Locators",
    "data": [
        {
            "id": "                                    ",
            "name": "                    .",
            "asset_id": "                                    ",
            "warehouse_ref_id":  ,
            "category": "locator",
            "x": null,
            "y": null,
            "z": null,
            "locator_ref_id":  ,
            "is_active": true,
            "deleted_at": null,
            "created_at": "2024-08-02T13:21:13+07:00",
            "updated_at": "2024-08-02T13:21:13+07:00",
            "asset": {
                "id": "                                    ",
                "asset_ref_id":  ,
                "name": "                    .",
                "code": "    .",
                "type": "                ",
                "description": null,
                "is_active": true,
                "created_at": "2023-05-12T12:03:48.310Z",
                "updated_at": "2023-06-22T08:45:24.842Z"
            }
        },
```


Gambar 3.5. Contoh *response get all locators*

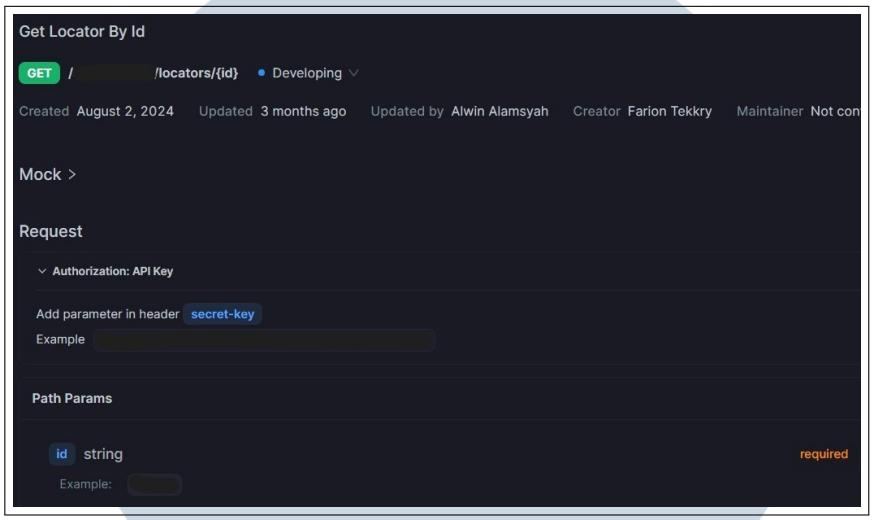
Gambar 3.5 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.4.



Gambar 3.6. Flowchart *get get locator by id*

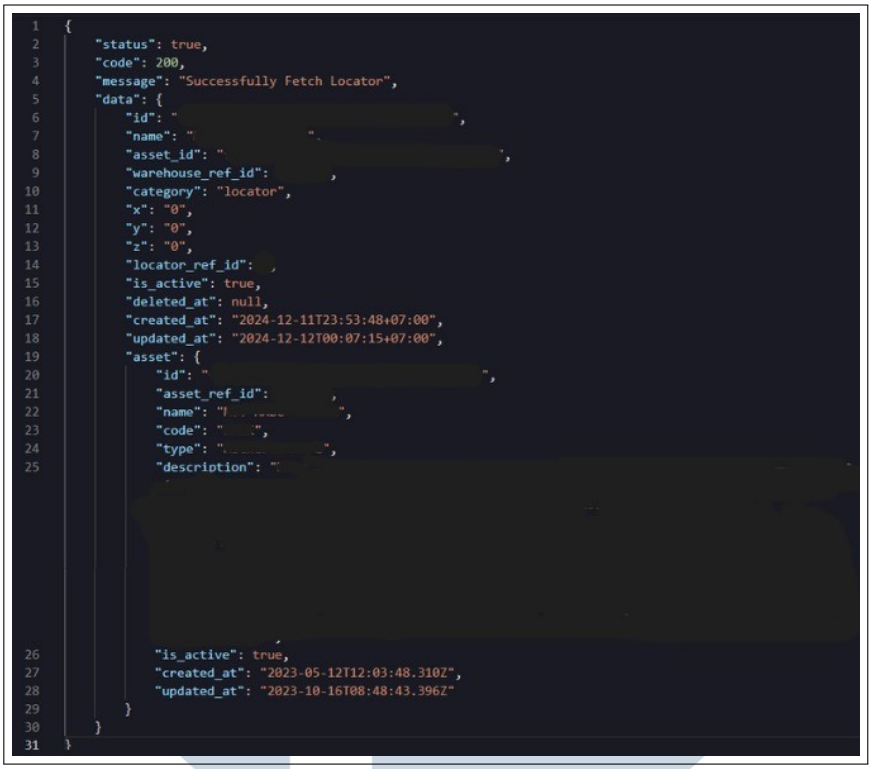
Gambar 3.6 merupakan flowchart ketika pengguna ingin mendapatkan data locator berdasarkan id yang diinginkan. Proses diawali dengan mengecek apakah

pengguna yang sedang *login* terautentikasi. Jika pengguna berhasil diautentikasi, akan dilakukan pengecekan terhadap id yang dikirim. Apabila id tersebut valid, maka locator dengan id tersebut akan dikembalikan.



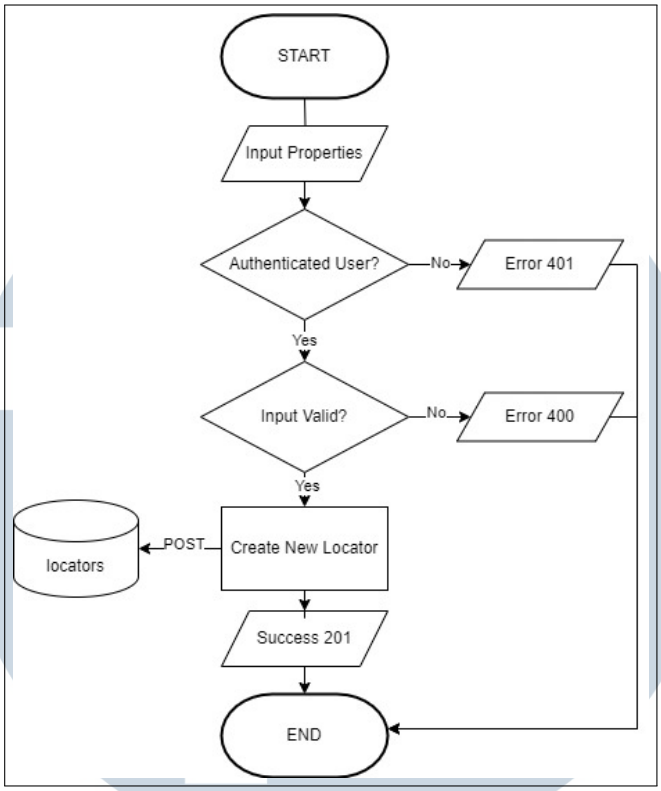
Gambar 3.7. Contoh *request get locator by id*

Gambar 3.7 merupakan contoh *request* yang dikirimkan ketika ingin mendapatkan data *locator* berdasarkan id yang diinginkan dengan meletakkan id yang ingin dicari pada *path params*.



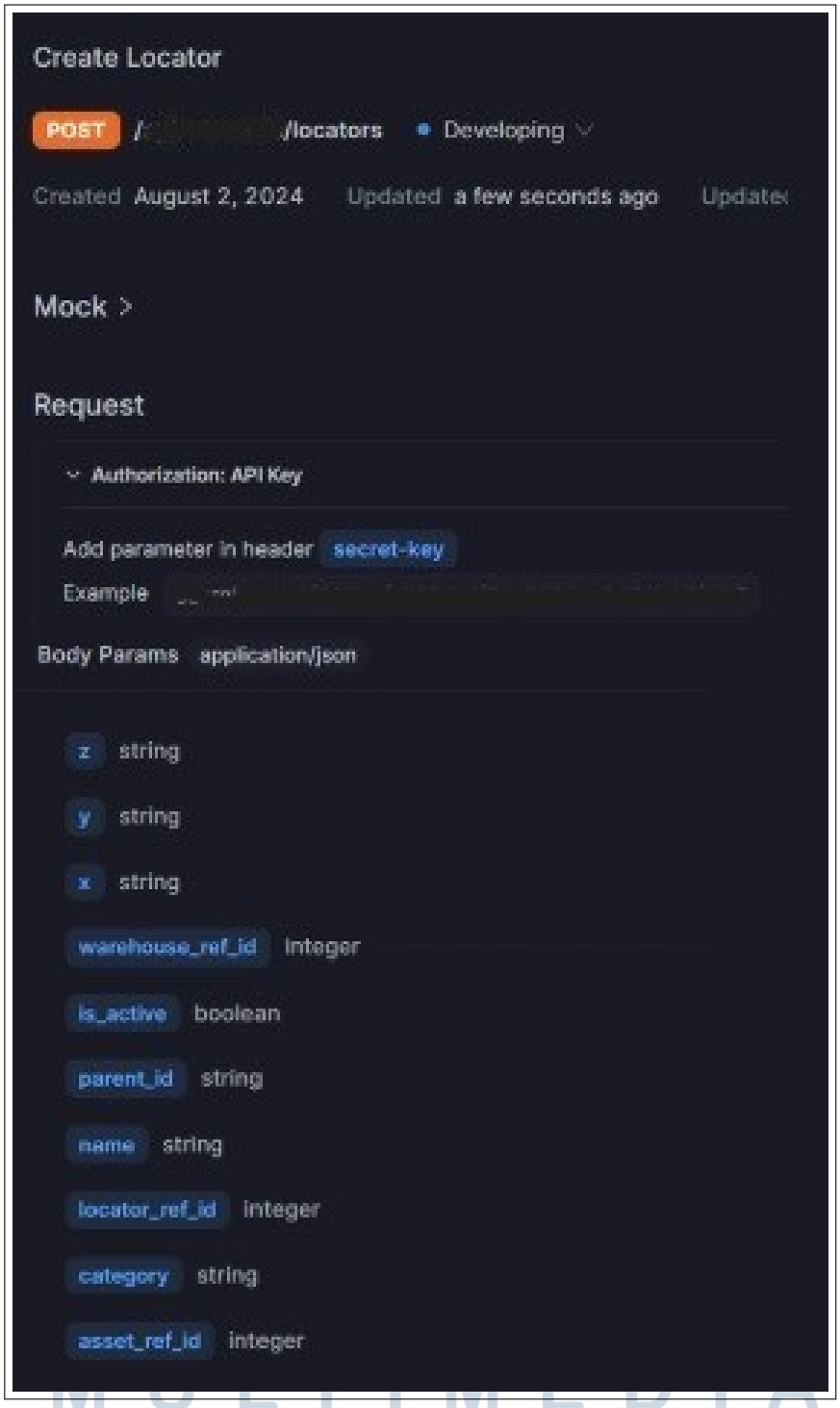
Gambar 3.8. Contoh *response get locator by id*

Gambar 3.24 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.7.



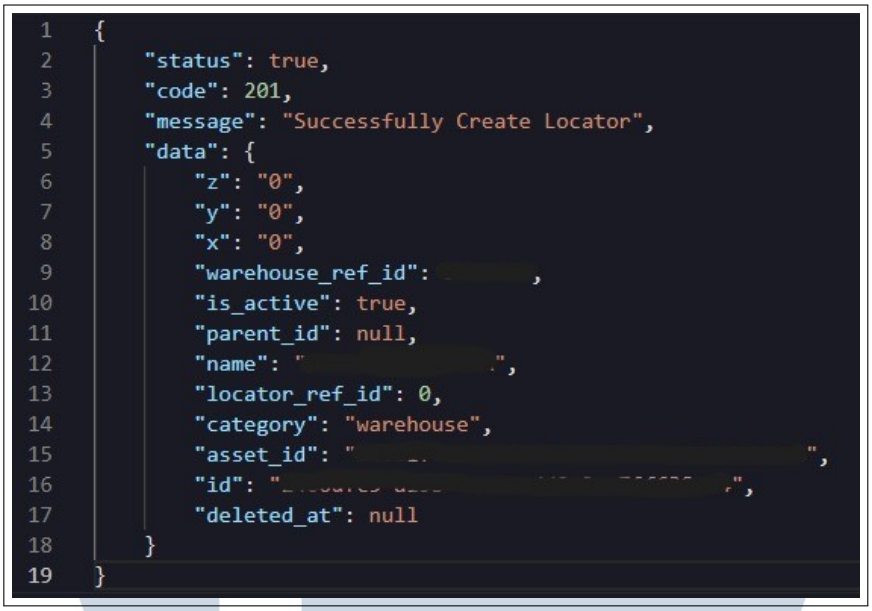
Gambar 3.9. Flowchart *create locator*

Gambar 3.9 merupakan flowchart ketika pengguna ingin membuat suatu locator. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi. Jika pengguna berhasil diautentikasi, akan dilakukan pengecekan terhadap input atau *body* yang dikirim. Apabila *body* valid, locator yang baru akan terbuat.



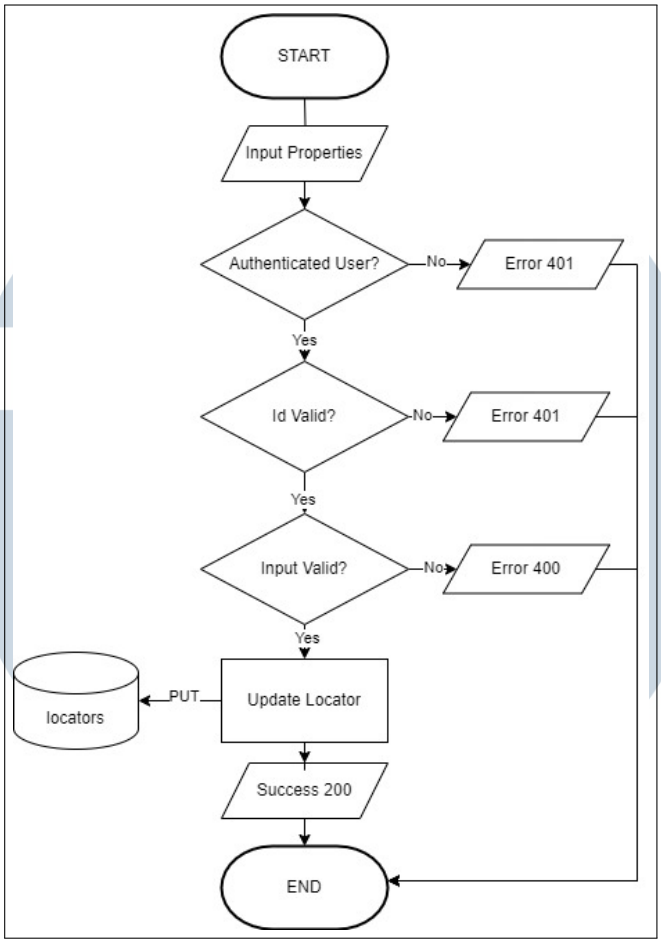
Gambar 3.10. Contoh *request create locator*

Gambar 3.10 merupakan contoh *request* yang dikirimkan ketika ingin membuat sebuah *locator*. Data yang dikirimkan adalah *x*, *y*, *warehouse_ref_id* (opsional), *parent_id* (opsional), *name*, *locator_ref_id* (opsional), *category*, dan *asset_ref_id*.



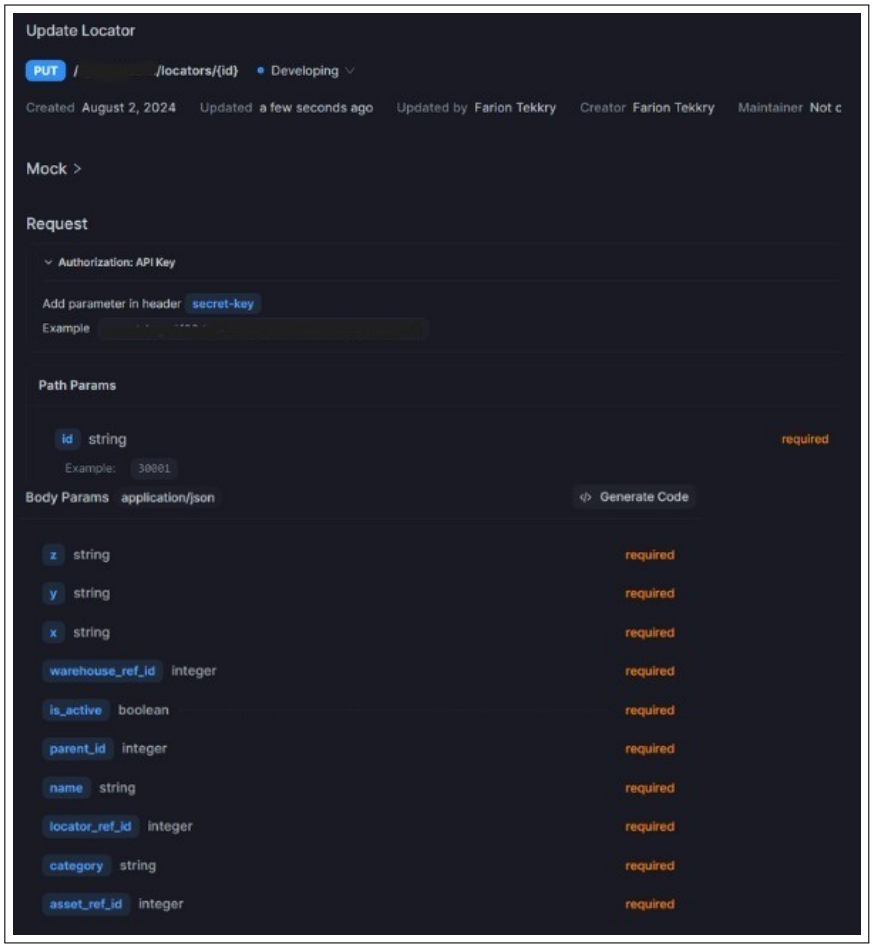
Gambar 3.11. Contoh *response create locator*

Gambar 3.11 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.10.



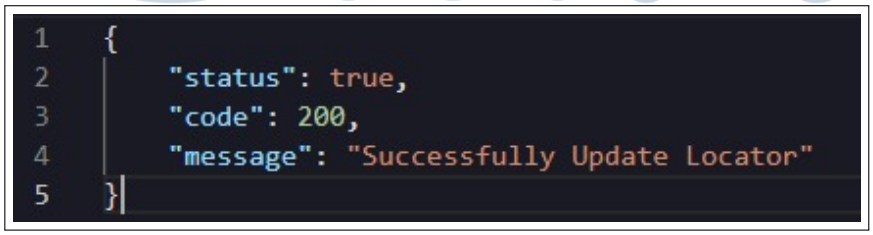
Gambar 3.12. Flowchart *update locator*

Gambar 3.12 merupakan flowchart ketika pengguna ingin melakukan *update* terhadap suatu locator. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi. Jika pengguna berhasil diautentikasi, akan dilakukan pengecekan terhadap id locator yang dikirimkan. Apabila id locator valid, maka akan dilakukan pengecekan input atau *body* yang dikirim, jika valid, maka locator dengan id tersebut akan di *update*.



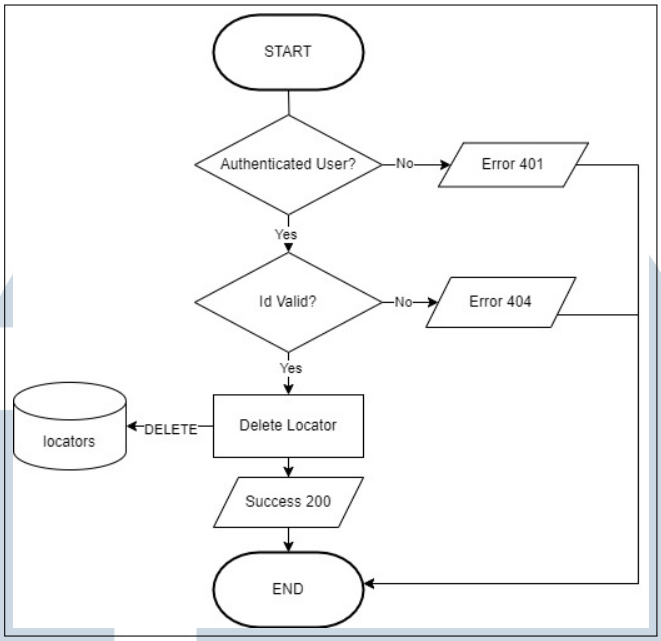
Gambar 3.13. Contoh *request update locator*

Gambar 3.13 merupakan contoh *request* yang dikirimkan ketika ingin melakukan suatu *update* terhadap sebuah *locator*. Data yang dikirimkan adalah *x*, *y*, *warehouse_ref_id* (opsional), *parent_id* (opsional), *name*, *locator_ref_id* (opsional), *category*, dan *asset_ref_id*.



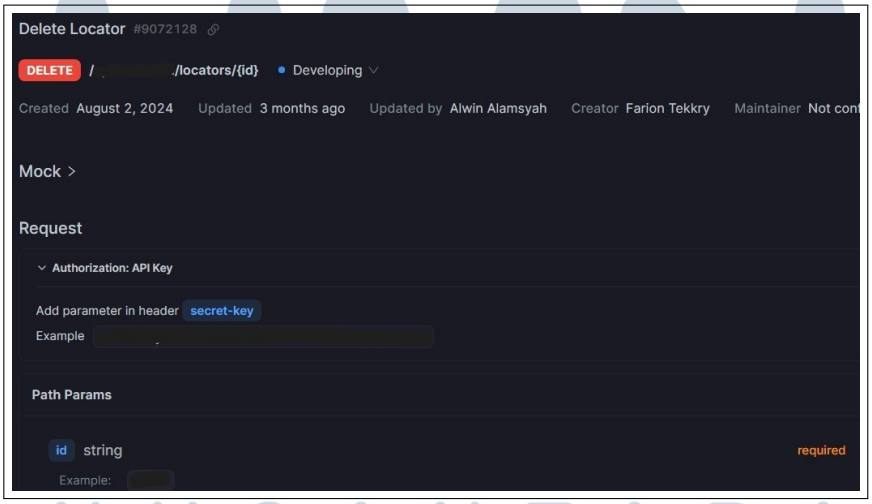
Gambar 3.14. Contoh *response update locator*

Gambar 3.14 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.13.



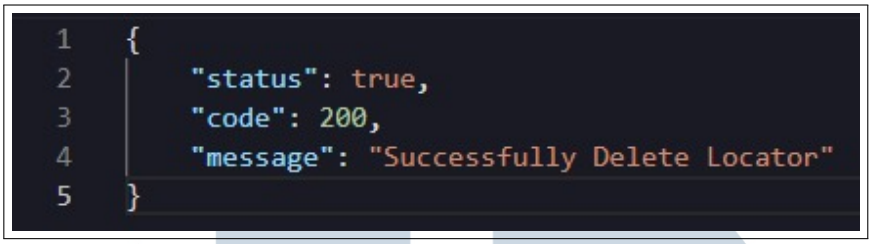
Gambar 3.15. Flowchart *delete locator*

Gambar 3.15 merupakan flowchart ketika pengguna ingin melakukan menghapus suatu locator. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi. Jika pengguna terautenetikasi, akan dilakukan pengecekan terhadap id locator yang dikirimkan. Apabila id locator valid, maka locator dengan id tersebut akan dihapus.



Gambar 3.16. Contoh *request delete locator*

Gambar 3.16 merupakan contoh *request* yang dikirimkan ketika ingin menghapus sebuah *locator* berdasarkan id yang diinginkan dengan meletakkan id pada *path params*.

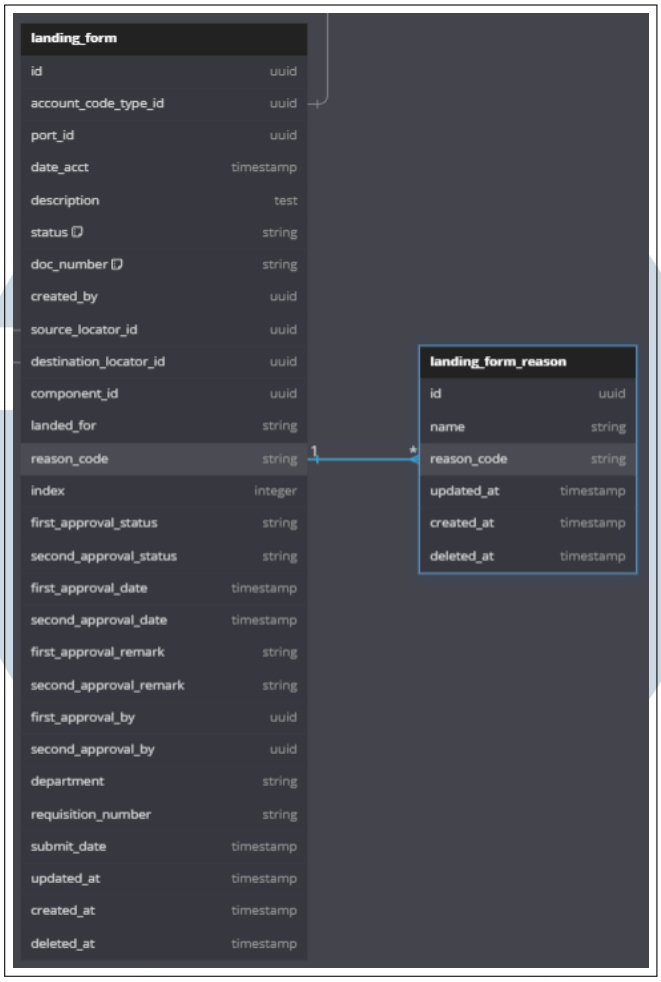


Gambar 3.17. Contoh *response delete locator*

Gambar 3.17 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.16.

B. Landing Form Reason CRUD

Landing form reason pada landing form merepresentasikan alasan suatu barang ketika diterima pada suatu kapal. Dalam pembuatan *landing form reason*, terdapat *database schema* yang dapat dilihat pada Gambar 3.18



Gambar 3.18. *Database schema landing form reason*

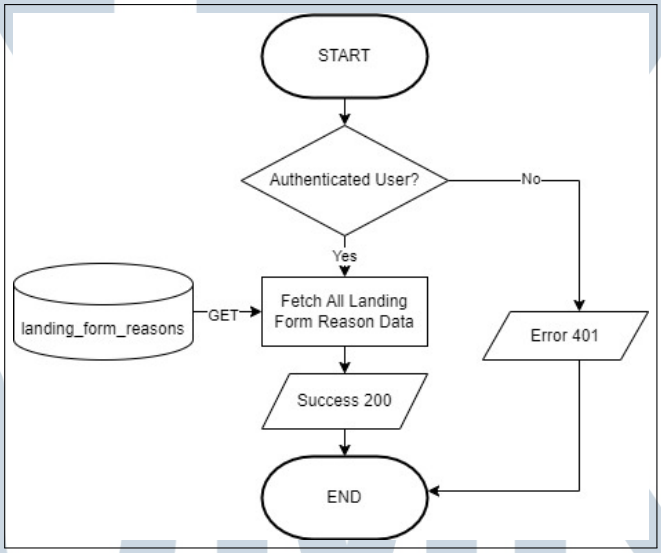
Landing form reason pada landing form merepresentasikan alasan suatu barang ketika diterima pada suatu kapal, terdapat beberapa *endpoint* yang dapat dilihat pada Tabel 3.3

Tabel 3.3. Tabel *endpoint* pada *landing form reason*

Nama API	Endpoint	Metode	Deskripsi
Get All Landing Form Reason	/landing-forms/reasons	GET	API ini berfungsi untuk mendapatkan semua data *warehouse*.
Get Landing Form Reason By Id	/landing-forms/reasons/:id	GET	API ini berfungsi untuk mendapatkan satu data *warehouse* yang diinginkan.

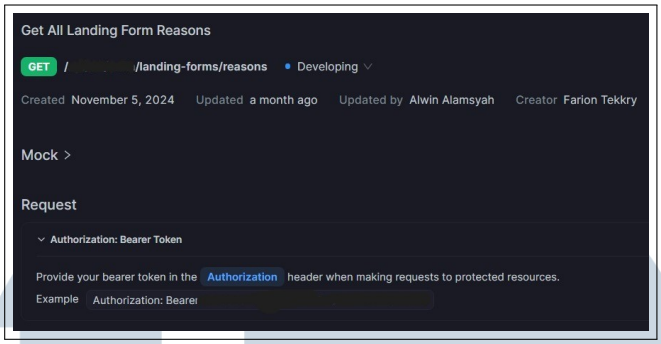
Tabel 3.3. Tabel *endpoint* pada *landing form reason* (Lanjutan)

Nama API	Endpoint	Metode	Deskripsi
Create Landing Form Reason	/landing-forms/reasons/	POST	API ini berfungsi untuk membuat satu *warehouse* baru.
Update Landing Form Reason	/landing-forms/reasons/:id	PUT	API ini berfungsi untuk memperbarui satu *warehouse* yang diinginkan.
Delete Landing Form Reason	/landing-forms/reasons/:id	DELETE	API ini berfungsi untuk menghapus satu *warehouse* yang diinginkan.



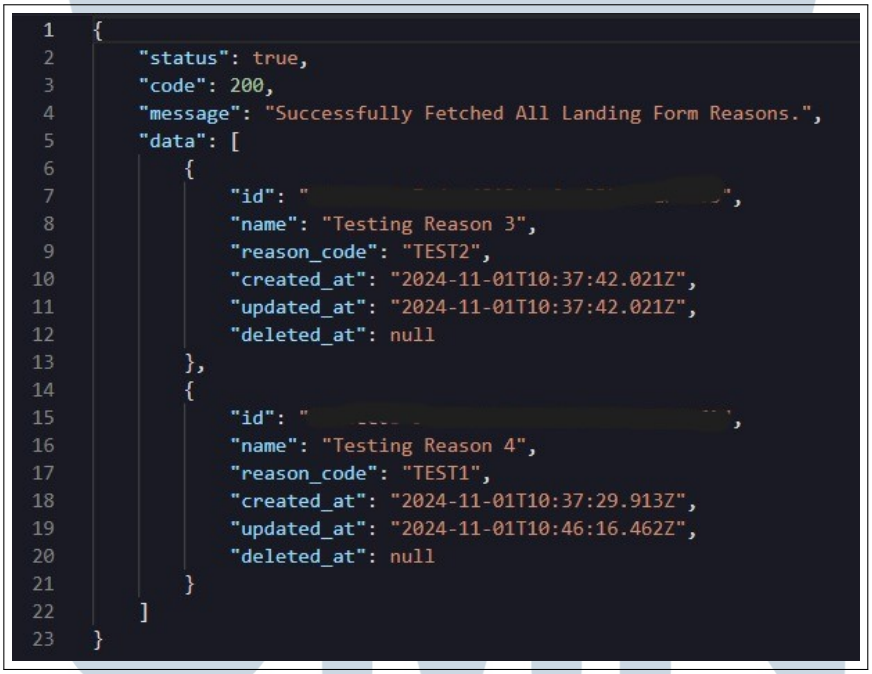
Gambar 3.19. Flowchart *get all landing form reasons*

Gambar 3.19 merupakan flowchart ketika pengguna ingin mendapatkan semua data landing form reason. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi. Jika pengguna terautentikasi, maka semua data *landing form reason* akan muncul dan begitu pula sebaliknya.



Gambar 3.20. Contoh *request get all landing form reasons*

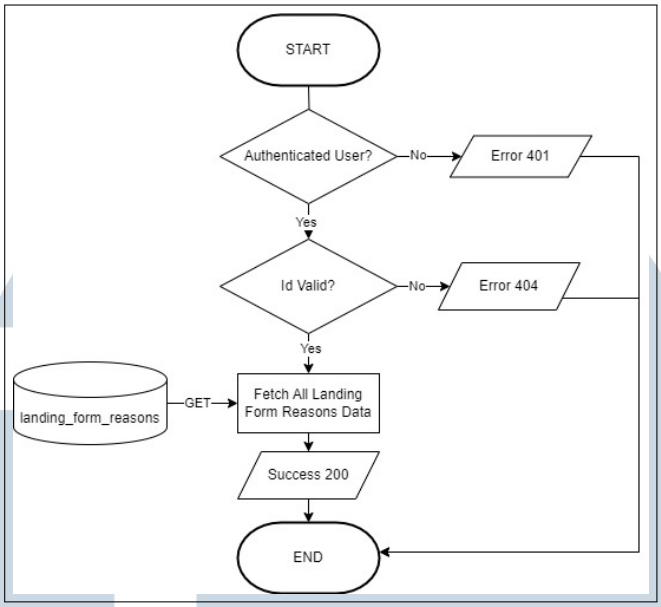
Gambar 3.20 merupakan contoh *request* yang dikirimkan ketika ingin mendapatkan semua data *landing form reasons*.



```
{
    "status": true,
    "code": 200,
    "message": "Successfully Fetched All Landing Form Reasons.",
    "data": [
        {
            "id": "[redacted]",
            "name": "Testing Reason 3",
            "reason_code": "TEST2",
            "created_at": "2024-11-01T10:37:42.021Z",
            "updated_at": "2024-11-01T10:37:42.021Z",
            "deleted_at": null
        },
        {
            "id": "[redacted]",
            "name": "Testing Reason 4",
            "reason_code": "TEST1",
            "created_at": "2024-11-01T10:37:29.913Z",
            "updated_at": "2024-11-01T10:46:16.462Z",
            "deleted_at": null
        }
    ]
}
```

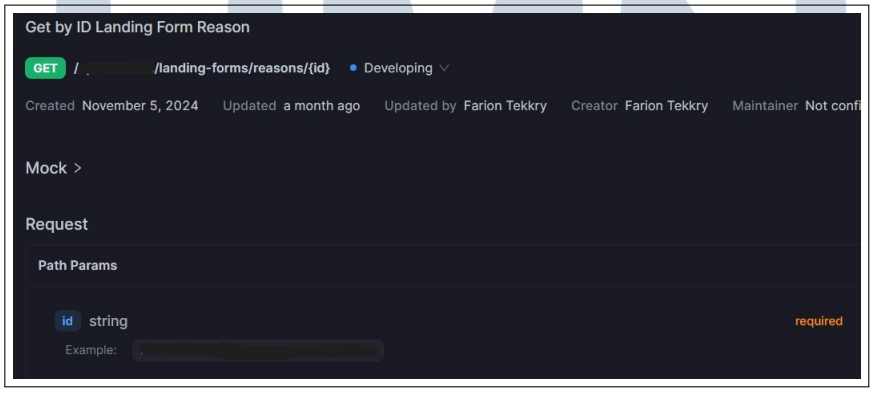
Gambar 3.21. Contoh *response get all landing form reasons*

Gambar 3.5 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.20.



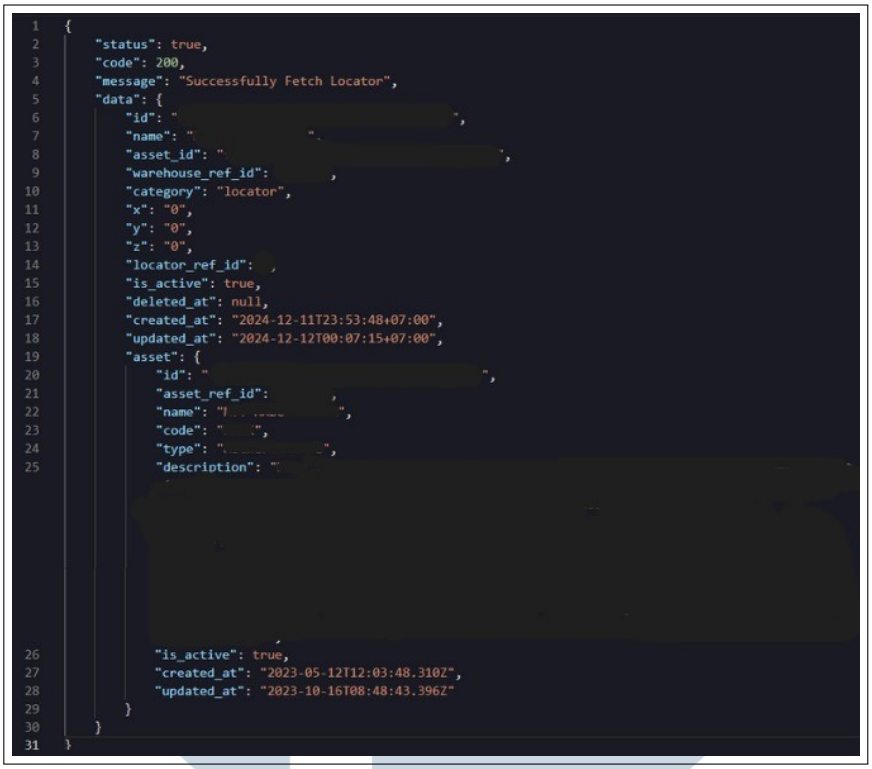
Gambar 3.22. Flowchart *get landing form reason by id*

Gambar 3.22 merupakan flowchart ketika pengguna ingin mendapatkan data landing form reason berdasarkan id yang diinginkan. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi. Jika pengguna berhasil diautentikasi, maka akan dilakukan pengecekan terhadap id yang dikirim. Apabila id tersebut valid, *landing form reason* dengan id tersebut akan dikembalikan.



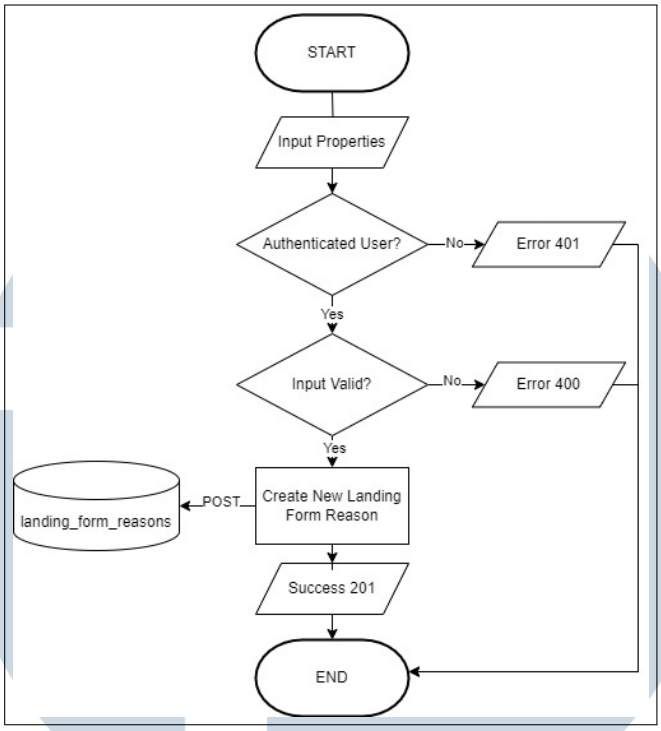
Gambar 3.23. Contoh *request get landing form reason by id*

Gambar 3.7 merupakan contoh *request* yang dikirimkan ketika ingin mendapatkan data *landing form reason* berdasarkan id yang diinginkan dengan meletakkan id yang ingin dicari pada *path params*.



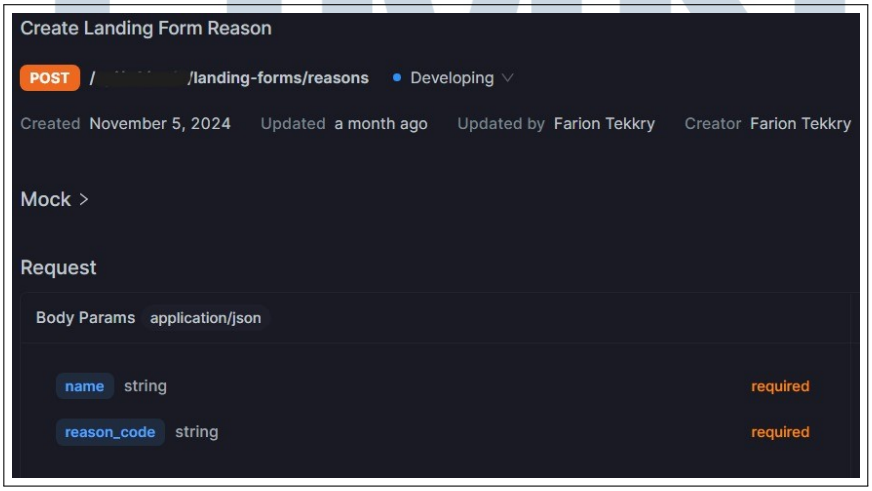
Gambar 3.24. Contoh *response get landing form reason by id*

Gambar 3.24 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.23.



Gambar 3.25. Flowchart *create landing form reason*

Gambar 3.25 merupakan flowchart ketika pengguna ingin membuat suatu landing form reason. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi. Jika pengguna berhasil diautentikasi, akan dilakukan pengecekan terhadap input atau *body* yang dikirim. Apabila *body* valid, maka *landing form reason* yang baru akan terbuat.



Gambar 3.26. Contoh *request create landing form reason*

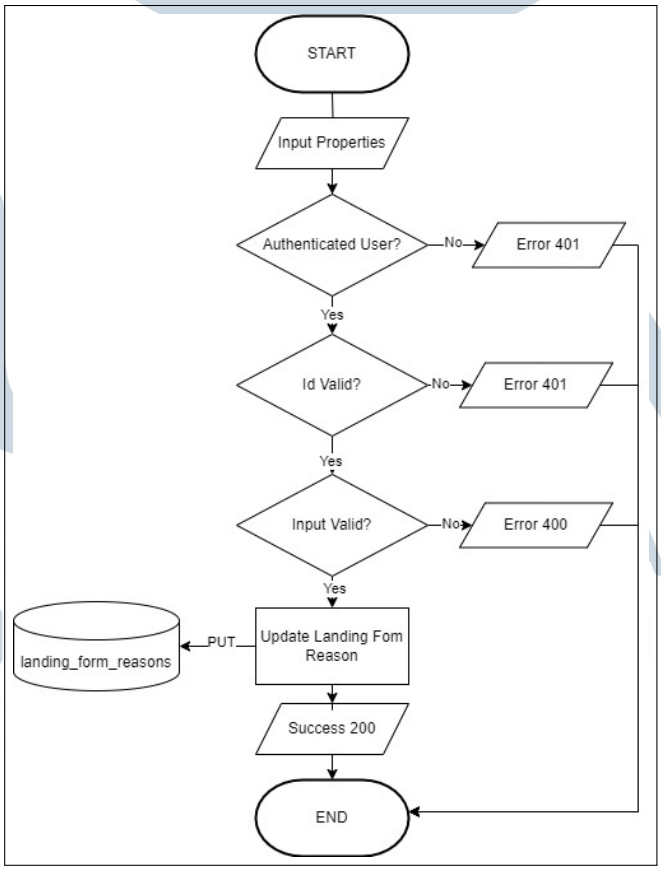
Gambar 3.26 merupakan contoh *request* yang dikirimkan ketika ingin

membuat sebuah *landing form reason*. Data yang dikirimkan adalah *x*, *name*, dan *reason_code* pada *request body*.


```
{
    "status": true,
    "code": 201,
    "message": "Successfully Create Landing Form Reason",
    "data": {
        "id": "[redacted]",
        "name": "Reason 1",
        "reason_code": "TEST 1",
        "updated_at": "2024-12-11T17:15:00.247Z",
        "created_at": "2024-12-11T17:15:00.247Z",
        "deleted_at": null
    }
}
```

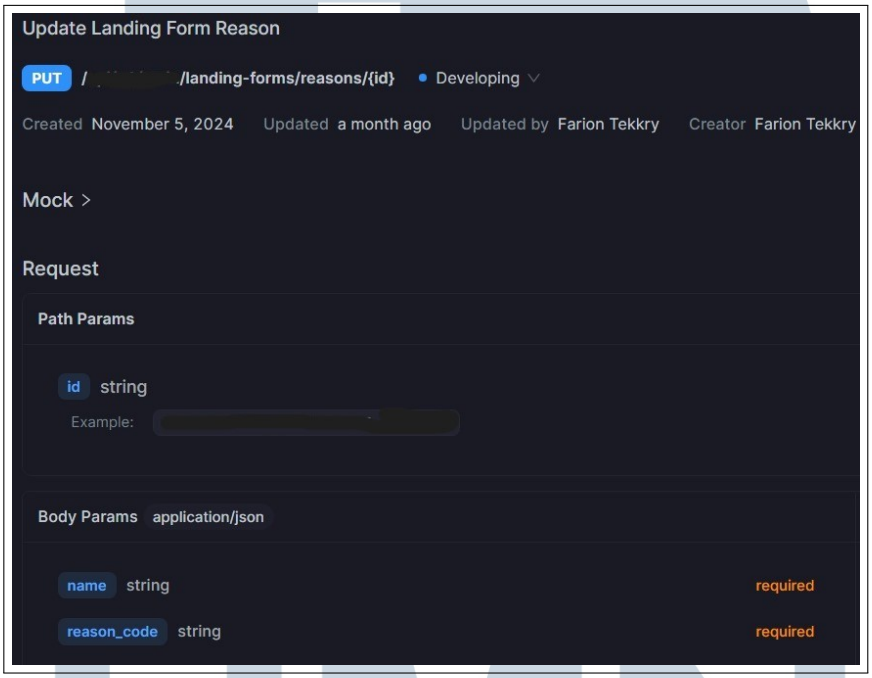

Gambar 3.27. Contoh *response create landing form reason*

Gambar 3.27 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.26.



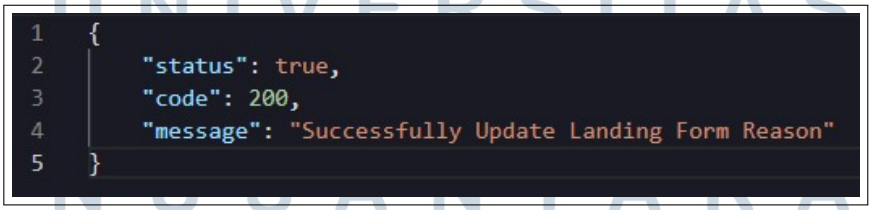
Gambar 3.28. Flowchart *update landing form reason*

Gambar 3.28 merupakan flowchart ketika pengguna ingin melakukan *update* terhadap suatu landing form reason. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi. Jika pengguna berhasil diautentikasi, akan dilakukan pengecekan terhadap id *landing form reason* yang dikirimkan. Apabila id *landing form reason* valid, maka akan dilakukan pengecekan input atau *body* yang dikirim, jika valid, maka *landing form reason* dengan id tersebut akan di *update*.



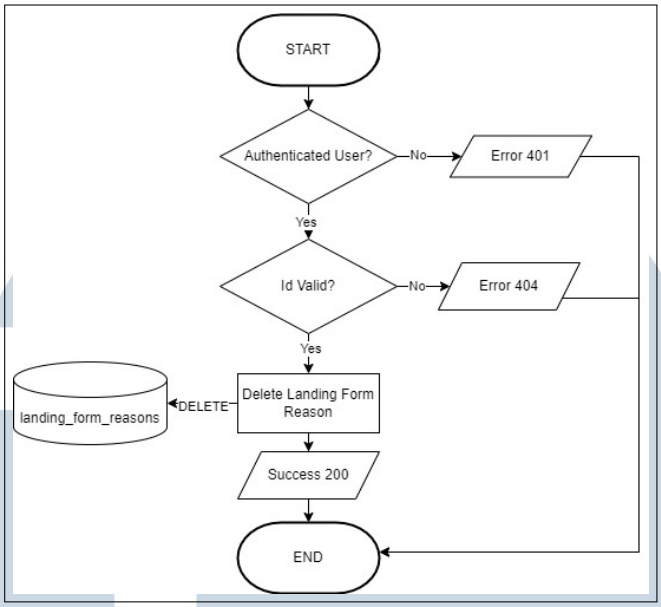
Gambar 3.29. Contoh *request update landing form reason*

Gambar 3.29 merupakan contoh *request* yang dikirimkan ketika ingin melakukan suatu *update* terhadap sebuah *landing form reason*. Data yang dikirimkan adalah *name*, dan *reason_code* pada *request body*, sedangkan pengguna juga harus memasukkan id yang diinginkan pada *path params*.



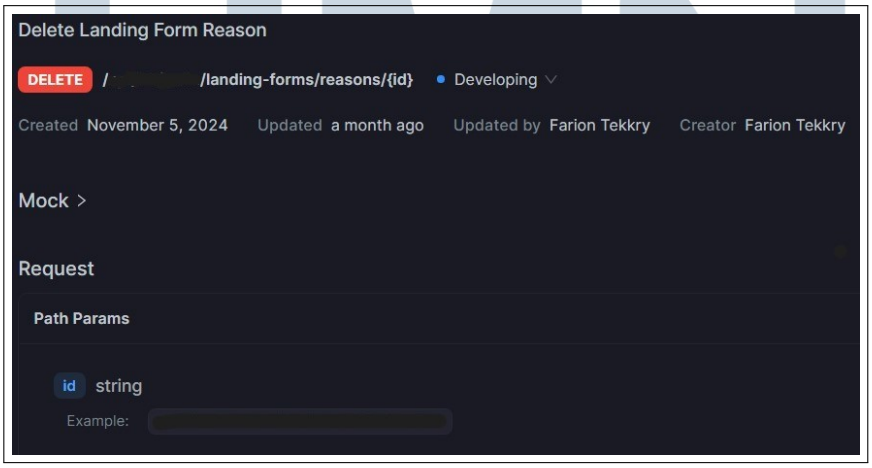
Gambar 3.30. Contoh *response update landing form reason*

Gambar 3.30 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.29.



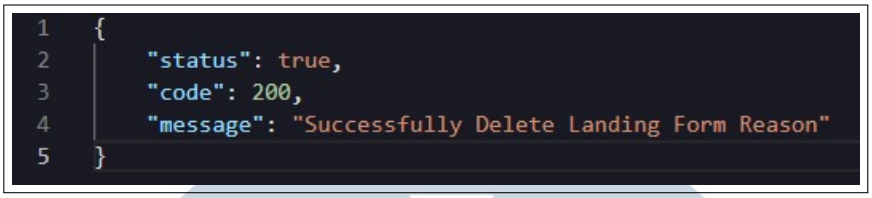
Gambar 3.31. Flowchart *delete landing form reason*

Gambar 3.31 merupakan flowchart ketika pengguna ingin melakukan menghapus suatu landing form reason. Proses diawali dengan mengecek apakah pengguna yang sedang *login* terautentikasi. Jika pengguna berhasil diautentikasi, akan dilakukan pengecekan terhadap id *landing form reason* yang dikirimkan. Apabila id *landing form reason* valid, maka *landing form reason* dengan id tersebut akan dihapus.



Gambar 3.32. Contoh *request delete landing form reason*

Gambar 3.32 merupakan contoh *request* yang dikirimkan ketika ingin menghapus sebuah *landing form reason* berdasarkan id yang diinginkan dengan meletakkan id pada *path params*.

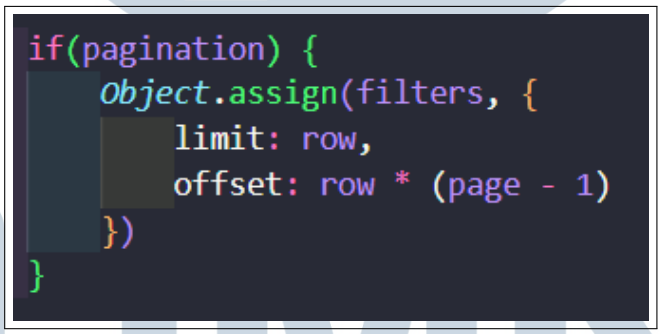


Gambar 3.33. Contoh *response delete landing form reason*

Gambar 3.17 merupakan contoh *response* yang didapatkan ketika pengguna mengirim *request* dengan bentuk seperti pada Gambar 3.32.

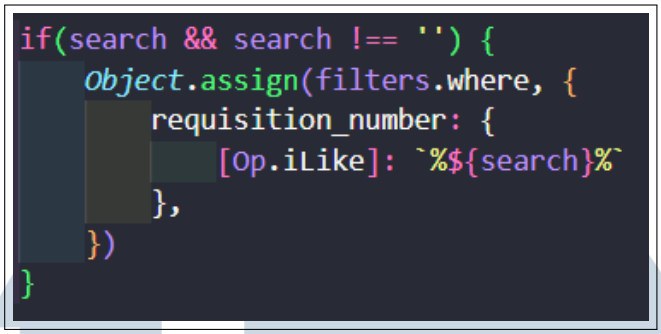
C. *Pagination* dan *Filtering*

Salah satu fitur yang ditambahkan adalah penambahan *pagination* dan *filtering* pada semua modul *inventory*, seperti *Vessel Requisition*, *Material Receipt*, *Miscellaneous Receipt*, *Intra Warehouse*, *Stock Card*, *Stock Opname*, *Internal Use*, *Landing Form*. Hal yang dilakukan adalah dengan menambahkan *pagination*, *filtering*, *sorting*, *searching* sesuai dengan kebutuhan.



Gambar 3.34. Pagination pada Listing

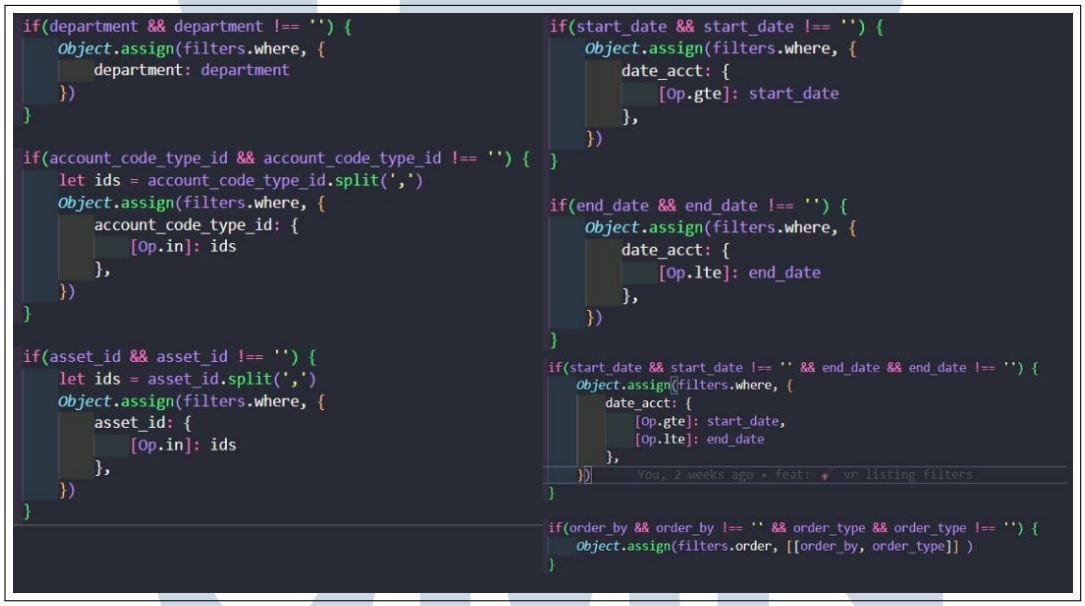
Pada potongan kode di Gambar 3.34,pengguna dapat memilih opsi untuk mengimplementasikan *pagination* atau tidak. Jika pengguna memilih untuk mengimplementasikan *pagination*, maka data yang dikirim akan diberi batas sesuai dengan jumlah baris yang ingin ditampilkan (*row*), dan dimulai dari halaman berapa (*offset*).



```
if(search && search !== '') {
    Object.assign(filters.where, {
        requisition_number: {
            [Op.iLike]: `%${search}%`
        },
    })
}
```

Gambar 3.35. Search pada Listing

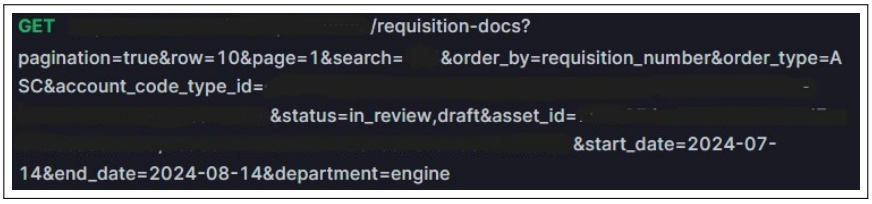
Gambar 3.35 merupakan potongan kode *search* berdasarkan nomor suatu dokumen. Untuk *pagination* dan *search* telah diterapkan di seluruh modul *inventory*.



```
if(department && department !== '') {
    Object.assign(filters.where, {
        department: department
    })
}

if(account_code_type_id && account_code_type_id !== '') {
    let ids = account_code_type_id.split(',')
    Object.assign(filters.where, {
        account_code_type_id: {
            [Op.in]: ids
        },
    })
}

if(asset_id && asset_id !== '') {
    let ids = asset_id.split(',')
    Object.assign(filters.where, {
        asset_id: {
            [Op.in]: ids
        },
    })
}

if(start_date && start_date !== '') {
    Object.assign(filters.where, {
        date_acct: {
            [Op.gte]: start_date
        },
    })
}

if(end_date && end_date !== '') {
    Object.assign(filters.where, {
        date_acct: {
            [Op.lte]: end_date
        },
    })
}

if(start_date && start_date !== '' && end_date && end_date !== '') {
    Object.assign(filters.where, {
        date_acct: {
            [Op.gte]: start_date,
            [Op.lte]: end_date
        },
    })
}

if(order_by && order_by !== '' && order_type && order_type !== '') {
    Object.assign(filters.order, [[order_by, order_type]] )
}
```

Gambar 3.36. Filter pada Listing VR

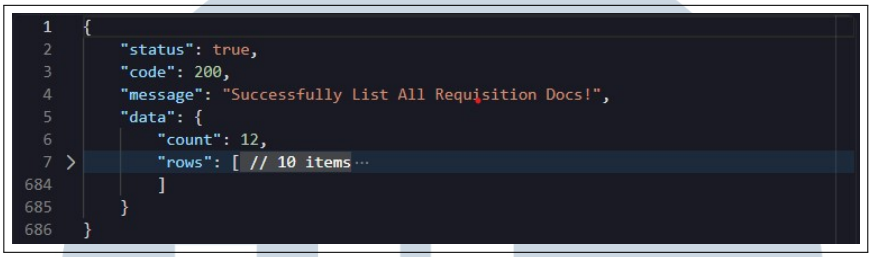
Gambar 3.36 adalah potongan kode *filter* yang bisa diterapkan pada *listing Vessel Requisition*. Pengguna dapat memilih untuk melakukan filter berdasarkan *department*, *account code type*, *asset*, *start date*, *end date*, dan *order*.



```
GET                                    /requisition-docs?
pagination=true&row=10&page=1&search=      &order_by=requisition_number&order_type=A
SC&account_code_type_id=
                                &status=in_review,draft&asset_id=
                                                        &start_date=2024-07-
14&end_date=2024-08-14&department=engine
```

Gambar 3.37. Contoh Request Listing VR

Gambar 3.37 adalah contoh *query params* yang dapat dikirimkan ketika ingin melakukan *filtering* pada *Veseel Requisition*. Tiap *query* dapat dikirimkan secara opsional.



Gambar 3.38. Contoh Response Listing VR

Gambar 3.38 adalah contoh *response* penerapan *pagination*, dan *filtering* pada *listing Vessel Requisition*.

3.3.3 Testing

Software Testing adalah pengeksekusian suatu aplikasi maupun program yang telah dikerjakan oleh *developer* atau pengembang. Pada umumnya, hal ini dilakukan untuk menguji perangkat lunak, serta menemukan *bug* sehingga kualitas dari *code* dapat ditingkatkan [6]. Testing pada aplikasi MRO ini dilakukan dengan menggunakan *blackbox testing*. *Blackbox testing* merupakan suatu pengujian yang hanya berfokus kepada kebutuhan fungsional aplikasi tersebut untuk memastikan bahwa fungsi, input, dan output telah sesuai dengan kebutuhan. *Blackbox testing* juga tidak memerlukan pemahaman mengenai bahasa pemrograman [7]. Pada *testing* yang dilakukan selama pelaksanaan kerja magang, *test cases* telah disiapkan sebelumnya. Selama *testing*, terdapat sejumlah 44 *test cases* dengan 41 kasus dengan status *passed*, 3 kasus dengan status *failed*, dan 1 kasus dengan status *untested*. Hasil *testing* yang dilakukan terhadap aplikasi dapat dilihat pada Tabel 3.4.

Tabel 3.4. Tabel *blackbox testing*

Test Case ID	**Title**	**Steps**	**Expected Result**	**Actual Result**	**Status**
WEB001	Create Material Receipt Document as Draft	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Material Receipt menu. 4) Click on Create New Material Receipt. 5) Fill in the required fields (Delivery Order No. & Port). 6) Click on Save Draft.	Draft Created	Draft Created	Passed
WEB002	Create Material Receipt Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Material Receipt menu. 4) Click on Create New Material Receipt. 5) Fill in the required fields (Delivery Order No. & Port). 6) Adjust the number of items received on the Qty Received column to match the number ordered from the column Qty Ordered. 7) Click on Add Remarks to add remarks regarding the product. 8) Click on Upload Document. 9) Click on Click to select file or drag and drop here. 10) Select document from local. 11) Click Add Document. 12) Click on Submit for Approval.	Document Created	Document Created	Passed
WEB003	Approval on Material Receipt Document (First Level Approval)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Material Receipt menu. 4) Click on icon on the Material Receipt document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Material Receipt / Reject Material Receipt.	Document Updated	Document Updated	Passed
WEB004	Approval on Material Receipt Document (Second Level Approval)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Material Receipt menu. 4) Click on icon on the Material Receipt document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Material Receipt / Reject Material Receipt.	Document Updated	Document Updated	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	**Title**	**Steps**	**Expected Result**	**Actual Result**	**Status**
WEB005	Print Material Receipt Document List as PDF	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Material Receipt menu. 4) Click on icon on the Material Receipt document.	PDF Printed with Document Attachment	PDF Printed with Document Attachment	Passed
WEB006	Export Material Receipt Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Material Receipt menu. 4) Click on the Export List button.	Excel Exported	Excel Exported	Passed
WEB007	View Stock Card	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Card menu. 4) Select Vessel then click on the View button.	Data Shown	Data Shown	Passed
WEB008	Export Stock Card List	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Card menu. 4) Select Vessel then click on the View button. 5) Click on the Export List button.	Excel Exported	Excel Exported Note: quantity is empty	Passed
WEB009	View Stock Card Detail	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Card menu. 4) Select Vessel then click on the View button. 5) Click on icon on the selected product.	Data Shown	Data Shown	Passed
WEB010	Export Stock Card Detail	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Card menu. 4) Select Vessel then click on the View button. “‘latex 5) Click on icon on the selected product. 6) Click on the Export List button.	Excel Exported	Excel Exported	Passed
WEB011	Create Stock Opname Document as Draft	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Opname menu. 4) Click on Create New Stock Opname. 5) Fill in the required field (Vessel). 6) Click on Save Draft.	Draft Created	Draft Created	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	Title	Steps	Expected Result	Actual Result	Status
WEB012	Create Stock Opname Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Opname menu. 4) Click on Create New Stock Opname. 5) Fill in the required field (Vessel). 6) Click on Add Product to add part in the Stock Opname document. 7) Select the product to be included in the Stock Opname document. 8) Click on Add Product. 9) Click on Add Remarks to add remarks regarding the product selected. 10) Click on Upload Document. 11) Click on Click to select file or drag and drop here. 12) Select document from local. 13) Click Add Document. 14) Click on Submit for Approval.	Document Created	Document Created	Passed
WEB013	Approval on Stock Opname Document (First Level Approval)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Opname menu. 4) Click on icon on the Stock Opname document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Stock Opname / Reject Stock Opname.	Document Updated	Document Updated	Passed
WEB014	Print Stock Opname Document as PDF	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Opname menu. 4) Click on icon on the Stock Opname document.	PDF Printed with Document Attachment	No Export Button when logged in as Manager	Failed
WEB015	Export Stock Opname Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Stock Opname menu. 4) Click on the Export List button.	Excel Exported	Excel Exported	Passed
WEB016	Create Internal Use Document as Draft	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Internal Use menu. 4) Click on Create Internal Use. 5) Fill in the required field (Account Code Type & Account code). 6) Click on Save Draft.	Draft Created	Draft Created	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	Title	Steps	Expected Result	Actual Result	Status
WEB017	Create Internal Use Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Internal Use menu. 4) Click on Create Internal Use. 5) Fill in the required field (Requisition Type & Account code). 6) Click on Add Part/Items to add part in the Internal Use document. 7) Choose First Locator, Second Locator, Drawing and Catalogue. 8) Click on View to list all products in the chosen catalogue. 9) Select the product to be included in the Internal Use document. 10) Click on Add Part/Item(s). 11) Click on Add Remarks to add remarks regarding the product selected. 12) Click on Upload Document. 13) Click on Click to select file or drag and drop here. 14) Select document from local. 15) Click Add Document. 16) Click on Submit for Approval.	Document Created	Document Created	Passed
WEB018	Approval on Internal Use Document (First Level Approval)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Internal Use menu. 4) Click on icon on the Internal use document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Internal use / Reject Internal Use.	Document Updated	Document Updated	Passed
WEB019	Approval on Internal Use Document (Second Level Approval)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Internal Use menu. 4) Click on icon on the Internal use document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Internal use / Reject Internal Use.	Document Updated	Document Updated	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	Title	Steps	Expected Result	Actual Result	Status
WEB020	Print Internal Use Document as PDF	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Internal Use menu. 4) Click on icon on the Internal Use document.	PDF Printed with Document Attachment	PDF Printed with Document Attachment	Passed
WEB021	Export Internal Use Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Internal Use menu. 4) Click on the Export List button.	Excel Exported	Excel Exported	Passed
WEB022	Create Misc. Receipt Document as Draft	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Misc. Receipt menu. 4) Click on Create Misc. Receipt. 5) Fill in the required field (Requisition Type & Account code). 6) Click on Save Draft.	Draft Created	Draft Created	Passed
WEB023	Create Misc. Receipt Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Misc. Receipt menu. 4) Click on Create Misc. Receipt. 5) Fill in the required field (Requisition Type & Account code). 6) Click on Add Part/Items to add part in the Misc. Receipt document. 7) Choose First Locator, Second Locator, Drawing and Catalogue. 8) Click on View to list all products in the chosen catalogue. 9) Select the product to be included in the Misc. Receipt document. 10) Click on Add Part/Item(s). 11) Click on Add Remarks to add remarks regarding the product selected. 12) Click on Upload Document. 13) Click on Click to select file or drag and drop here. 14) Select document from local. 15) Click Add Document. 16) Click on Submit.	Document Created	Document Created	Passed
WEB024	Print Misc. Receipt Document as PDF	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Misc. Receipt menu. 4) Click on icon on the Misc. Receipt document.	PDF Printed with Document Attachment	PDF Printed with Document Attachment	Passed
WEB025	Export Misc. Receipt Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Misc. Receipt menu. 4) Click on the Export List button.	Excel Exported	Excel Exported	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	Title	Steps	Expected Result	Actual Result	Status
WEB026	Create Intra Warehouse Document as Draft	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Intra Warehouse menu. 4) Click on Create Intra Warehouse. 5) Fill in the required field (Receiving Vessel / Warehouse & Receiving Department). 6) Click on Save Draft.	Draft Created	Draft Created	Passed
WEB027	Create Intra Warehouse Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Intra Warehouse menu. 4) Click on Create Intra Warehouse. 5) Fill in the required field (Receiving Vessel / Warehouse & Receiving Department). 6) Click on Add Product to add part in the Intra Warehouse document. 7) Select the product to be included in the Intra Warehouse document. 8) Click on Add Product. 9) Click on Add Remarks to add remarks regarding the product selected. 10) Click on Upload Document. 11) Click on Click to select file or drag and drop here. 12) Select document from local. 13) Click Add Document. 14) Click on Submit for Approval.	Document Created	Document Created Note: Draft can be edited by another user	Passed
WEB028	Approval on Intra Warehouse Document (First Level Approval - Source Warehouse)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Intra Warehouse menu. 4) Click on icon on the Intra Warehouse document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Intra Warehouse / Reject Intra Warehouse.	Document Updated	Document Updated	Passed
WEB029	Approval on Intra Warehouse Document (Second Level Approval - Destination Warehouse)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Intra Warehouse menu. 4) Click on icon on the Intra Warehouse document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Intra Warehouse / Reject Intra Warehouse.	Document Updated	Document Updated	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	Title	Steps	Expected Result	Actual Result	Status
WEB030	Approval on Intra Warehouse Document (Third Level Approval - Destination Warehouse)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Intra Warehouse menu. 4) Click on icon on the Intra Warehouse document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Intra Warehouse / Reject Intra Warehouse.	Document Updated	Document Updated	Passed
WEB031	Print Intra Warehouse Document as PDF	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Intra Warehouse menu. 4) Click on icon on the Intra Warehouse document.	PDF Printed with Document Attachment	PDF Printed with Document Attachment	Passed
WEB032	Export Intra Warehouse Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Intra Warehouse menu. 4) Click on the Export List button.	Excel Exported	Excel Exported	Passed
WEB033	Create Landing Form Document as Draft	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Landing Form menu. 4) Click on Create Landing Form. 5) Fill in the required field (Requisition Type, Account code & Warehouse). 6) Click on Save Draft.	Draft Created	Draft Created	Passed
WEB034	Create Landing Form Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Landing Form menu. 4) Click on Create Landing Form. 5) Fill in the required field (Requisition Type, Account code & Warehouse). 6) Click on Add Product to add part in the Landing Form document. 7) Select the product to be included in the Landing Form document. 8 “‘latex) Click on Add Product. 9) Click on Add Remarks to add remarks regarding the product selected. 10) Click on Upload Document. 11) Click on Click to select file or drag and drop here. 12) Select document from local. 13) Click Add Document. 14) Click on Submit for Approval.	Document Created	Document Created	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	Title	Steps	Expected Result	Actual Result	Status
WEB035	Approval on Landing Form Document (First Level Approval)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Landing Form menu. 4) Click on icon on the Landing Form document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Landing Form / Reject Landing Form.	Document Updated	Document updated but document's second approval is checked instead of first approval	Failed
WEB036	Approval on Landing Form Document (Second Level Approval)	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Landing Form menu. 4) Click on icon on the Landing Form document that Needs Review. 5) Click on Upload Document. 6) Click on Click to select file or drag and drop here. 7) Select document from local. 8) Click Add Document. 9) Click on Approve Landing Form / Reject Landing Form.	Document Updated		Untested
WEB037	Print Landing Form Document as PDF	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Landing Form menu. 4) Click on icon on the Landing Form document.	PDF Printed with Document Attachment	PDF Printed with Document Attachment	Passed
WEB038	Export Landing Form Document	1) Log on to the App. 2) Click on the Maintenance menu. 3) Click on the Landing Form menu. 4) Click on the Export List button.	Excel Exported	Excel Exported	Passed
WEB039	Print to PDF Feature on Vessel Requisition Document	1) Log on to the App. 2) Click on the Vessel Requisition menu. 3) Click on icon on the Vessel Requisition document.	PDF Printed with Document Attachment	PDF Printed with Document Attachment	Passed
WEB040	Addition of New Numbering Format for Vessel Requisition Document	1) Log on to the App. 2) Click on the Vessel Requisition menu.	Data Shown	Data Shown	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	Title	Steps	Expected Result	Actual Result	Status
WEB041	Categorization between Vessel Requisition Document for Docking or Maintenance Purpose	1) Log on to the App. 2) Click on the Vessel Requisition menu. 3) Click on Create New Requisition. 4) Fill in the required field (Requisition Type, Account code, Requisition Description). 5) Fill in Category to select between Docking or Maintenance in the Requisition document. 6) Click on Add Part/Items to add part in the Requisition document. 7) Choose First Locator, Second Locator, Drawing and Catalogue. 8) Click on View to list all products in the chosen catalogue. 9) Select the product to be included in the Vessel Requisition document. 10) Click on Add Part/Item(s). 11) Click on Add Remarks to add remarks regarding the product selected. 12) Click on Submit for Approval. 13) Continue approval process	Data Shown	Data Shown	Passed
WEB042	Last Order Date on product section when inputting Vessel Requisition Document	1) Log on to the App. 2) Click on the Vessel Requisition menu. 3) Click on Create New Requisition. 4) Fill in the required field (Requisition Type, Account code, Requisition Description). 5) Fill in Category to select between Docking or Maintenance in the Requisition document. 6) Click on Add Part/Items to add part in the Requisition document. 7) Choose First Locator, Second Locator, Drawing and Catalogue. 8) Click on View to list all products in the chosen catalogue. 9) Select the product to be included in the Vessel Requisition document. 10) Click on Add Part/Item(s).	Data Shown	Data Shown	Passed

Tabel 3.4. Tabel *blackbox testing* (Lanjutan)

Test Case ID	Title	Steps	Expected Result	Actual Result	Status
WEB043	Adding multiple product with the same product code when inputting Vessel Requisition document	1) Log on to the App. 2) Click on the Vessel Requisition menu. 3) Click on Create New Requisition. 4) Fill in the required field (Requisition Type, Account code, Requisition Description). 5) Fill in Category to select between Docking or Maintenance in the Requisition document. 6) Click on Add Part/Items to add part in the Requisition document. 7) Choose First Locator, Second Locator, Drawing and Catalogue. 8) Click on View to list all products in the chosen catalogue. 9) Select the product to be included in the Vessel Requisition document. Only product with Expense Type that can be added multiple into the requisition document. 10) Click on Add Part/Item(s).	Data Shown	Data Shown	Passed
WEB044	Backdate Feature on Vessel Requisition Document	1) Log on to the App. 2) Click on the Vessel Requisition menu. 3) Click on icon on the Vessel Requisition document that Needs Review. 4) Choose a date for the requisition document date. 5) Click on Approve Requisition / Reject Requisition.	Document Updated	Document Updated	Passed

3.4 Kendala dan Solusi yang Ditemukan

Selama melakukan kegiatan magang, terdapat beberapa kendala yang ditemukan, yaitu:

1. *Code* sebelumnya yang berantakan, mulai dari banyaknya *code* yang ditulis berulang, dan penulisan *code* yang tidak efektif pada beberapa *file*.
2. Kurangnya komunikasi antar anggota dalam tim sehingga menghambat alur pengerjaan, seperti isi *response* API yang tidak sesuai dengan kebutuhan divisi *frontend*.

Selain kendala, terdapat pula solusi yang dilakukan untuk mengatasi kendala tersebut, yaitu:

1. Menanyakan kepada *supervisor* mengenai *code* yang berantakan sehingga dapat lebih cepat dimengerti.

2. Meningkatkan komunikasi antar anggota dalam tim dengan sering membahas *project*, khususnya pada API.