

BAB IV

IMPLEMENTASI DAN PENJELASAN

4.1 Implementasi dan Uji Coba Sistem

Berdasarkan analisis dan perancangan sistem yang telah dibuat pada bab sebelumnya, maka untuk bab ini akan dibahas mengenai implementasi dan pembahasan secara menyeluruh dengan menyertakan tampilan aplikasi dan potongan kode program.

4.1.1 User Services Admin

```
public function login(Request $request)
{
    $this->validate($request, [
        'username' => 'required|string',
        'password' => 'required|string',
    ]);

    $credentials = $request->only(['username',
    'password']);

    if (! $token = auth()->attempt($credentials)) {
        $request->message = 'Unauthorized';
        return (new ErrorResource($request))->response()-
>setStatusCode(401);
    }

    $request->message = 'Success';
    $request->data = [
        'access_token' => $token,
        'token_type' => 'bearer',
        'expires_in' => auth()->factory()->getTTL() * 60
    ];
    return (new GlobalResource($request))->response()-
>setStatusCode(200);
}
```

Gambar 4. 1 User Service Admin Login

Pada Gambar 4.1 adalah fungsi login yang memvalidasi input pengguna (username dan password), mencoba mengautentikasi menggunakan kredensial yang

diberikan, dan menghasilkan token JWT jika berhasil. Jika autentikasi gagal, fungsi mengembalikan respons error dengan status 401 (Unauthorized). Jika berhasil, fungsi mengembalikan respons sukses dengan status 200, berisi token akses, tipe token, dan waktu kedaluwarsa token.

4.1.2 Rest Api Admin Register

```
public function create(Request $request)
{
    // Only ADMIN
    $payload = auth()->payload();
    $user = User::find($payload->get('sub'));
    if ($user->role != 'ADMIN') {
        $request->message = 'Only ADMIN users can
create';
        return (new ErrorResource($request))-
>response()->setStatusCode(400);
    }

    // Check if username is already
    $user = User::firstWhere('username', $request-
>username);
    if ($user) {
        $request->message = 'Username already exists';
        return (new ErrorResource($request))-
>response()->setStatusCode(400);
    }

    $newUser = new User();
    $newUser->username = $request->username;
```

```

        $newUser->password = Hash::make($request->password,
[
            'rounds' => 15,
        ]);
        $newUser->role = "USER";

        $newUser->save();

        $request->message = 'Success';
        $request->data = $newUser;
        return (new GlobalResource($request))->response()-
>setStatusCode(201);
    }

```

Gambar 4. 2 Admin Register

Pada Gambar 4.2 merupakan adalah fungsi untuk membuat pengguna baru, hanya dapat dilakukan oleh pengguna dengan peran *ADMIN*. Fungsi memeriksa apakah pengguna yang mencoba membuat akun adalah *ADMIN*, lalu memastikan username yang dimasukkan belum digunakan. Jika valid, fungsi membuat akun baru dengan peran *USER*, menyimpan data ke database, dan mengembalikan respons sukses dengan status 201. Jika ada pelanggaran aturan, respons error dikembalikan dengan status 400.

4.1.3 Rest Api GET Item

```

public function index(Request $request)
{
    $qSort = $request->query('sort');
    $qName = $request->query('name');
}

```

```

        $items = Item::select('*');

        if ($qSort != null) $items = $items-
>orderBy('updated_at', $qSort);

        else $items = $items->orderBy('updated_at',
'desc');

        if ($qName != null) $items = $items-
>where('items.name', 'ilike', '%'.$qName.'%');

        $items = $items->get();

        $request->message = 'Success';

        $request->data = $items;

        $request->meta = null;

        return (new GlobalResource($request))->response()-
>setStatusCode(200);

    }

```

Gambar 4.3 GET Item

Pada Gambar 4.3 merupakan fungsi untuk menampilkan daftar item dengan fitur pencarian berdasarkan nama dan pengurutan berdasarkan waktu pembaruan. Jika parameter sort disediakan, daftar item akan diurutkan sesuai arahnya, jika tidak, diurutkan secara default dalam urutan menurun. Jika parameter name disediakan, fungsi akan memfilter item berdasarkan nama yang mengandung kata kunci tersebut. Hasilnya dikembalikan dengan respons sukses berstatus 200.

4.1.4 Rest Api Get Total Jumlah Item

```

public function countTotalItems(Request $request)
{
    $items = Item::select(

```

```

        'items.*',

        DB::raw("CAST(coalesce((select sum(total) from
transactions where items_id = items.id and type = 'IN'), 0)
- coalesce((select sum(total) from transactions where
items_id = items.id and type = 'OUT'), 0) as INTEGER) as
total")

    )->get();

    $request->message = 'Success';

    $request->data = $items;

    $request->meta = null;

    return (new GlobalResource($request))->response()-
>setStatusCode(200);

}

```

Gambar 4.4 Total Jumlah Item

Pada Gambar 4.4 merupakan fungsi untuk menghitung total jumlah setiap item dengan menghitung selisih antara total transaksi masuk (type = 'IN') dan keluar (type = 'OUT) berdasarkan item ID. Hasilnya berupa daftar item dengan total yang dihitung, yang dikembalikan dalam respons sukses berstatus 200.

4.1.5 Rest Api Post Item

```

public function create(Request $request)
{
    // Check if item is already
    $checkItem = Item::firstWhere('name', $request-
>name);
    if ($checkItem) {
        $request->message = 'Item already exists';
        return (new ErrorResource($request))-
>response()->setStatusCode(400);
    }
}

```

```

        $newItem = new Item();
        $newItem->name = $request->name;
        $newItem->created_by = $request->created_by;
        $newItem->save();

        $request->message = 'Success';
        $request->data = $newItem;
        return (new GlobalResource($request))->response()->setStatusCode(201);
    }

```

Gambar 4.5 Api Post Item

Pada Gambar 4.5 merupakan fungsi untuk menambahkan item baru. Fungsi memeriksa apakah item dengan nama yang sama sudah ada di database. Jika ya, respons error dengan status 400 dikembalikan. Jika tidak, item baru dibuat dan disimpan, lalu respons sukses dengan data item dan status 201 dikembalikan.

4.1.6 Rest Api Put Item

```

public function update(Request $request, $id)
    {
        // Check if item is already
        $checkItem = Item::firstWhere('name', $request->name);
        if ($checkItem) {
            $request->message = 'Item already exists';
            return (new ErrorResource($request))->response()->setStatusCode(400);
        }
    }

```

```

        $newItem = Item::find($id);
        $newItem->name = $request->name;
        $newItem->save();

        $request->message = 'Success';
        $request->data = $newItem;

        return (new GlobalResource($request))->response()-
>setStatusCode(200);
    }

```

Gambar 4. 6 Put Item

Pada Gambar 4.6 merupakan fungsi untuk memperbarui data item berdasarkan ID. Fungsi memeriksa apakah nama item yang dimasukkan sudah ada di database. Jika ya, respons error dengan status 400 dikembalikan. Jika tidak, item dengan ID yang sesuai diperbarui namanya, disimpan, dan respons sukses dengan data item yang diperbarui dikembalikan dengan status 200.

4.1.7 Rest Api Delete Item

```

public function delete(Request $request, $id)
    {
        $checkItem = Item::find($id);
        if (!$checkItem) {
            $request->message = 'Item not found';
            return (new ErrorResource($request))-
>response()->setStatusCode(404);
        }
        $checkItem->delete();
    }

```

```

        $request->message = 'Success';

        $request->data = $checkItem;

        return (new GlobalResource($request))->response()-
>setStatusCode(200);

    }

```

Gambar 4.7 Delete Item

Pada Gambar 4.7 merupakan fungsi untuk menghapus item berdasarkan ID. Fungsi memeriksa apakah item dengan ID yang diberikan ada di database. Jika tidak ditemukan, respons error dengan status 404 dikembalikan. Jika ditemukan, item dihapus, dan respons sukses dengan data item yang dihapus dikembalikan dengan status 200.

4.1.8 Rest Api Get Transaction

```

public function index(Request $request)
{
    $qSort = $request->query('sort');
    $qName = $request->query('name');

    $transactions = Transaction::join('items',
'items.id', '=', 'transactions.items_id');

    if ($qSort != null) $transactions = $transactions-
>orderBy('date', $qSort);
    else $transactions = $transactions->orderBy('date',
'desc');

    if ($qName != null) $transactions = $transactions-
>where('items.name', 'ilike', '%'.$qName.'%');
}

```



```

        $transactions = $transactions-
>get(['transactions.*', 'items.name as items_name']);

        $request->message = 'Success';

        $request->data = $transactions;

        $request->meta = null;

        return (new GlobalResource($request))->response()-
>setStatusCode(200);

    }

```

Gambar 4. 8 Get Transaction

Pada Gambar 4.8 merupakan fungsi untuk menampilkan daftar transaksi dengan fitur pencarian berdasarkan nama item dan pengurutan berdasarkan tanggal. Jika parameter sort disediakan, transaksi akan diurutkan sesuai arahnya, jika tidak, diurutkan dalam urutan menurun. Jika parameter name disediakan, transaksi akan difilter berdasarkan nama item yang mengandung kata kunci tersebut. Hasilnya, termasuk nama item, dikembalikan dalam respons sukses dengan status 200.

4.1.9 Rest Api Post Transaction

```

public function create(Request $request)
{
    // check trx type out
    if ($request->type == "OUT") {
        $checkTrxIn =
Transaction::select(DB::raw('SUM(total) as total'))-
>where("items_id", "=", $request->items_id)->where("type",
"=", "IN")->first();

        $checkTrxOut =
Transaction::select(DB::raw('SUM(total) as total'))-
>where("items_id", "=", $request->items_id)->where("type",
"=", "OUT")->first();
    }
}

```

```

        // check no stock
        $total = $checkTrxIn->total - $checkTrxOut-
>total;

        if ($total < 1) {
            $request->message = 'No Stocks';
            return (new ErrorResource($request))-
>response()->setStatusCode(400);
        }

        // check request total
        $total = $total - $request->total;
        if ($total < 0) {
            $request->message = 'No Stocks';
            return (new ErrorResource($request))-
>response()->setStatusCode(400);
        }
    }

    // Save Data
    $transaction = new Transaction();
    $transaction->items_id = $request->items_id;
    $transaction->users_id = $request->users_id;
    $transaction->type = $request->type;
    $transaction->total = $request->total;
    $transaction->date = $request->date;
    $transaction->created_by = $request->created_by;
    $transaction->save();

    $request->message = 'Success';
    $request->data = $transaction;
    return (new GlobalResource($request))->response()-
>setStatusCode(201);
}

```

Gambar 4.9 Post Transaction

Pada Gambar 4.9 merupakan fungsi untuk membuat transaksi, dengan pengecekan stok item jika jenis transaksi adalah OUT. Fungsi memeriksa total transaksi masuk (IN) dan keluar (OUT) untuk memastikan ada stok yang cukup sebelum menyetujui transaksi keluar. Jika stok tidak mencukupi, respons error dengan status 400 dan pesan No Stocks dikembalikan. Jika stok cukup, data transaksi disimpan, dan respons sukses dengan status 201 beserta detail transaksi dikembalikan.

4.1.10 Rest Api Delete Transaction

```
public function delete(Request $request, $id)
{
    $transaction = Transaction::find($id);
    if (!$transaction) {
        $request->message = 'Transaction not found';
        return (new ErrorResource($request))-
>response()->setStatusCode(404);
    }
    $transaction->delete();

    $request->message = 'Success';
    $request->data = $transaction;
    return (new GlobalResource($request))->response()-
>setStatusCode(200);
}
```

Gambar 4. 10 Delete Transaction

Pada Gambar 4.10 merupakan fungsi untuk menghapus transaksi berdasarkan ID. Fungsi memeriksa apakah transaksi dengan ID yang diberikan ada di database. Jika tidak ditemukan, respons error dengan status 404 dan pesan Transaction not found

dikembalikan. Jika ditemukan, transaksi dihapus, dan respons sukses dengan status 200 beserta data transaksi yang dihapus dikembalikan.

4.1.11 Server.js Untuk Vue Admin

```
const express = require('express')
const serveStatic = require('serve-static')
const path = require('path')

const app = express()

//here we are configuring dist to serve app files
app.use('/', serveStatic(path.join(__dirname, '/dist')))

// this * route is to serve project on different page
routes except root /
app.get(/.*/ , function (req, res) {
  res.sendFile(path.join(__dirname, '/dist/index.html'))
})

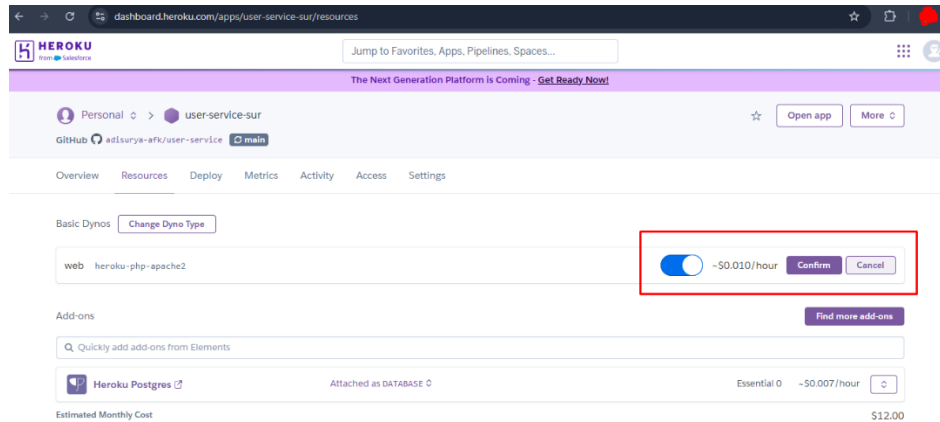
const port = process.env.PORT || 8080
app.listen(port)
console.log('app is listening on port: ${port}')
```

Gambar 4. 11 Vue Admin

Pada gambar 4.11 merupakan program untuk menyajikan aplikasi web kita yang sudah jadi. Semua berkas statis seperti HTML, CSS, dan JavaScript diambil dari direktori. Jika pengguna meminta halaman yang tidak ada, secara otomatis akan diarahkan ke halaman utama (index.html), yang biasanya menjadi pintu masuk utama aplikasi.

4.2 Pembahasan Sistem

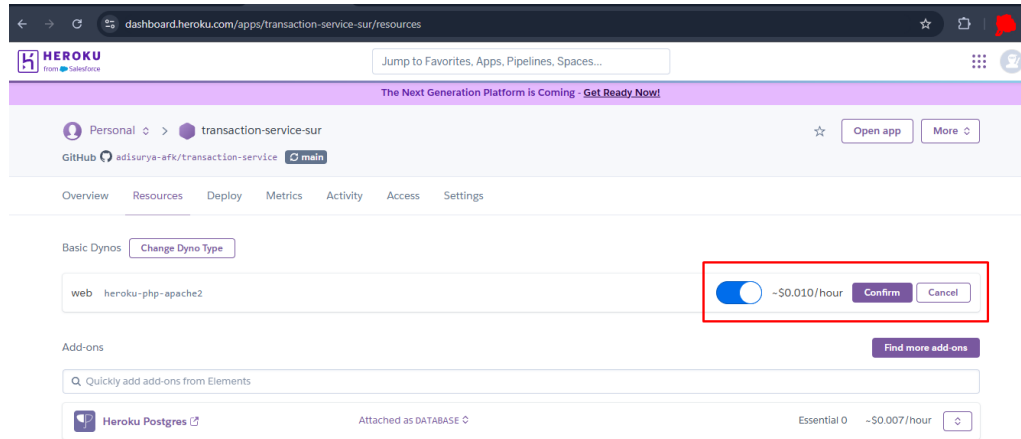
4.2.1 Heroku User Services



Gambar 4. 12 User Serice

Pada Gambar 4.12 merupakan layanan backend yang mengelola data pengguna. Untuk menjalankan aplikasi ini di Heroku, perlu memastikan bahwa user-service sudah dideploy dengan benar menggunakan Node.js. Perintah `npm start` pada user-service akan memulai server dan melayani permintaan terkait pengguna, seperti autentikasi, pendaftaran, dan manajemen profil.

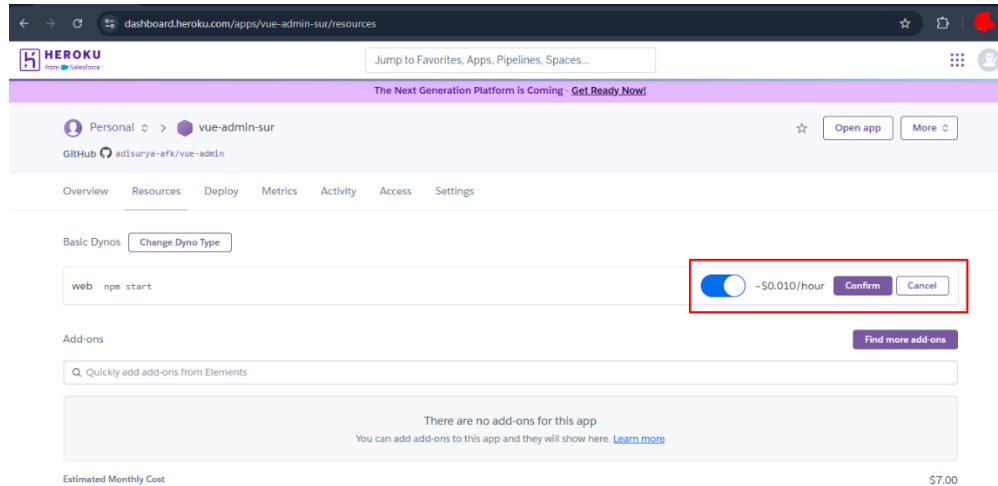
4.2.2 Heroku Transaction Services



Gambar 4. 13 Halaman Transaction-Service

Pada Gambar 4.13 merupakan Layanan backend ini menangani semua transaksi yang dilakukan oleh pengguna. Dengan menjalankan `npm start` pada `transaction-service`, Heroku akan mengaktifkan server yang bertanggung jawab untuk mencatat dan memproses transaksi, mengelola data transaksi, serta berinteraksi dengan database atau API eksternal untuk memastikan proses transaksi berjalan dengan lancar.

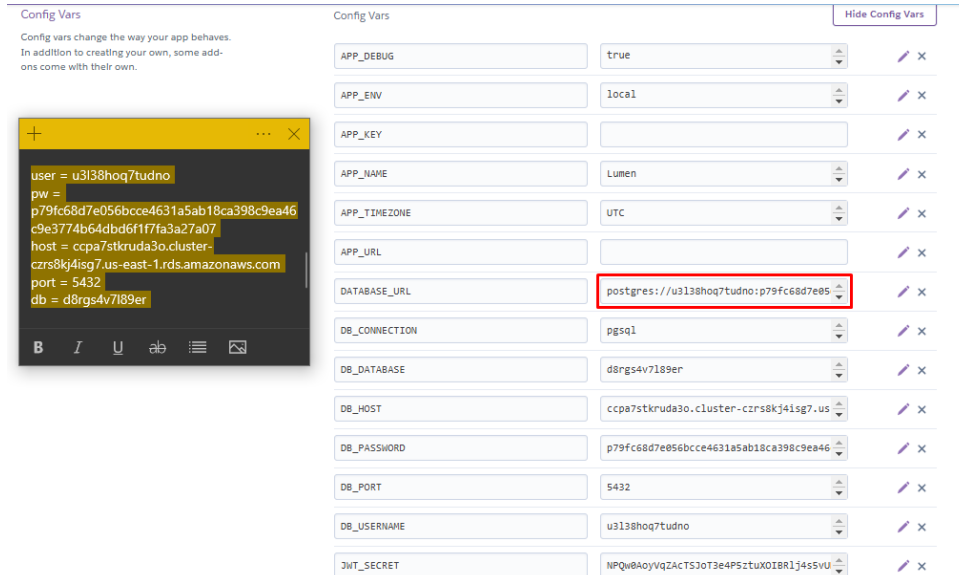
4.2.3 Heroku Vue Admin



Gambar 4. 14 Heroku Vue Admin

Pada Gambar 4.14 merupakan aplikasi frontend berbasis Vue.js yang digunakan untuk mengelola dan menampilkan data dari user-service dan transaction-service. Setelah aplikasi di-build dengan `npm run build`, file statis frontend akan didploy ke Heroku. Ketika dijalankan dengan `npm start`, frontend akan diaktifkan dan siap untuk berinteraksi dengan backend untuk menampilkan informasi dan melakukan manajemen data secara dinamis melalui antarmuka pengguna (UI).

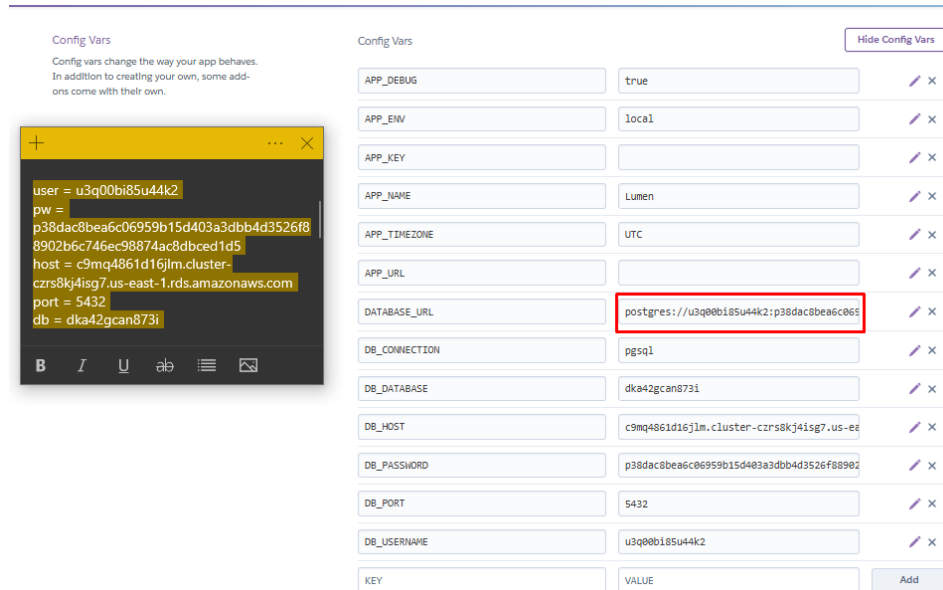
4.2.4 Halaman Config Vars User



Gambar 4.15 Config Vars User

Pada Gambar 4.15 merupakan halaman Config Vars di Heroku berada di menu Settings pada aplikasi Anda. Halaman ini digunakan untuk menambahkan, melihat, dan mengelola variabel konfigurasi yang berisi data yang serupa dengan file .env, seperti kredensial database atau API keys. Dengan mengklik tombol Reveal Config Vars, Anda dapat melihat atau menambahkan key-value pair yang digunakan oleh aplikasi Anda. Halaman ini memastikan informasi sensitif tersimpan dengan aman dan mudah diakses melalui environment variables dalam aplikasi.

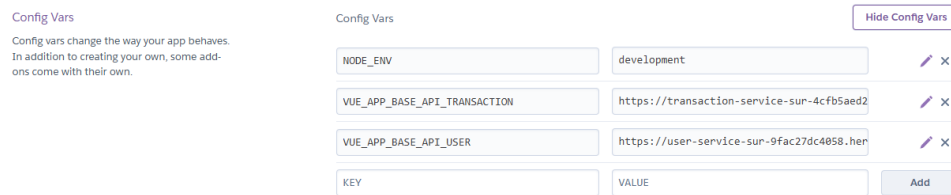
4.2.5 Halaman Config Vars Transaction



Gambar 4. 16 Convig Vars Transaction

Pada Gambar 4.16 merupakan halaman Config Vars di Heroku berada di menu Settings pada aplikasi Anda. Halaman ini digunakan untuk menambahkan, melihat, dan mengelola variabel konfigurasi yang berisi data yang serupa dengan file .env, seperti kredensial database atau API keys. Dengan mengklik tombol Reveal Config Vars, Anda dapat melihat atau menambahkan key-value pair yang digunakan oleh aplikasi Anda. Halaman ini memastikan informasi sensitif tersimpan dengan aman dan mudah diakses melalui environment variables dalam aplikasi.

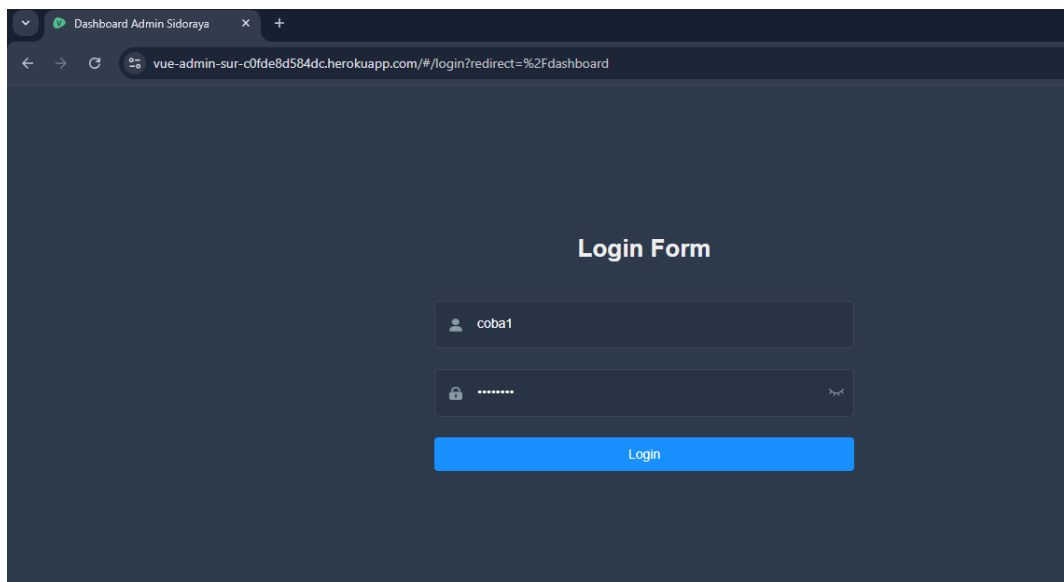
4.2.6 Halaman Config Vars Vue Admin



Gambar 4. 17 Config Vars Vue Admin

Pada Gambar 4.17 merupakan halaman Config Vars di Heroku berada di menu Settings pada aplikasi Anda. Halaman ini digunakan untuk menambahkan, melihat, dan mengelola variabel konfigurasi yang berisi data yang serupa dengan file .env

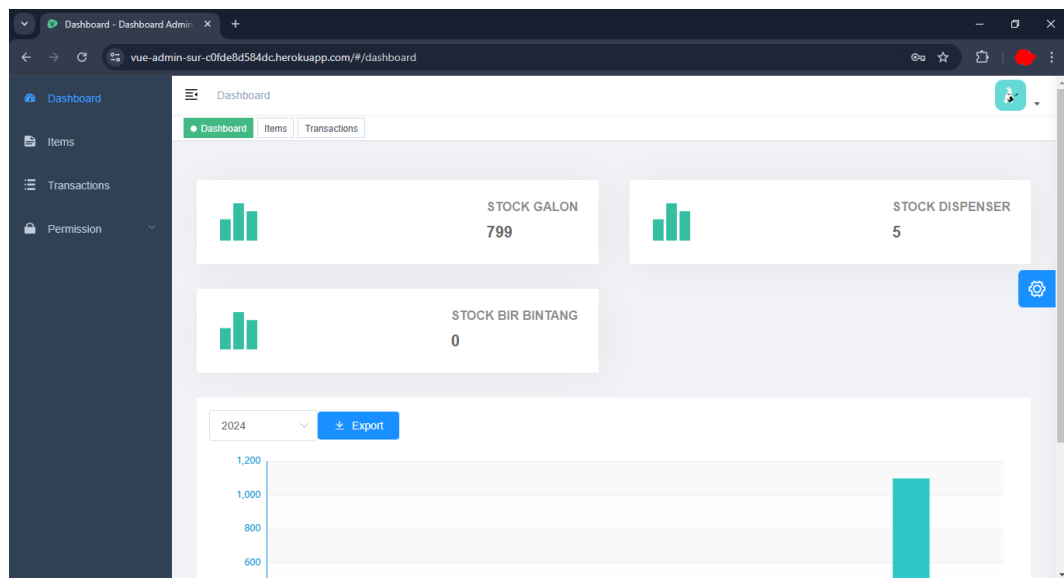
4.2.7 Halaman Login (Admin)



Gambar 4. 18 Halaman Login

Gambar 4.18 merupakan template Vue.js untuk form login menggunakan Element UI. Form login terdiri dari dua input, yaitu untuk username dan password, dengan validasi dan aturan form yang disesuaikan. Input password memiliki fitur untuk menampilkan atau menyembunyikan teks password dengan ikon eye. Jika caps lock aktif, akan muncul tooltip untuk mengingatkan pengguna. Form ini juga mendukung login menggunakan tombol yang memanggil metode `handleLogin`. Selain itu, ada dialog yang dapat muncul untuk menyarankan metode login sosial (meskipun tidak dapat disimulasikan secara lokal).

4.2.8 Halaman Dashbord (Admin)

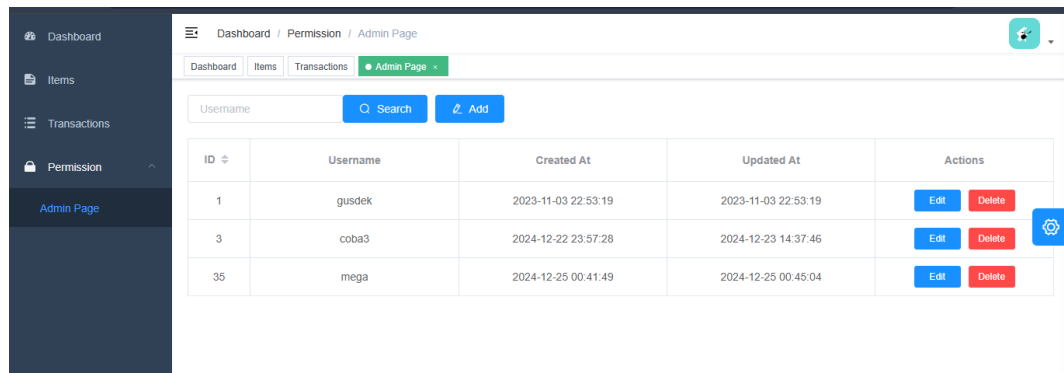


Gambar 4. 19 Halaman Dashboard

Pada Gambar 4.19 template untuk halaman dasbor admin menggunakan Vue.js dan Element UI. Halaman ini terdiri dari beberapa komponen: `PanelGroup`,

ExportReport, dan BarChart. Komponen PanelGroup ditempatkan di bagian atas halaman, diikuti oleh dua kolom utama dalam sebuah el-row: satu untuk menampilkan komponen ExportReport dan satu lagi untuk menampilkan grafik batang menggunakan BarChart. Beberapa komponen lain seperti LineChart, RaddarChart, PieChart, dan TransactionTable telah diimpor tetapi tidak digunakan dalam template ini. Dasbor ini dirancang untuk menampilkan data dan memungkinkan ekspor laporan melalui ExportReport.

4.2.9 Halaman Data User (Admin)



The screenshot shows a web application interface for an Admin Page. On the left is a dark sidebar with navigation items: Dashboard, Items, Transactions, Permission, and Admin Page (highlighted). The main content area has a breadcrumb trail: Dashboard / Permission / Admin Page. Below the breadcrumb are tabs: Dashboard, Items, Transactions, and Admin Page (active). A search bar with 'Username' placeholder and 'Search' and 'Add' buttons is present. Below the search bar is a table with the following data:

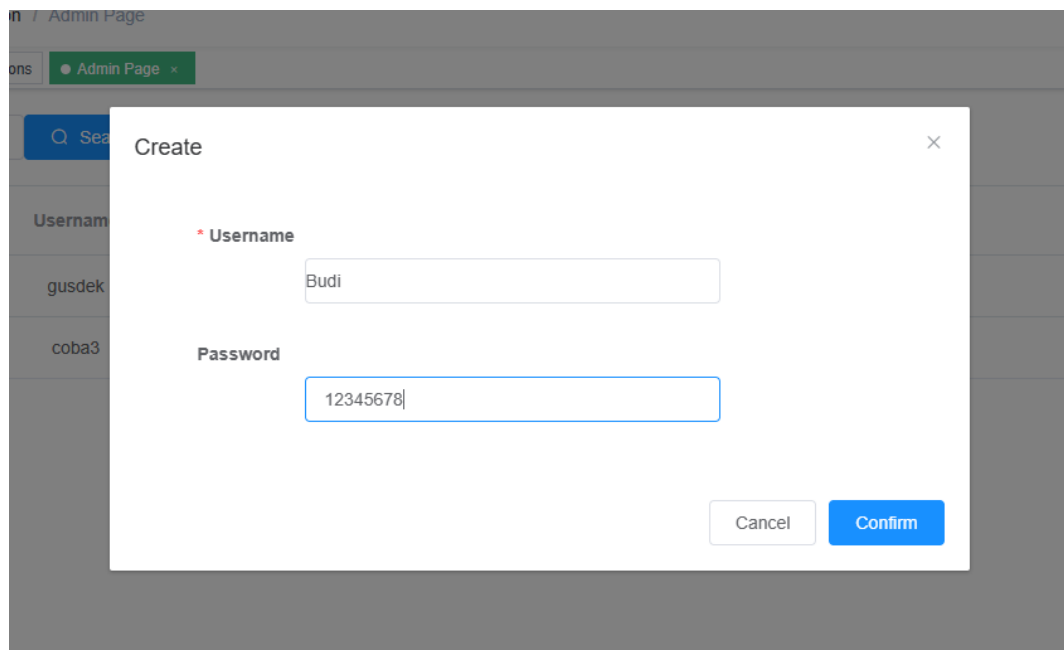
ID	Username	Created At	Updated At	Actions
1	gusdek	2023-11-03 22:53:19	2023-11-03 22:53:19	Edit Delete
3	coba3	2024-12-22 23:57:28	2024-12-23 14:37:46	Edit Delete
35	mega	2024-12-25 00:41:49	2024-12-25 00:45:04	Edit Delete

Gambar 4. 20 Halaman Data User

Pada Gambar 4.20 adalah Halaman data admin ini menggunakan Vue.js dan Element UI untuk menampilkan informasi profil pengguna dengan komponen-komponen yang terstruktur rapi. Di sisi kiri halaman, terdapat komponen *UserCard* yang menunjukkan data pengguna seperti nama, avatar, dan peran. Di sisi kanan, terdapat *el-card* yang berisi *el-tabs* dengan tiga tab: *Activity*, *Timeline*, dan *Account*. Setiap tab menampilkan komponen yang relevan, seperti *Activity* untuk riwayat

aktivitas, *Timeline* untuk garis waktu, dan *Account* untuk informasi akun pengguna. Data pengguna diambil dari store *Vuex* dan ditampilkan di dalam halaman menggunakan metode *getUser()*.

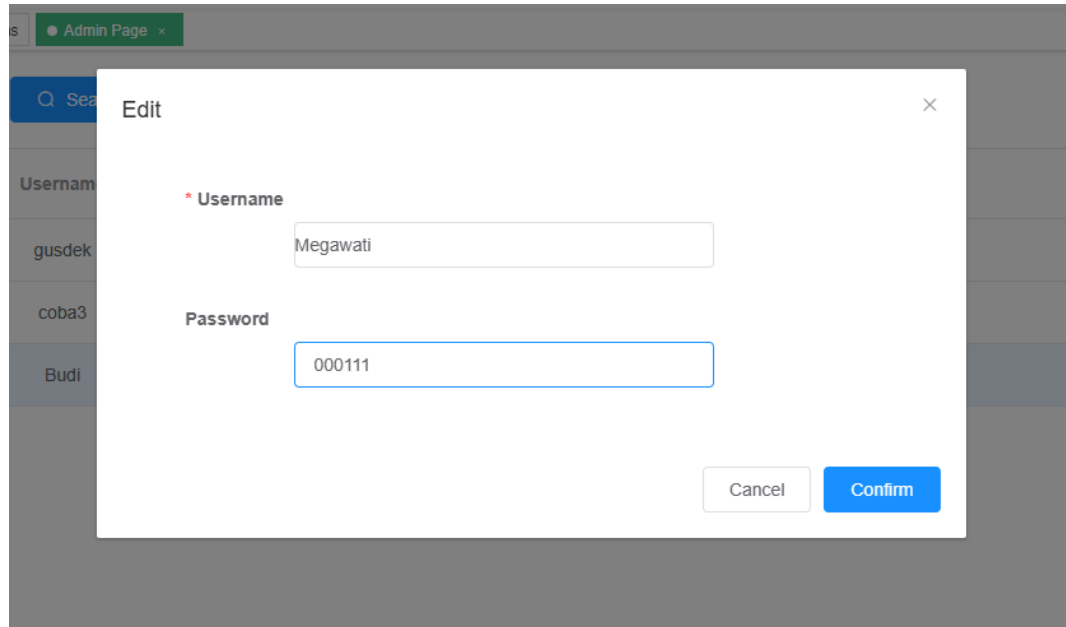
4.2.10 Halaman Registrasi Account User (Admin)

A screenshot of a web application interface showing a modal window titled "Create". The modal contains two form fields: "Username" with the value "Budi" and "Password" with the value "12345678". There are "Cancel" and "Confirm" buttons at the bottom right of the modal. The background shows a sidebar with "Admin Page" selected and a search bar.

Gambar 4. 21 Registrasi Account User Oleh Admin

Pada Gambar 4.21 merupakan halaman registrasi akun admin ini menggunakan Vue.js dan Element UI untuk menangani pembaruan data pengguna. Terdapat el-form dengan dua form item: satu untuk Name dan satu lagi untuk Email, keduanya mengikat data menggunakan v-model.

4.2.11 Halaman Edit Account User (Admin)

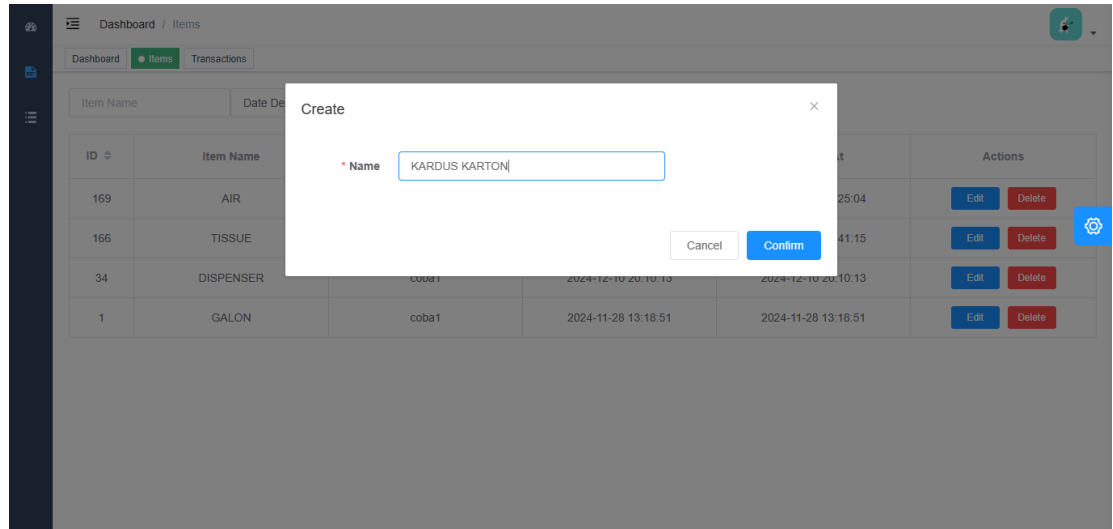


The image shows a web browser window with a tab titled 'Admin Page'. A modal dialog box titled 'Edit' is open, containing two input fields. The first field is labeled '* Username' and contains the text 'Megawati'. The second field is labeled 'Password' and contains the text '000111'. At the bottom right of the modal, there are two buttons: 'Cancel' and 'Confirm'.

Gambar 4. 22 Edit Account User Oleh Admin

Pada Gambar 4.22 merupakan halaman edit data pengguna mengklik tombol Update, metode submit() dijalankan, yang menampilkan pesan sukses menggunakan this.\$message(), mengindikasikan bahwa informasi pengguna telah berhasil diperbarui. Komponen ini menerima objek user sebagai properti, yang memungkinkan pengisian nilai input dengan data yang sudah ada sebelumnya.

4.2.12 Halaman Create Data Item



Gambar 4. 23 Create Data Item

Pada Gambar 4.23 merupakan halaman ini menyediakan fungsionalitas untuk membuat, menampilkan, mengedit, dan menghapus data menggunakan Vue.js dengan integrasi Element UI. Untuk membuat data, pengguna dapat mengklik tombol "Add", yang membuka dialog form untuk memasukkan informasi baru. Data yang dimasukkan akan disubmit menggunakan metode `createData()`, yang melakukan validasi input dan kemudian memanggil API untuk menyimpan data. Jika berhasil, pengguna akan diberi notifikasi dan data yang ditambahkan akan ditampilkan di tabel.

4.2.13 Halaman Data Item

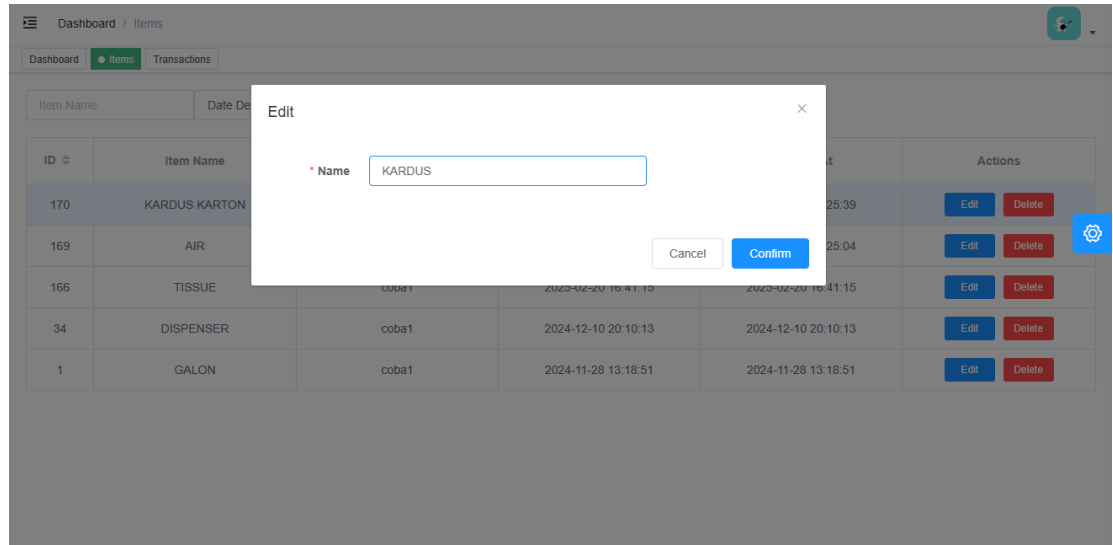
The screenshot shows a web application interface for managing items. At the top, there is a navigation bar with 'Dashboard / Items' and a user profile icon. Below this is a breadcrumb trail with 'Dashboard', 'Items', and 'Transactions'. A search bar is located above the table, with a dropdown menu set to 'Date Descendi'. The table itself has six columns: 'ID', 'Item Name', 'Created by', 'Created At', 'Updated At', and 'Actions'. Each row represents an item with its respective details and 'Edit' and 'Delete' buttons. A settings gear icon is visible on the right side of the table.

ID	Item Name	Created by	Created At	Updated At	Actions
170	KARDUS KARTON	coba1	2025-02-21 12:25:39	2025-02-21 12:25:39	Edit Delete
169	AIR	coba1	2025-02-20 19:25:04	2025-02-20 19:25:04	Edit Delete
166	TISSUE	coba1	2025-02-20 16:41:15	2025-02-20 16:41:15	Edit Delete
34	DISPENSER	coba1	2024-12-10 20:10:13	2024-12-10 20:10:13	Edit Delete
1	GALON	coba1	2024-11-28 13:18:51	2024-11-28 13:18:51	Edit Delete

Gambar 4. 24 Halaman Data Item

Pada Gambar 4.24 adalah tabel menampilkan data yang sudah ada dalam format yang mudah dibaca, termasuk kolom ID, Nama Item, Waktu Dibuat, dan Waktu Diperbarui. Pengguna dapat melakukan pencarian dengan memfilter berdasarkan nama item atau memilih urutan pengurutan berdasarkan tanggal. Setiap baris memiliki tombol aksi untuk mengedit atau menghapus item. Mengklik tombol "Edit" akan membuka dialog dengan data item yang dapat diubah, dan mengklik tombol "Delete" akan menghapus item tersebut setelah konfirmasi. Pembaruan dan penghapusan data dilakukan melalui API, dan tabel akan diperbarui setelah aksi tersebut berhasil.

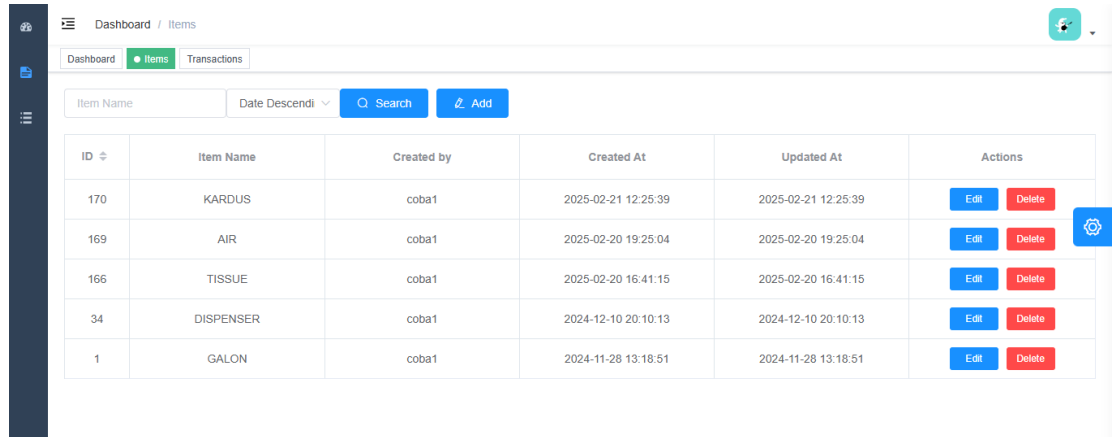
4.2.14 Halaman Edit Item



Gambar 4. 25 Halaman Edit Item

Pada Gambar 4.25 adalah dialog form juga digunakan untuk mengedit data yang sudah ada. Ketika tombol "Edit" diklik, data yang dipilih akan dimuat ke dalam form, memungkinkan pengguna untuk memperbaiki informasi. Setelah pengeditan selesai, tombol "Confirm" akan memanggil metode `updateData()`, yang melakukan validasi dan mengirimkan permintaan pembaruan ke API. Setelah data diperbarui, tabel akan diperbarui dengan data terbaru. Selain itu, fitur pagination memungkinkan pengguna untuk menjelajahi daftar data yang lebih besar, dan fungsionalitas ekspor memungkinkan data untuk diunduh dalam format Excel.

4.2.15 Halaman User Supervisor

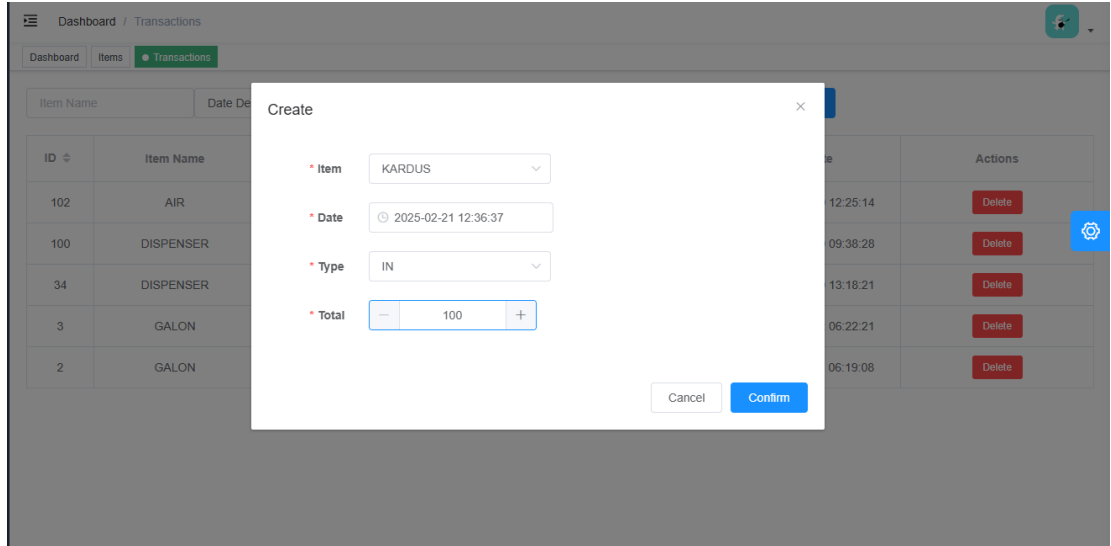


ID	Item Name	Created by	Created At	Updated At	Actions
170	KARDUS	coba1	2025-02-21 12:25:39	2025-02-21 12:25:39	Edit Delete
169	AIR	coba1	2025-02-20 19:25:04	2025-02-20 19:25:04	Edit Delete
166	TISSUE	coba1	2025-02-20 16:41:15	2025-02-20 16:41:15	Edit Delete
34	DISPENSER	coba1	2024-12-10 20:10:13	2024-12-10 20:10:13	Edit Delete
1	GALON	coba1	2024-11-28 13:18:51	2024-11-28 13:18:51	Edit Delete

Gambar 4. 26 Halaman User Supervisor

Pada Gambar 4.26 merupakan peran Supervisor memiliki tanggung jawab untuk memantau, mengelola, dan mengawasi aktivitas pengguna lainnya dalam sistem. Supervisor dapat mengakses data yang lebih mendalam, melakukan verifikasi dan validasi atas proses yang dilakukan oleh anggota tim lainnya, serta memastikan bahwa semua tindakan sesuai dengan prosedur yang berlaku. Supervisor juga memiliki hak untuk menyetujui atau menolak permintaan terkait data atau aktivitas penting, serta memberikan arahan atau bimbingan untuk meningkatkan kinerja operasional. Peran ini bertujuan untuk menjaga kualitas dan konsistensi dalam alur kerja sistem.

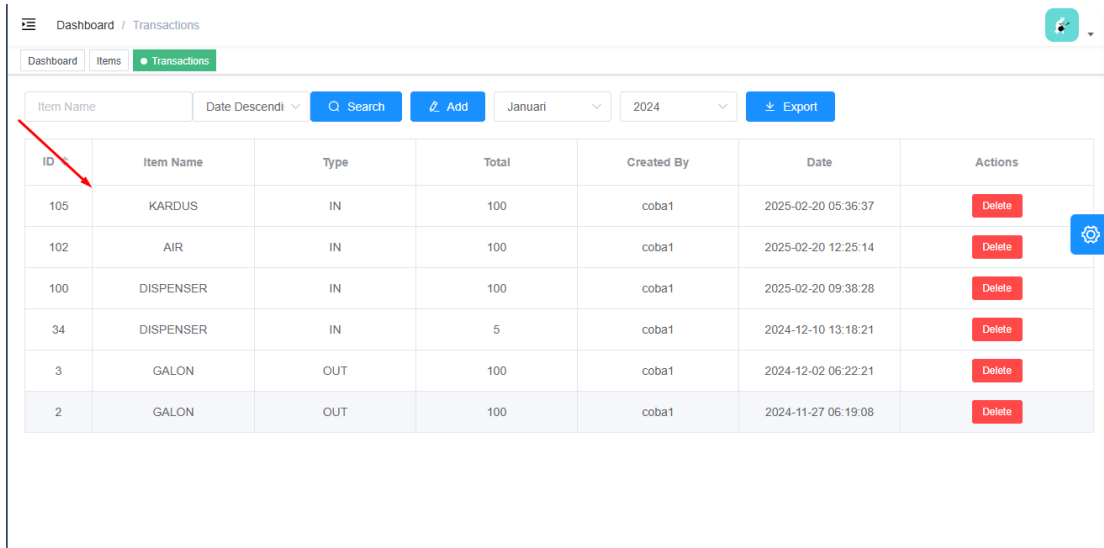
4.2.16 Halaman User Create Transaction (Supervisor)



Gambar 4. 27 Halaman User Create Transaction

Pada Gambar 4.27 merupakan halaman ini terdapat tombol "Add" yang memungkinkan supervsior untuk menambahkan transaksi baru. Ketika tombol ini diklik, formulir untuk memasukkan data transaksi akan muncul dalam dialog box. Supervesior dapat memilih item, jenis transaksi, tanggal, dan total transaksi yang ingin ditambahkan. Setelah formulir diisi, data akan dikirim ke server untuk disimpan melalui metode `createData()`, yang memvalidasi input sebelum mengirim permintaan untuk membuat transaksi baru.

4.2.17 Halaman User Data Transaction (Supervisor)

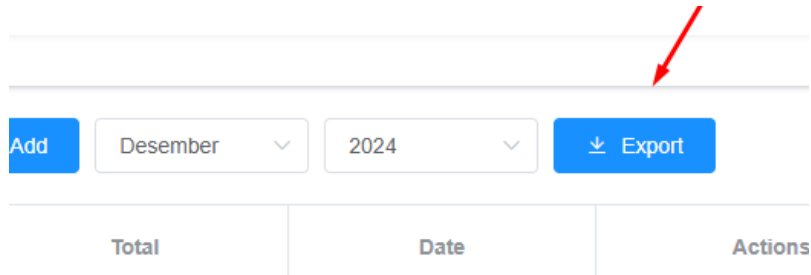


ID	Item Name	Type	Total	Created By	Date	Actions
105	KARDUS	IN	100	coba1	2025-02-20 05:36:37	Delete
102	AIR	IN	100	coba1	2025-02-20 12:25:14	Delete
100	DISPENSER	IN	100	coba1	2025-02-20 09:38:28	Delete
34	DISPENSER	IN	5	coba1	2024-12-10 13:18:21	Delete
3	GALON	OUT	100	coba1	2024-12-02 06:22:21	Delete
2	GALON	OUT	100	coba1	2024-11-27 06:19:08	Delete

Gambar 4. 28 Halaman User Data Transaction

Pada Gambar 4.28 merupakan transaksi yang sudah ada ditampilkan dalam tabel dengan menggunakan komponen el-table. Tabel ini menampilkan berbagai informasi seperti ID transaksi, nama item, tipe transaksi (IN/OUT), total, dan tanggal transaksi. Data ditarik dari server melalui fungsi `getList()`, yang memanggil API untuk mendapatkan daftar transaksi berdasarkan query pencarian yang dipilih oleh pengguna (seperti bulan, tahun, dan jenis transaksi).

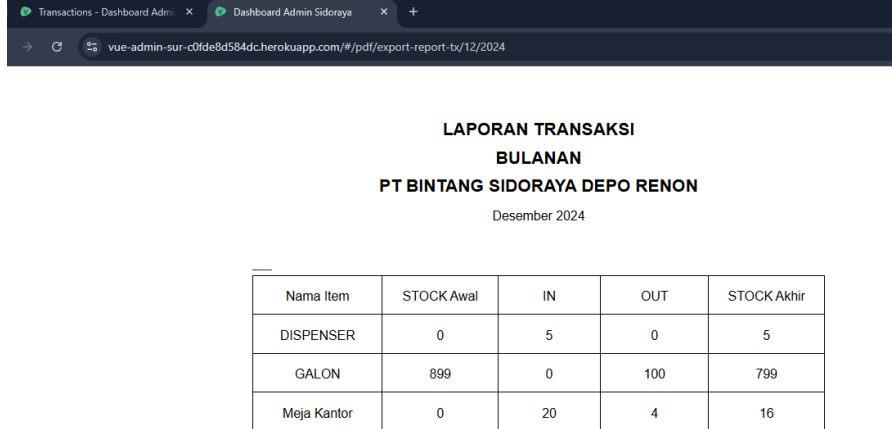
4.2.18 Halaman User Export Data Transaction (Supervisor)



Gambar 4. 29 Halaman Export Data Transaction (Supervisor)

Pada Gambar 4.29 merupakan Fitur ekspor data transaksi memungkinkan pengguna untuk mengunduh laporan transaksi dalam format PDF berdasarkan bulan dan tahun yang dipilih. Tombol "Export" yang terletak di bagian atas halaman memungkinkan pengguna untuk mengekspor data yang difilter ke dalam file PDF. Link untuk ekspor ini dibangun menggunakan router-link yang mengarahkan pengguna ke URL untuk mengunduh laporan PDF.

4.2.19 Halaman User Laporan PDF Data Transaction (Supervisor)



LAPORAN TRANSAKSI
BULANAN
PT BINTANG SIDORAYA DEPO RENON
Desember 2024

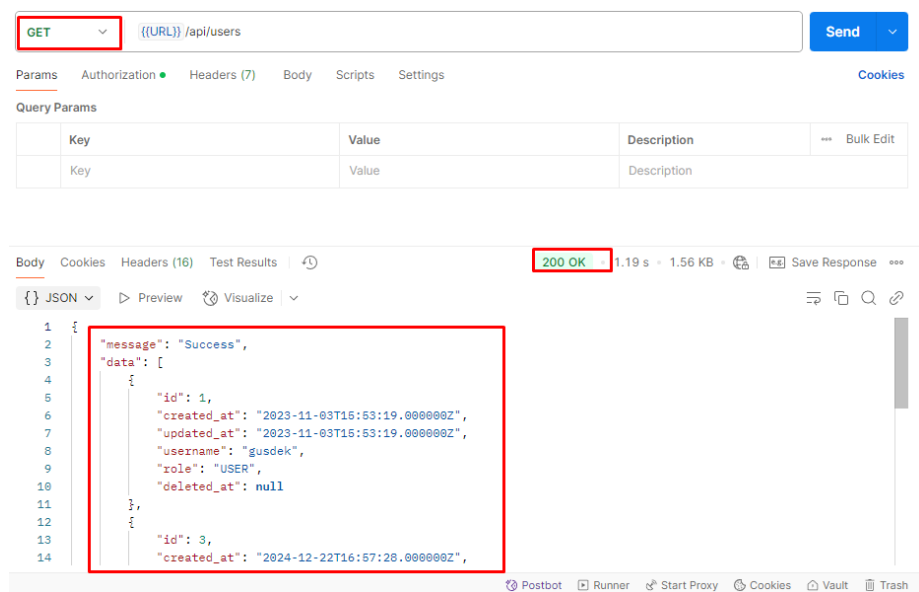
Nama Item	STOCK Awal	IN	OUT	STOCK Akhir
DISPENSER	0	5	0	5
GALON	899	0	100	799
Meja Kantor	0	20	4	16

Gambar 4. 30 Halaman Laporan PDF Data Transaction (Supervisor)

Pada Gambar 4.30 merupakan Halaman laporan transaksi ini menyajikan informasi mengenai transaksi barang dalam format bulanan untuk PT Bintang Sidoraya Depo Renon. Pada bagian atas, terdapat informasi bulan dan tahun yang dilaporkan, yang diambil berdasarkan nilai bulan dan tahun yang ditentukan. Laporan kemudian menampilkan tabel dengan lima kolom utama: Nama Item, Stock Awal, IN (masuk), OUT (keluar), dan Stock Akhir. Setiap baris data dalam tabel tersebut berisi informasi transaksi yang mencakup perubahan stock barang selama periode tersebut. Variabel $\${data}$ akan digantikan dengan data transaksi yang relevan, sehingga laporan ini dapat menampilkan transaksi yang aktual berdasarkan data yang diambil. Setelah laporan selesai, halaman ini memberikan gambaran menyeluruh tentang status inventaris barang selama periode yang dilaporkan.

4.3 Uji Coba Sistem

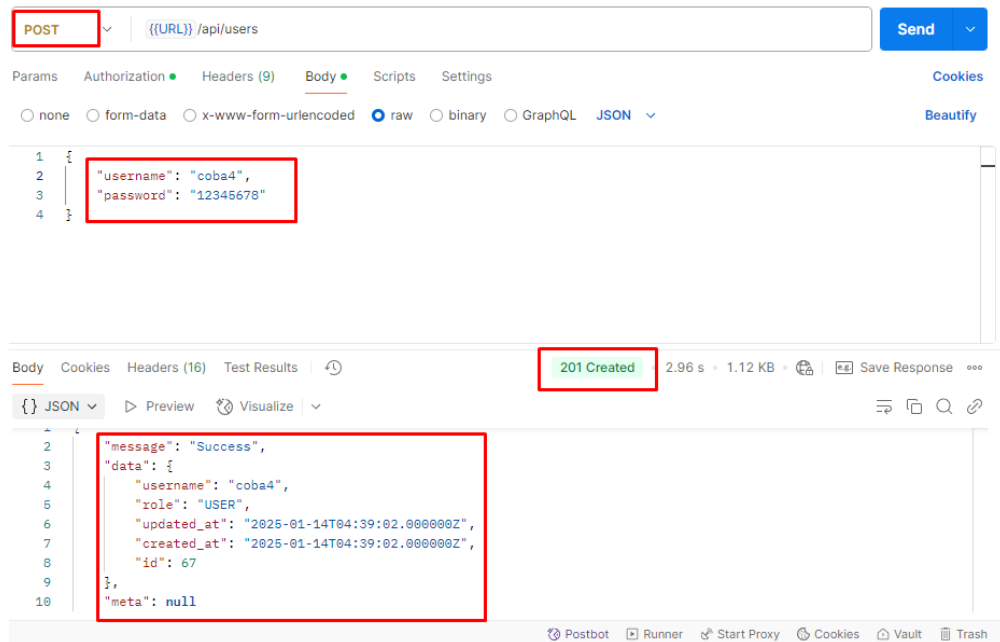
4.3.1 Tampilan Get List



Gambar 4. 31 Tampilan GET List User

Pada Gambar 4.31 merupakan endpoint API untuk mengambil daftar pengguna dari sistem. Endpoint ini menggunakan metode GET dengan URL *http://user-service-sur-9fac27dc4058.herokuapp.com*. Jika permintaan berhasil, server akan merespons dengan status kode 200 OK dan mengembalikan data dalam format JSON. Respons berisi pesan "Success" serta daftar pengguna dengan informasi detail, seperti id, created_at, updated_at, username, role, dan deleted_at. Sebagai contoh, data pengguna dengan nama "gusdek" sebagai USER. Respons ini juga dapat menyertakan metadata tambahan, tetapi dalam contoh ini, nilai meta adalah null.

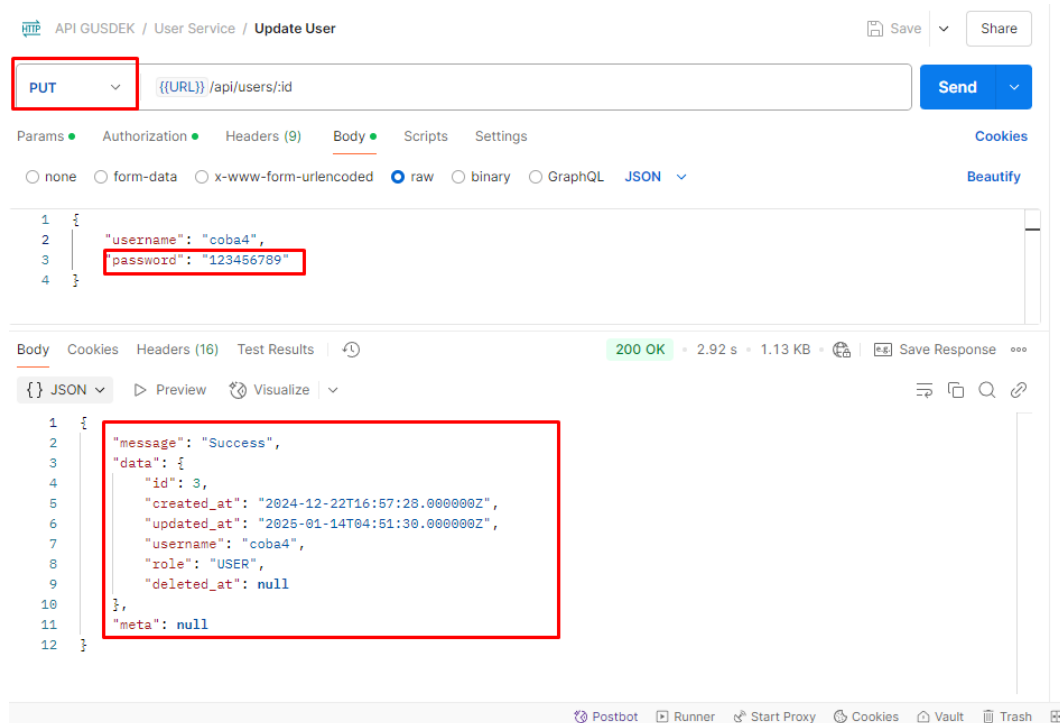
4.3.2 Tampilan Post Create User



Gambar 4. 32 Tampilan POST Create User

Pada Gambar 4.32 merupakan endpoint API untuk membuat pengguna baru dengan metode POST di URL *http://user-service-sur-9fac27dc4058.herokuapp.com*. Jika berhasil, server merespons dengan status 201 Created, pesan "User Successfully Created!", dan detail pengguna baru, seperti username, role, id, serta waktu pembuatan dan pembaruan. Jika permintaan dilakukan oleh pengguna yang tidak memiliki hak akses yang sesuai (selain ADMIN), server merespons dengan status 400 Bad Request, pesan "Only ADMIN users can create", dan data kosong.

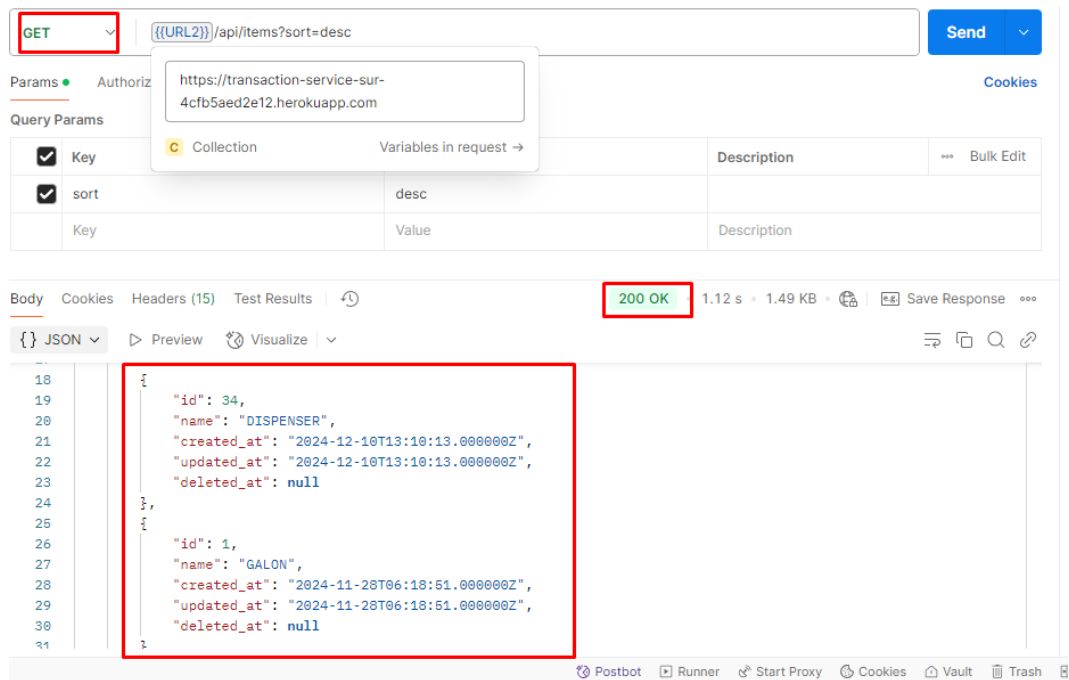
4.3.3 Tampilan Put Update User



Gambar 4. 33 Tampilan PUT Update User

Pada Gambar 4.33 merupakan endpoint API untuk memperbarui data pengguna dalam sistem menggunakan metode PUT di URL *http://user-service-sur-9fac27dc4058.herokuapp.com*. Jika permintaan berhasil, server merespons dengan status 200 Created, pesan "User Successfully Created!", dan detail pengguna yang diperbarui, termasuk username, role, id, serta waktu pembuatan dan pembaruan. Namun, jika permintaan dilakukan oleh pengguna dengan hak akses yang tidak sesuai (selain ADMIN), server merespons dengan status 400 Bad Request, pesan "Only ADMIN users can create", dan data kosong.

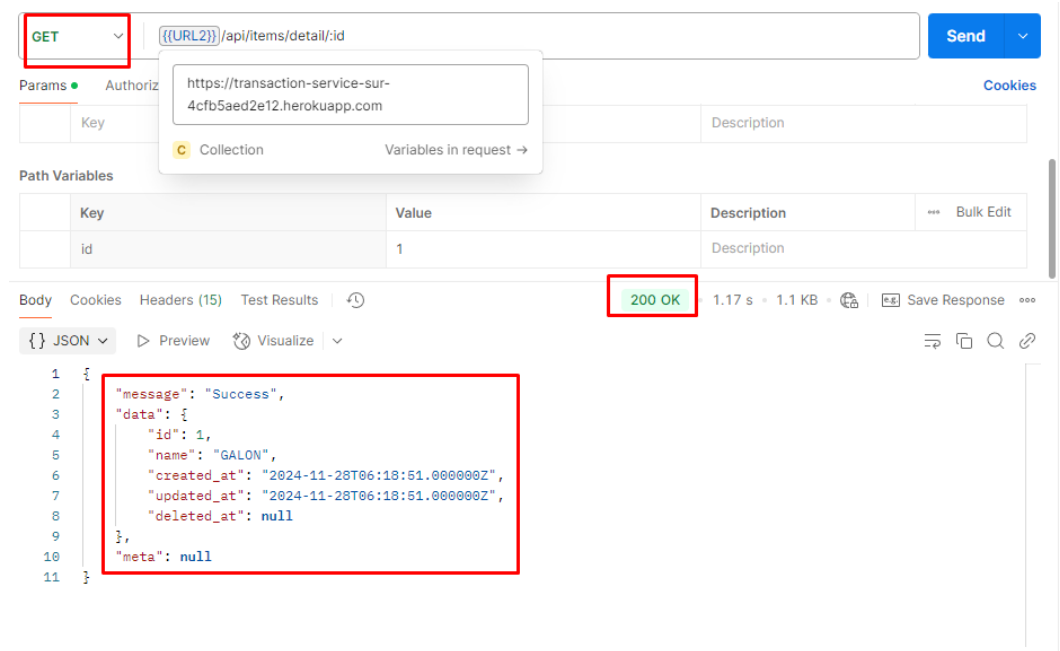
4.3.4 Tampilan Get List Item



Gambar 4. 34 Tampilan GET List Item

Pada Gambar 4.34 merupakan endpoint API untuk mengambil daftar barang dari sistem menggunakan metode GET di URL `https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items?sort=desc`. Jika permintaan berhasil, server merespons dengan status **200 OK** dan mengembalikan data dalam format JSON. Respons mencakup pesan "Success" serta daftar barang dengan detail seperti *id*, *name*, *created_at*, *updated_at*, dan *deleted_at*. Sebagai contoh, data mencakup barang "GALON" dan "DISPENSER". Respons ini juga dapat menyertakan metadata tambahan, namun pada contoh ini, nilai *meta* adalah null.

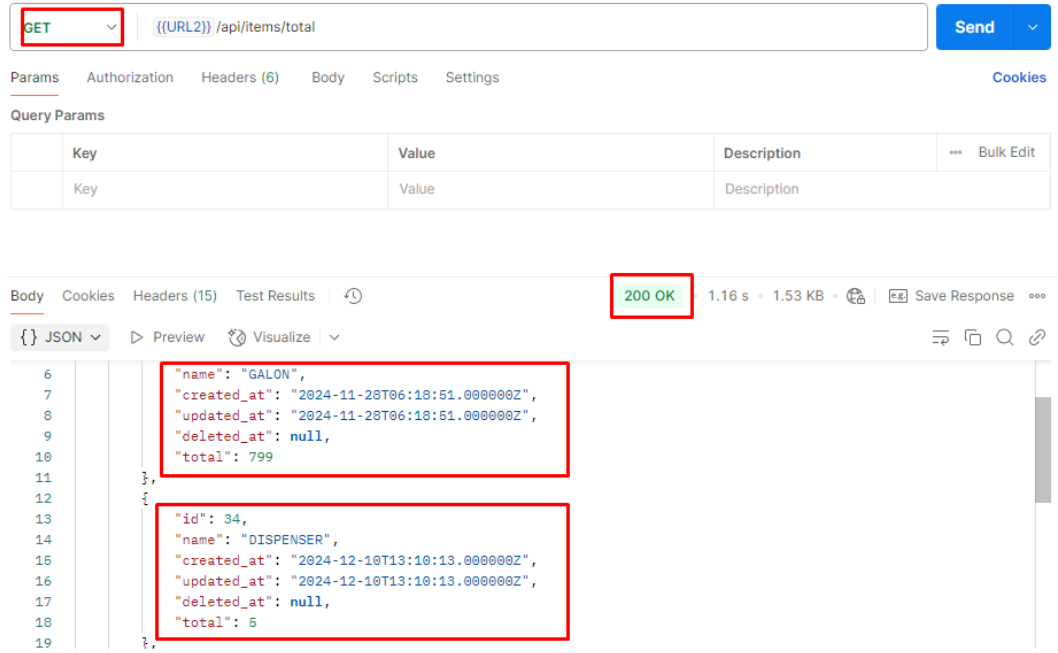
4.3.5 Tampilan Get Item Detail



Gambar 4. 35 Tampilan GET Item Detail

Pada Gambar 4.35 merupakan endpoint API untuk mengambil detail barang tertentu berdasarkan id menggunakan metode GET di URL `https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/detail/:id`. Jika permintaan berhasil, server merespons dengan status **200 OK** dan mengembalikan data dalam format JSON. Respons mencakup pesan "Success" dan detail barang yang diminta, termasuk *id*, *name*, *created_at*, *updated_at*, dan *deleted_at*. Sebagai contoh, data mencakup barang "Galon" dengan *id* 1. Respons ini juga dapat menyertakan metadata tambahan, namun pada contoh ini, nilai *meta* adalah null.

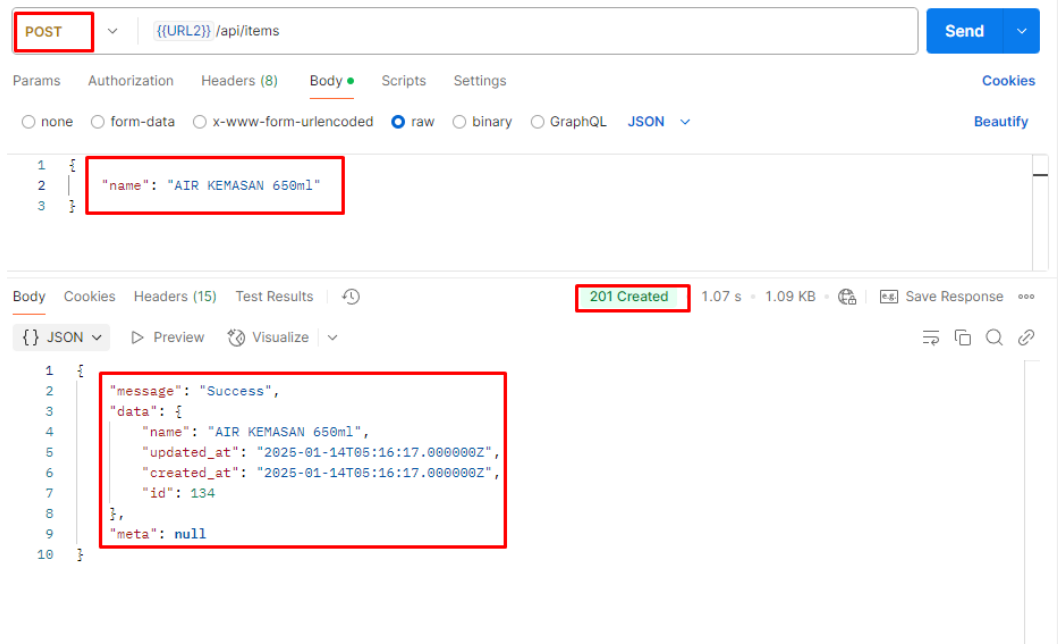
4.3.6 Tampilan Get Item Total



Gambar 4. 36 Tampilan GET Item Total

Pada Gambar 4.36 merupakan endpoint API untuk mengambil total barang dalam sistem menggunakan metode GET di URL *https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/total*. Jika permintaan berhasil, server merespons dengan status 200 OK dan mengembalikan data dalam format JSON. Respons mencakup pesan "Success" serta daftar barang dengan detail seperti *id*, *name*, *created_at*, *updated_at*, *deleted_at*, dan *total*. Sebagai contoh, data mencakup barang "GALON" dengan total "799" dan "DISPENSER " dengan total "5". Respons ini juga dapat menyertakan metadata tambahan, namun pada contoh ini, nilai meta adalah null.

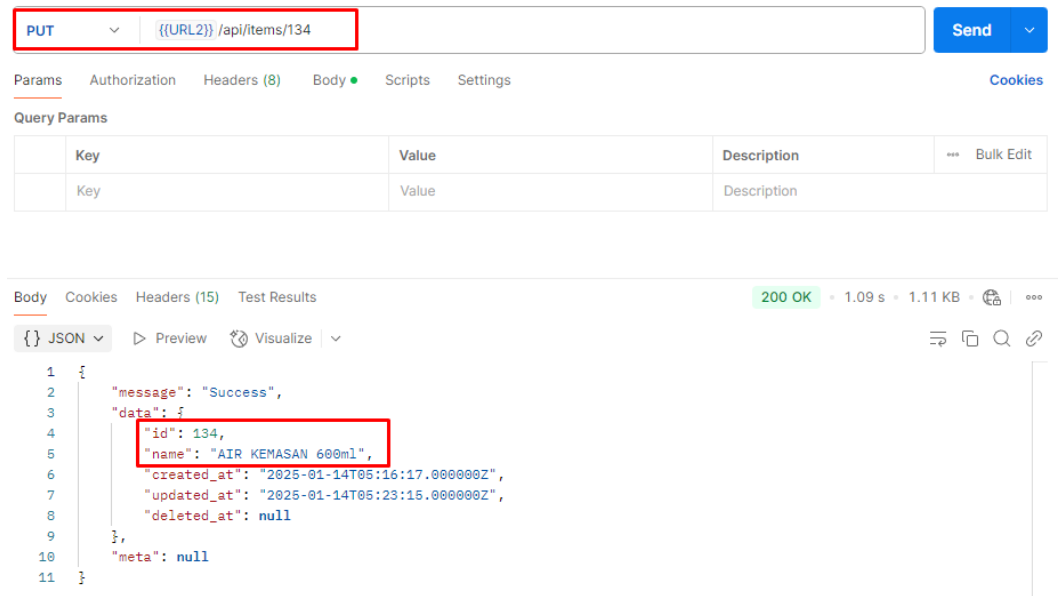
4.3.7 Tampilan Post Create Item



Gambar 4. 37 Tampilan POST Create Item

Pada Gambar 4.37 merupakan endpoint API untuk membuat barang baru dalam sistem menggunakan metode POST di URL `https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items`. Jika permintaan berhasil, server merespons dengan status 201 Created, pesan "Success", dan detail barang yang baru dibuat, termasuk name, id, serta waktu pembuatan dan pembaruan. Sebagai contoh, data mencakup barang "AIR KEMASAN 650ml" dengan id 134. Namun, jika barang yang akan dibuat sudah ada dalam sistem, server merespons dengan status 400 Bad Request, pesan "Item already exists", dan data kosong.

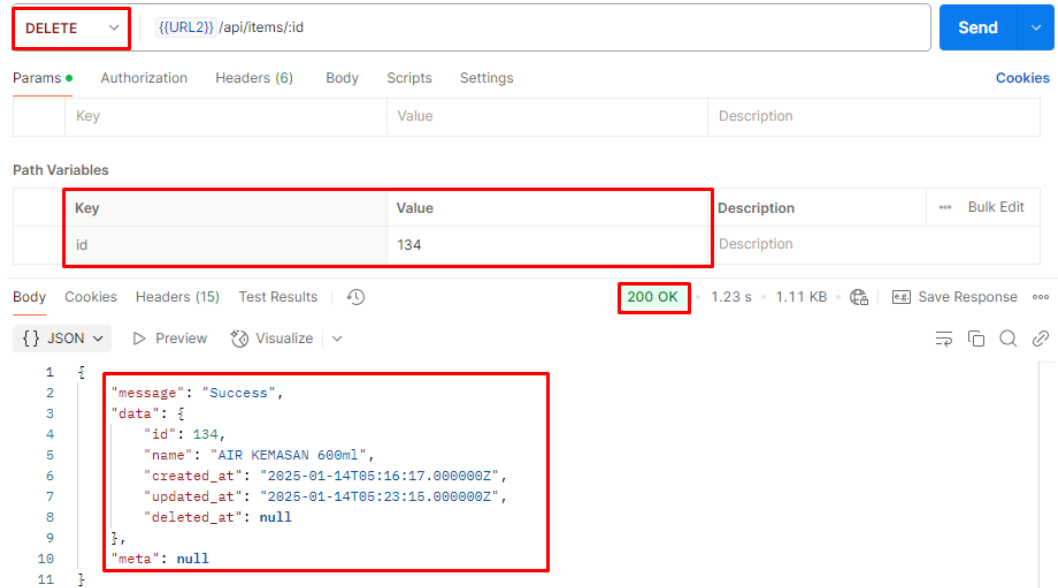
4.3.8 Tampilan Put Update Item



Gambar 4. 38 Tampilan PUT Update Item

Pada Gambar 4.38 merupakan endpoint API untuk memperbarui data barang tertentu menggunakan metode PUT di URL *https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/134*, di mana 134 merupakan ID barang yang akan diperbarui. Jika permintaan berhasil, server merespons dengan status 200 OK, pesan "Success", dan detail barang yang diperbarui, termasuk id, name, created_at, updated_at, dan deleted_at. Sebagai contoh, data mencakup barang "AIR KEMASAN 600ml" dengan ID 134. Namun, jika barang yang diperbarui sudah ada dalam sistem dengan data yang sama, server merespons dengan status 400 Bad Request, pesan "Item already exists", dan data kosong.

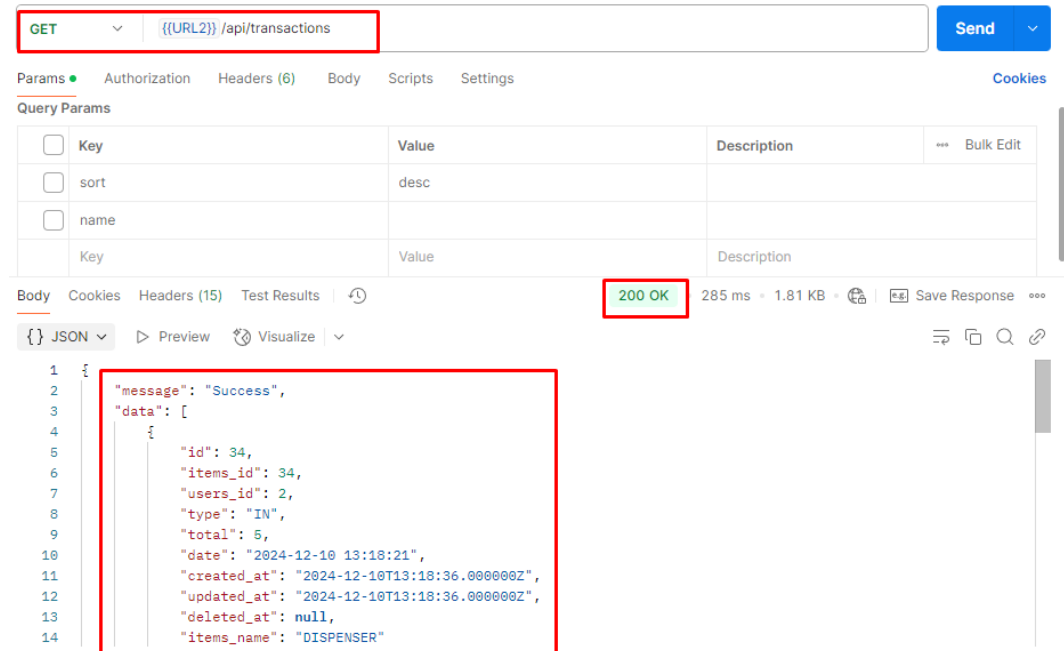
4.3.9 Tampilan Delete Item



Gambar 4. 39 Tampilan DELETE Item

Pada Gambar 4.39 merupakan endpoint API untuk menghapus barang tertentu menggunakan metode DELETE di URL `https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/:id`, di mana `:id` merupakan ID barang yang akan dihapus. Jika permintaan berhasil, server merespons dengan status 200 OK, pesan "Success", dan detail barang yang telah dihapus, termasuk `id`, `name`, `created_at`, `updated_at`, dan `deleted_at`. Sebagai contoh, data mencakup barang "AIR KEMASAN 600ml" dengan ID 134. Namun, jika barang dengan ID yang diberikan tidak ditemukan dalam sistem, server merespons dengan status 404 Not Found, pesan "Item not found", dan data kosong.

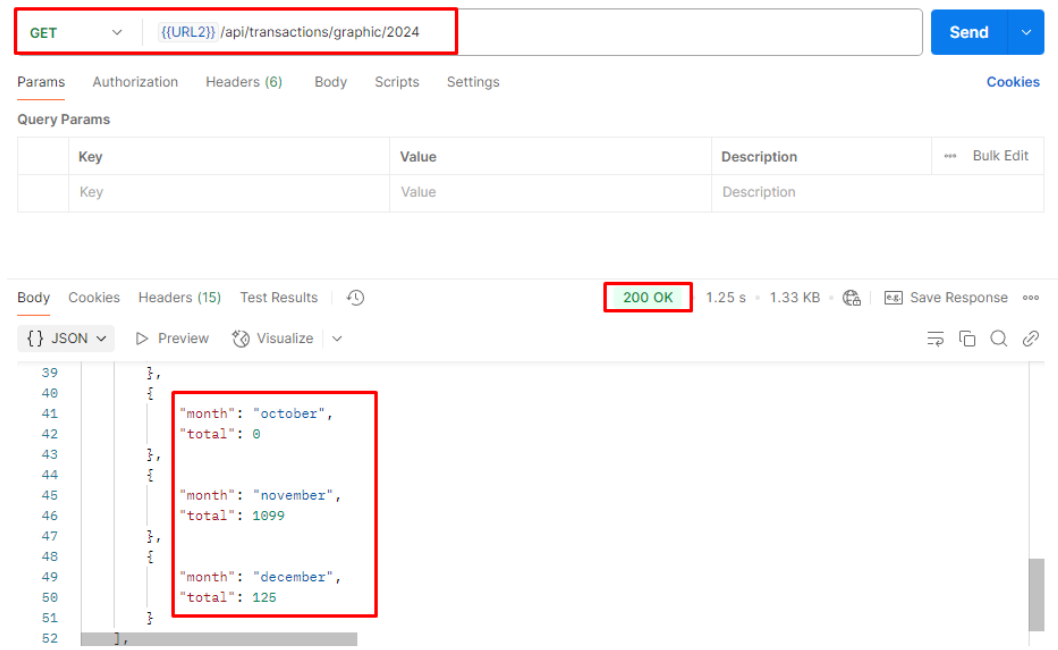
4.3.10 Tampilan Get Transaction List



Gambar 4. 40 Tampilan GET Transaction List

Pada Gambar 4.40 merupakan endpoint API untuk mengambil daftar transaksi menggunakan metode GET di URL `https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/transactions`. Jika permintaan berhasil, server merespons dengan status 200 OK dan mengembalikan data dalam format JSON. Respons mencakup pesan "Success" dan detail transaksi, termasuk id, items_id, users_id, type, total, date, created_at, updated_at, serta items_name. Sebagai contoh, data mencakup transaksi dengan barang "DISPENSER", yang menunjukkan transaksi masuk (IN) dengan total 5 unit. Respons ini juga dapat menyertakan metadata tambahan, namun pada contoh ini, nilai meta adalah null.

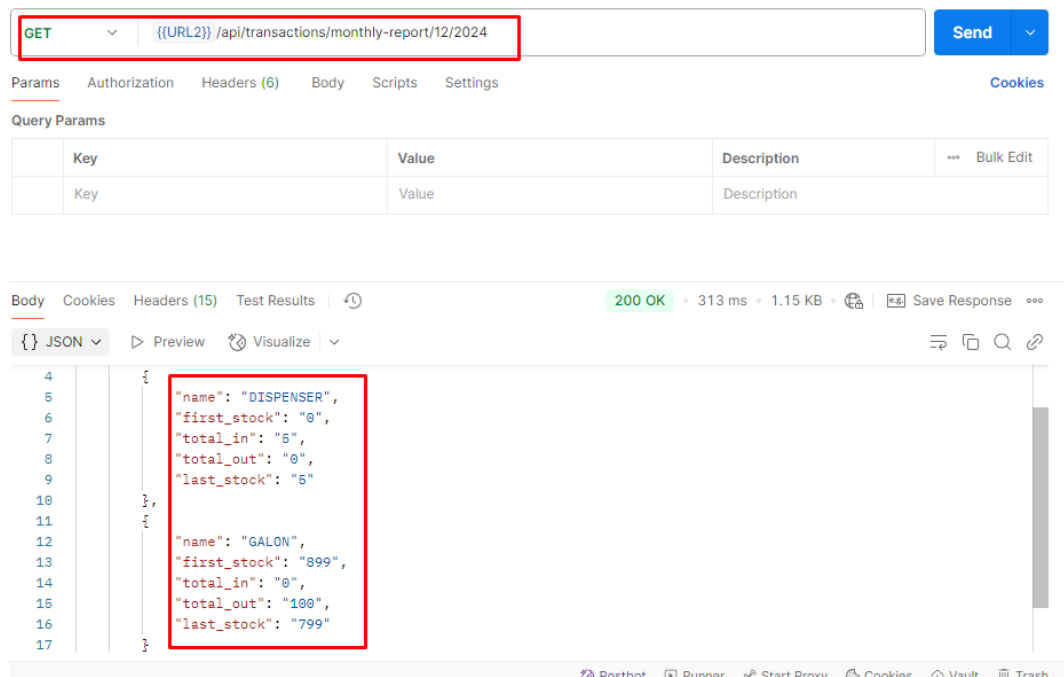
4.3.11 Tampilan Get Graphic



Gambar 4. 41 Tampilan GET Graphic

Pada Gambar 4.41 merupakan endpoint API untuk mengambil data grafik transaksi berdasarkan tahun menggunakan metode GET di URL `https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/transactions/graphic/2024`. Jika permintaan berhasil, server merespons dengan status **200 OK** dan mengembalikan data dalam format JSON. Respons mencakup pesan "Success" dan data transaksi per bulan untuk tahun 2024, dengan total transaksi yang terjadi setiap bulan. Sebagai contoh, data menunjukkan bahwa pada bulan November terdapat 1099 transaksi, Desember terdapat 125 transaksi, dan bulan lainnya tidak ada transaksi (total 0). Respons ini juga dapat menyertakan metadata tambahan, namun pada contoh ini, nilai meta adalah null.

4.3.12 Tampilan Get laporan Transaction Bulanan

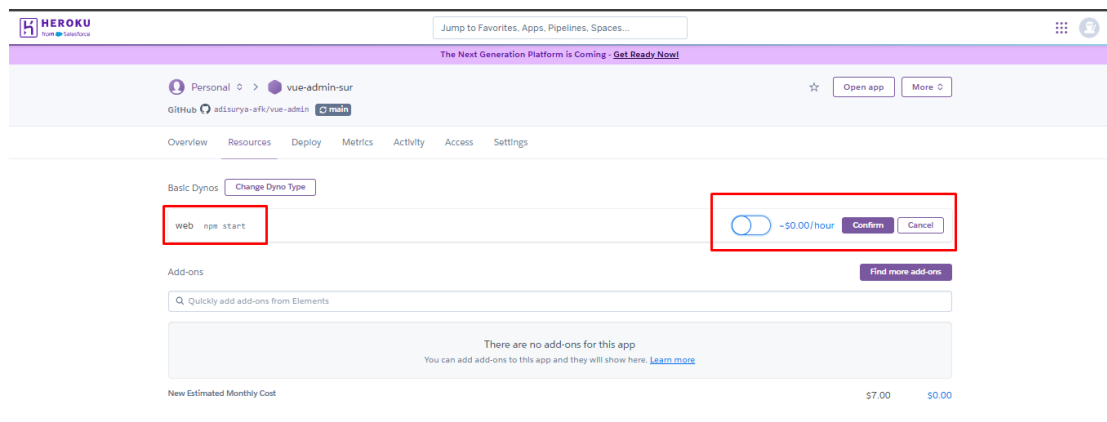


Gambar 4. 42 Tampilan GET Laporan Transaction Bulanan

Pada Gambar 4.42 merupakan endpoint API untuk mengambil laporan transaksi bulanan berdasarkan bulan dan tahun tertentu menggunakan metode GET di URL <https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/transactions/monthly-report/03/2024>. Jika permintaan berhasil, server merespons dengan status 200 OK dan mengembalikan data dalam format JSON. Respons mencakup pesan "Success" dan data laporan transaksi untuk bulan Maret 2024, dengan rincian barang yang diproses, termasuk name, first_stock, total_in, total_out, dan last_stock. Sebagai contoh, laporan menunjukkan

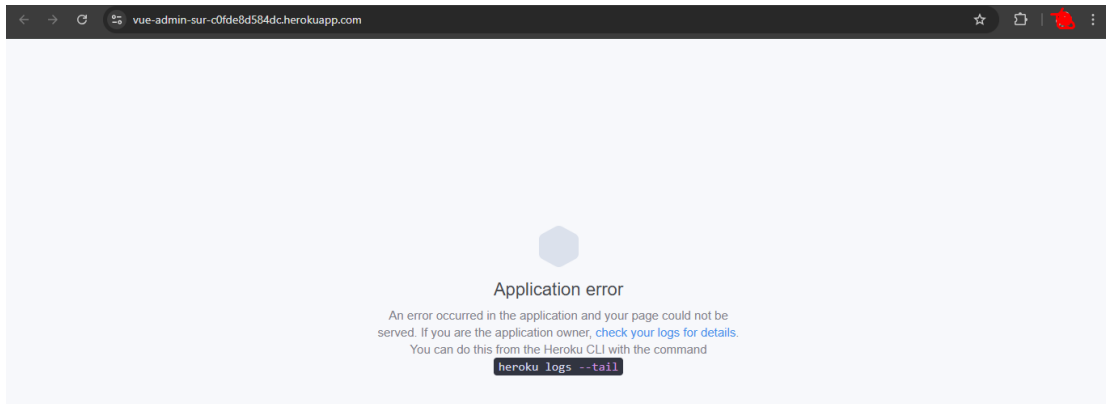
barang "GALON" dengan stok awal 899, total masuk 0, total keluar 100, dan stok akhir 799; sedangkan "DISPENSER" memiliki stok awal 0, total masuk 5, total keluar 0, dan stok akhir 5. Respons ini juga dapat menyertakan metadata tambahan, namun pada contoh ini, nilai meta adalah null.

4.3.13 Service Vue Admin Non-Aktif



Gambar 4. 43 Vue Admin Non-Aktif

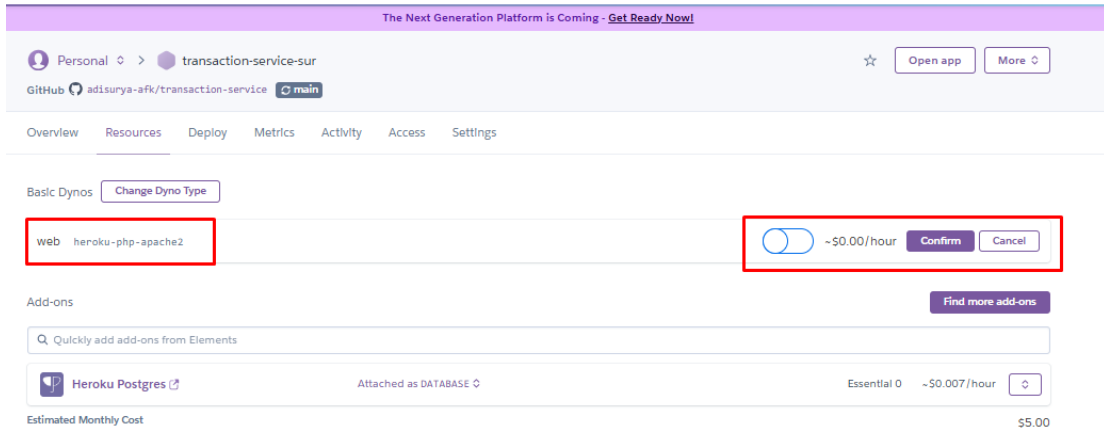
Gambar 4.43 ini menunjukkan halaman Heroku untuk aplikasi vue-admin-sur, yang merupakan aplikasi frontend Vue.js. Service web aplikasi ini dimatikan (akan dimatikan setelah konfirmasi) untuk tujuan pengujian microservice. Alasan spesifiknya dapat bervariasi, dalam kasus ini terkait dengan isolasi pengujian, simulasi kegagalan, penghematan biaya, atau fokus pengembangan.



Gambar 4. 44 Tampilan Vue Admin

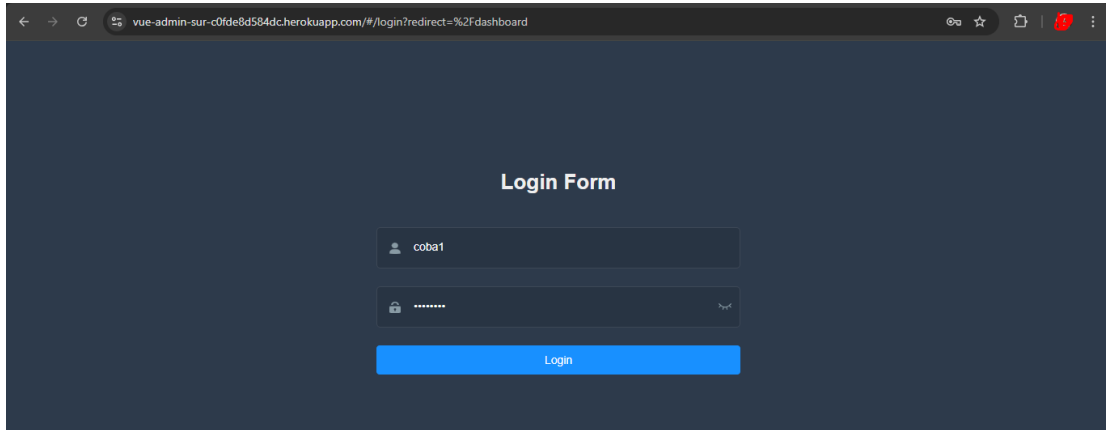
Gambar 4.44 tersebut menampilkan pesan kesalahan "Application error" pada aplikasi Heroku. Pesan ini muncul karena layanan web (dyno) aplikasi dimatikan atau mengalami masalah. Penyebabnya antara lain dyno yang tidak aktif, kesalahan pada kode aplikasi, masalah saat deployment, atau masalah pada infrastruktur Heroku. Dalam kasus ini service web aplikasi ini sengaja dimatikan untuk melakukan simulasi pengujian penerapan microservice.

4.3.14 Service Transaction Non-Aktif



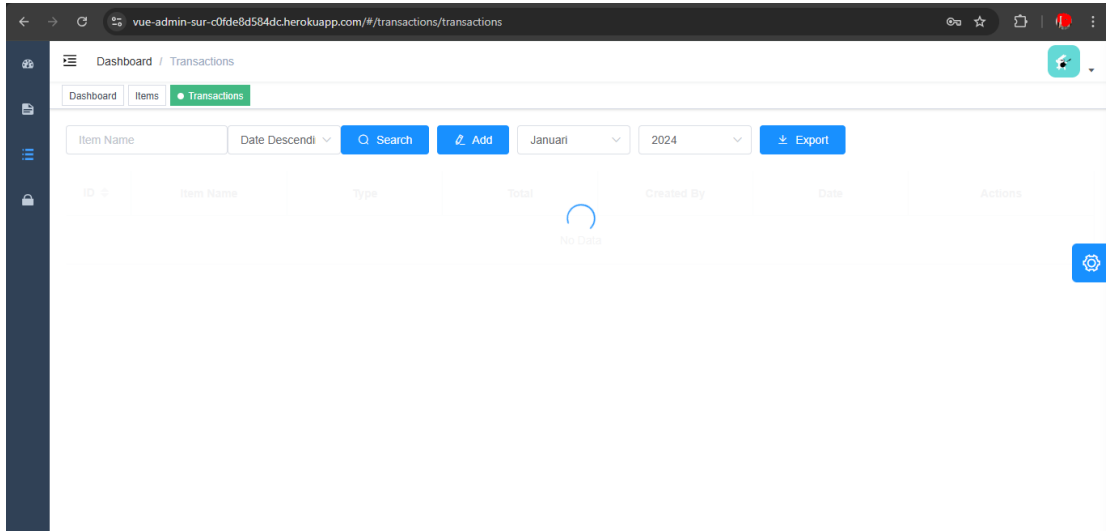
Gambar 4. 45 Transaction Non-Aktif

Gambar 4.45 ini menunjukkan halaman Heroku untuk Transaction service yang merupakan layanan aplikasi untuk bagian item dan transaksi. Service web aplikasi ini dimatikan (akan dimatikan setelah konfirmasi) untuk tujuan pengujian microservice. Alasan spesifiknya dapat bervariasi, dalam kasus ini terkait dengan isolasi pengujian, simulasi kegagalan, penghematan biaya, atau fokus pengembangan.



Gambar 4. 46 Tampilan Log-in

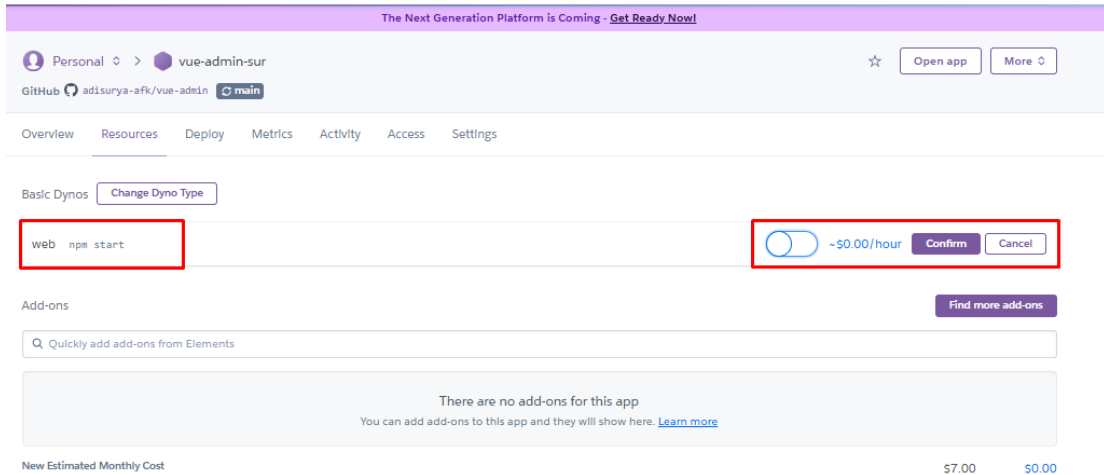
Pada Gambar 4.46 merupakan kondisi tampilan login saat service Transactionnya dimatikan, ini dikarenakan penerapan Microservice pada aplikasi web ini. Dalam arsitektur microservice, aplikasi dipecah menjadi layanan-layanan kecil dan independen yang berkomunikasi satu sama lain. Setiap layanan dapat di-deploy secara terpisah. Jika satu layanan dinonaktifkan atau mengalami masalah, layanan lain tetap berjalan namun dengan keterbatasan dalam hal konten aplikasi atau lainnya. Hal ini dimungkinkan karena setiap layanan memiliki dependensi minimal dan beroperasi secara terpisah. Arsitektur microservice memberikan ketahanan dan fleksibilitas, memungkinkan aplikasi tetap berfungsi sebagian meskipun ada gangguan pada beberapa komponennya.



Gambar 4. 47 Tampilan Halaman Transaction

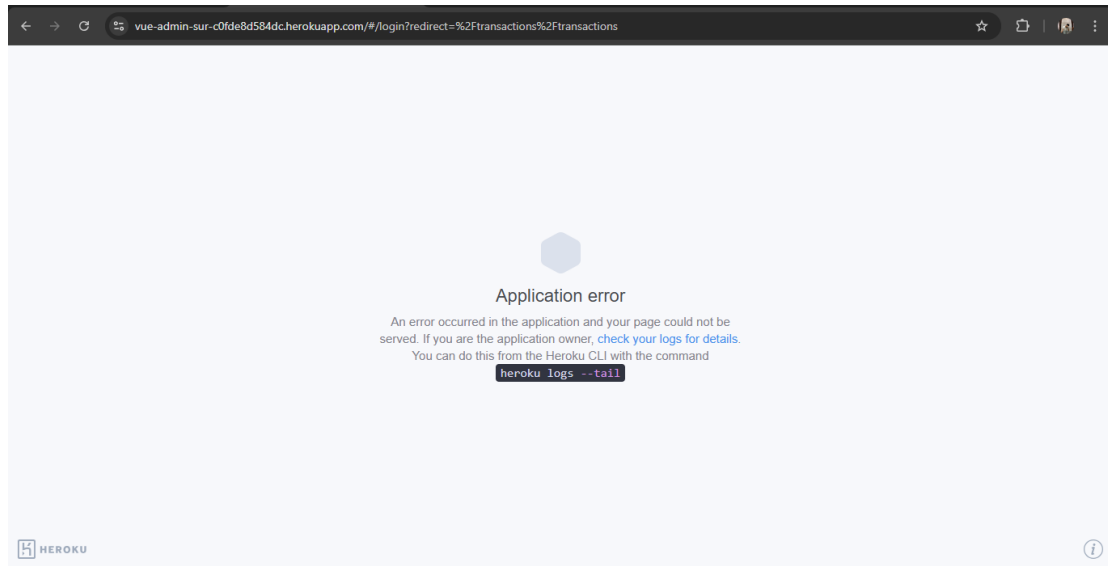
Gambar 4.47 merupakan tampilan halaman Transaction, gambar tersebut menampilkan halaman aplikasi web inventory tanpa ada konten Transaction yang diakibatkan oleh service yang mengalami gangguan atau dinonaktifkan. Hal ini terjadi karena setiap layanan memiliki dependensi minimal dan beroperasi secara terpisah. Arsitektur microservice memungkinkan aplikasi tetap berfungsi sebagian meskipun ada gangguan pada beberapa komponennya.

4.3.15 Service User Non-Aktif



Gambar 4. 48 User Service Non-Aktif

Gambar 4.48 ini menunjukkan halaman Heroku untuk bagian User service yang merupakan layanan aplikasi untuk bagian pengelola akun pengguna. Service web aplikasi ini dimatikan (akan dimatikan setelah konfirmasi) untuk tujuan pengujian microservice. Alasan spesifiknya dapat bervariasi, dalam kasus ini terkait dengan isolasi pengujian, simulasi kegagalan, penghematan biaya, atau fokus pengembangan.



Gambar 4. 49 Tampilan Halaman Log-in

Gambar 4.49 tersebut menampilkan pesan kesalahan "Application error" pada aplikasi Heroku. Pesan ini muncul karena layanan web (dyno) aplikasi dimatikan atau mengalami masalah. Penyebabnya antara lain dyno yang tidak aktif, kesalahan pada kode aplikasi, masalah saat deployment, atau masalah pada infrastruktur Heroku. Dalam kasus ini service web aplikasi ini sengaja dimatikan untuk melakukan simulasi pengujian penerapan microservice.

4.3.16 Pengujian Blackbox Testing

Tabel 4. 1 Blackbox Testing

No	Fitur	Skenario	Input	Harapan Hasil	Hasil Pengujian	Kesimpulan
1.	Halaman Login (Admin)	Validasi login dengan username dan password	Username dan Password	Sistem mengarahkan pengguna ke dashboard jika data valid	Berhasil	Normal
2.	Halaman Dashboard (Admin)	Memuat komponen dashboard (PanelGroup, ExportReport, BarChart)	Tidak ada input	Semua komponen tampil sesuai desain	Berhasil	Normal
3.	Halaman Data Admin	Menampilkan informasi profil pengguna	Tidak ada input	Data profil (nama, avatar, peran) ditampilkan dengan benar	Berhasil	Normal
4.	Halaman Registrasi Akun Admin	Mendaftarkan admin baru	Nama dan Email	Admin baru berhasil didaftarkan	Berhasil	Normal
5.	Halaman Edit Akun Admin	Mengubah data pengguna	Nama atau Email baru	Data pengguna diperbarui dan notifikasi sukses ditampilkan	Berhasil	Normal
6.	Halaman Create Data Item	Menambahkan data baru	Informasi data item	Data berhasil ditambahkan dan muncul di tabel	Berhasil	Normal
7.	Halaman Data Item	Menampilkan data item yang ada, termasuk pencarian dan filter	Kata kunci pencarian atau filter tanggal	Data sesuai filter/pencarian ditampilkan, aksi edit/hapus dapat dilakukan	Berhasil	Normal
8.	Halaman Edit Item	Mengedit data item yang sudah ada	Perubahan data item	Data item diperbarui dan tabel	Berhasil	Normal

				menampilkan informasi terbaru		
9.	Halaman User Supervisor	Menampilkan data pengguna dengan peran Supervisor	Tidak ada input	Data Supervisor ditampilkan lengkap dengan fungsi validasi/verifikasi	Berhasil	Normal
10.	Halaman Create Transaction (Supervisor)	Menambahkan transaksi baru	Detail transaksi baru	Transaksi berhasil ditambahkan dan muncul di tabel	Berhasil	Normal
11.	Halaman Data Transaction (Supervisor)	Menampilkan tabel transaksi, termasuk pencarian dan filter	Kata kunci pencarian atau filter tanggal	Data sesuai filter/pencarian ditampilkan	Berhasil	Normal
12.	Halaman Edit Data Transaction	Mengedit data transaksi	Perubahan data transaksi	Data transaksi diperbarui dan tabel menampilkan informasi terbaru	Berhasil	Normal
13.	Halaman Export Data Transaction	Mengekspor data transaksi ke file PDF	Pilihan bulan dan tahun	File PDF berhasil diunduh dengan data transaksi yang sesuai	Berhasil	Normal
14.	Halaman Laporan PDF Data Transaction	Menampilkan laporan PDF data transaksi	Pilihan bulan dan tahun	Laporan menampilkan informasi transaksi sesuai periode yang dipilih	Berhasil	Normal
15.	Tampilan GET List User	Mengambil daftar pengguna dalam sistem	GET: https://user-service-sur-9fac27dc4058.herokuapp.com/api/users	Status 200 OK, pesan "Success", data JSON	Berhasil	Normal

				daftar pengguna		
16.	Tampilan POST Create User	Membuat pengguna baru dengan hak akses ADMIN	POST: https://user-service-sur-9fac27dc4058.herokuapp.com/api/users dengan data valid (username, role)	Status 201 Created, pesan "User Successfully Created!", detail pengguna baru	Berhasil	Normal
17.		Membuat pengguna baru dengan hak akses selain ADMIN	POST: https://user-service-sur-9fac27dc4058.herokuapp.com/api/users dengan data valid	Status 400 Bad Request, pesan "Only SUPER_ADMIN users can create", data kosong	Berhasil	Normal
18.	Tampilan PUT Update User	Memperbarui data pengguna dengan hak akses ADMIN	PUT: https://user-service-sur-9fac27dc4058.herokuapp.com/api/users dengan data valid (id, username, role)	Status 201 Created, pesan "User Successfully Created!", detail pengguna yang diperbarui	Berhasil	Normal
19.		Memperbarui data pengguna dengan hak akses selain ADMIN	PUT: https://user-service-sur-9fac27dc4058.herokuapp.com/api/users dengan data valid	Status 400 Bad Request, pesan "Only SUPER_ADMIN users can create", data kosong	Berhasil	Normal
20.	Tampilan GET List Item	Mengambil daftar barang dari sistem	GET: http://localhost:8001/api/items	Status 200 OK, pesan "Success", data JSON daftar barang	Berhasil	Normal
21.	Tampilan GET Item Detail	Mengambil detail barang berdasarkan id	GET: https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/detail/:id	Status 200 OK, pesan "Success", data JSON detail barang	Berhasil	Normal

22.	Tampilan GET Item Total	Mengambil total barang dari sistem	GET: https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/total	Status 200 OK, pesan "Success", data JSON daftar barang dengan total	Berhasil	Normal
23.	Tampilan POST Create Item	Membuat barang baru	POST: https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items dengan data valid (name)	Status 201 Created, pesan "Success", detail barang baru	Berhasil	Normal
24.		Membuat barang yang sudah ada	POST: https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items dengan data valid (name)	Status 400 Bad Request, pesan "Item already exists", data kosong	Berhasil	Normal
25.	Tampilan PUT Update Item	Memperbarui data barang	PUT: https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/:id dengan data valid (id, name)	Status 200 OK, pesan "Success", detail barang yang diperbarui	Berhasil	Normal
26.		Memperbarui barang dengan data yang sudah ada	PUT: https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/:id dengan data valid (id, name)	Status 400 Bad Request, pesan "Item already exists", data kosong	Berhasil	Normal
27.	Tampilan DELETE Item	Menghapus barang berdasarkan id	DELETE: https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/:id	Status 200 OK, pesan "Success", detail barang yang dihapus	Berhasil	Normal
28.		Menghapus barang dengan id yang tidak ditemukan	DELETE: https://transaction-service-sur-4cfb5aed2e12.herokuapp.com/api/items/:id	Status 404 Not Found, pesan "Item not found", data kosong	Berhasil	Normal

29.	Tampilan GET Transaction List	Mengambil daftar transaksi	GET: https://transaction- service-sur- 4cfb5aed2e12.herokuapp pp.com/api/transactions	Status 200 OK, pesan "Success", data JSON daftar transaksi	Berhasil	Normal
30.	Tampilan GET Graphic	Mengambil data grafik transaksi berdasarkan tahun	GET: https://transaction- service-sur- 4cfb5aed2e12.herokuapp pp.com/api/transactions /graphic/:year	Status 200 OK, pesan "Success", data JSON grafik transaksi	Berhasil	Normal
31.	Tampilan GET Laporan Transaction Bulanan	Mengambil laporan transaksi bulanan berdasarkan bulan dan tahun	GET: https://transaction- service-sur- 4cfb5aed2e12.herokuapp pp.com/api/transactions /monthly- report/:month/:year	Status 200 OK, pesan "Success", data JSON laporan transaksi	Berhasil	Normal