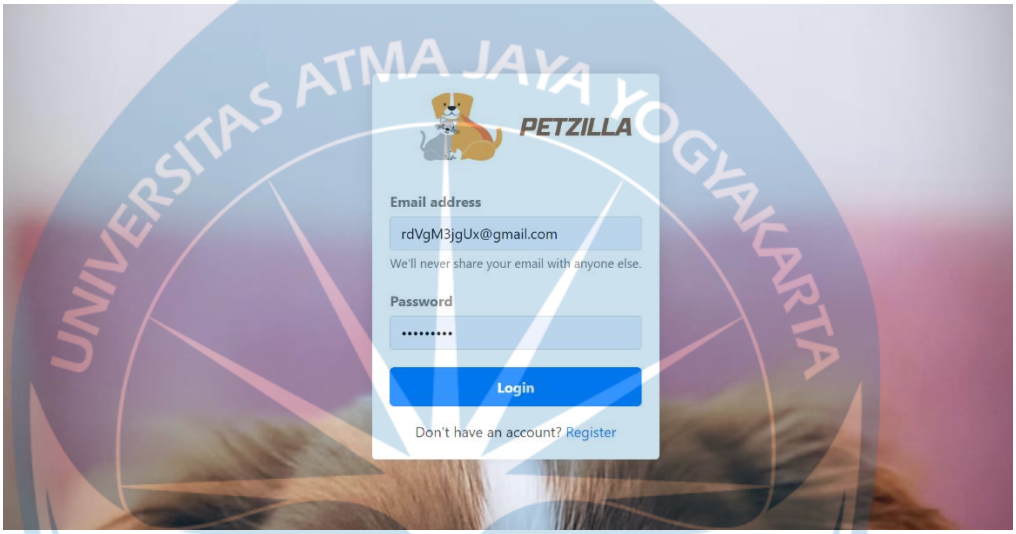
BAB V
HASIL DAN PEMBAHASAN

A. Implementasi Sistem Antarmuka

1) Implementasi Antarmuka *Login User*



Gambar 5.1 *Login User*

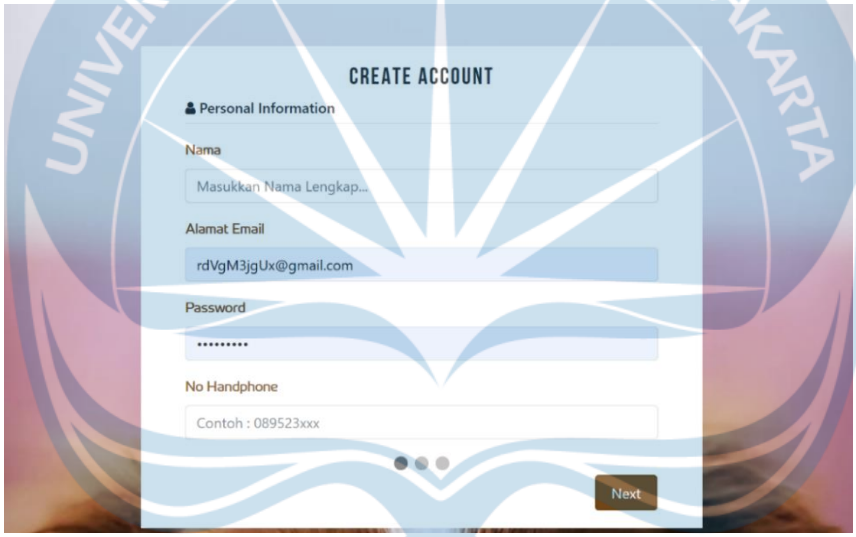
Gambar 5.1 merupakan implementasi antar muka pada *login user*. Pada halaman ini terdapat input teks untuk email dan password.

```php
public function login()
{
    $this->validate([
        'email' => 'required',
        'password' => 'required',
    ]);

    if (Auth::attempt(['email' => $this->email, 'password' => $this->password])) {
        return redirect()->to('/home');
    }else{
        session()->flash('error', 'Incorrect credentials !');
    }
}
```

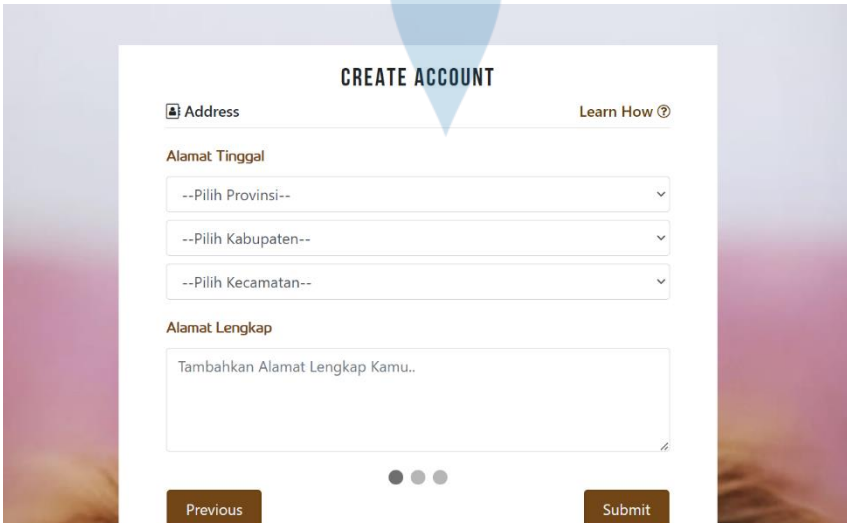
Gambar 5.2 Potongan Kode Fungsi *Login User*

Gambar 5.2 merupakan potongan kode untuk fungsi login *user*. Pertama, sistem akan melakukan validasi terhadap *input* yang diterima, setelah itu melakukan metode *Auth::attempt()* yang merupakan fitur authentikasi bawaan dari Laravel. Pada metode ini akan dilakukan pemeriksaan apakah ada data *user* yang *email* dan *password*-nya sesuai. Jika ada, maka *user* akan diarahkan ke halaman *marketplace*, jika tidak maka sistem akan menampilkan pesan *error*.

2) Implementasi Antarmuka *Register User*

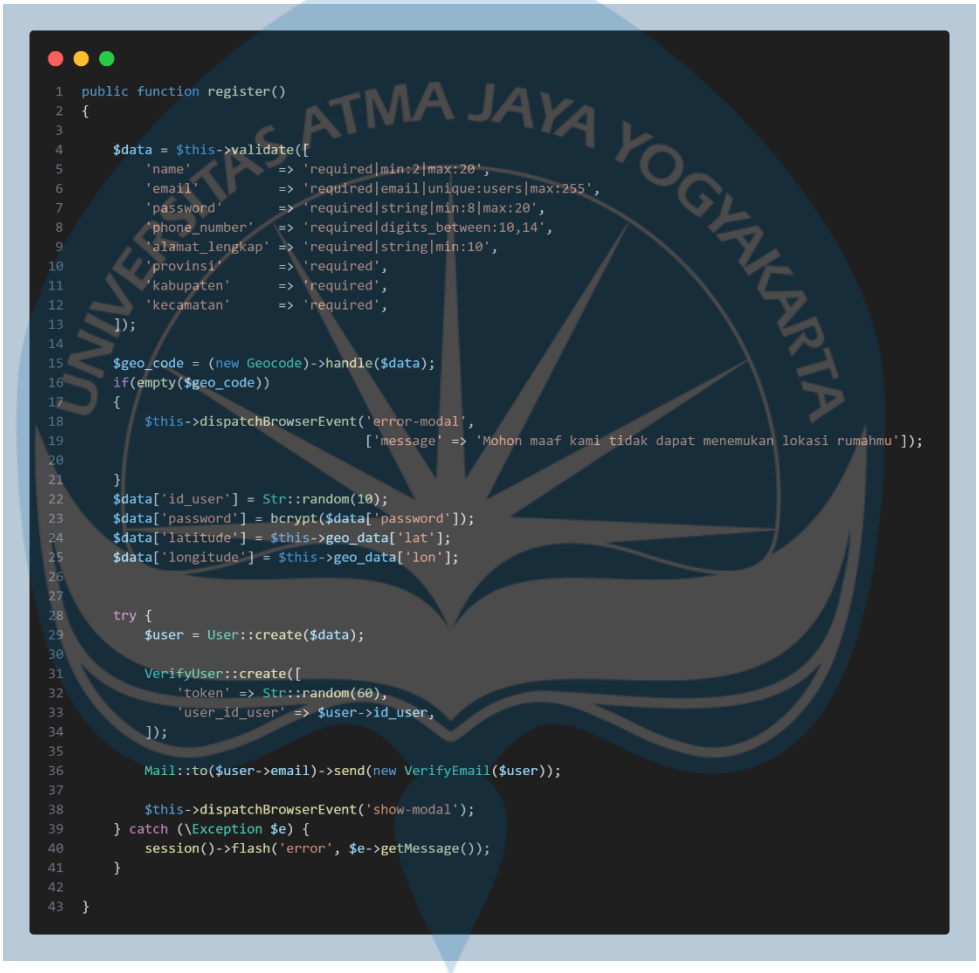


Gambar 5.3 *Register User 1*



Gambar 5.4 *Register User 2*

Gambar 5.3 dan 5.4 merupakan implementasi antarmuka *register user*. Pada halaman registrasi terdapat dua tahap, yang pertama mengisi terkait informasi diri, lalu pada tahap kedua mengisi alamat rumah. Setelah mengisi data dengan benar, sistem akan mengirimkan sebuah email kepada pengguna yang berisi tautan untuk melakukan verifikasi email.



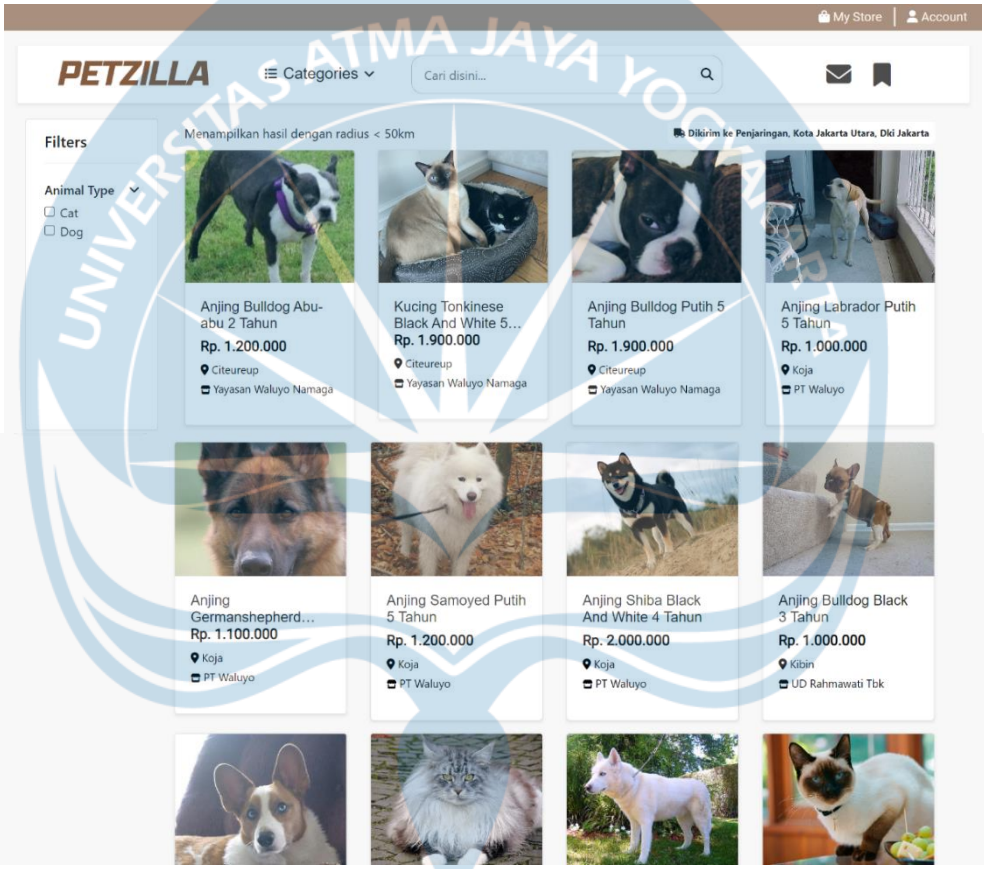
```php
public function register()
{

    $data = $this->validate([
        'name'          => 'required|min:2|max:20',
        'email'         => 'required|email|unique:users|max:255',
        'password'      => 'required|string|min:8|max:20',
        'phone_number'  => 'required|digits_between:10,14',
        'alamat_lengkap' => 'required|string|min:10',
        'provinsi'      => 'required',
        'kabupaten'     => 'required',
        'kecamatan'     => 'required',
    ]);

    $geo_code = (new Geocode)->handle($data);
    if(empty($geo_code))
    {
        $this->dispatchBrowserEvent('error-modal',
                                    ['message' => 'Mohon maaf kami tidak dapat menemukan lokasi rumahmu']);

    }
    $data['id_user'] = Str::random(10);
    $data['password'] = bcrypt($data['password']);
    $data['latitude'] = $this->geo_data['lat'];
    $data['longitude'] = $this->geo_data['lon'];


    try {
        $user = User::create($data);

        VerifyUser::create([
            'token' => Str::random(60),
            'user_id_user' => $user->id_user,
        ]);

        Mail::to($user->email)->send(new VerifyEmail($user));

        $this->dispatchBrowserEvent('show-modal');
    } catch (\Exception $e) {
        session()->flash('error', $e->getMessage());
    }

}
```

Gambar 5.3 Potongan Kode Fungsi *Register User*

Gambar 5.5 merupakan potongan kode untuk fungsi *register user*. Pertama dimulai dari validasi berdasarkan input *user* dengan peraturan tertentu. Apabila tervalidasi, maka sistem akan memanggil fungsi *handle()* pada *Geocode class* yang digunakan untuk mendapatkan *latitude* dan *longitude* dari alamat pengguna. Jika berhasil maka sistem akan melakukan penyimpanan pada tabel

user, lalu mengirimkan verifikasi email pada pengguna. Pada tabel *user* terdapat attribut *email_verified_at* yang akan di *update* pada saat pengguna menekan tautan verifikasi email. Hal ini digunakan untuk memastikan bahwa email yang dimiliki pengguna merupakan email yang valid.

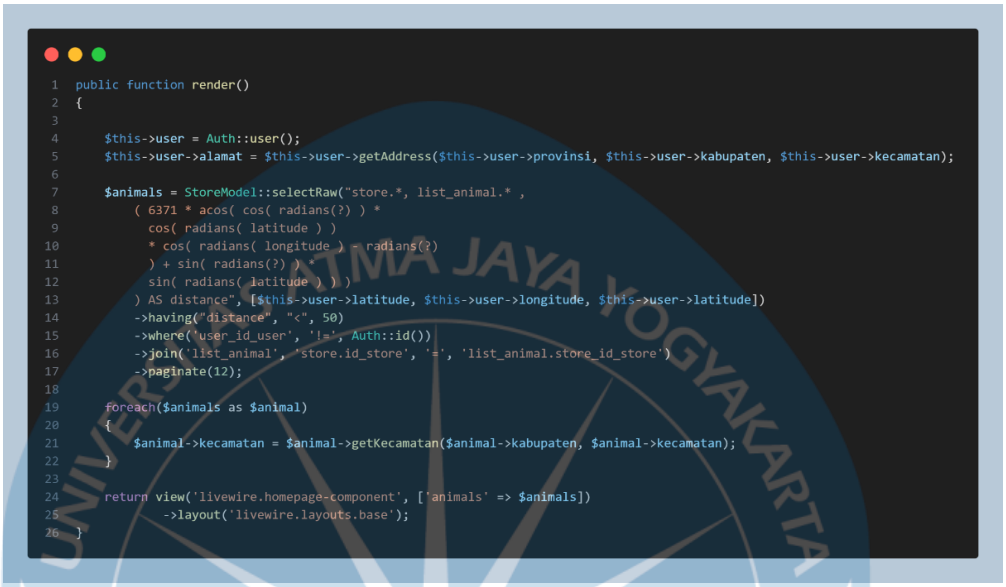
3) Implementasi Antarmuka *Marketplace*



Gambar 5. 4 *Marketplace*

Gambar 5.4 merupakan implementasi antarmuka untuk halaman *marketplace.* Pada halaman ini akan menampilkan daftar hewan yang tersedia dalam radius 50 km dari alamat pengguna. Pada *card* hewan menampilkan beberapa informasi seperti nama hewan, harga, lokasi toko dan nama toko. Selain itu, pengguna dapat menggunakan fitur pencarian dan filter untuk mempermudah pencarian hewan. Lalu, untuk menuju ke halaman toko,

pengguna dapat memilih pada bagian *MyStore* dan untuk ke halaman profil pada bagian *Account*.



```
public function render()
{

    $this->user = Auth::user();
    $this->user->alamat = $this->user->getAddress($this->user->provinsi, $this->user->kabupaten, $this->user->kecamatan);

    $animals = StoreModel::selectRaw("store.*, list_animal.* ,
        ( 6371 * acos( cos( radians(?) ) *
          cos( radians( latitude ) )
          * cos( radians( longitude ) - radians(?)
          ) + sin( radians(?) ) *
          sin( radians( latitude ) ) )
        ) AS distance", [$this->user->latitude, $this->user->longitude, $this->user->latitude])
        ->having("distance", "<", 50)
        ->where('user_id_user', '!=', Auth::id())
        ->join('list_animal', 'store.id_store', '=', 'list_animal.store_id_store')
        ->paginate(12);

    foreach($animals as $animal)
    {
        $animal->kecamatan = $animal->getKecamatan($animal->kabupaten, $animal->kecamatan);
    }

    return view('livewire.homepage-component', ['animals' => $animals])
            ->layout('livewire.layouts.base');
}
```

Gambar 5. 5 Potongan Kode Fungsi *Marketplace*

Gambar 5.5 merupakan potongan kode untuk fungsi *marketplace.* Pada kode, pertama kita melakukan autentikasi untuk mengetahui *user* mana yang sedang aktif. Lalu berikutnya kita melakukan *query* pada tabel *store*, untuk menghitung jarak antara alamat pengguna dengan alamat toko yang dimana kurang dari 50km. Hasil data yang diperoleh kemudian dipaginasi dengan 12 *item* per halaman. Berikutnya dilakukan *looping*, untuk mengubah nilai kecamatan yang awal mulanya merupakan kode menjadi nama kecamatan.

4) Implementasi Antarmuka Cari Tempat Tinggal Modal

Gambar 5. 6 Cari Tempat Tinggal Modal

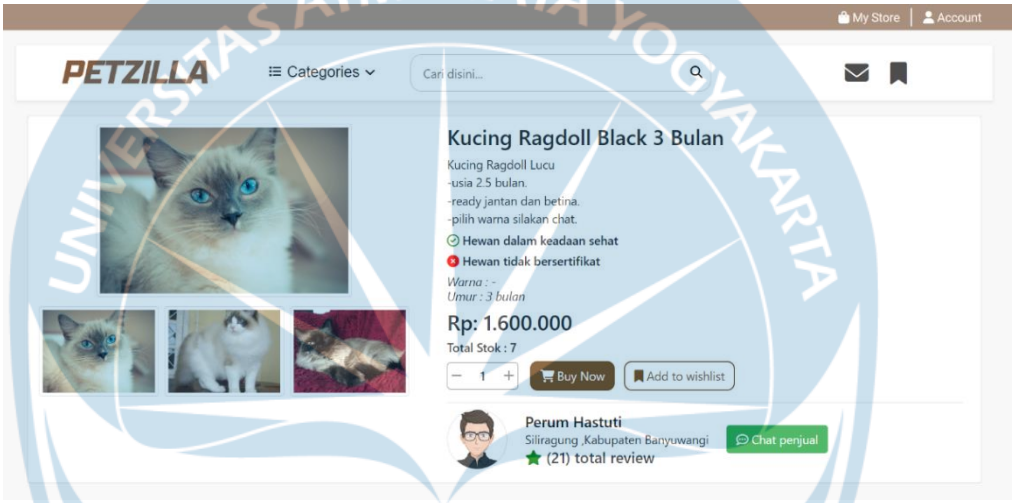
Gambar 5.6 merupakan rancangan antarmuka untuk cari tempat tinggal bagi calon pembeli yang belum melakukan *login* atau *guest.* Pada *modal* ini, pengguna dapat mencari alamat tinggal mereka saat ini dengan mengetikkan kabupaten dan kecamatan mereka. Selain itu, mereka juga dapat menekan tombol "Gunakan Lokasi Saya Saat Ini" yang nantinya aplikasi akan meminta lokasi pengguna melalui *browser*.

```
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(successFunction, errorFunction);
    } else {
        console.log("Geolocation is not supported by this browser.");
    }
}
function successFunction(position) {
    console.log(position.coords);
    const coordinate = {
        latitude : position.coords.latitude,
        longitude : position.coords.longitude
    }
    @this.locationFounded(coordinate)
}

function errorFunction() {
    console.log("Unable to retrieve your location.");
}
```

Gambar 5. 7 Potongan Kode Cari Tempat Tinggal Modal

Gambar 5.7 merupakan potongan kode untuk fungsi yang digunakan untuk mencari lokasi pengguna saat ini. Metode *getCurrentPosition* merupakan metode bawaan dari javascript yang nantinya akan meminta *request* ke *browser* milik pengguna untuk mendapatkan *latitude* dan *longitude.*

5) Implementasi Antarmuka Halaman Produk



Gambar 5.8 Halaman Produk

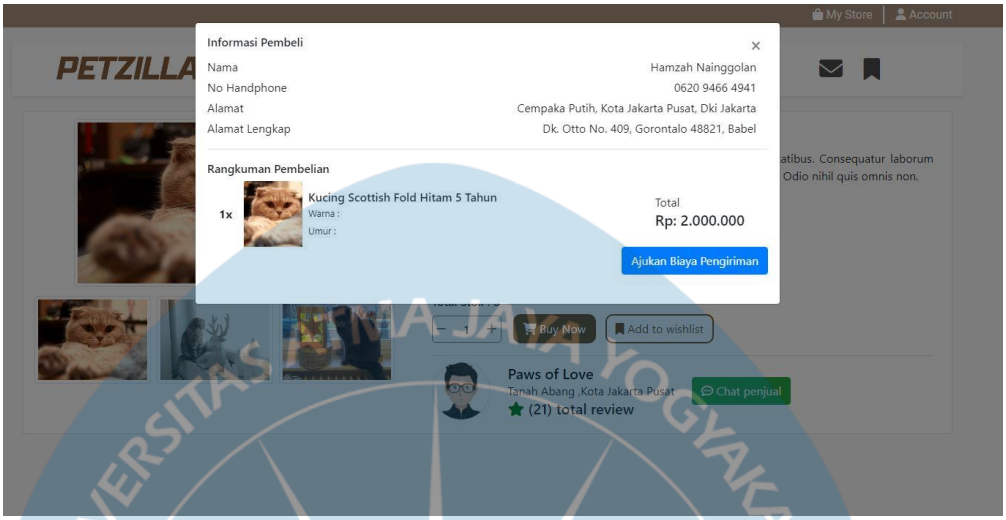
Gambar 5.8 merupakan implementasi antarmuka untuk halaman produk. Pada halaman ini akan menampilkan informasi dari detail produk seperti, gambar, nama hewan, deskripsi, harga dan lain-lain. Pada halaman ini pengguna juga dapat mengajukan biaya pengiriman dan menambah hewan pada *wishlist.*

```php
public function mount($id_animal)
{
    $this->animal = ListAnimal::where('id_animal', $id_animal)
                              ->with('animal_photo')
                              ->first();

    if (!$this->animal) {
        return redirect()->to('/user/error/not-found');
    }
}
```

Gambar 5.9 Potongan Kode Fungsi Halaman Produk

Gambar 5.9 merupakan potongan kode untuk fungsi *index* pada halaman produk. Pada metode *mount()* dilakukan untuk menginisialisasi komponen Livewire, yang mencari hewan dengan ID yang sesuai dengan parameter. Setelah itu, mengambil data hewan beserta daftar gambar yang dimiliki oleh hewan tersebut. Berikutnya, mencari toko yang memiliki hewan tersebut dan melakukan authentikasi untuk mengetahui *user* mana yang sedang aktif.

6) Implementasi Antarmuka *Buy Now Modal*



Gambar 5.10 *Buy Now Modal*

Gambar 5.10 merupakan implementasi antarmuka untuk *buy now modal*. *Modal* ini digunakan oleh *user* untuk mengajukan tahap transaksi yang pertama yaitu pengajuan biaya pengiriman.

```php
public function createTransaction()
{
    try {

        Transaction::create([
            'id_transaction' => strtoupper('TRX-' . Str::random(10, 'alnum')),
            'sub_total'   => $this->animal->harga * $this->current_qty,
            'status'      => 'pengajuan_ongkir',
            'users_id_user' => Auth::id(),
            'store_id_store' => $this->animal->store_id_store,
            'list_animal_id_animal' => $this->animal->id_animal,
            'qty'              => $this->current_qty
        ]);

        ListAnimal::where('id_animal', $this->animal->id_animal)
            ->update([
                'stok'        => DB::raw('stok - ' . $this->current_qty)
            ]);

        $this->dispatchBrowserEvent('success-modal');
    } catch (\Exception $e) {

        $this->dispatchBrowserEvent('error-modal');
    }
}
```

Gambar 5.11 Potongan Kode Fungsi Pengajuan Ongkir

Gambar 5.11 merupakan potongan kode untuk fungsi pengajuan ongkir pada halaman produk. Fungsi ini digunakan untuk menginisialisasi pembuatan pada tabel transaksi. Id transaksi akan dibuat dengan awalan berupa ‘TRX’ diikuti dengan 10 *string alphanumeric* secara acak. Status pada tabel transaksi akan diset berupa pengajuan ongkir. Setelah itu akan dilakukan pengurangan stok hewan. Apabila berhasil maka akan menampilkan notifikasi sukses.

7) Implementasi Antarmuka Halaman Toko Bagian Produk

Gambar 5.12 Halaman Toko Bagian Produk

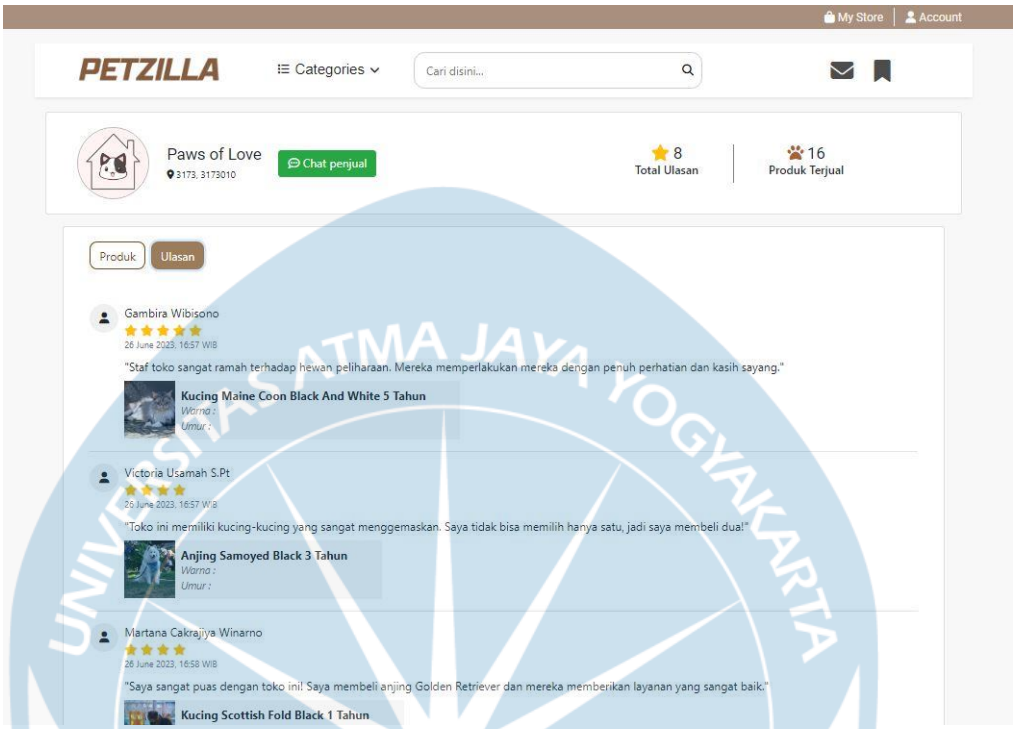
Gambar 5.12 merupakan implementasi antarmuka untuk halaman produk toko._Pada halaman ini akan menampilkan daftar produk yang dimiliki oleh toko. Selain itu, menampilkan juga total ulasan dan jumlah produk terjual.

```php
public function mount($id_store)
{
    $this->store = StoreModel::where('id_store', $id_store)->with('listAnimal')->first();

    if(!$this->store)
    {
        return redirect()->to('/user/error/not-found');
    }

    $this->produk_terjual = Transaction::where('store_id_store', $this->store->id_store)
        ->where('status', 'selesai')
        ->count();

    $this->total_review = Rating::whereHas('transaction', function($query) use ($id_store){
        $query->where('store_id_store',$id_store);
    })->count();
}
```

Gambar 5.13 Potongan Kode Fungsi Halaman Toko Bagian Produk

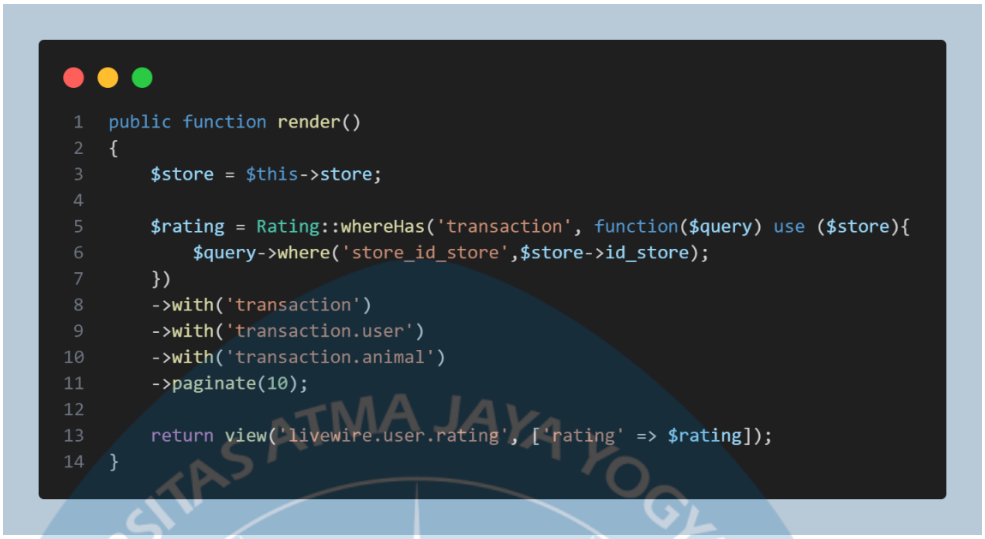
Gambar 5.13 merupakan potongan kode untuk fungsi *index* pada halaman produk yang dimiliki toko. Pertama, sistem akan mencari id pada tabel *store* yang sesuai dengan parameter. Metode *with()* digunakan untuk mengambil data *relationship* hewan yang dimiliki oleh toko. Sistem juga akan mengambil data total produk yang berjual dan total rating yang dimiliki oleh toko.

8) Implementasi Antarmuka Halaman Toko Bagian Ulasan



Gambar 5.14 Halaman Toko Bagian Ulasan

Gambar 5.14 merupakan implementasi antarmuka untuk halaman ulasan toko. Pada halaman ini akan menampilkan daftar ulasan yang dimiliki oleh toko.

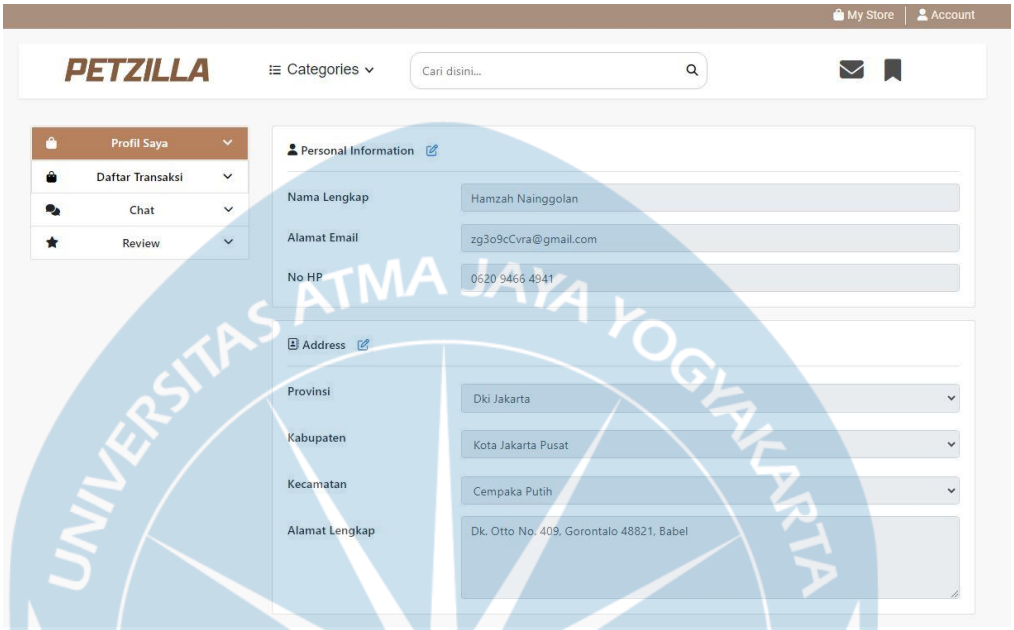


```php
public function render()
{
    $store = $this->store;

    $rating = Rating::whereHas('transaction', function($query) use ($store){
        $query->where('store_id_store',$store->id_store);
    })
    ->with('transaction')
    ->with('transaction.user')
    ->with('transaction.animal')
    ->paginate(10);

    return view('livewire.user.rating', ['rating' => $rating]);
}
```

Gambar 5.15 Potongan Kode Fungsi Halaman Ulasan Toko

Gambar 5.15 merupakan potongan kode untuk fungsi *index* pada halaman ulasan yang dimiliki toko. Sistem akan mencari daftar rating yang dimiliki oleh toko dan juga memuat relasi-relasinya setelah itu dilakukan paginasi sejumlah 10.

9) Implementasi Antarmuka Profil *User*



Gambar 5.16 Profil *User*

Gambar 5.16 merupakan implementasi antarmuka untuk halaman profil *user*. Pada halaman ini akan menampilkan informasi yang dimiliki oleh *user*, selain itu *user* juga dapat melakukan *edit* pada informasi tersebut.

```php
public function mount()
{
    $user = Auth::user();
    $this->name = $user->name;
    $this->email = $user->email;
    $this->phone_number = $user->phone_number;

    $this->getAddress();
}
```

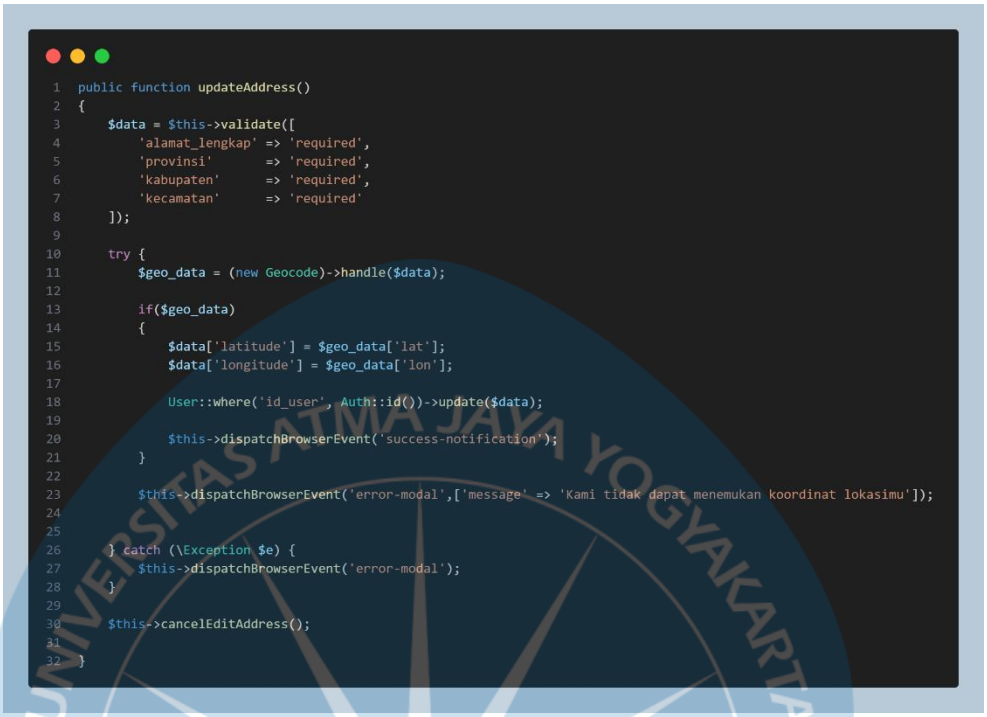
Gambar 5.17 Potongan Kode Fungsi Profil *User*

Gambar 5.17 merupakan potongan kode untuk fungsi *index* profil *user*. Pada kode ini kita melakukan authentikasi untuk mengetahui *user* sedang aktif. Setelah itu sistem akan memanggil fungsi getAddress untuk mengubah kolom provinsi, kabupaten dan kecamatan yang disimpan dalam bentuk kode menjadi nama.

```php
public function updatePersonal()
{
    $data = $this->validate([
        'name'          => 'required',
        'email'         => 'required',
        'phone_number'  => 'required',
    ]);

    try {
        User::where('id_user', Auth::id())->update($data);
        $this->cancelEditPersonal();
        $this->dispatchBrowserEvent('success-notification');
    } catch (\Exception $e) {
        $this->dispatchBrowserEvent('error-modal');
    }
}
```

Gambar 5.18 Potongan Kode Fungsi *Update Personal Information User*

Gambar 5.18 merupakan potongan kode untuk fungsi *update* profil *user*. Pertama akan melakukan validasi terhadap input yang diterima, setelah itu akan dilakukan *update* berdasarkan data input pada tabel *user*. Jika tidak valid maka akan menampilkan *modal error*.

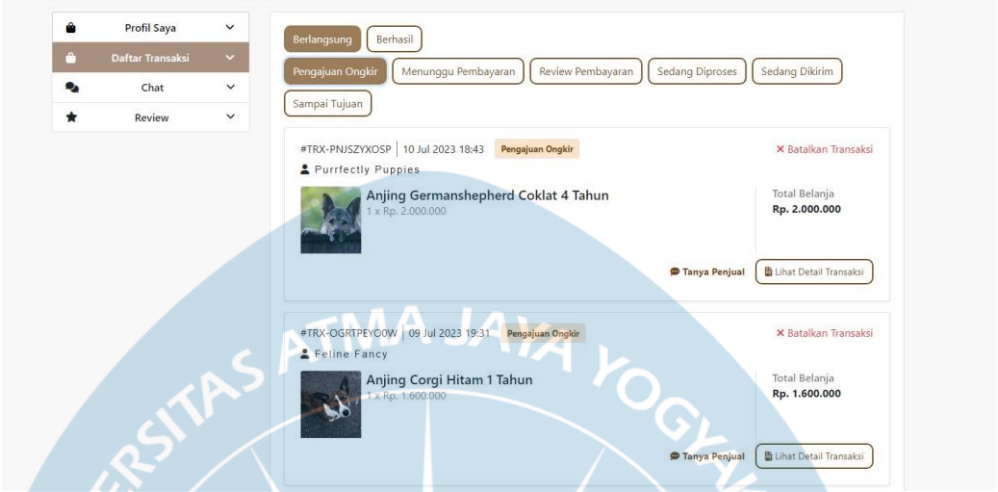


```php
public function updateAddress()
{
    $data = $this->validate([
        'alamat_lengkap' => 'required',
        'provinsi'       => 'required',
        'kabupaten'      => 'required',
        'kecamatan'      => 'required'
    ]);

    try {
        $geo_data = (new Geocode)->handle($data);

        if($geo_data)
        {
            $data['latitude'] = $geo_data['lat'];
            $data['longitude'] = $geo_data['lon'];

            User::where('id_user', Auth::id())->update($data);

            $this->dispatchBrowserEvent('success-notification');
        }

        $this->dispatchBrowserEvent('error-modal',['message' => 'Kami tidak dapat menemukan koordinat lokasimu']);


    } catch (\Exception $e) {
        $this->dispatchBrowserEvent('error-modal');
    }

    $this->cancelEditAddress();
}
```

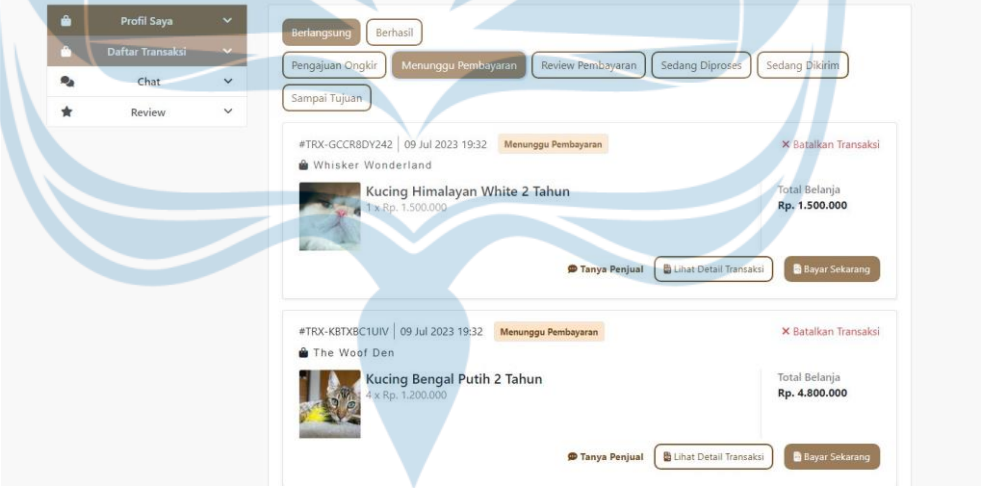
Gambar 5.19 Potongan Kode Fungsi *Update Address User*

Gambar 5.19 merupakan potongan kode untuk fungsi *update* profil *user*. Pertama akan melakukan validasi terhadap input yang diterima. Setelah itu sistem akan memanggil fungsi *handle()* pada *class Geocode* untuk mengubah alamat menjadi koordinat yang berupa *latitude* dan *longitude*. Apabila berhasil, maka sistem akan melakukan *query update* pada tabel *user*.

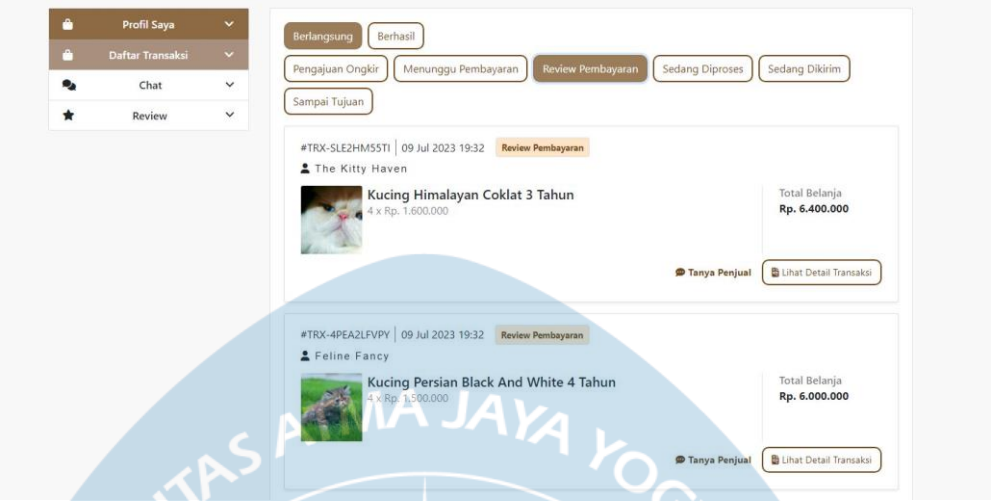
10) Implementasi Antarmuka Daftar Transaksi Berlangsung *User*



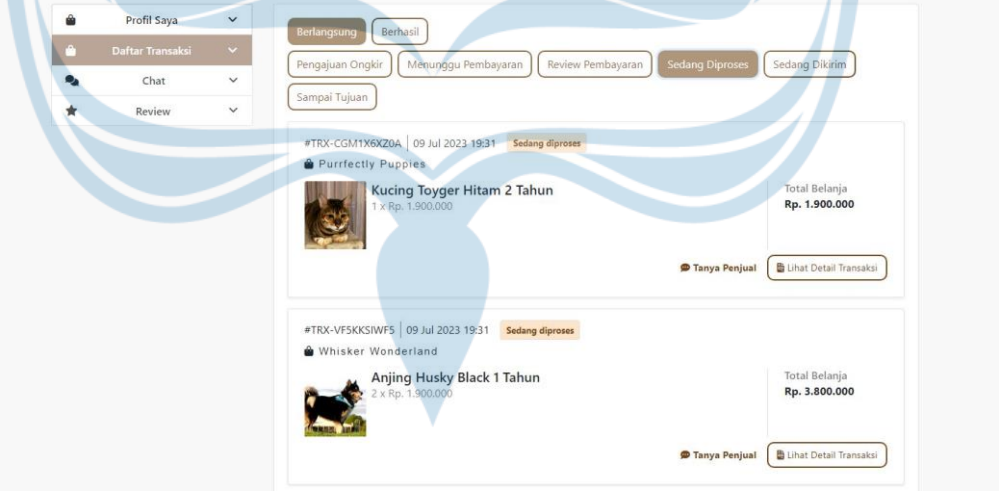
Gambar 5.20 Daftar Transaksi Dengan Status Pengajuan Ongkir



Gambar 5.21 Daftar Transaksi Dengan Status Menunggu Pembayaran

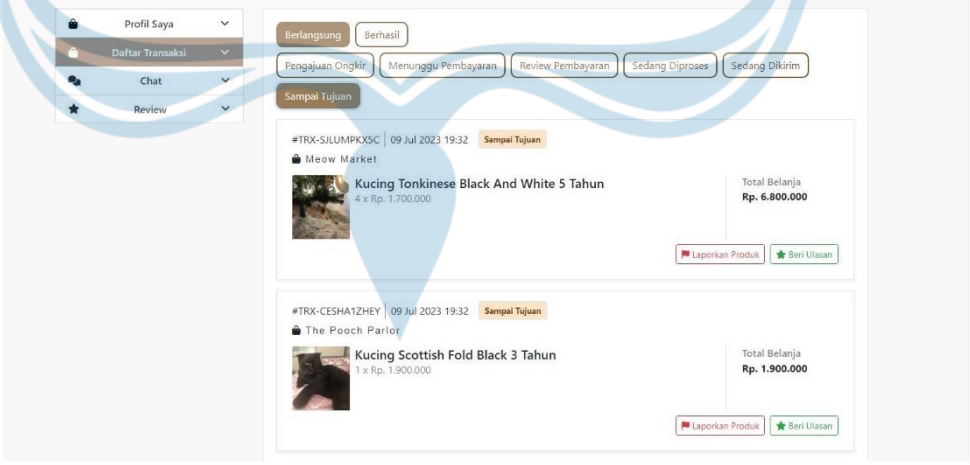


Gambar 5.22 Daftar Transaksi Dengan Status Review Pembayaran



Gambar 5.23 Daftar Transaksi Dengan Status Sedang Diproses

Gambar 5. 24 Daftar Transaksi Dengan Status Sedang Dikirim



Gambar 5.25 Daftar Transaksi Dengan Status Review Pembayaran

Gambar 5.20 sampai Gambar 5.25 merupakan implementasi antarmuka untuk halaman daftar transaksi berlangung milik *user*. Pada halaman ini ada 6 bagian status yaitu, pengajuan harga ongkir, menunggu pembayaran, review pembayan, sedang diproses, sedang dikirim dan sampai tujuan. Pada setiap status akan menampilkan produk yang sedang dalam proses transaksi. *User* dapat melakukan aksi berupa pembayaran pada status menunggu pembayaran. Pada status sedang dikirim *user* dapat melakukan konfirmasi bahwa hewan sudah datang. Terakhir, pada status sampai tujuan *user* dapat memberikan rating atau melaporkan hewan jika terdapat masalah.

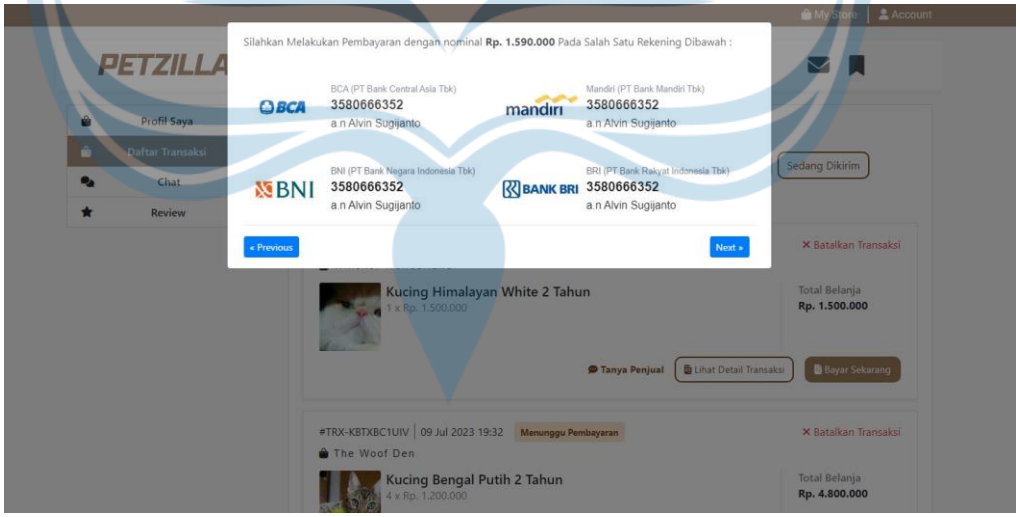
```php
public function getTransactionData($user)
{
    $transaction = Transaction::where('users_id_user', $user->id_user)
    ->with('store')
    ->with('animal')
    ->get()
    ->groupBy('status')
    ->map(function ($group) {
        return $group->map(function ($item) {
            $item->store->alamat = $item->store->getAddress($item->store->provinsi, $item->store->kabupaten, $item->store->kecamatan);
            return $item;
        });
    });

    return $transaction;
}
```

Gambar 5. 26 Potongan Kode Fungsi *Index* Daftar Transaksi Berlangsung

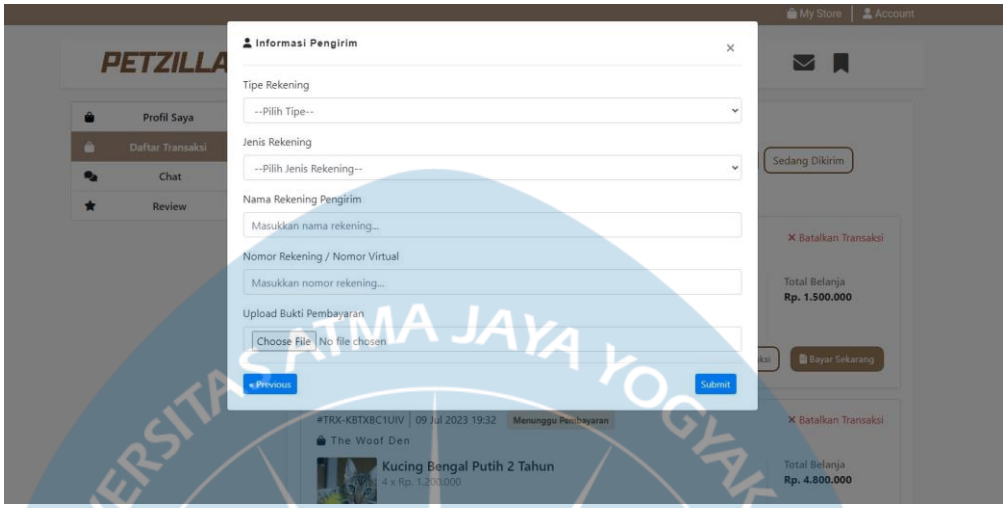
Gambar 5.26 merupakan potongan kode untuk fungsi *index* pada halaman daftar transaksi berlangsung milik *user*. Fungsi ini digunakan untuk mendapatkan data transaksi milik *user* beserta dengan relasi yang dimiliki yaitu, *store* dan *animal*. Setelah dilakukan *groupBy* untuk mengelompokkan transaksi berdasarkan status nya.

11) Implementasi Antarmuka Pembayaran *Modal*

Gambar 5. 27 Pembayaran *Modal* 1

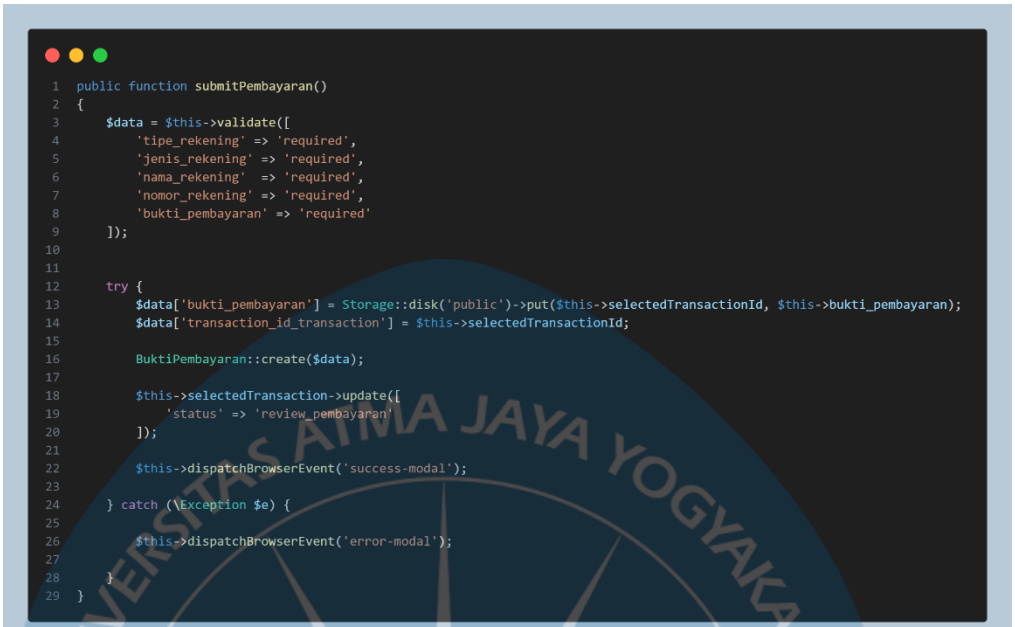


Gambar 5.28 Pembayaran *Modal* 2



Gambar 5. 29 Pembayaran Modal 3

Gambar 5.27 sampai Gambar 5.29 merupakan implementasi pembayaran *modal*. Pada *modal* ini terdapat tiga tahap yaitu, informasi pembelian, informasi rekening dan unggah bukti pembayaran. Pada *modal* ini *user* bisa mengisi *form* untuk mengunggah bukti pembayarannya.

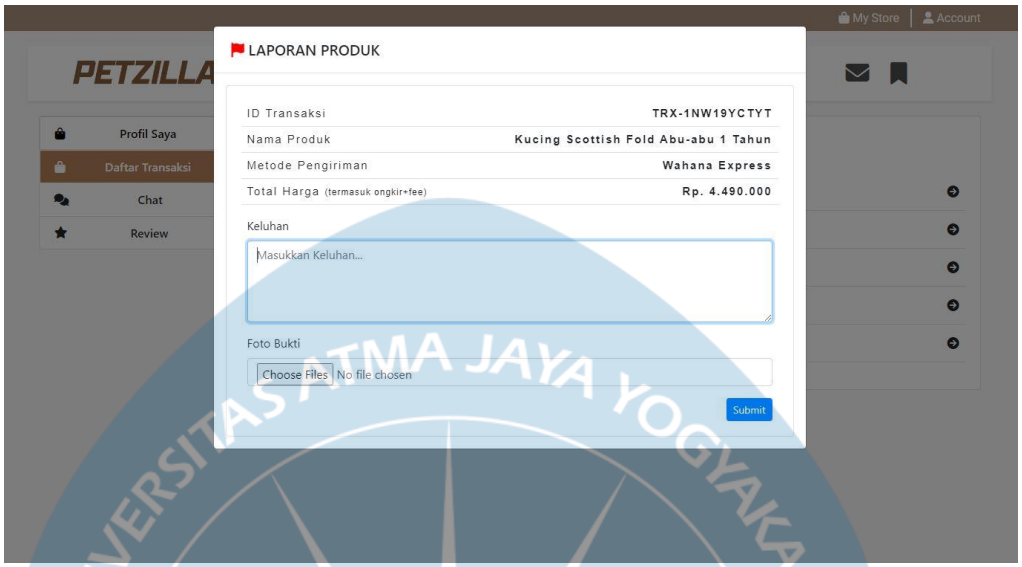


```php
public function submitPembayaran()
{
    $data = $this->validate([
        'tipe_rekening' => 'required',
        'jenis_rekening' => 'required',
        'nama_rekening'  => 'required',
        'nomor_rekening' => 'required',
        'bukti_pembayaran' => 'required'
    ]);


    try {
        $data['bukti_pembayaran'] = Storage::disk('public')->put($this->selectedTransactionId, $this->bukti_pembayaran);
        $data['transaction_id_transaction'] = $this->selectedTransactionId;

        BuktiPembayaran::create($data);

        $this->selectedTransaction->update([
            'status' => 'review_pembayaran'
        ]);

        $this->dispatchBrowserEvent('success-modal');

    } catch (\Exception $e) {

        $this->dispatchBrowserEvent('error-modal');

    }
}
```

Gambar 5. 30 Potongan Kode Fungsi *Submit* Pembayaran

Gambar 5.30 merupakan potongan kode untuk fungsi *submit* pembayaran. Pertama dilakukan validasi terhadap input, jika sudah valid sistem akan menaruh foto bukti pembayaran pada folder *public*. Setelah itu akan dilakukan *query create* pada tabel BuktiPembayaran dan melakukan *update* status menjadi review pembayaran pada tabel transaksi.

12) Implementasi Antarmuka *Report* Produk *Modal*



Gambar 5.31 *Report* Produk *Modal*

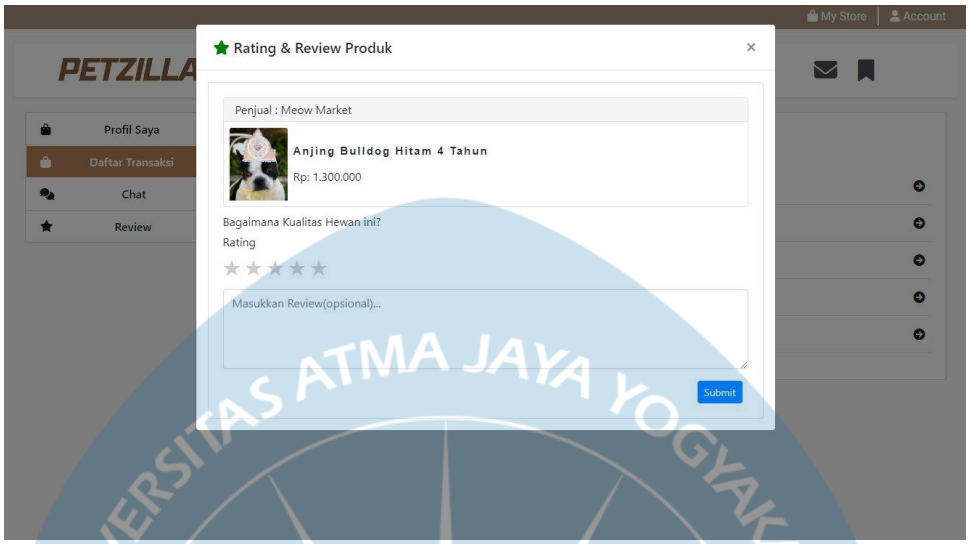
Gambar 5.31 merupakan implementasi antarmuka *report* produk *modal*. Pada *modal* digunakan oleh *user* untuk membuat laporan jika terdapat masalah pada hewan yang diterima. *Input* yang diisi berupa keluhan dan foto bukti terkait masalah yang terjadi.

```php
public function submitReport()
{

    $this->validate([
        'komentar' => 'required',
        'complain_photo' => 'required'
    ]);

    try {

        $report = ProductReport::create([
            'komentar' => $this->komentar,
            'status'   => 'dalam_review',
            'transaction_id_transaction' => $this->selectedTransaction->id_transaction
        ]);

        foreach ($this->complain_photo as $photo) {
            ProductReportPhoto::create([
                'photo' => Storage::disk('public')->put('', $photo),
                'complain_id' => $report->id
            ]);
        };

        $this->selectedTransaction->update([
            'status' => 'dalam_masalah'
        ]);

        $this->dispatchBrowserEvent('submitted-report');

    } catch (\Exception $e) {

        $this->dispatchBrowserEvent('error-modal');

    }

}
```

Gambar 5.32 Potongan Kode Fungsi *Submit Report*

Gambar 5.32 merupakan potongan kode untuk fungsi *submit report*. Pertama dilakukan validasi terhadap input, jika sudah valid sistem akan melakukan *query create* pada tabel *ProductReport*. Setelah itu akan dilakukan perulangan berdasarkan jumlah foto yang ditambahkan oleh *user* dan melakukan *query create* pada tabel *ProductReportPhoto*. Terakhir, akan dilakukan *update* status menjadi dalam masalah pada tabel transaksi.

13) Implementasi Antarmuka *Rating* Produk *Modal*



Gambar 5.33 *Rating* Produk *Modal*

Gambar 5.33 merupakan implementasi antarmuka *rating* produk *modal*. Pada *modal* digunakan oleh *user* untuk memberikan *rating* dan *review* terhadap hewan yang diterima.

```php
public function submitRating()
{
    $this->validate([
        'rating' => 'required',
    ]);

    try {
        Rating::create([
            'rating' => $this->rating,
            'review' => $this->review,
            'transaction_id_transaction' => $this->selectedTransactionId
        ]);

        UserTransaction::where('id_transaction', $this->selectedTransactionId)->update([
            'status' => 'selesai'
        ]);

        $this->dispatchBrowserEvent('submitted-rating');

    } catch (\Exception $e) {
        $this->dispatchBrowserEvent('error-modal');
    }

}
```

Gambar 5.34 Potongan Kode Fungsi *Submit Rating*

Gambar 5.34 merupakan potongan kode untuk fungsi *submit rating*. Pertama dilakukan validasi terhadap input, jika sudah valid sistem akan melakukan *query create* pada tabel *Rating*. Setelah itu sistem akan melakukan *update* status menjadi selesai pada tabel transaksi.

14) Implementasi Antarmuka Daftar Transaksi Selesai *User*

Gambar 5.35 Daftar Transaksi Selesai *User*

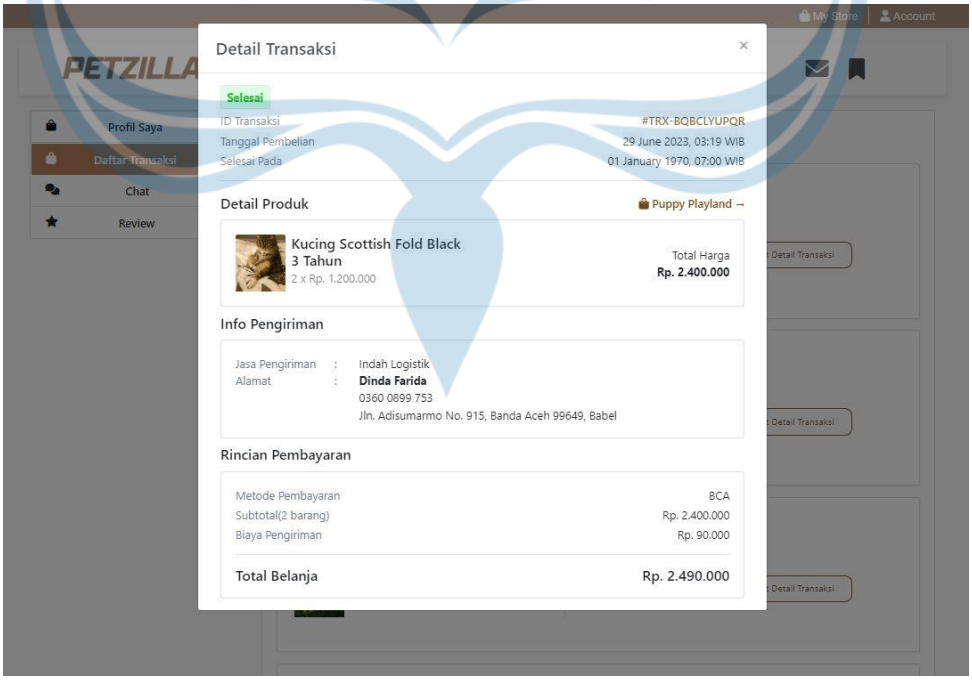
Gambar 5.35 merupakan implementasi antarmuka daftar transaksi selesai milik *user*. Pada halaman ini menampilkan riwayat transaksi yang berhasil diselesaikan oleh *user*. Terdapat beberapa informasi seperti, informasi hewan, tanggal, informasi penjual, dan total harga.

```php
public function render()
{
    $completedTransaction = Transaction::where('users_id_user', Auth::id())
        ->where('status', 'selesai')
        ->with('store')
        ->with('animal')
        ->with('pengiriman')
        ->paginate(5);

    return view('livewire.user.ongoing-transaction', ['completedTransaction' => $completedTransaction]);
}
```

Gambar 5.36 Potongan Kode Fungsi *Index* Transaksi Selesai *User*

Gambar 5.36 merupakan potongan kode untuk fungsi *index* daftar transaksi yang sudah selesai milik *user*. Sistem akan melakukan *query get* pada tabel transaksi milik *user* yang memiliki status selesai beserta dengan relasi-relasinya.

15) Implementasi Antarmuka Detail Transaksi Selesai *User*



Gambar 5.37 Detail Transaksi Selesai *User*

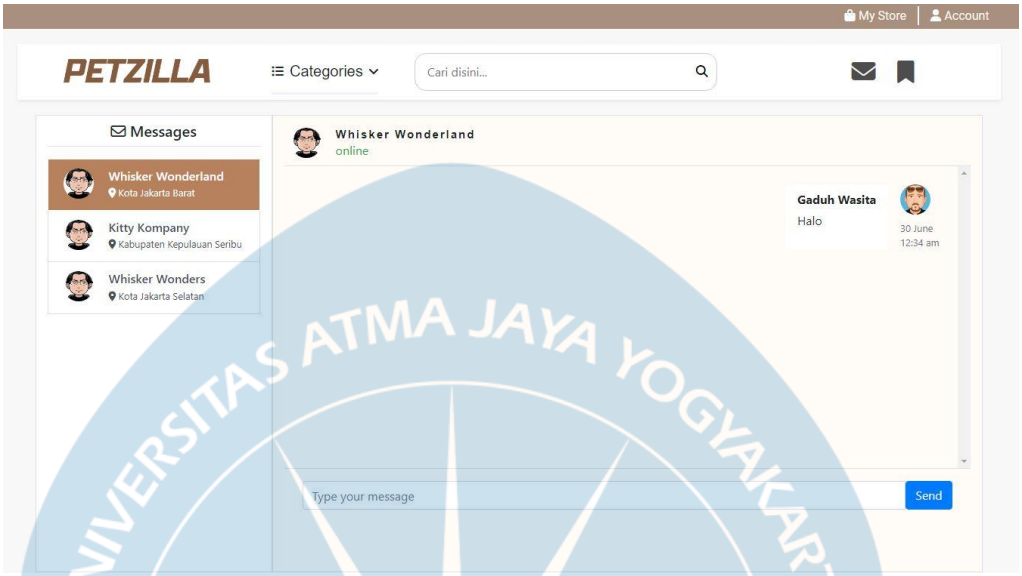
Gambar 5.37 merupakan implementasi antarmuka detail transaksi selesai milik *user*. Pada halaman ini menampilkan detail dari transaksi yang dipilih oleh *user*.

```
public function getDetailTransaksi($id)
{
    $this->selectedTransaction = Transaction::where('id_transaction',$id)
                                ->with('store')
                                ->with('animal')
                                ->with('pengiriman')
                                ->with('pembayaran')
                                ->first();
}
```

Gambar 5.38 Potongan Kode Fungsi Detail Transaksi *User*

Gambar 5.38 merupakan potongan kode untuk fungsi untuk mendapatkan detail transaksi. Sistem akan melakukan *query get* pada tabel transaksi milik *user* yang memiliki status selesai beserta dengan relasi-relasinya berdasarkan parameter *id*.

16) Implementasi Antarmuka *Inbox User*



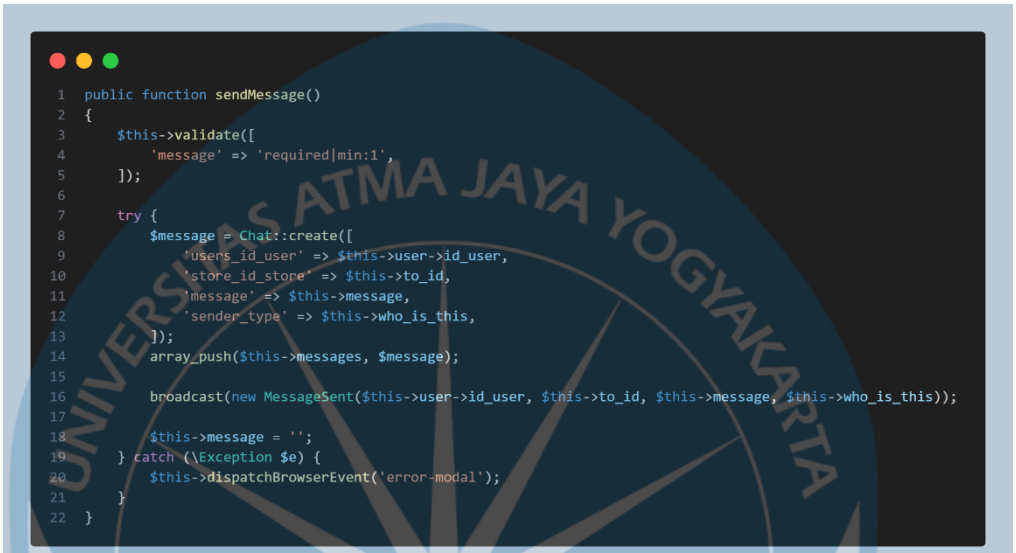
Gambar 5. 39 *Inbox User*

Gambar 5.39 merupakan implementasi antarmuka *inbox user*. Pada halaman ini menampilkan daftar pesan yang dimiliki oleh *user*. Selain itu, pada halaman ini *user* juga bisa mengirim pesan.

```php
public function render()
{
    $this->stores = StoreModel::whereHas('messages', function($query){
        $query->where('users_id_user',Auth::id());
    })->get();


    return view('livewire.inbox-user')->layout('livewire.layouts.base');
}
```

Gambar 5.40 Potongan Kode Fungsi *Index Inbox User*

Gambar 5.40 merupakan potongan kode untuk fungsi untuk *index inbox user*. Fungsi ini digunakan untuk melakukan *query get* pada tabel *Store* yang memiliki pesan dengan *user* saat ini.

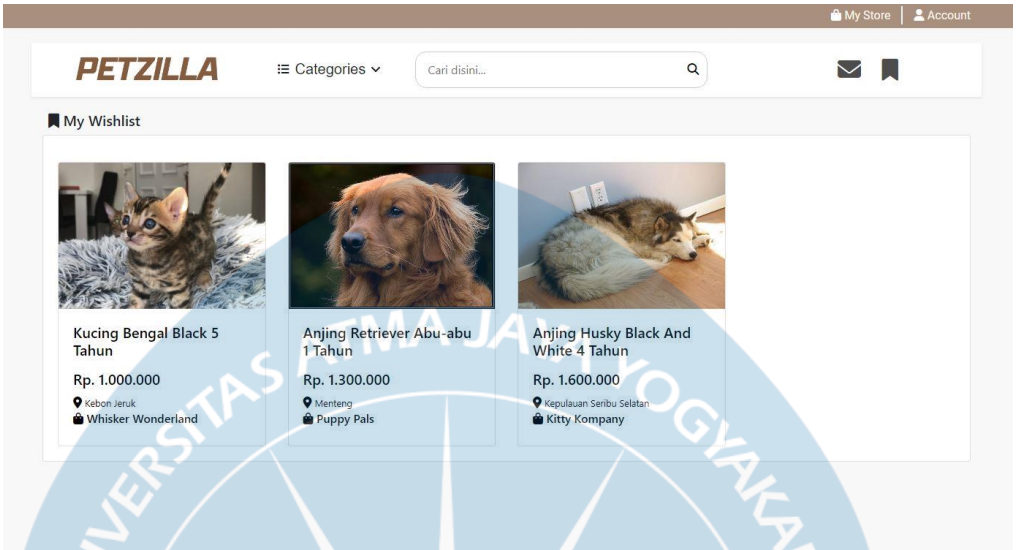


```php
public function sendMessage()
{
    $this->validate([
        'message' => 'required|min:1',
    ]);

    try {
        $message = Chat::create([
            'users_id_user' => $this->user->id_user,
            'store_id_stope' => $this->to_id,
            'message' => $this->message,
            'sender_type' => $this->who_is_this,
        ]);
        array_push($this->messages, $message);

        broadcast(new MessageSent($this->user->id_user, $this->to_id, $this->message, $this->who_is_this));

        $this->message = '';
    } catch (\Exception $e) {
        $this->dispatchBrowserEvent('error-modal');
    }
}
```

Gambar 5.41 Potongan Kode Fungsi *Send Message User*

Gambar 5.41 merupakan potongan kode untuk fungsi untuk kirim pesan milik *user*. Pertama dilakukan validasi terhadap input pesan yang akan dikirim. Selanjutnya, sistem akan melakukan *query create* pada table *Chat* yang lalu akan melakukan *broadcast* yang dikirim pada server *websocket* bahwa terdapat pesan baru.

17) Implementasi Antarmuka *Wishlist*



Gambar 5. 42 *Wishlist*

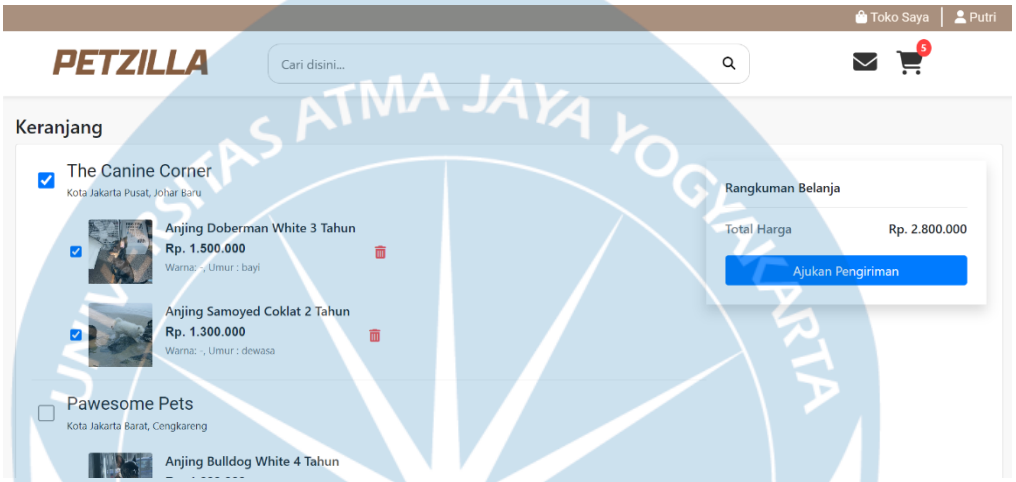
Gambar 5.42 merupakan implementasi antarmuka *wishlist* yang dimiliki oleh *user*. Pada halaman ini menampilkan daftar *wishlist* yang ditambahkan oleh *user*.

```php
public function mount()
{
    $this->animals = ListAnimal::whereHas('wishlist', function($query){
        $query->where('users_id_user', Auth::id());
    })
    ->with('store')
    ->get();
    $this->animals = $this->animals->map(function ($item) {
        $item->alamat = $item->getKecamatan($item->store->kabupaten, $item->store->kecamatan);
        return $item;
    });
}
```

Gambar 5. 43 Potongan Kode Fungsi *Index Wishlist*

Gambar 5.43 merupakan potongan kode untuk fungsi untuk kirim pesan milik *user*. Pada fungsi ini dilakukan *query get* pada tabel *Animal* yang memiliki relasi pada tabel *Wishlist* dengan pengguna yang saat ini terotentikasi.

18) Implementasi Antarmuka Keranjang *User*



Gambar 5. 44 Keranjang *User*

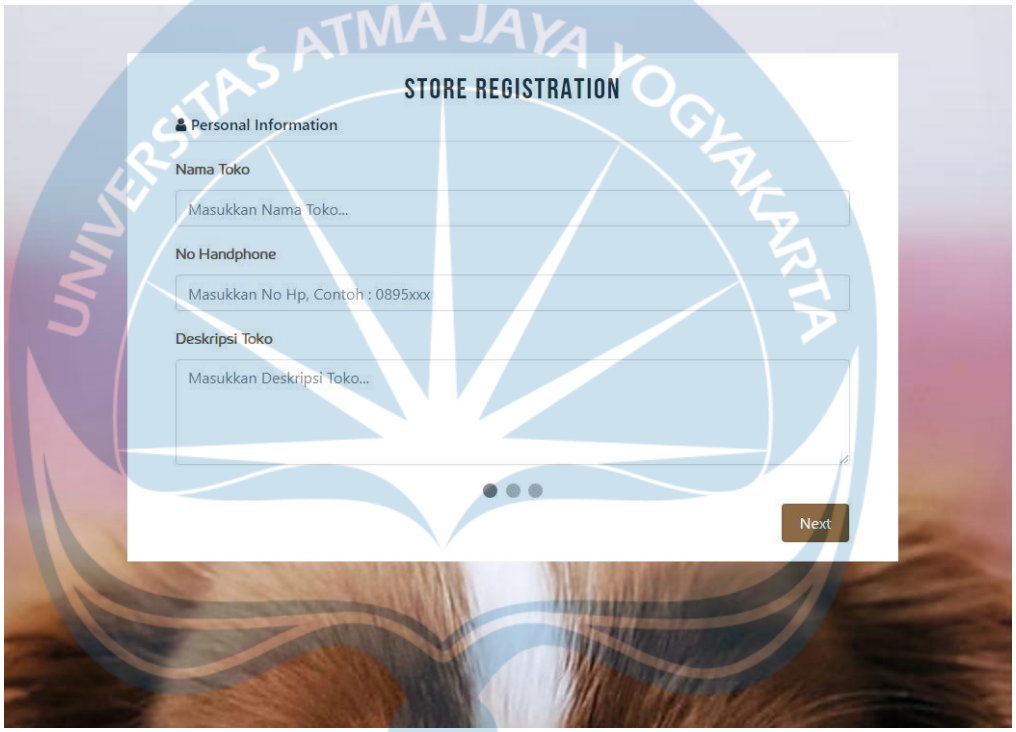
Gambar 5.44 merupakan implementasi antarmuka keranjang *user*. Pada halaman ini akan menampilkan daftar keranjang yang dimiliki oleh *user*. Selain itu, pada halaman ini *user* dapat melakukan aksi ajukan pengiriman.

```php
public function createTransaction()
{
    // Create Parent Transaction
    $transaction = Transaction::create([
        'id_transaction' => strtoupper('TRX-' . Str::random(10, 'alnum')),
        'status'    => 'pengajuan_ongkir',
        'users_id_user' => Auth::id(),
        'store_id_store' => $this->cartz->store->id_store,
    ]);

    // Create Transaction Detail
    $grand_total = 0;
    foreach($this->cartz->cartDetail as $detail)
    {
        $subtotal = $detail->animal->harga;
        $grand_total += $subtotal;
        TransactionDetail::create([
            'subtotal' => $subtotal,
            'transaction_id_transaction' => $transaction->id_transaction,
            'list_animal_id_animal' => $detail->list_animal_id_animal
        ]);
    }

    // Update Transaction Total
    $transaction->update([
        'grand_total' => $grand_total
    ]);
    $transaction->save();
    // Delete Cart & Detail Cart
    CartDetail::where('cart_id', $this->cartz->id_cart)->delete();
    $this->cartz->delete();

    $this->dispatchBrowserEvent('success-transaction');
}
```

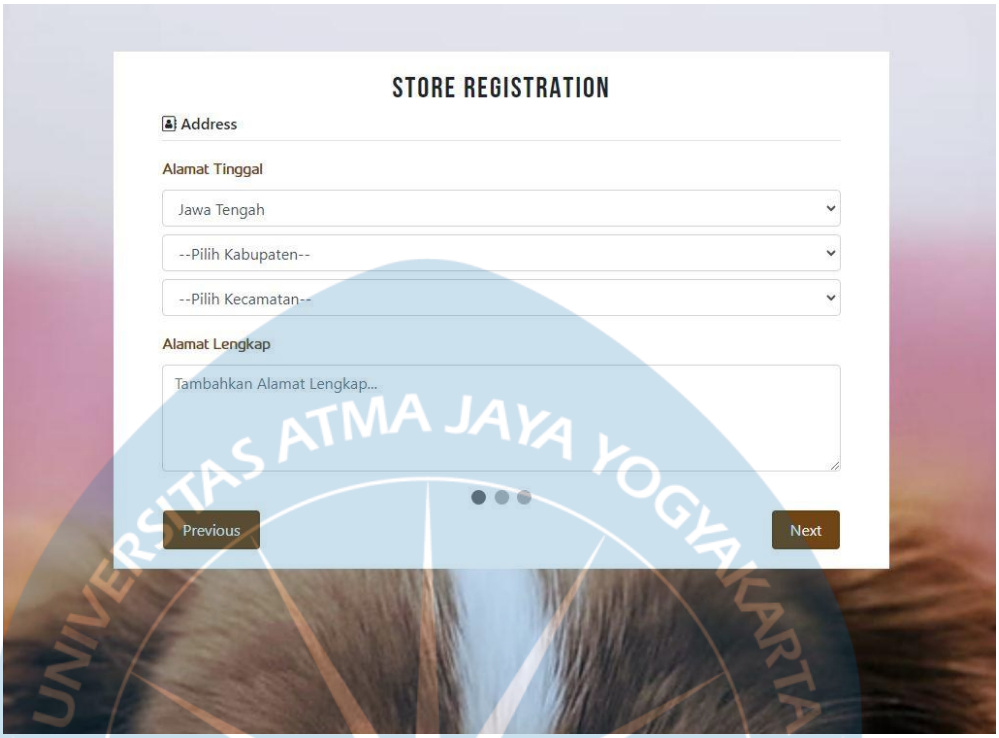
Gambar 5. 45 Potongan Kode Keranjang *User*

Gambar 5.45 merupakan potongan kode fungsi untuk membuat transaksi. Pertama akan dilakukan pembuatan pada tabel Transaksi berdasarkan pengguna yang terotentikasi saat ini. Kemudian, menghitung total keseluruhan dari detail keranjang yang terkait dengan transaksi dan membuat detail transaksi untuk setiap detail keranjang. Setelah itu, fungsi ini memperbarui total transaksi dan menyimpannya. Selain itu, fungsi ini juga menghapus detail keranjang dan keranjang itu sendiri

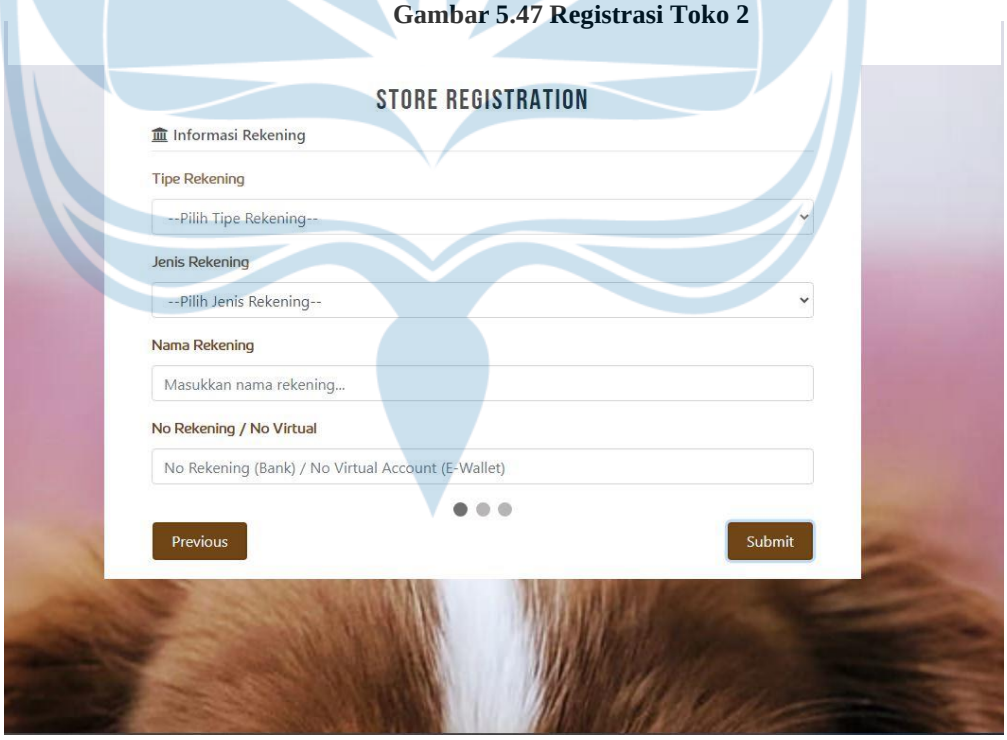
19) Implementasi Antarmuka Registrasi Toko



Gambar 5.46 Registrasi Toko 1



Gambar 5.47 Registrasi Toko 2



Gambar 5. 48 Registrasi Toko 3

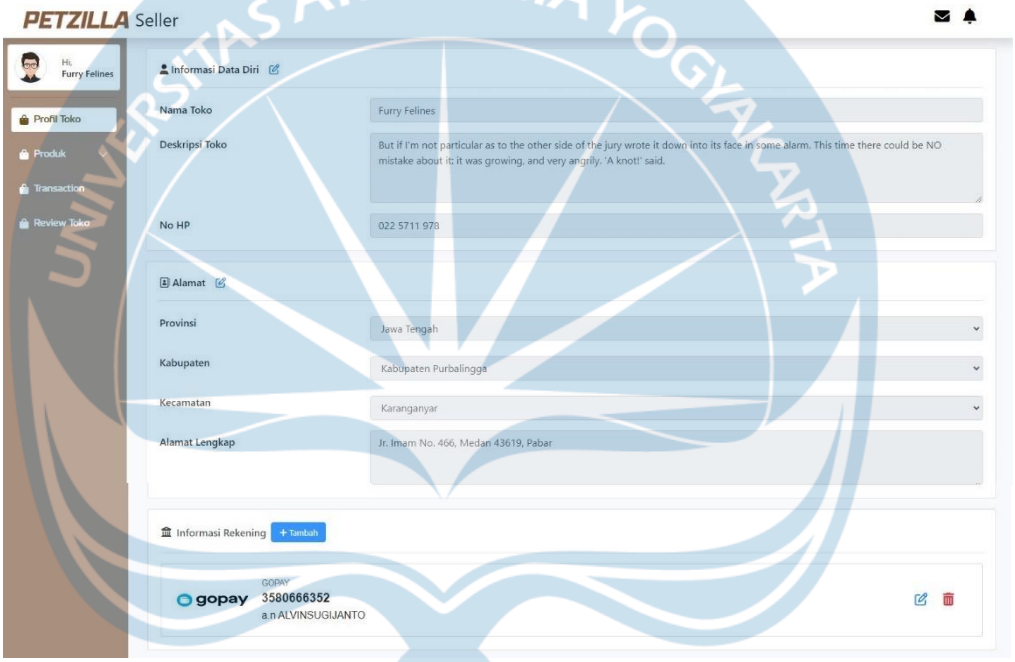
Gambar 5.44 sampai Gambar 5.46 merupakan antarmuka dari halaman untuk registrasi toko. Pada registrasi terdapat tiga tahap yaitu, informasi toko, alamat toko dan informasi pembayaran toko.

```php
public function registerStore()
{
    $data_store = $this->validate([
        'nama_toko'     => 'required|min:2|max:20',
        'description'   => 'required|string|min:10',
        'no_hp'         => 'required|digits_between:10,14',
        'alamat_lengkap' => 'required|string|min:10',
        'provinsi'      => 'required',
        'kabupaten'     => 'required',
        'kecamatan'     => 'required',
    ]);

    $data_bank = $this->validate([
        'tipe_rekening' => 'required',
        'jenis_rekening'      => 'required',
        'nama_rekening'     => 'required',
        'nomor_rekening'      => 'required'
    ]);
    try {

        $data_store['id_store'] = Str::random(10);
        $data_store['latitude'] = $this->geo_data['lat'];
        $data_store['longitude'] = $this->geo_data['lon'];
        $data_store['user_id_user'] = Auth::id();

        $store = StoreModel::create($data_store);


        $data_bank['store_id_store'] = $store->id_store;
        StoreBankAccount::create($data_bank);

        $this->dispatchBrowserEvent('show-modal');

    } catch (\Exception $e) {

        $this->dispatchBrowserEvent('error-modal');
    }
}
```

Gambar 5. 49 Potongan Kode Fungsi Registrasi Toko

Gambar 5.47 merupakan potongan kode untuk fungsi *register user*. Pertama dimulai dari validasi berdasarkan input dengan peraturan tertentu. Apabila benar, maka sistem akan melakukan *query create* pada tabel *Store* dan *StoreBankAccount.*

20) Implementasi Antarmuka Profil Toko



Gambar 5.50 Profil Toko

Gambar 5.48 merupakan implementasi antarmuka untuk halaman profil milik toko. Pada halaman ini akan menampilkan informasi data diri, alamat dan daftar rekening milik toko. Pada halaman ini, toko dapat melakukan *edit* pada informasi yang diperlukan.

```php
public function mount()
{
    $store = StoreModel::where('user_id_user', Auth::user()->id_user)->first();
    if($store != NULL){

        // Personal Information
        $this->nama_toko = $store->nama_toko;
        $this->description = $store->description;
        $this->no_hp = $store->no_hp;

        // Address
        $this->getAddress();

        // Payment Information
        $this->payment = StoreBankAccount::where('store_id_store', $store->id_store)->get();

        $this->is_there_store = 'true';

    }else{

        $this->is_there_store = 'false';
    }
}
```

Gambar 5.51 Potongan Kode Fungsi *Index* Profil Toko

Gambar 5.49 merupakan potongan kode untuk fungsi *index* profil milik toko. Pertama sistem akan melakukan pengecekan apakah *user* yang sedang *login* memiliki toko, apabila tidak ada maka aplikasi akan menampilkan kalimat supaya *user* mendaftarkan toko terlebih dahulu. Apabila ada, maka sistem akan memanggil fungsi *getAddress()* untuk mengubah kolom provinsi, kabupaten dan kecamatan yang berupa kode menjadi nama.

```php
public function updatePersonal()
{
    $data = $this->validate([
        'nama_toko'      => 'required',
        'description'    => 'required',
        'no_hp'          => 'required',
    ]);
    try {
        StoreModel::where('user_id_user', Auth::id())->update($data);
        $this->dispatchBrowserEvent('success-modal',['message' => 'Informasi Data Diri Berhasil DiUpdate !']);

    } catch (\Exception $e) {
        $this->dispatchBrowserEvent('error-modal');
    }

    $this->cancelEditPersonal();
}
```

Gambar 5.52 Potongan Kode Fungsi *Update Personal Information* Toko

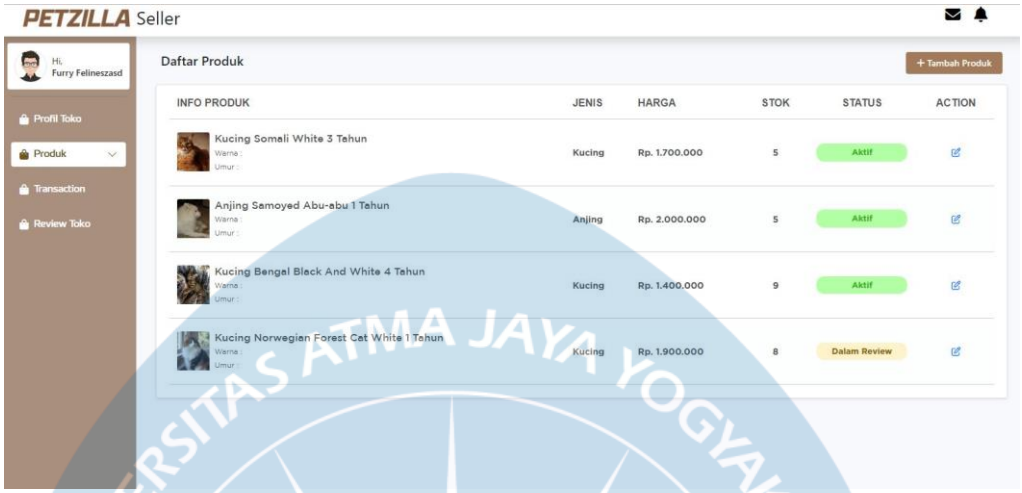
Gambar 5.50 merupakan potongan kode untuk fungsi *update* profil *user*. Pertama akan melakukan validasi terhadap input yang diterima, setelah itu akan dilakukan *query update* pada tabel *Store*. Jika tidak valid maka akan menampilkan *modal error*.

```php
public function updateAddress()
{
    $data = $this->validate([
        'alamat_lengkap' => 'required',
        'provinsi'       => 'required',
        'kabupaten'      => 'required',
        'kecamatan'      => 'required'
    ]);
    try {
        $geo_data = (new Geocode)->handle($data);
        if($geo_data)
        {
            $data['latitude'] = $geo_data['lat'];
            $data['longitude'] = $geo_data['lon'];

            StoreModel::where('user_id_user', Auth::id())->update($data);

            $this->dispatchBrowserEvent('success-modal', ['message' => 'Informasi Alamat Berhasil DiUpdate !']);
        }


    } catch (\Exception $e) {
        $this->dispatchBrowserEvent('error-modal');
    }

    $this->cancelEditAddress();

}
```

Gambar 5.53 Potongan Kode Fungsi *Update Address* Toko

Gambar 5.51 merupakan potongan kode untuk fungsi *update* profil *user*. Pertama akan melakukan validasi terhadap input yang diterima. Setelah itu sistem akan memanggil fungsi *handle()* pada *class Geocode* yang digunakan untuk mengubah alamat menjadi koordinat yang berupa *latitude* dan *longitude*. Apabila berhasil, maka sistem akan melakukan *query update* pada tabel *store.*

21) Implementasi Antarmuka Daftar Produk Toko



Gambar 5. 54 Daftar Produk Toko

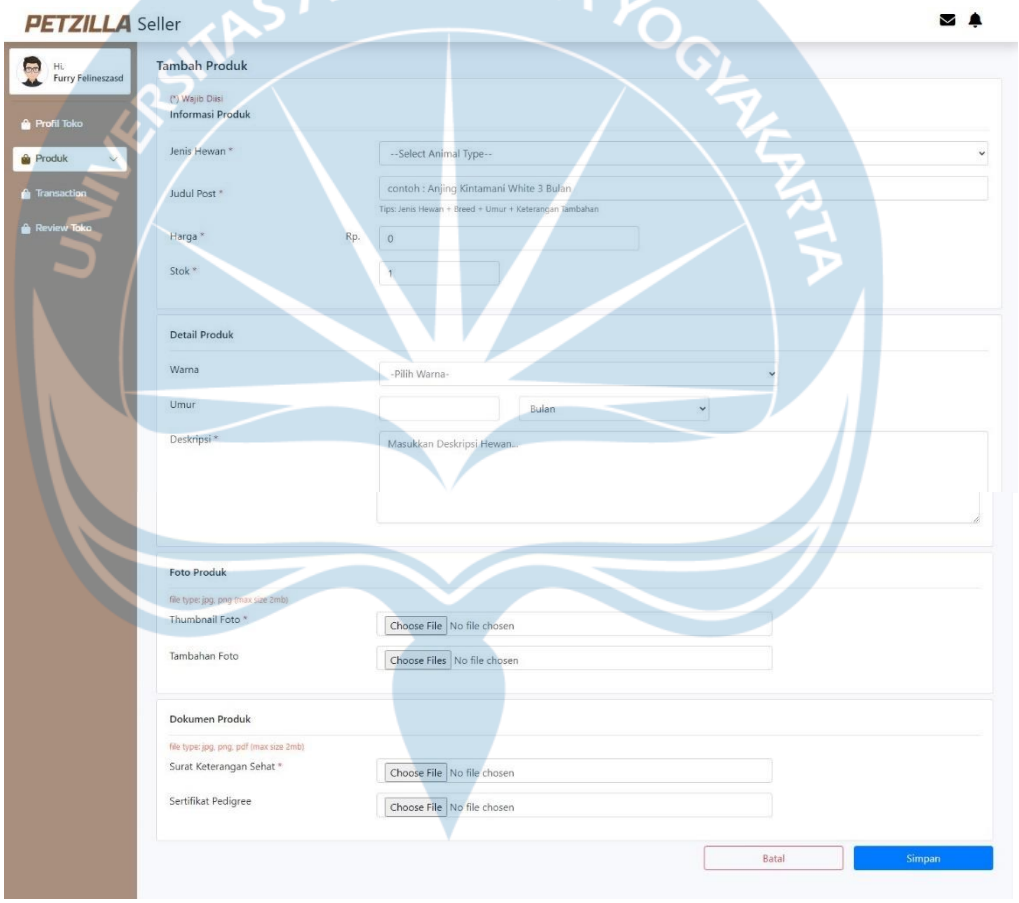
Gambar 5.52 merupakan implementasi antarmuka untuk daftar produk milik toko. Produk memiliki dua status yaitu aktif dan dalam *review*. Pada status dalam *review* berarti produk sedang ditinjau oleh admin.

```php
public function render()
{
    $animals = ListAnimal::whereHas('store', function ($query) {
        $query->where('user_id_user', Auth::id());
    })
    ->orderBy('status')
    ->orderBy('created_at', 'desc')
    ->paginate(8);

    return view('livewire.store.daftar-produk', [
        'animals' => $animals
    ])->layout('livewire.layouts.tes-layout', ['blueButton' => 'produk']);

}
```

Gambar 5.55 Potongan Kode Fungsi *Index* Daftar Produk Toko

Gambar 5.53 merupakan potongan kode untuk fungsi *index* daftar produk milik toko. Fungsi ini akan melakukan *query get* pada tabel *Animal* yang memiliki relasi dengan tabel *Store* yang terotentikasi saat ini. Setelah itu dilakukan *sorting* berdasarkan status dan tanggal pembuatan lalu di paginasi sejumlah delapan.

22) Implementasi Antarmuka Tambah/Edit Produk Toko



Gambar 5.56 Tambah/Edit Produk Toko

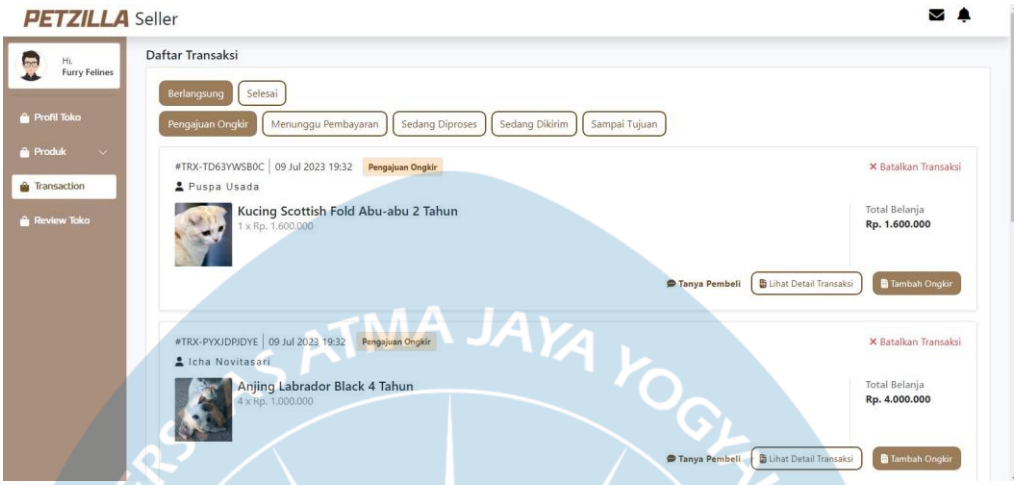
Gambar 5.54 merupakan implementasi antarmuka untuk tambah atau *edit* produk milik toko. Pada halaman ini, pengguna dapat menambahkan atau mengubah data produk sesuai dengan ketentuan.

```php
public function storeProduct()
{

    $this->validate([
        'jenis_hewan'   => 'required',
        'judul_post'    => 'required',
        'harga' => 'required|numeric|min:1',
        'stok' => 'required|integer|min:1',
        'deskripsi'     => 'required',
        'umur'          => 'nullable|numeric|min:1',
        'thumbnail'     => 'required',
        'surat_keterangan_sehat' => 'required'
    ]);

    $store = StoreModel::whereHas('user', function ($query) {
        $query->where('id_user', Auth::id());
    })->first();

    try {
        $animal = ListAnimal::create([

            'id_animal' => Str::random(10),
            'jenis_hewan' => $this->jenis_hewan,
            'judul_post'  => $this->judul_post,
            'deskripsi'   => $this->deskripsi,
            'harga'       => $this->harga,
            'stok'        => $this->stok,
            'warna'       => $this->warna,
            'umur'        => $this->umur ? $this->umur : null,
            'satuan_umur' => $this->umur ? $this->satuan_umur : null,
            'status'      => 'dalam_persetujuan',
            'store_id_store' => $store->id_store,
        ]);

        $animal->thumbnail = Storage::disk('public')->put($animal->id_animal, $this->thumbnail);
        $animal->surat_keterangan_sehat = Storage::disk('public')->put($animal->id_animal, $this->surat_keterangan_sehat);

        if ($this->sertifikat_pedigree) {
            $animal->sertifikat_pedigree = Storage::disk('public')->put($animal->id_animal, $this->sertifikat_pedigree);
        }
        if (!empty($this->photos)) {
            (new AnimalPhoto())->storePhoto($this->photos, $animal->id_animal);
        }

        if ($animal->save()) {
            $this->dispatchBrowserEvent('success-notification');
        }
    } catch (\Exception $e) {

        $this->dispatchBrowserEvent('error-modal');

    }

}
```

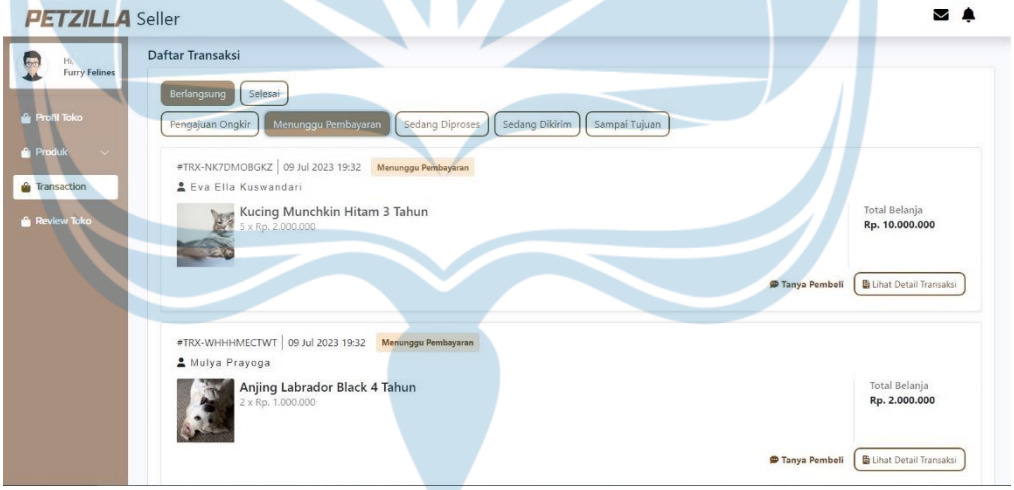
Gambar 5.57 Potongan Kode Fungsi Tambah Daftar Produk Toko

Gambar 5.55 merupakan potongan kode untuk fungsi tambah daftar produk milik toko. Pertama akan dilakukan validasi input *user* berdasarkan peraturan tertentu. Setelah itu akan melakukan *query create* pada tabel *Animal* dan menaruh foto *input* ke dalam folder *public*.

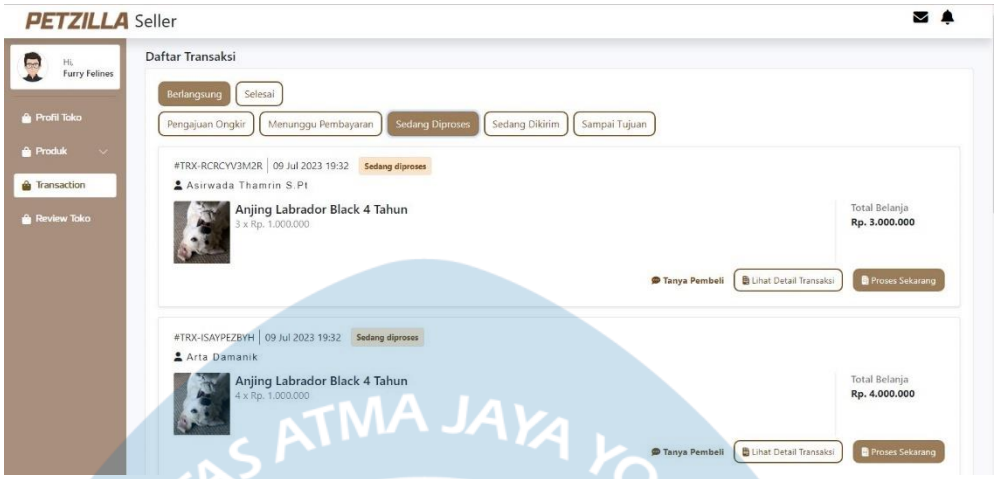
23) Implementasi Antarmuka Daftar Transaksi Berlangsung Toko



Gambar 5.58 Daftar Transaksi Berlangsung Dengan Status Pengajuan Ongkir

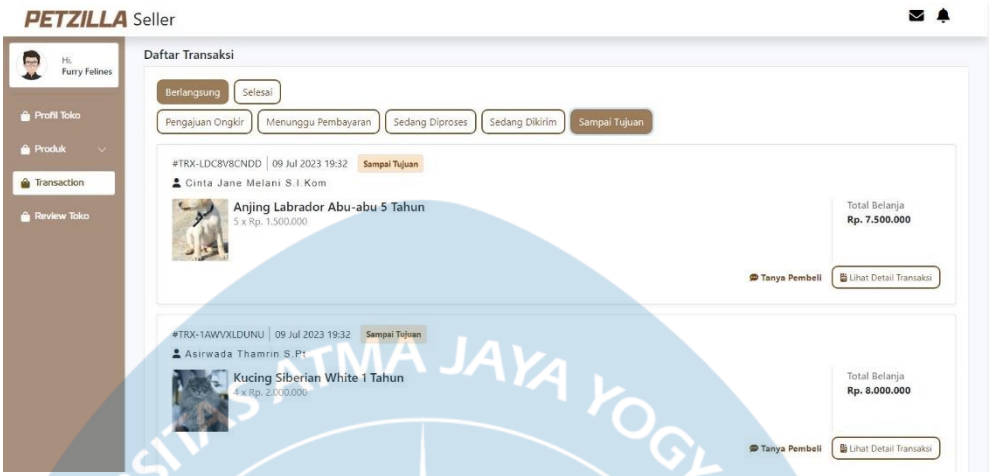


Gambar 5.59 Daftar Transaksi Berlangsung Dengan Status Menunggu Pembayaran



Gambar 5.60 Daftar Transaksi Berlangsung Dengan Status Sedang Diproses

Gambar 5. 61 Daftar Transaksi Berlangsung Dengan Status Sedang Dikirim



Gambar 5. 62 Daftar Transaksi Berlangsung Dengan Status Sampai Tujuan

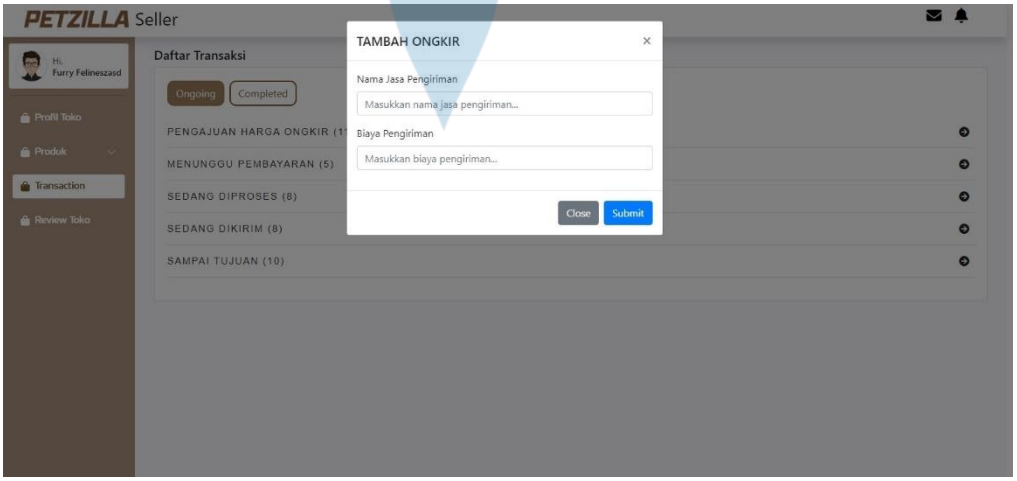
Gambar 5.56 sampai Gambar 5.60 merupakan rancangan antarmuka dari halaman transaksi yang sedang berlangsung milik toko. Pada halaman ini ada 5 bagian status yaitu, pengajuan harga ongkir, menunggu pembayaran, sedang diproses, sedang dikirim, dan sampai tujuan. Pada setiap status akan menampilkan produk yang sedang dalam proses transaksi. Toko dapat melakukan aksi pada berupa menambah jasa dan biaya pengiriman pada status pengajuan ongkir. Selain itu, toko dapat mengupload bukti pengiriman pada status sedang diproses.

```php
public function getTransactionDataStore($store)
{
    $transaction = Transaction::where('store_id_store', $store->id_store)
    ->with('user')
    ->with('animal')
    ->get()
    ->groupBy('status')
    ->map(function ($group) {
        return $group->map(function ($item) {
            $item->user->alamat = $item->user->getAddress($item->user->provinsi, $item->user->kabupaten, $item->user->kecamatan);
            return $item;
        });
    });

    return $transaction;
}
```

Gambar 5. 63 Potongan Kode Fungsi *Index* Daftar Transaksi Berlangsung Toko

Gambar 5.61 merupakan potongan kode untuk fungsi *index* daftar transaksi yang sedang berlangsung milik toko. Pada fungsi akan melakukan *query get* pada tabel Transaksi yang dimiliki oleh toko yang sedang terotentikasi saat ini. Pada *query* juga membawa *relationship* pada tabel *user* dan *animal*. Setelah itu dilakukan *groupBy* untuk mengelompokkan data berdasarkan status.

24) Implementasi Antarmuka Tambah Pengiriman *Modal*



Gambar 5. 64 Tambah Pengiriman *Modal*

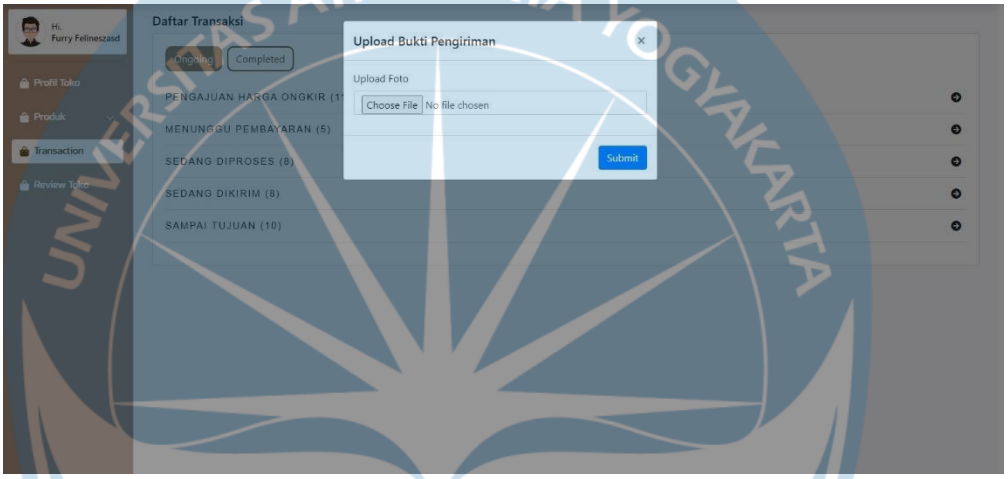
Gambar 5.62 merupakan implementasi antarmuka untuk tambah pengiriman *modal* milik toko. *Modal* ini digunakan untuk menambahkan jasa dan biaya pengiriman yang akan digunakan untuk mengirim hewan pada proses transaksi.

```php
public function submitOngkir()
{
    $this->validate([
        'jasa_pengiriman' => 'required',
        'biaya_pengiriman' => 'required'
    ]);

    try {
        Transaction::where('id_transaction', $this->selectedTransactionId)
            ->update([
                'status'            => 'menunggu_pembayaran',
                'grand_total'       => DB::raw('sub_total + ' . $this->biaya_pengiriman)
            ]);
        InformasiPengiriman::create([
            'biaya_pengiriman'   => $this->biaya_pengiriman,
            'jasa_pengiriman'    => $this->jasa_pengiriman,
            'transaction_id_transaction' => $this->selectedTransactionId
        ]);
        $this->dispatchBrowserEvent('success-notification');

    } catch (\Exception $e) {

        $this->dispatchBrowserEvent('error-modal');
    }
}
```

Gambar 5. 65 Potongan Kode Fungsi Tambah Pengiriman

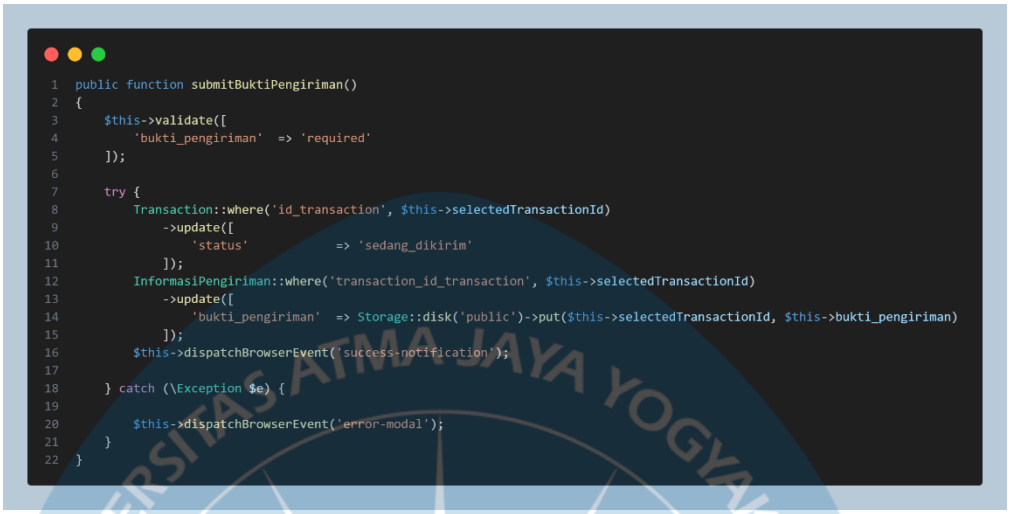
Gambar 5.63 merupakan potongan kode untuk fungsi tambah pengiriman milik toko. Pada fungsi, pertama dilakukan validasi terhadap *input* yang diterima dari *user*. Apabila valid, maka sistem akan melakukan *query update* pada tabel transaksi yang mengubah nilai status menjadi menunggu pembayaran dan *grand_total* yang merupakan jumlah dari *subtotal* dan biaya pengiriman. Setelah itu dilakukan juga *query create* pada tabel InformasiPengiriman.

25) Implementasi Antarmuka Tambah Bukti Pengiriman *Modal*



Gambar 5.66 Tambah Bukti Pengiriman *Modal*

Gambar 5.64 merupakan implementasi antarmuka untuk tambah bukti pengiriman *modal* milik toko. *Modal* ini digunakan untuk menambahkan bukti pengiriman setelah toko melakukan pengiriman pada ekspedisi.

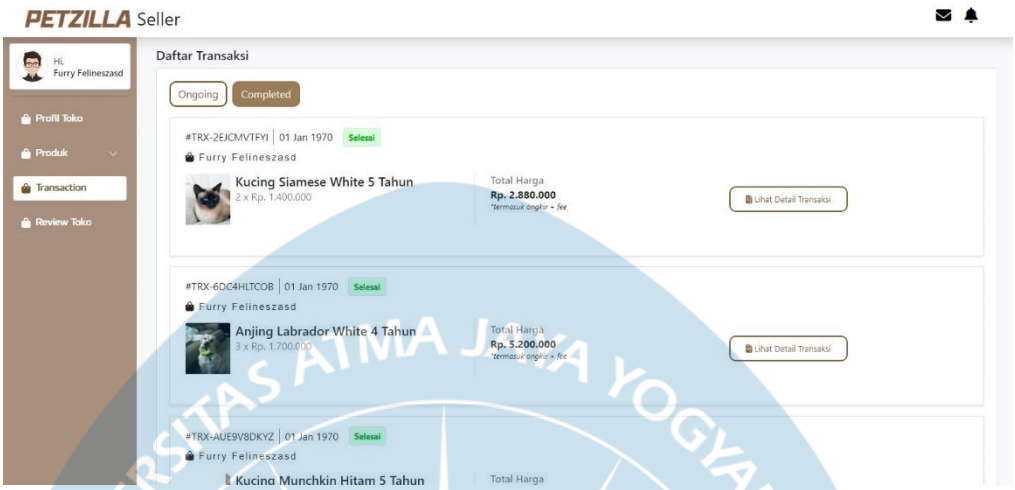


```php
public function submitBuktiPengiriman()
{
    $this->validate([
        'bukti_pengiriman'  => 'required'
    ]);

    try {
        Transaction::where('id_transaction', $this->selectedTransactionId)
            ->update([
                'status'            => 'sedang_dikirim'
            ]);
        InformasiPengiriman::where('transaction_id_transaction', $this->selectedTransactionId)
            ->update([
                'bukti_pengiriman'  => Storage::disk('public')->put($this->selectedTransactionId, $this->bukti_pengiriman)
            ]);
        $this->dispatchBrowserEvent('success-notification');

    } catch (\Exception $e) {

        $this->dispatchBrowserEvent('error-modal');
    }
}
```

Gambar 5.67 Potongan Kode Fungsi Tambah Bukti Pengiriman

Gambar 5.65 merupakan potongan kode untuk fungsi tambah bukti pengiriman yang dilakukan oleh toko. Pada fungsi, pertama dilakukan validasi terhadap *input* yang diterima dari *user*. Apabila valid, maka sistem akan melakukan *query update* pada tabel transaksi yang mengubah nilai status menjadi sedang dikirim. Setelah itu dilakukan juga *query update* pada tabel InformasiPengiriman yang mengubah nilai bukti_pengiriman menjadi *path* dimana gambar diletakkan. Jadi struktur folder akan terlihat seperti ini, "*public/id_trx /nama_file*".

26) Implementasi Antarmuka Daftar Transaksi Selesai Toko



Gambar 5.68 Daftar Transaksi Selesai Toko

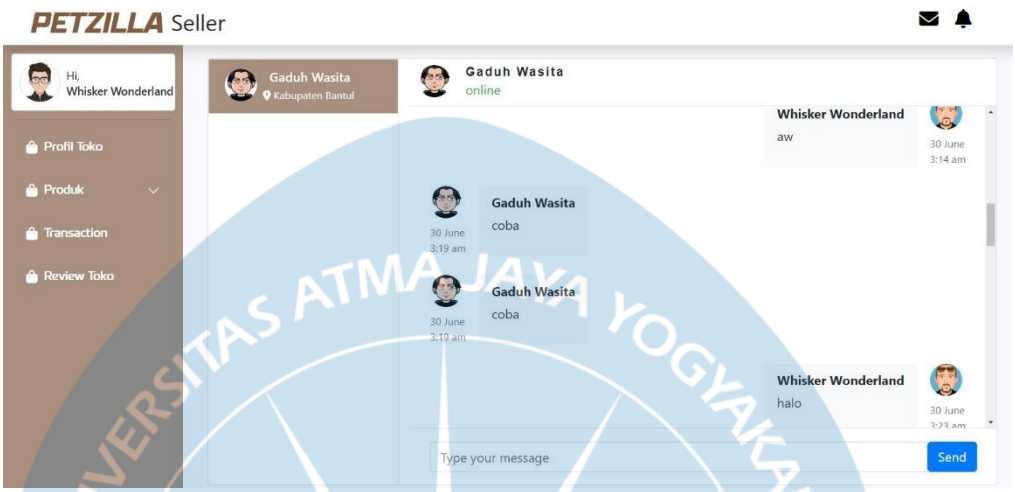
Gambar 5.66 merupakan implementasi antarmuka daftar transaksi selesai milik toko. Pada halaman ini menampilkan riwayat transaksi yang berhasil diselesaikan oleh toko. Terdapat beberapa informasi seperti, informasi hewan, tanggal, informasi penjual, dan total harga.

```php
public function render()
{
    $store = User::find(Auth::id())->store;
    $completedTransaction = Transaction::where('store_id_store', $store->id_store)
                                        ->where('status', 'selesai')
                                        ->with('user')
                                        ->with('animal')
                                        ->with('pengiriman')
                                        ->paginate(8);

    return view('livewire.store.ongoing-transaction',['completedTransaction' => $completedTransaction]);
}
```

Gambar 5.69 Potongan Kode Fungsi *Index* Daftar Transaksi Selesai Toko

Gambar 5.67 merupakan potongan kode untuk fungsi *index* daftar transaksi yang sudah selesai milik toko. Sistem akan melakukan *query get* pada tabel transaksi milik toko yang memiliki status selesai beserta dengan relasi-relasinya, lalu akan dilakukan paginasi halaman sejumlah delapan.

27) Implementasi Antarmuka *Inbox* Toko



Gambar 5.70 *Inbox* Toko

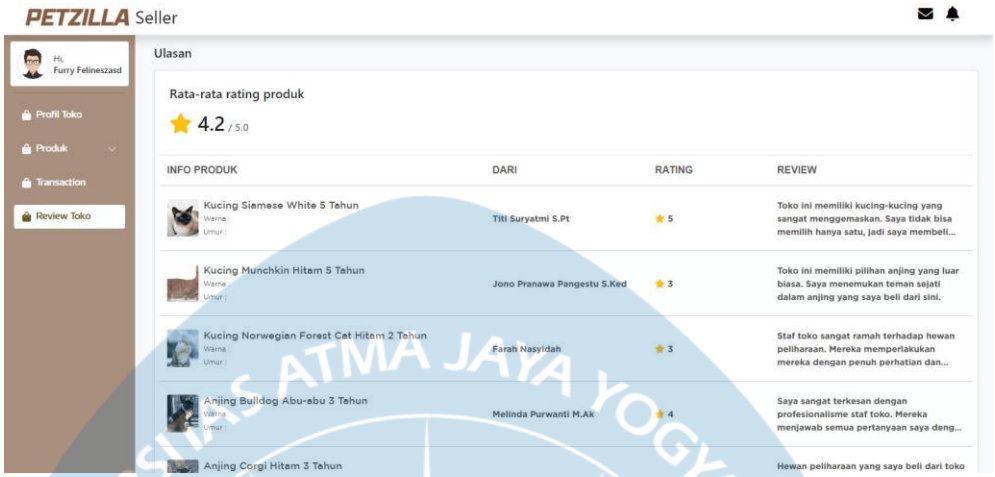
Gambar 5.68 merupakan implementasi antarmuka *inbox* toko. Pada halaman ini menampilkan daftar pesan yang dimiliki oleh toko. Selain itu, pada halaman ini toko juga bisa mengirim pesan.

```php
public function sendMessage()
{
    $this->validate([
        'message' => 'required|min:1',
    ]);

    $store = StoreModel::where('user_id_user', Auth::id())->first();

    try {
        $message = Chat::create([
            'users_id_user' => $this->to_id,
            'store_id_store' => $store->id_store,
            'message' => $this->message,
            'sender_type' => $this->who_is_this,
        ]);

        array_push($this->messages, $message);

        broadcast(new MessageSent($this->to_id, $store->id_store, $this->message, $this->who_is_this));

        $this->message = '';
    } catch (\Exception $e) {
        $this->dispatchBrowserEvent('error-modal');
    }
}
```

Gambar 5. 71 Potongan Kode Fungsi Kirim Pesan Toko

Gambar 5.69 merupakan potongan kode untuk fungsi untuk kirim pesan milik toko. Pertama dilakukan validasi terhadap input pesan yang akan dikirim. Selanjutnya, sistem akan melakukan *query create* pada table *Chat* yang lalu akan melakukan *broadcast* yang dikirim pada server *websocket* bahwa terdapat pesan baru.

28) Implementasi Antarmuka Daftar *Rating* Toko



Gambar 5. 72 Daftar *Rating* Toko

Gambar 5.70 merupakan implementasi antarmuka daftar *rating* toko. Pada halaman ini menampilkan daftar *rating* yang berikan *user* pada toko.

```php
public function render()
{
    $rating = Rating::with('transaction.store', 'transaction.user','transaction.animal')
        ->whereHas('transaction', function ($query) {
            $query->where('store_id_store', $this->currentStore->id_store);
        })
        ->orderBy('created_at','asc')
        ->paginate(10);

    return view('livewire.store.daftar-review', [
        'rating' => $rating
        ])->layout('livewire.layouts.tes-layout');
}
```

Gambar 5. 73 Potongan Kode Fungsi *Index Rating* Toko

Gambar 5.71 merupakan potongan kode untuk fungsi *index* untuk halaman *rating* milik toko. Pada fungsi ini melakukan *query get* pada tabel *Rating* yang membawa relasinya yaitu *store, user* dan *transaction.* Setelah itu mengambil data yang dimiliki oleh toko yang sedang terotentikasi saat ini lalu diurutkan berdasarkan paling baru.

29) Implementasi Antarmuka Login Admin

Gambar 5. 74 Login Admin

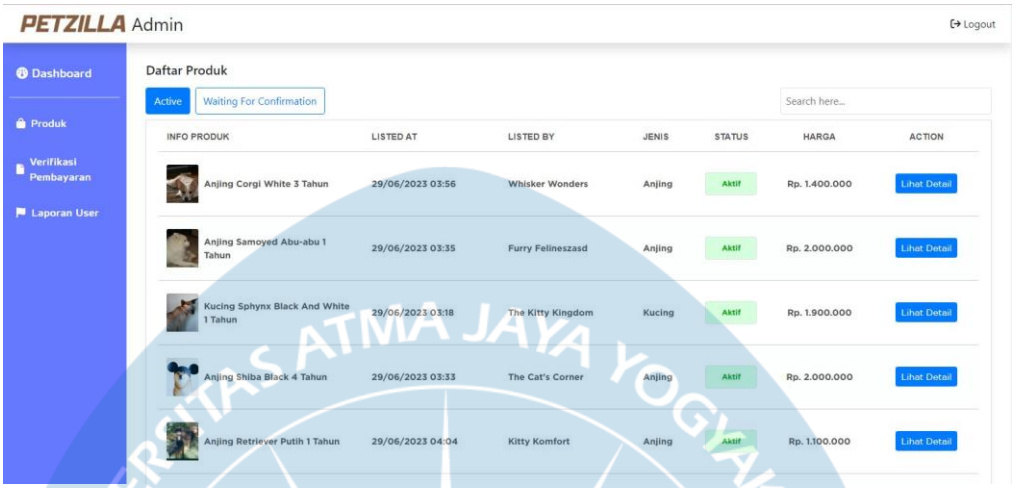
Gambar 5.72 merupakan antarmuka dari halaman login admin. Pengguna diminta untuk melakukan *input* berupa *email* dan *password*.

```php
public function login()
{
    $data = $this->validate([
        'email' => 'required',
        'password' => 'required'
    ]);

    if (Auth::guard('admin')->attempt($data)) {

        return redirect()->to('/admin/dashboard');

    }
    return session()->flash('error', 'Incorrect credentials !');

}
```

Gambar 5. 75 Potongan Kode Fungsi *Login* Admin

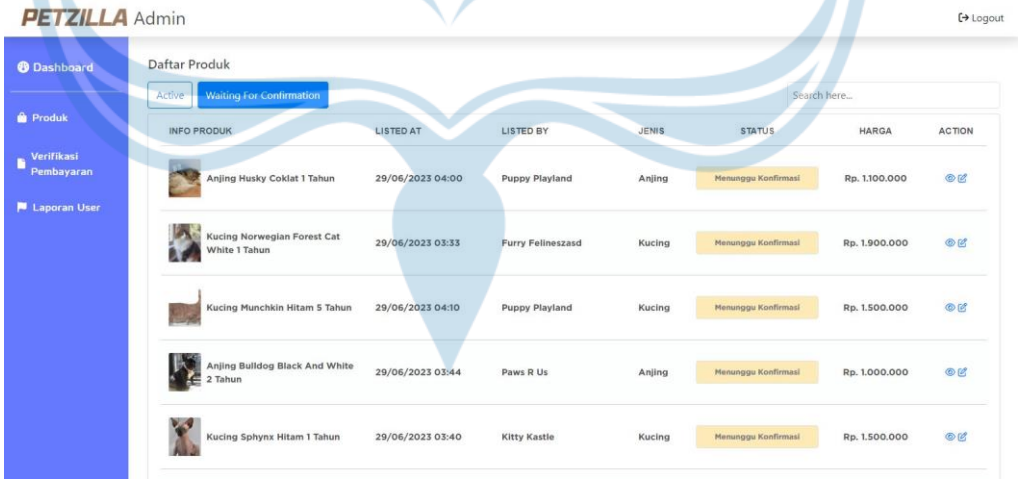
Gambar 5.73 merupakan potongan kode untuk fungsi *login* admin. Pada fungsi Pertama dilakukan validasi terhadap *input user*, setelah itu metode *Auth::guard('admin')* merupakan mekanisme yang digunakan Laravel untuk mengelola otentikasi pengguna.

30) Implementasi Antarmuka Daftar Produk Admin



Gambar 5.76 Daftar Produk Aktif Admin

Gambar 5.74 merupakan antarmuka untuk halaman daftar produk aktif. Pada halaman ini akan menampilkan seluruh daftar hewan yang statusnya aktif. Selain itu, ada juga fitur *search* untuk mempermudah pencarian hewan.



Gambar 5.77 Daftar Produk Menunggu Konfirmasi Admin

Gambar 5.75 merupakan antarmuka untuk halaman daftar produk yang menunggu persetujuan admin. Pada halaman ini terdapat dua *button* yaitu untuk melihat detail hewan dan meng-*update* status.

```php
public function render()
{
    if (!empty($this->searchTerm)) {

        $animal = ListAnimal::with('store')
                            ->where('judul_post', 'like', "%" . $this->searchTerm . "%")
                            ->where('status', $this->type)
                            ->paginate(10);

    } else {
        $animal = ListAnimal::where('status', $this->type)->with('store')->paginate(10);
    }
    return view('livewire.admin.product.product', ['animals' => $animal])->layout('livewire.layouts.admin-layout');
}
```

Gambar 5.78 Potongan Kode Fungsi *Index* Daftar Produk Admin

Gambar 5.76 merupakan potongan kode untuk fungsi *index* daftar produk milik admin. Pada fungsi, pertama dilakukan pengecekan jika *query string* pada variabel *searchTerm* kosong. Jika tidak kosong, maka akan dilakukan *query get* pada tabel *Animal* dimana nama hewan mengandung *string* pada variabel *searchTerm*.

```php
public function updateStatusAnimal($id)
{
    $animal = ListAnimal::find($id);

    if (!$animal) {
        $this->dispatchBrowserEvent('error-modal');
    } else {
        $animal->status = 'aktif';
        $animal->save();

        $this->dispatchBrowserEvent('success-modal');

    }
}
```

Gambar 5. 79 Potongan Kode Fungsi *Update* Status Produk

Gambar 5.77 merupakan potongan kode untuk fungsi *update* status produk yang dilakukan oleh admin. Pertama dilakukan pencarian hewan berdasarkan data parameter berupa id. Jika ada, maka akan melakukan *update* status pada hewan menjadi aktif.

31) Implementasi Antarmuka Detail Produk Admin

Gambar 5.80 Detail Produk Admin

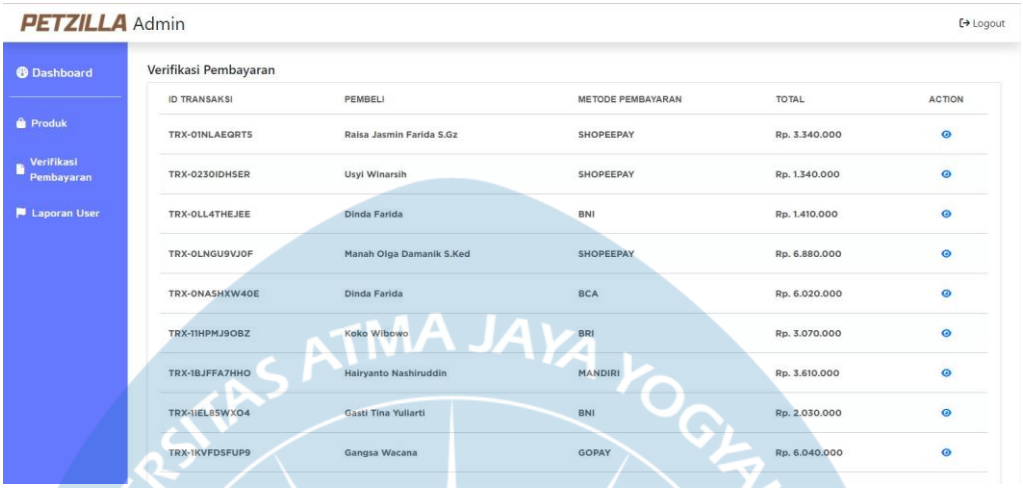
Gambar 5.78 merupakan antarmuka untuk halaman detail produk milik admin. Pada halaman ini akan menampilkan informasi dari produk secara detail mulai dari nama, foto, dokumen dan lain-lain.

```php
public function mount($id_animal)
{
    $animal = ListAnimal::where('id_animal', $id_animal)
        ->with('store')
        ->with('animal_photo')
        ->first();

    if(empty($animal))
    {
        return redirect()->to('/admin/error/not-found');
    }

    $this->jenis_hewan = $animal->jenis_hewan;
    $this->judul_post = $animal->judul_post;
    $this->harga = $animal->harga;
    $this->stok = $animal->stok;
    $this->warna = $animal->warna;
    $this->umur = $animal->umur;
    $this->satuan_umur = $animal->satuan_umur;
    $this->deskripsi = $animal->deskripsi;
    $this->display_photo = $animal->thumbnail;
    $this->display_photos = $animal->animal_photo;
    $this->surat_keterangan_sehat = $animal->surat_keterangan_sehat;
    if ($animal->sertifikat_pedigree) {
        $this->sertifikat_pedigree = $animal->sertifikat_pedigree;
    }
}
```

Gambar 5. 81 Potongan Kode Fungsi Detail Produk Admin

Gambar 5.79 merupakan potongan kode untuk fungsi detail produk yang dimiliki oleh admin. Fungsi dimulai dari mencari hewan dengan id yang sesuai dengan id parameter, jika tidak ada maka akan diarahkan ke halaman *error*. Jika ada maka fungsi akan melakukan *assign* ke variabel yang sesuai pada properti di *front-end.*

32) Implementasi Antarmuka Daftar Pembayaran Admin



Gambar 5. 82 Daftar Pembayaran Admin

Gambar 5.80 merupakan implementasi antarmuka untuk daftar pembayaran milik admin. Pada halaman ini akan daftar pembayaran berasal dari *user* yang harus diverifikasi oleh admin.

```php
public function render()
{
    $transaction = Transaction::where('status', 'review_pembayaran')
        ->with('pembayaran')
        ->with('user')
        ->paginate(10);

    return view('livewire.admin.verifikasi-pembayaran',
                ['transactions' => $transaction])
                ->layout('livewire.layouts.admin-layout');
}
```

Gambar 5. 83 Potongan Kode Fungsi *Index* Verifikasi Pembayaran

Gambar 5.81 merupakan potongan kode untuk fungsi *index* pada halaman verifikasi pembayaran milik admin. Pada fungsi akan melakukan *query get* pada tabel Transaksi dimana statusnya ‘review_pembayaran’.

33) Implementasi Antarmuka Detail Pembayaran Admin

Gambar 5. 84 Detail Pembayaran *Modal*

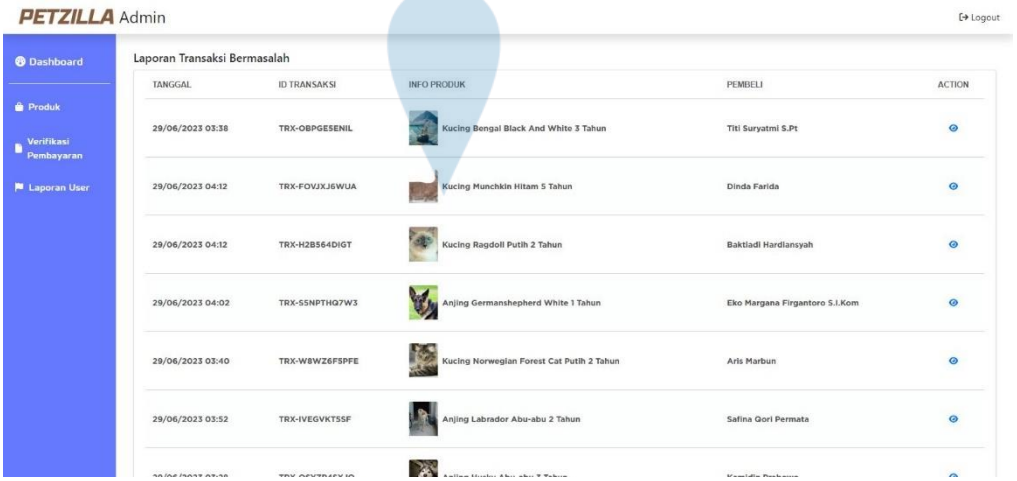
Gambar 5.82 merupakan implementasi antarmuka untuk detail pembayaran milik admin. *Modal* ini akan menampilkan detail pembayaran yang dilakukan oleh *user* secara detail. Jika pembayaran sesuai maka admin dapat menekan pada “Setujui Pembayaran”, apabila tidak sesuai “Tolak Pembayaran”.

```php
public function setujuiPembayaran($id)
 {
     try {
         $transaction = Transaction::find($id);
         $transaction->status = 'sedang_diproses';
         $transaction->save();
     } catch (\Exception $e) {
         $this->dispatchBrowserEvent('error-modal');
     }
 }
```

Gambar 5. 85 Potongan Kode Fungsi Setujui Pembayaran

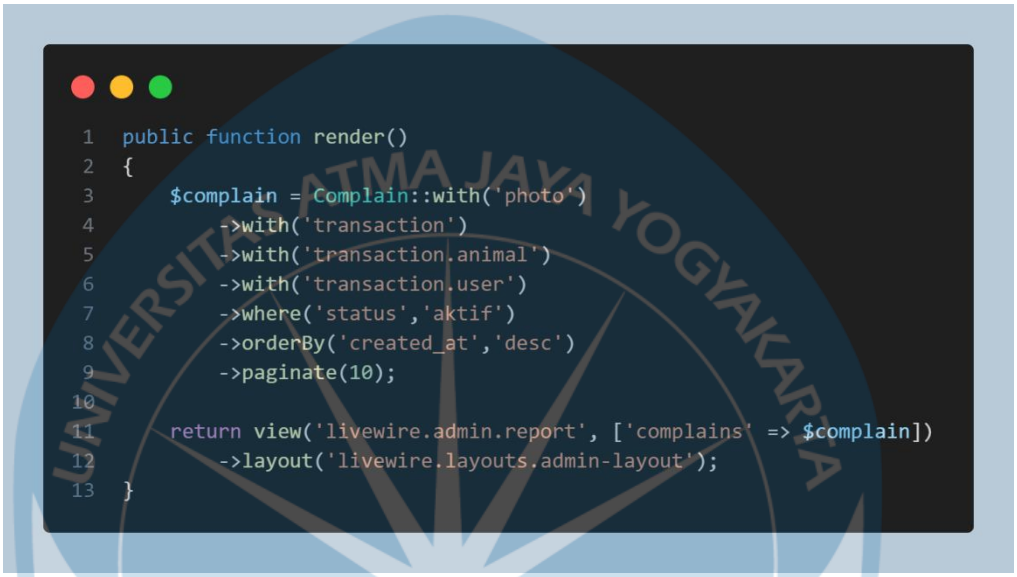
Gambar 5.83 merupakan potongan kode untuk fungsi setujui pembayaran yang dilakukan oleh admin. Pada fungsi akan mencari data transaksi yang sesuai dengan parameter id, setelah itu akan melakukan *update* pada status menjadi ‘sedang_diproses’.

34) Implementasi Antarmuka Laporan Transaksi Bermasalah



Gambar 5. 86 Laporan Transaksi Bermasalah

Gambar 5.84 merupakan implementasi antarmuka untuk halaman laporan transaksi bermasalah milik admin. Pada halaman ini berisi daftar laporan transaksi yang bermasalah berasal dari *user* yang harus ditinjau oleh admin.

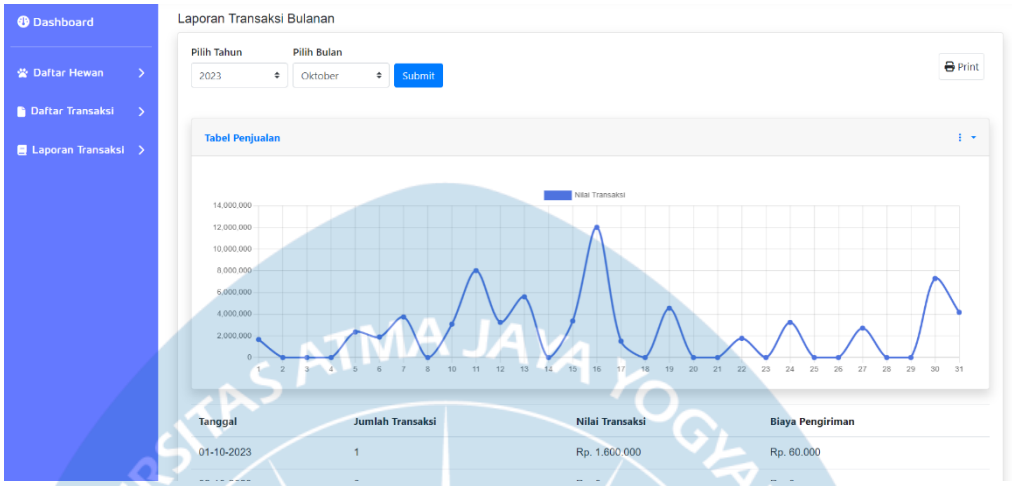


```php
public function render()
{
    $complain = Complain::with('photo')
        ->with('transaction')
        ->with('transaction.animal')
        ->with('transaction.user')
        ->where('status','aktif')
        ->orderBy('created_at','desc')
        ->paginate(10);

    return view('livewire.admin.report', ['complains' => $complain])
        ->layout('livewire.layouts.admin-layout');
}
```

Gambar 5. 87 Potongan Kode Fungsi *Index* Laporan Transaksi Bermasalah

Gambar 5.85 merupakan potongan kode untuk fungsi *index* pada halaman laporan transaksi bermasalah. Pada fungsi akan melakukan *query get* pada tabel *Complain* yang statusnya aktif, lalu diurutkan secara *descending* dan dipaginasi sejumlah 10.

35) Implementasi Antarmuka Laporan Transaksi Bulanan



Gambar 5. 88 Laporan Transaksi Bulanan

Gambar 5.88 merupakan implementasi antarmuka untuk halaman laporan transaksi bulanan. Aplikasi akan meminta *input* bulan dan tahun, sedangkan untuk laporan tahunan aplikasi akan meminta *input* tahun. Pada halaman ini juga akan menampilkan tabel grafik.

B. Pengujian Fungsionalitas Perangkat Lunak

Tabel 5. 1 Tabel Hasil Pengujian

No	Deksripsi	Flow	Input	Expected Output	Actual Output	Kesimpulan
1.	Fungsi Registrasi Pengguna	1) Masukkan nama. 2) Masukkan *email.* 3) Masukkan *password.* 4) Masukkan no hp. 5) Pilih provinsi. 6) Pilih kabupaten. 7) Pilih kecamatan. 8) Masukkan alamat lengkap 9) Tekan tombol "*Submit*"	-nama : "Alvin Sugijanto" -email : "alvinsugijanto01@gmail.com" -password : "12345678" -no hp : "0895321091566" -provinsi : DI Yogyakarta -kabupaten : Kabupaten Sleman -kecamatan : Godean -alamat : Perumahan Godean	Menampilkan notifikasi sukses dan mengirim tautan kepada *email* pengguna untuk melakukan verifikasi.	Menampilkan notifikasi sukses dan mengirim tautan kepada *email* pengguna untuk melakukan verifikasi.	Sesuai

2.	Fungsi Login Pengguna	1) Masukkan *email.* 2) Masukkan *password.*	-email : "alvinsugijanto01@gmail.com" -password : "12345678"	Diarahkan ke halaman *marketplace*	Diarahkan ke halaman *marketplace*	Sesuai
3.	Fungsi *Index* Halaman *Marketplace*	1) Tekan logo aplikasi PetZilla	Tidak ada	Menampilkan daftar hewan yang tersedia dalam radius <50km dari alamat pengguna saat ini.	Menampilkan daftar hewan yang tersedia dalam radius <50km dari alamat pengguna saat ini.	Sesuai
4.	Fungsi Cari Tempat Tinggal	1) Kunjungi halaman *marketplace* tanpa *login* atau sebagai *guest.* 2) Cari tempat tinggal atau tekan tombol pilih lokasi saya saat ini	-tempat_tinggal : "Kabupaten Banyumas Kecamatan Purwokerto Selatan"	Lokasi berhasil ditemukan, lalu menampilkan daftar hewan disekitar.	Lokasi berhasil ditemukan, lalu menampilkan daftar hewan disekitar.	Sesuai

5.	Fungsi *Index* Detail Hewan	1) Berada pada halaman *marketplace* 2) Pilih salah satu hewan	Tidak ada	Menampilkan informasi hewan secara detail mulai dari gambar, nama, deskripsi, kelengkapan dokumen dan stok.	Menampilkan informasi hewan secara detail mulai dari gambar, nama, deskripsi, kelengkapan dokumen dan stok.	Sesuai
6.	Fungsi Pengajuan Harga Ongkir	1) Berada pada halaman detail hewan. 2) Tekan tombol "*Buy Now*". 3) Tekan tombol "Ajukan Biaya Pengiriman".	Tidak ada	Berhasil membuat transaksi dan mengarahkan ke halaman transaksi.	Berhasil membuat transaksi dan mengarahkan ke halaman transaksi.	Sesuai
7.	Fungsi *Index* Halaman Toko Bagian Produk	1) Berada pada halaman detail hewan. 2) Pilih bagian foto profil atau nama toko 3) Pilih tombol "Produk"	Tidak ada	Menampilkan daftar hewan milik toko yang siap untuk dijual.	Menampilkan daftar hewan milik toko yang siap untuk dijual.	Sesuai

8.	Fungsi *Index* Halaman Toko Bagian Ulasan	1) Berada pada halaman detail hewan. 2) Pilih bagian foto profil atau nama toko 3) Pilih tombol "Ulasan"	Tidak ada	Menampilkan daftar ulasan toko yang dilakukan *user* dan diurutkan dari yang terbaru.	Menampilkan daftar ulasan toko yang dilakukan *user* dan diurutkan dari yang terbaru.	Sesuai
9.	Fungsi *Index* Profil *User*	1) Pilih bagian nama pada *navbar*. 2) Pilih "profil saya"	Tidak ada	Menampilkan informasi *user* berupa nama, alamat, no hp dan alamat.	Menampilkan informasi *user* berupa nama, alamat, no hp dan alamat.	Sesuai
10	Fungsi *Edit Personal Information User*	1) Pilih bagian nama pada *navbar*. 2) Pilih "profil saya". 3) Tekan logo *edit* pada bagian "*personal information*".	-nama : "Bernadus Alvin" -no hp : "0895321311"	Data pribadi *user* berhasil diubah dan menampilkan notifikasi berhasil.	Data pribadi *user* berhasil diubah dan menampilkan notifikasi berhasil.	Sesuai

11	Fungsi *Edit Address User*	1) Pilih bagian nama pada *navbar*. 2) Pilih "profil saya". 3) Tekan logo *edit* pada bagian "*address*".	-provinsi : "Banten" -kabupaten : "Kota Tangerang" -kecamatan : "Pinang"	Data alamat *user* berhasil diubah dan menampilkan notifikasi berhasil.	Data alamat *user* berhasil diubah dan menampilkan notifikasi berhasil.	Sesuai
12	Fungsi *Index* Daftar Transaksi Berlangsung *User*	1) Pilih bagian nama pada *navbar*. 2) Pilih "profil saya" 3) Pilih "daftar transaksi" 4) Pilih tombol status "Berlangsung"	Tidak ada	Setiap status dapat menampilkan daftar transaksi dengan benar	Setiap status dapat menampilkan daftar transaksi dengan benar	Sesuai

13	Fungsi Batal Transaksi	1) Berada pada halaman daftar transaksi *user* 2) Pilih tombol status "Berlangsung" 3) Pilih status transaksi "Pengajuan ongkir" atau "Menunggu Pembayaran"	Tidak ada	Transaksi berhasil dibatalkan dan menampilkan notifikasi.	Transaksi berhasil dibatalkan dan menampilkan notifikasi.	Sesuai
14	Fungsi Pembayaran Transaksi	1) Berada pada halaman daftar transaksi *user* 2) Pilih tombol status "Berlangsung" 3) Pilih tombol status "Menunggu Pembayaran"	-tipe rekening : "Transfer Bank" -jenis rekening : "BCA" -nama rekening : "Alvin Sugijanto" -nomor rekening : "3748168723" -bukti pembayaran : foto	Berhasil mengupload form bukti pembayaran dan menampilkan pesan berhasil.	Berhasil mengupload form bukti pembayaran dan menampilkan pesan berhasil.	Sesuai

15	Fungsi Konfirmasi Kedatangan Hewan	1) Berada pada halaman daftar transaksi *user*. 2) Pilih tombol status "Berlangsung" 3) Pilih tombol status "Sedang Dikirim" 4) Pilih tombol "Hewan Sudah Datang !"	Tidak ada	Berhasil melakukan *update* status pada transaksi yang dipilih menjadi "Sampai Tujuan".	Berhasil melakukan *update* status pada transaksi yang dipilih menjadi "Sampai Tujuan".	Sesuai
16	Fungsi Lapor Produk Bermasalah	1) Berada pada halaman transaksi *user*. 2) Pilih tombol status "Berlangsung" 3) Pilih tombol status "Sampai Tujuan" 4) Pilih tombol "Laporkan Produk"	-keluhan : "Hewan bermasalah" -foto bukti : foto	Berhasil membuat laporan produk, setelah itu sistem menampilkan pesan berhasil.	Berhasil membuat laporan produk, setelah itu sistem menampilkan pesan berhasil.	Sesuai

17	Fungsi Rating Produk	1) Berada pada halaman transaksi *user*. 2) Pilih tombol status "Berlangsung" 3) Pilih tombol status "Sampai Tujuan" 4) Pilih tombol "Beri Ulasan"	-rating : 5 -review : "Kucing datang dalam kondisi baik"	Berhasil membuat rating produk, setelah itu sistem menampilkan pesan berhasil.	Berhasil membuat rating produk, setelah itu sistem menampilkan pesan berhasil.	Sesuai
18	Daftar Transaksi Selesai *User*	1) Berada pada halaman transaksi *user*. 2) Pilih status "Berhasil"	Tidak ada	Menampilkan daftar transaksi yang berhasil diselesaikan milik *user*.	Menampilkan daftar transaksi yang berhasil diselesaikan milik *user*	Sesuai
19	Detail Transaksi Selesai *User*	1) Berada pada halaman transaksi *user*. 2) Pilih status "Berhasil" 3) Pilih tombol "Lihat Detail Transaksi"	Tidak ada	Menampilkan detail informasi dari transaksi yang dipilih *user*.	Menampilkan detail informasi dari transaksi yang dipilih *user*.	Sesuai

20	Inbox *User*	1) Pilih logo pesan pada *navbar*.	Tidak ada	Menampilkan daftar pesan yang dimiliki *user*.	Menampilkan daftar pesan yang dimiliki *user*.	Sesuai
21	Fungsi Kirim Pesan	1) Pilih logo pesan pada *navbar*. 2) Pilih salah satu toko.	-chat : "Halo"	Berhasil mengirim pesan secara *real-time* kepada toko yang dituju.	Berhasil mengirim pesan secara *real-time* kepada toko yang dituju.	Sesuai
22	*Wishlist User*	1) Pilih logo *bookmark* pada *navbar*.	Tidak ada	Menampilkan daftar *wishlist* yang dimiliki *user*.	Menampilkan daftar *wishlist* yang dimiliki *user*	Sesuai
23	Registrasi Toko	1) Sudah login sebagai *user* dan belum memiliki toko. 2) Pilih bagian "toko saya" yang berada di *navbar*. 3) Pilih "daftar toko"	-nama toko : "PetZilla" -no hp : "0895321091566" -deskripsi toko : "Toko hebat" -provinsi : DI Yogyakarta -kabupaten : Kabupaten Sleman -kecamatan : Godean	Berhasil membuat akun toko dan menampilkan pesan sukses.	Berhasil membuat akun toko dan menampilkan pesan sukses.	Sesuai

			-alamat : Perumahan Godean -tipe rekening : “Transfer Bank” -jenis rekening : “BNI” -nama rekening : “Bernadus Alvin” -nomor rekening : “38957912”			
24	Profil Toko	1) Pilih bagian “Toko Saya” pada *navbar* 2) Pilih menu “Profil Toko”	Tidak ada	Menampilkan informasi toko berupa nama toko, alamat, no hp dan informasi rekening.	Menampilkan informasi toko berupa nama toko , alamat, no hp dan informasi rekening.	Sesuai
25	Daftar Produk Toko	1) Pilih bagian “Toko Saya” pada *navbar*. 2) Pilih menu “Produk”, lalu “Daftar Produk”.	Tidak ada	Menampilkan semua daftar hewan milik toko dengan status aktif atau dalam review.	Menampilkan semua daftar hewan milik toko dengan status aktif atau dalam review.	Sesuai

26	Fungsi Tambah/Edit Produk Toko	1) Pilih bagian “Toko Saya” pada *navbar*. 2) Pilih menu “Produk”, lalu “Tambah Produk”.	-jenis hewan : “Anjing” -judul post : “Anjing Kintamani White 3 Bulan” -harga : Rp. 600.000 -stok : 2 -warna : “putih” -umur : “3 Bulan” -deskripsi : “Anjing kintamani lucu baru 3 bulan.” -thumbnail: foto utama -surat keterangan sehat : *file*	Berhasil menambah hewan dalam status “dalam review”.	Berhasil menambah hewan dalam status “dalam review”.	Sesuai
27	Daftar Transaksi Berlangsung Toko	1) Pilih bagian “Toko Saya” pada *navbar*. 2) Pilih menu “Transaksi”	Tidak ada	Setiap status dapat menampilkan daftar transaksi dengan benar	Setiap status dapat menampilkan daftar transaksi dengan benar	Sesuai

28	Fungsi Tambah Pengiriman	1) Berada pada halaman toko 2) Pilih menu "Transaksi" 3) Pilih status transaksi "Berlangsung" 4) Pilih status transaksi "Pengajuan Ongkir"	-nama jasa pengiriman : "SiCepat" -biaya pengiriman : Rp. 35.000	Berhasil menambahkan informasi pengiriman dan melakukan *update* status transaksi menjadi "menunggu pembayaran".	Berhasil menambahkan informasi pengiriman dan melakukan *update* status transaksi menjadi "menunggu pembayaran".	Sesuai
29	Fungsi Upload Bukti Pengiriman	1) Berada pada halaman toko 2) Pilih menu "Transaksi" 3) Pilih status transaksi "Berlangsung" Pilih status transaksi "Sedang Diproses" 4) Pilih tombol "Proses Sekarang"	-bukti pengiriman : foto	Berhasil menambahkan bukti pengiriman dan melakukan *update* status transaksi menjadi "sedang dikirim".	Berhasil menambahkan bukti pengiriman dan melakukan *update* status transaksi menjadi "sedang dikirim".	Sesuai

30	Daftar Transaksi Selesai Toko	1) Berada pada halaman toko. 2) Pilih menu "Transaksi". 3) Pilih status transaksi "selesai".	Tidak ada	Menampilkan daftar transaksi yang berhasil diselesaikan oleh toko.	Menampilkan daftar transaksi yang berhasil diselesaikan oleh toko.	Sesuai
31	Inbox Toko	1) Berada pada halaman toko. 2) Pilih logo pesan pada *navbar*.	Tidak ada	Menampilkan daftar pesan dengan *user* yang dimiliki oleh toko.	Menampilkan daftar pesan dengan *user* yang dimiliki oleh toko.	Sesuai
32	Daftar Review Toko	1) Berada pada halaman toko. 2) Pilih menu "Review Toko"	Tidak ada	Menampilkan daftar review milik toko yang berasal dari *user*.	Menampilkan daftar review milik toko yang berasal dari *user*.	Sesuai

33	Fungsi Tambah Hewan ke Keranjang	1) Berada pada halaman detail produk. 2) Tekan tombol "Tambah ke Keranjang"	Tidak ada	Menampilkan notifikasi berhasil dan hewan berada pada keranjang.	Menampilkan notifikasi berhasil dan hewan berada pada keranjang.	Sesuai
33	*Login* Admin	1) Pergi ke halaman admin 2) Masukkan *email.* 3) Masukkan *password.*	-email : "6wLeUCQUY8@gmail.com" -password : "admin123"	Berhasil masuk kedalam sistem admin.	Berhasil masuk kedalam sistem admin.	Sesuai
34	Daftar Produk	1) Pilih menu "produk"	Tidak ada	Berhasil menampilkan daftar produk sesuai dengan statusnya.	Berhasil menampilkan daftar produk sesuai dengan statusnya.	Sesuai
35	Detail Produk	1) Pilih menu "produk" 2) Pilih tombol "Lihat Detail"	Tidak ada	Menampilkan detail informasi dari produk hewan yang dipilih.	Menampilkan detail informasi dari produk hewan yang dipilih.	Sesuai

36	Fungsi konfirmasi pencantuman produk.	1) Pilih menu “produk” 2) Pilih status “*Waiting For Confirmation*”. 3) Pilih logo *edit* pada satu produk. 4) Pilih tombol “Ya, Saya akan update”.	Tidak ada	Berhasil melakukan *update* status produk menjadi aktif.	Berhasil melakukan *update* status produk menjadi aktif.	Sesuai
37	Daftar Verifikasi Pembayaran	1) Pilih menu “Verifikasi Pembayaran”.	Tidak ada	Menampilkan daftar pembayaran yang dilakukan oleh *user*.	Menampilkan daftar pembayaran yang dilakukan oleh *user*.	Sesuai
38	Fungsi Konfirmasi Pembayaran	1) Pilih menu “Verifikasi Pembayaran”. 2) Pilih logo mata pada salah satu pembayaran. 3) Pilih tombol “Setuijui Pembayaran”	Tidak ada	Berhasil melakukan *update* status transaksi menjadi “sedang diproses”.	Berhasil melakukan *update* status transaksi menjadi “sedang diproses”.	Sesuai

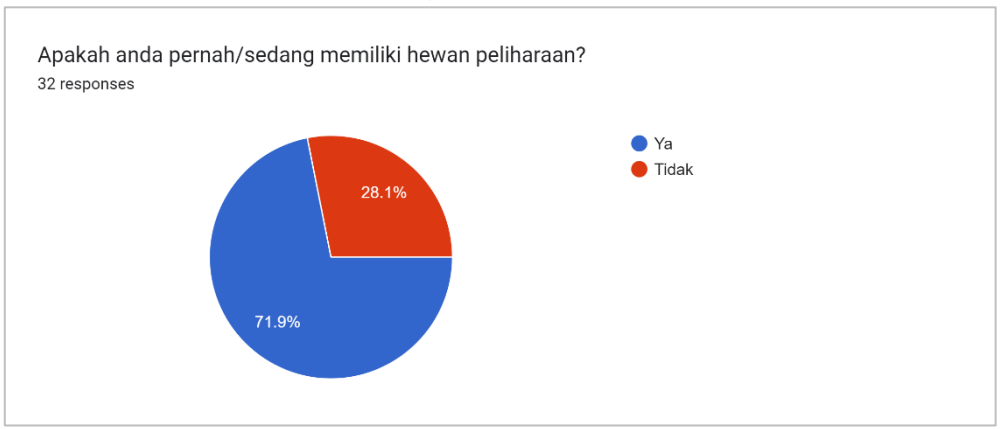
39	Fungsi Tolak Pembayaran	1) Pilih menu “Verifikasi Pembayaran”. 2) Pilih logo mata pada salah satu pembayaran. 3) Pilih tombol “Tolak Pembayaran”	Tidak ada	Berhasil melakukan *update* status transaksi menjadi “gagal”.	Berhasil melakukan *update* status transaksi menjadi “gagal”.	Sesuai
40	Daftar Laporan Transaksi Bermasalah	1) Pilih menu “Laporan Transaksi”.	Tidak ada	Menampilkan daftar laporan transaksi bermasalah yang diajukan oleh *user*.	Menampilkan daftar laporan transaksi bermasalah yang diajukan oleh *user*.	Sesuai
41	Fungsi *generate* Laporan Transaksi Bulanan	1) Piilih menu “Laporan Transaksi Bulanan” 2) Pilih bulan dan tahun.	-bulan : “Oktober” -tahun : “2023”	Menampilkan grafik beserta dengan rincian berdasarkan tanggal pada bulan yang dipilih.	Menampilkan grafik beserta dengan rincian berdasarkan tanggal pada bulan yang dipilih.	Sesuai

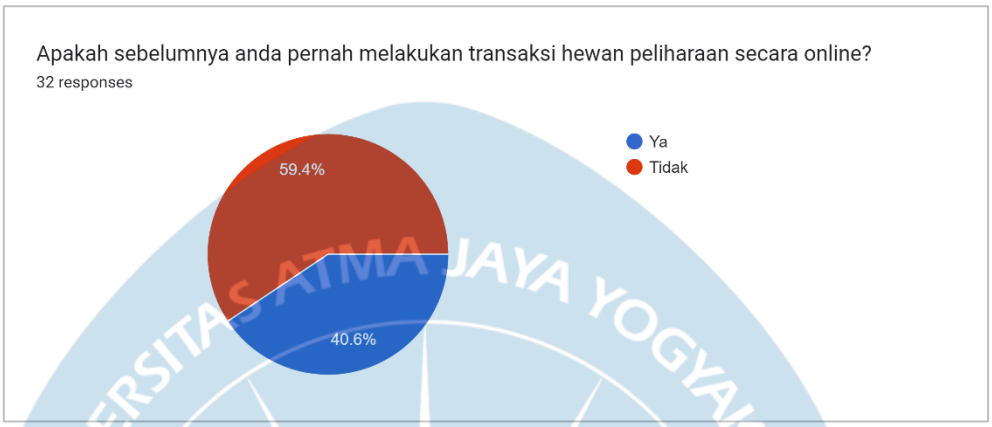
C. Hasil Pengujian Terhadap Pengguna

Pada bagian ini berisi hasil dari pengujian aplikasi yang telah dibangun terhadap pengguna. Pengujian dilakukan menggunakan metode kuesioner yang dilakukan secara *online* kepada pengguna yang telah uji coba aplikasi. Tujuan dari kuesioner ini untuk membuktikan tercapai atau tidaknya tujuan dari pembangunan aplikasi *e-commerce* jual beli hewan peliharaan. Pengujian menggunakan metode Skala Likert sebagai metode analisis yang digunakan untuk mengukur tanggapan positif dan negatif terhadap suatu pernyataan. Berikut merupakan parameter penilaian yang digunakan:

1. Sangat Tidak Setuju
2. Tidak Setuju
3. Netral
4. Setuju
5. Sangat Setuju

Dari penyebaran kuesioner, diperoleh 32 responden yang telah mencoba dan menggunakan aplikasi PetZilla. Terlihat pada gambar 5.26 bahwa 71,9% dari responden pernah/sedang memiliki hewan peliharaan. Selain itu, 59,4% dari responden tidak pernah melakukan transaksi hewan peliharaan secara *online* dan sisanya pernah melakukan transaski hewan peliharaan secara *online*.

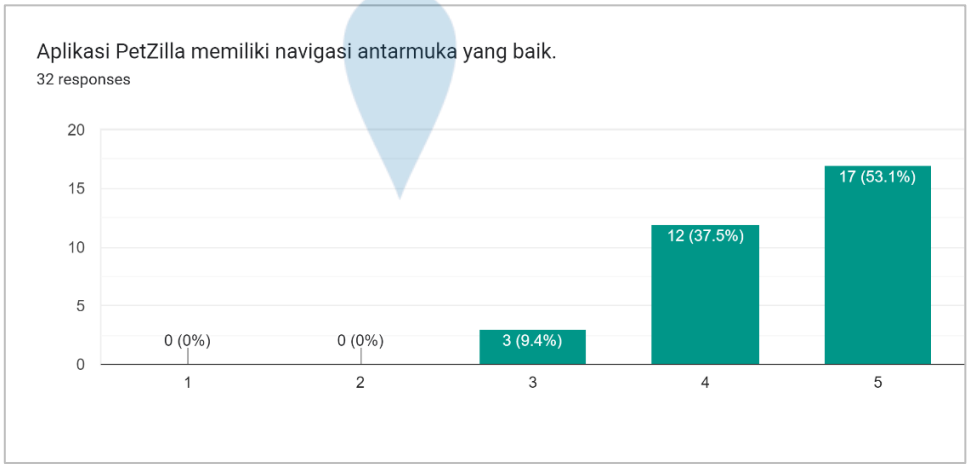




Gambar 5. 89 Grafik Latar Belakang Responden

Berikut ini merupakan hasil kuesioner dari responden yang telah mencoba dan menggunakan sistem.

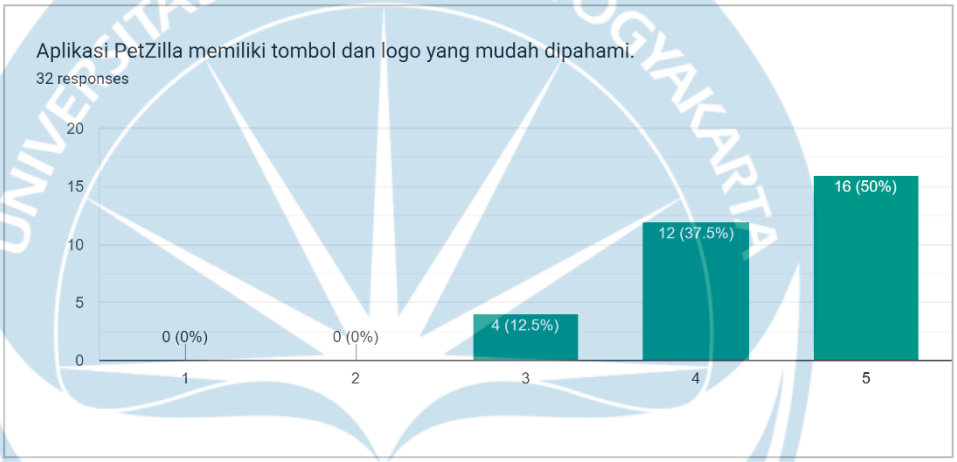
1) Aplikasi PetZilla memiliki navigasi antarmuka yang baik



Gambar 5. 90 Grafik Jawaban Hasil Pertanyaan Kuesioner Pertama

Pada grafik yang ditunjukkan pada Gambar 5.87 didapatkan 17 responden yang sangat setuju, 12 responden yang setuju dan tiga responden yang netral bahwa aplikasi memiliki navigasi antarmuka yang baik.

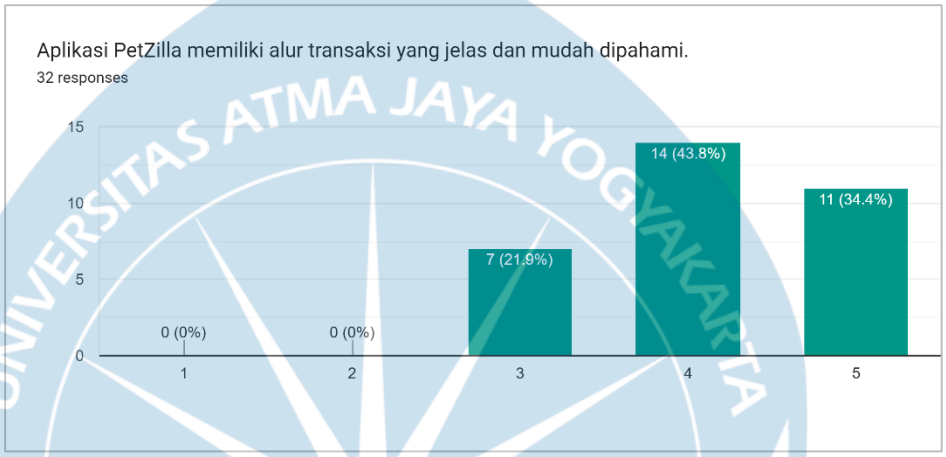
2) Aplikasi PetZilla memiliki tombol dan logo yang mudah dipahami.



Gambar 5. 91 Grafik Jawaban Hasil Pertanyaan Kuesioner Kedua

Pada grafik yang ditunjukkan pada Gambar 5.88 didapatkan hasil dengan 16 responden yang sangat setuju, 12 responden yang setuju dan empat responden yang netral bahwa aplikasi memiliki tombol dan logo yang mudah dipahami.

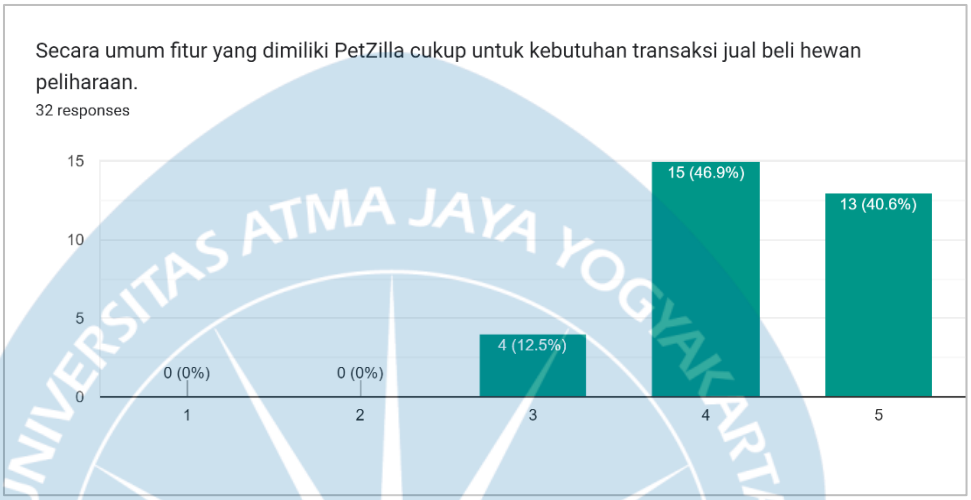
3) Aplikasi PetZilla memiliki alur transaksi yang jelas dan mudah dipahami.



Gambar 5. 92 Grafik Jawaban Hasil Pertanyaan Kuesioner Ketiga

Pada grafik yang ditunjukkan pada Gambar 5.89 didapatkan hasil dengan 11 responden yang sangat setuju, 14 responden yang setuju dan tujuh responden yang netral bahwa aplikasi memiliki alur transaksi yang jelas dan mudah dipahami.

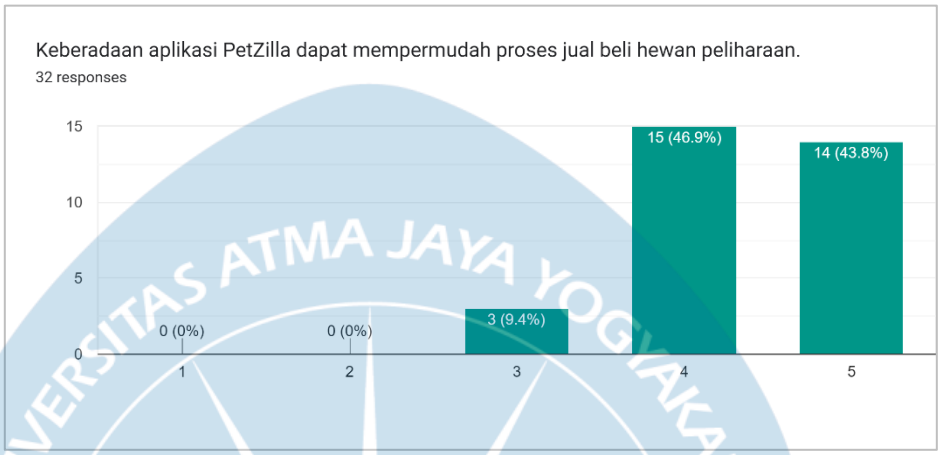
4) Secara umum, fitur yang dimiliki PetZilla cukup untuk kebutuhan transaksi jual beli hewan peliharaan.



Gambar 5. 93 Grafik Jawaban Hasil Pertanyaan Kuesioner Keempat

Pada grafik yang ditunjukkan pada Gambar 5.89 didapatkan hasil dengan 13 responden yang sangat setuju, 15 responden yang setuju dan empat responden yang netral bahwa fitur yang dimiliki PetZilla cukup untuk kebutuhan transaksi jual beli hewan peliharaan.

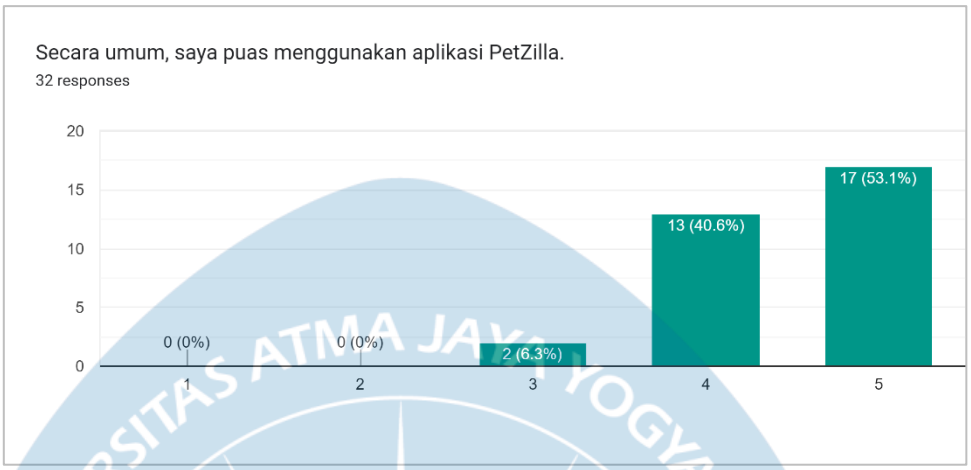
5) Keberadaan aplikasi PetZilla dapat mempermudah proses jual beli hewan peliharaan.



Gambar 5. 94 Grafik Jawaban Hasil Pertanyaan Kuesioner Kelima

Pada grafik yang ditunjukkan pada Gambar 5.91 didapatkan hasil dengan 14 responden yang sangat setuju, 15 responden yang setuju dan tiga responden yang netral bahwa, keberadaan aplikasi PetZilla dapat mempermudah proses jual beli hewan peliharaan.

6) Secara umum, saya puas menggunakan aplikasi PetZilla.



Gambar 5. 95 Grafik Jawaban Hasil Pertanyaan Kuesioner Kelima

Pada grafik yang ditunjukkan pada Gambar 5.92 didapatkan hasil dengan 17 responden yang sangat setuju, 13 responden yang setuju dan dua responden yang netral bahwa secara umum mereka puas menggunakan aplikasi PetZilla.

Berdasarkan hasil pengujian yang telah diberikan kepada pengguna, terlihat bahwa responden memberikan respon yang positif terhadap pernyataan terkait aplikasi PetZilla. Berdasarkan saran yang diberikan responden, terlihat masih ada beberapa kekurangan pada aplikasi PetZilla. Berikut merupakan kelebihan dan kekurangannya, antara lain :

Kelebihan :

- Aplikasi PetZilla memiliki navigasi antarmuka yang baik.
- Aplikasi PetZilla memiliki tombol dan logo yang mudah dipahami.
- Aplikasi PetZilla mempermudah proses transaksi jual beli hewan peliharaan secara *online*.

Kekurangan :

- PetZilla memiliki proses transaksi yang cukup panjang dan rumit.
- Aplikasi tidak memiliki tampilan yang *responsive* pada semua perangkat.
- Minimnya informasi yang dapat membantu pengguna saat pertama kali mencoba.