DOKUMENTASI PROJEK WEB PROGRAMMING
E-CATERING EATWELL

Oleh:
Agatha Gloria Tungky - 2702363304
Cynthia Antoni - 2702363355
Davin Alviano - 2702363374
Ivan Cornelius Saputra - 2702363443
Michael Onasis Hasri - 2702363506
William Valerie - 2702363632

DAFTAR ISI

BAB I INTRODUCTION

A. Latar Belakang

Dalam beberapa tahun terakhir, kesadaran masyarakat akan pentingnya gaya hidup sehat terus mengalami peningkatan. Banyak individu, terutama yang tinggal di perkotaan, mulai memperhatikan asupan makanan mereka sebagai bagian dari upaya menjaga kesehatan, meningkatkan energi, dan mencegah berbagai penyakit. Namun, meskipun minat terhadap pola makan sehat kian tinggi, kenyataannya tidak semua orang memiliki cukup waktu, pengetahuan gizi, atau kemampuan memasak untuk menyiapkan makanan sehat setiap harinya. Aktivitas yang padat, kesulitan akses makanan sehat, serta minimnya variasi resep yang sesuai dengan kebutuhan nutrisi menjadi tantangan utama bagi banyak orang untuk konsisten menjalani pola makan sehat.

Di sisi lain, proses pencarian dan pemesanan layanan catering tersebut seringkali belum praktis. Pengguna harus mencari satu per satu vendor melalui media sosial atau rekomendasi pribadi, tanpa jaminan kejelasan menu, komposisi gizi, sistem pembayaran yang aman, atau keandalan pengiriman. Hal ini menyebabkan pengalaman pengguna menjadi tidak efisien dan menyulitkan, terutama bagi mereka yang menginginkan solusi cepat dan terpercaya untuk kebutuhan makan sehat sehari-hari.

B. Tujuan dan Manfaat

Website EatWell bertujuan untuk menjadi platform digital yang mempertemukan masyarakat yang ingin menerapkan pola makan sehat dengan penyedia jasa catering makanan sehat secara praktis, aman, dan terpercaya. Melalui sistem yang dirancang menyerupai e-commerce, EatWell memudahkan pengguna dalam mencari, memilih, dan memesan menu makanan sehat sesuai kebutuhan harian maupun mingguan. Selain itu, platform ini juga bertujuan memberikan kemudahan bagi para pelaku usaha catering untuk mengelola menu, pesanan, dan pelanggan secara efisien dalam satu sistem terpusat.

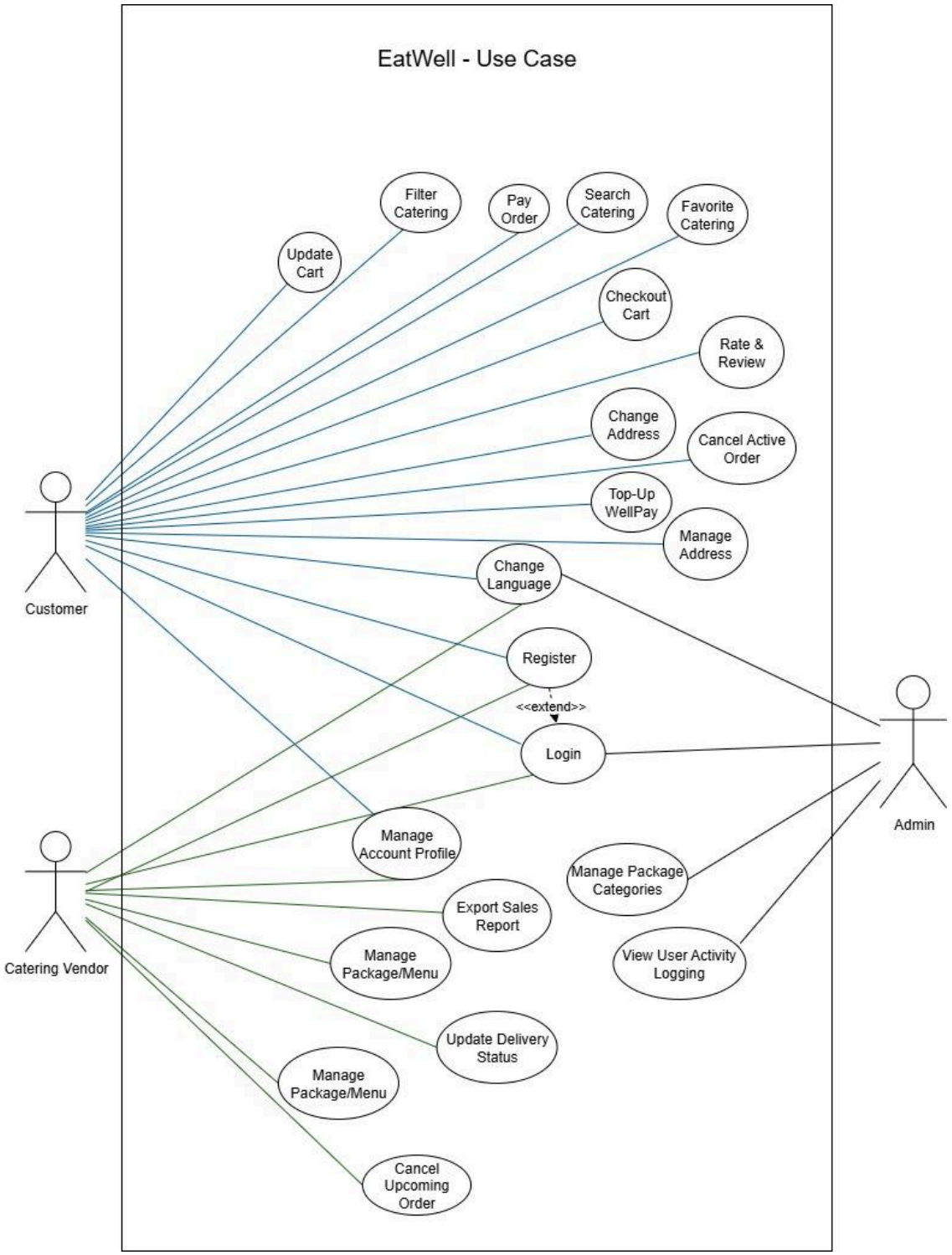
Dengan adanya website EatWell, masyarakat dapat menjalani gaya hidup sehat dengan lebih mudah karena tidak perlu lagi repot menyiapkan makanan sendiri atau bingung mencari vendor yang tepat. Pengguna bisa mengakses berbagai pilihan menu sehat dari vendor terpercaya, melihat informasi gizi, serta memesan dan memantau pengiriman dengan cepat dan aman. Di sisi lain, pelaku usaha catering mendapatkan manfaat berupa peningkatan jangkauan pasar, kemudahan operasional, dan dukungan digitalisasi usaha tanpa harus membangun platform sendiri, sehingga membuka peluang ekonomi baru di sektor kuliner sehat.

C. Tech Stack

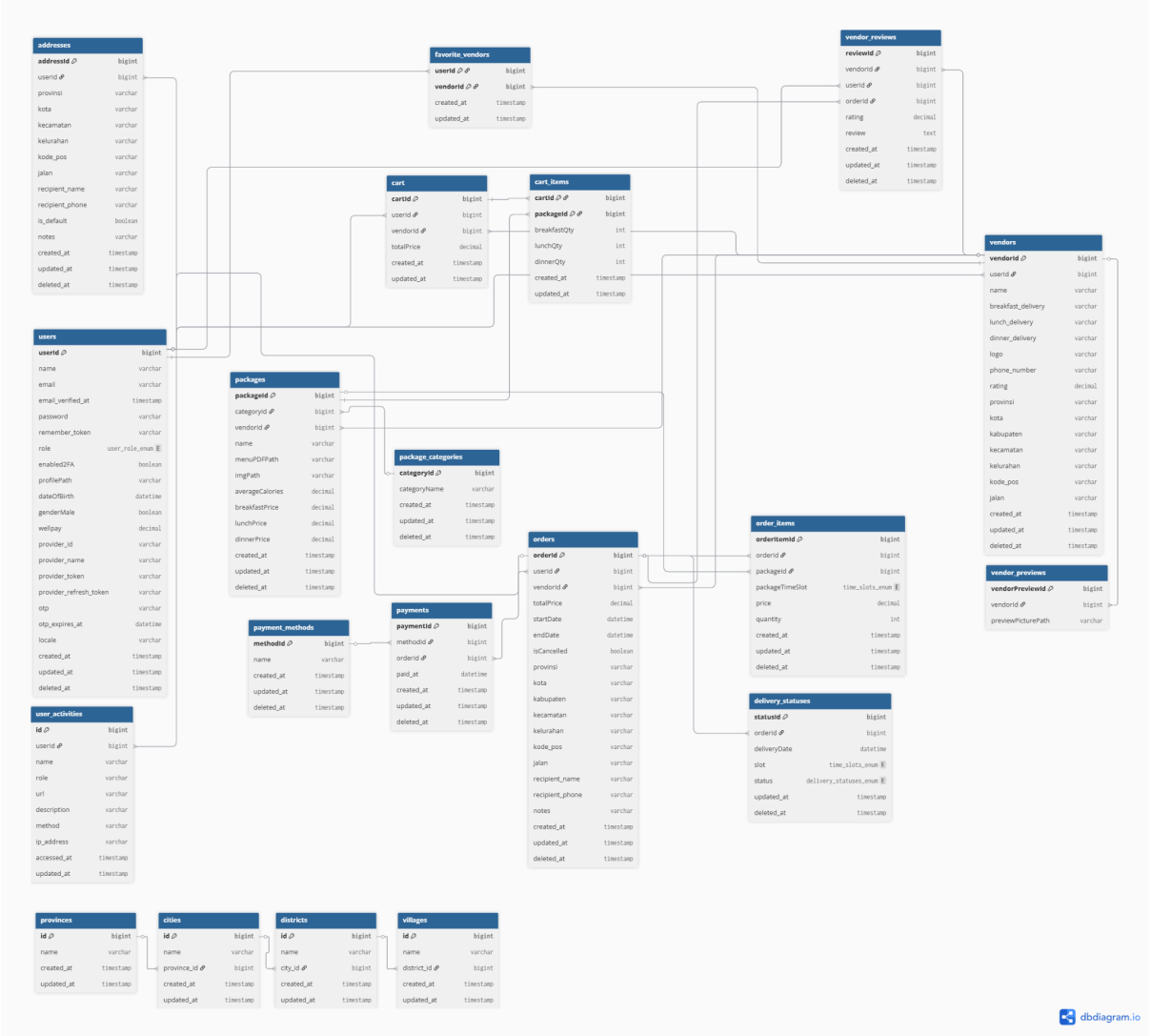
- Template Engine: Blade
- CSS Framework: Bootstrap
- Build Tool: Vite
- Icons & Font: Google Icons & Fonts
- Framework: Laravel 11 (PHP 8.4)
- Primary DB: MySQL 8
- PHP Unit: Feature tests
- Laravel Excel (Maatwebsite 3.1) : export dan import menu

BAB II SYSTEM DESIGN

A. Use Case Diagram



B. Entity Relationship Diagram



1. users

Menyimpan data seluruh akun pengguna: pelanggan, vendor, dan admin.
Atribut:

- userId: Primary key, ID unik pengguna
- name: Nama lengkap pengguna
- email: Alamat email pengguna (unik)
- email_verified_at: Waktu verifikasi email
- password: Kata sandi yang sudah di-hash
- role: Peran pengguna: customer, vendor, admin
- enabled2FA: Apakah otentikasi dua faktor diaktifkan
- remember_token: Token untuk fungsi 'remember me'
- dateOfBirth: Tanggal lahir pengguna
- genderMale: Jenis kelamin (true = laki-laki)
- locale: Preferensi bahasa pengguna direkam untuk keperluan lokalisasi email yang dikirim
- wellpay: Saldo virtual pengguna
- created_at: Waktu data dibuat
- updated_at: Waktu data diperbarui

Relasi:

- 1 user memiliki banyak addresses
- 1 user dapat menjadi 1 vendor
- 1 user memiliki 1 cart
- 1 user memiliki banyak orders
- 1 user memiliki banyak vendor_reviews
- 1 user memiliki banyak user_activities
- 1 user dapat menyukai banyak vendors (melalui favorite_vendors)

2. **Addresses**

Alamat milik user yang digunakan untuk pengiriman.
Atribut:

- addressId: Primary key, ID alamat
- userId: Foreign key ke users
- provinsi: Provinsi
- kota: Kota
- kecamatan: Kecamatan
- kelurahan: Kelurahan
- kode_pos: Kode pos
- jalan: Nama jalan dan nomor
- recipient_name: Nama penerima
- recipient_phone: Nomor telepon penerima
- is_default: Menandakan alamat default
- notes: Catatan tambahan

Relasi:
Setiap address dimiliki oleh satu user (belongsTo users)

3. **vendors**

Data profil bisnis dari vendor (penjual makanan).
Atribut:

- vendorId: Primary key, ID vendor
- userId: Foreign key ke users
- name: Nama usaha/vendor
- logo: Path logo
- phone_number: Nomor kontak vendor
- rating: Rata-rata penilaian vendor
- provinsi - jalan: Alamat lengkap vendor
- breakfast_delivery: Waktu antar sarapan
- lunch_delivery: Waktu antar makan siang
- dinner_delivery: Waktu antar malam

Relasi:

- 1 vendor dimiliki oleh 1 user
- 1 vendor memiliki banyak packages
- 1 vendor memiliki banyak orders dan reviews

- 1 vendor memiliki banyak vendor_previews
- 1 vendor memiliki 1 cart
- 1 vendor dapat difavoritkan oleh banyak user (melalui favorite_vendors)

4. **favorite_vendors**
   Pivot table antara user dan vendor yang disukai dengan atribut default saja, yaitu menyimpan userId, vendorId, created_at dan update_at.

5. **vendor_reviews**
   Review dan rating yang diberikan pengguna ke vendor.
   Atribut:
   - reviewId: Primary key
   - vendorId: FK ke vendors
   - userId: FK ke users
   - orderId: FK ke orders
   - rating: Nilai rating
   - review: Isi ulasan

   Relasi:
   belongsTo users, vendors, orders

6. **vendor_previews**
   Gambar preview dari vendor.
   Atribut:
   - vendorPreviewId: Primary key
   - vendorId: FK ke vendors
   - previewPicturePath: Path gambar

   Relasi: belongs to vendors

7. **package_categories**
   Kategori dari paket makanan, seperti diet, reguler, vegetarian, dll.
   Atribut:
   - categoryId: Primary key
   - categoryName: Nama kategori
   - created_at: Waktu dibuat
   - updated_at: Waktu diperbarui

   Relasi: 1 kategori terhubung ke banyak packages
8. **packages**
   Paket makanan yang dijual vendor.
   Atribut:
   - packageId: Primary key
   - vendorId: FK ke vendors
   - categoryId: FK ke package_categories
   - name: Nama paket
   - menuPDFPath: Link ke menu PDF
   - imgPath: Gambar paket

- averageCalories: Kalori rata-rata
- breakfastPrice: Harga sarapan
- lunchPrice: Harga makan siang
- dinnerPrice: Harga makan malam

Relasi:
- belongs to vendors dan package_categories
- has many ke cart_items dan order_items

9. **orders**
   Pesanan makanan yang dilakukan pengguna.
   Atribut:
   - orderId: Primary key
   - userId: FK ke users
   - vendorId: FK ke vendors
   - totalPrice: Total harga
   - startDate: Tanggal mulai
   - endDate: Tanggal akhir
   - isCancelled: Status pembatalan
   - alamat lengkap: Provinsi - Jalan
   - recipient_name: Nama penerima
   - recipient_phone: Nomor HP
   - notes: Catatan

   Relasi:
   - belongsTo users dan vendors
   - hasMany order_items dan delivery_statuses
   - hasOne payment dan vendor_review

10. **order_items**
    Item individual dalam suatu order, berisi detail paket dan kuantitas.
    Atribut:
    - orderItemId: Primary key
    - orderId: Foreign key ke orders
    - packageId: Foreign key ke packages
    - packageTimeSlot: Slot waktu: Morning, Afternoon, Night
    - price: Harga item
    - quantity: Jumlah yang dipesan

    Relasi: belongsTo orders dan packages

11. **delivery_statuses**
    Status pengantaran makanan per tanggal dan slot waktu.
    Atribut:
    - statusId: Primary key
    - orderId: FK ke orders
    - deliveryDate: Tanggal pengiriman

- slot: Slot waktu pengiriman
- status: Status pengiriman: Prepared, Delivered, Arrived, Cancelled
- updated_at: Waktu diperbarui

Relasi: belongs to orders

12. **payment_methods**
    Daftar metode pembayaran yang tersedia.
    Atribut:
    - methodId: Primary key
    - name: Nama metode pembayaran
    - created_at: Waktu dibuat
    - updated_at: Waktu diperbarui

    Relasi: terhubung ke banyak payments
13. **payments**
    Data pembayaran pesanan pengguna.
    Atribut:
    - paymentId: Primary key
    - methodId: FK ke payment_methods
    - orderId: FK ke orders
    - paid_at: Waktu pembayaran
    - created_at: Waktu dibuat
    - updated_at: Waktu diperbarui

    Relasi: belongsTo orders dan payment_methods
14. **cart**
    Keranjang sementara pengguna untuk menambahkan paket sebelum checkout.
    Atribut:
    - cartId: Primary key
    - userId: FK ke users
    - vendorId: FK ke vendors
    - totalPrice: Total harga dalam keranjang
    - created_at: Waktu dibuat
    - updated_at: Waktu diperbarui

    Relasi:
    - belongsTo users dan vendors
    - hasMany cart_items

15. **cart_items**
    Isi dari keranjang, berisi kuantitas paket berdasarkan waktu makan
    Atribut:
    - cartId: FK ke cart
    - packageId: FK ke packages
    - breakfastQty: Jumlah untuk sarapan
    - lunchQty: Jumlah untuk makan siang
    - dinnerQty: Jumlah untuk makan malam

Relasi: belongsTo cart and packages

16. **user_activities**
    Mencatat seluruh aktivitas pengguna di sistem untuk keperluan audit/log.
    Atribut:
    - userId: FK ke users
    - name: Nama pengguna
    - role: Peran pengguna
    - url: URL yang diakses
    - description: Deskripsi aktivitas
    - method: Metode HTTP
    - ip_address: Alamat IP pengguna
    - accessed_at: Waktu akses

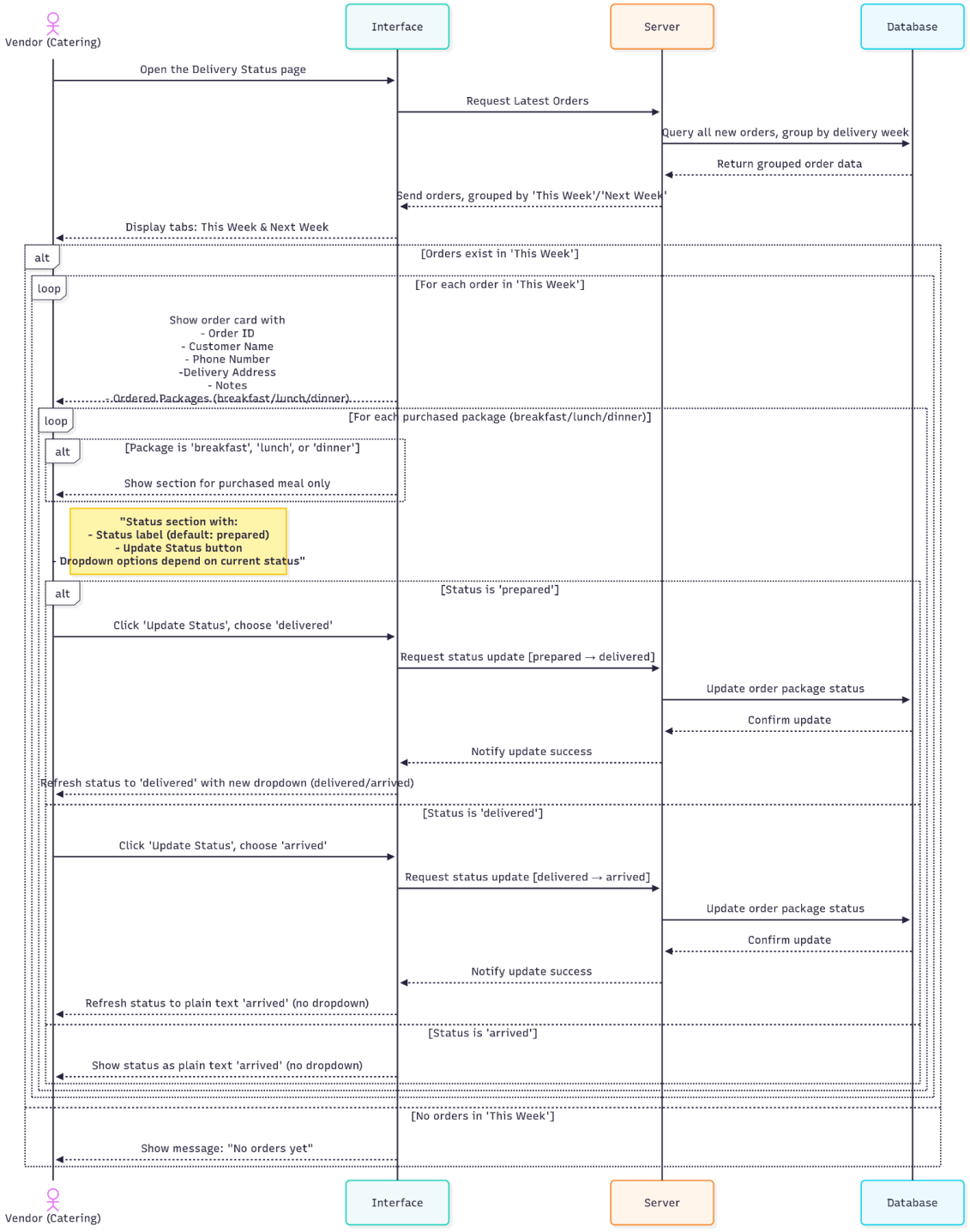
    Relasi: belongsTo users

17. **provinces, cities, district, villages**
    Tabel referensi wilayah administratif Indonesia secara hierarkis yang didapat dari API Wilayah Indonesia oleh Binderbyte lalu disimpan dalam database local EatWell.
    Atribut:
    - province_id: ID provinsi
    - city_id: ID kota
    - district_id: ID kecamatan
    - village_id: ID desa/kelurahan
    - name: Nama wilayah

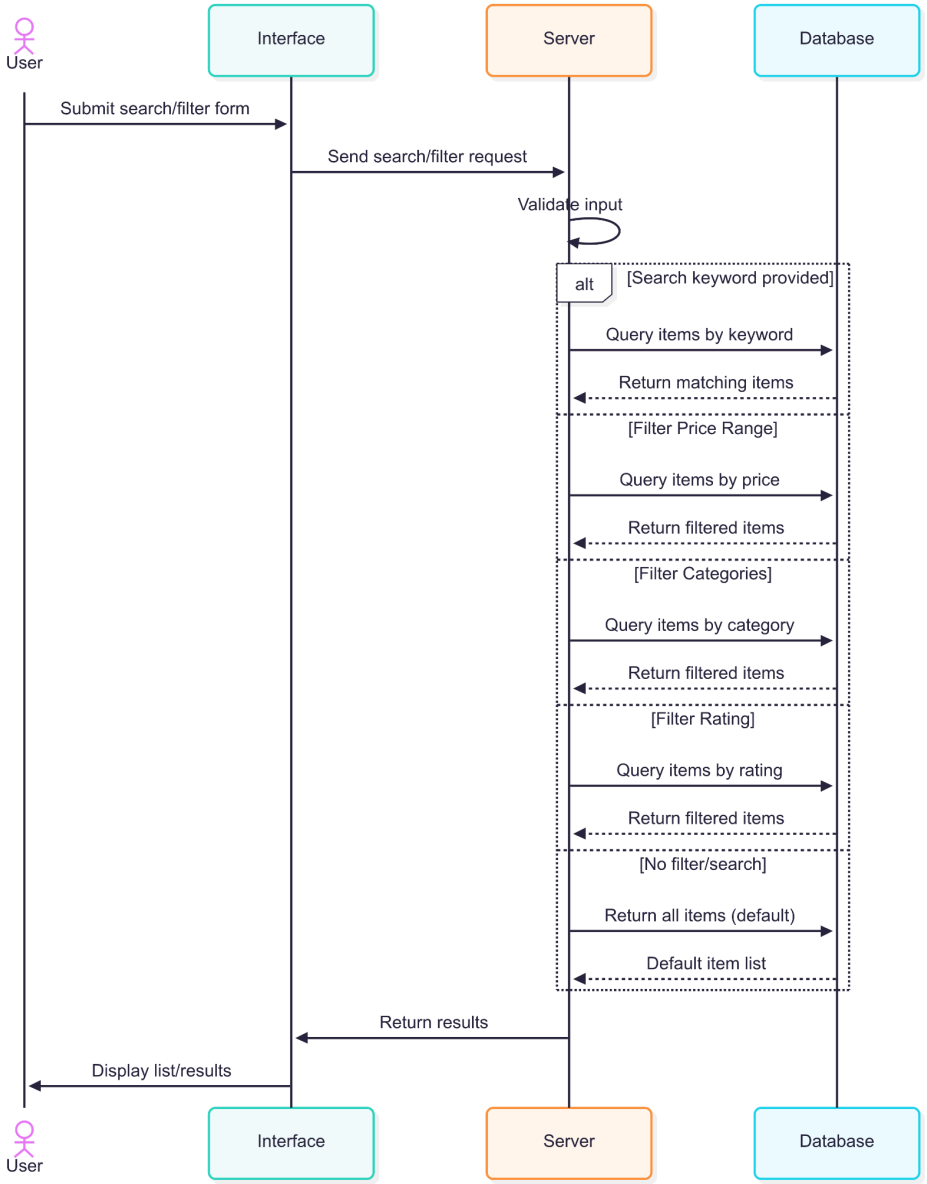
    Relasi:
    - 1 provinsi punya banyak kota/kabupaten
    - 1 kota/kabupaten punya banyak kecamatan
    - 1 kecamatan punya banyak kelurahan/desa

C. Sequence Diagram

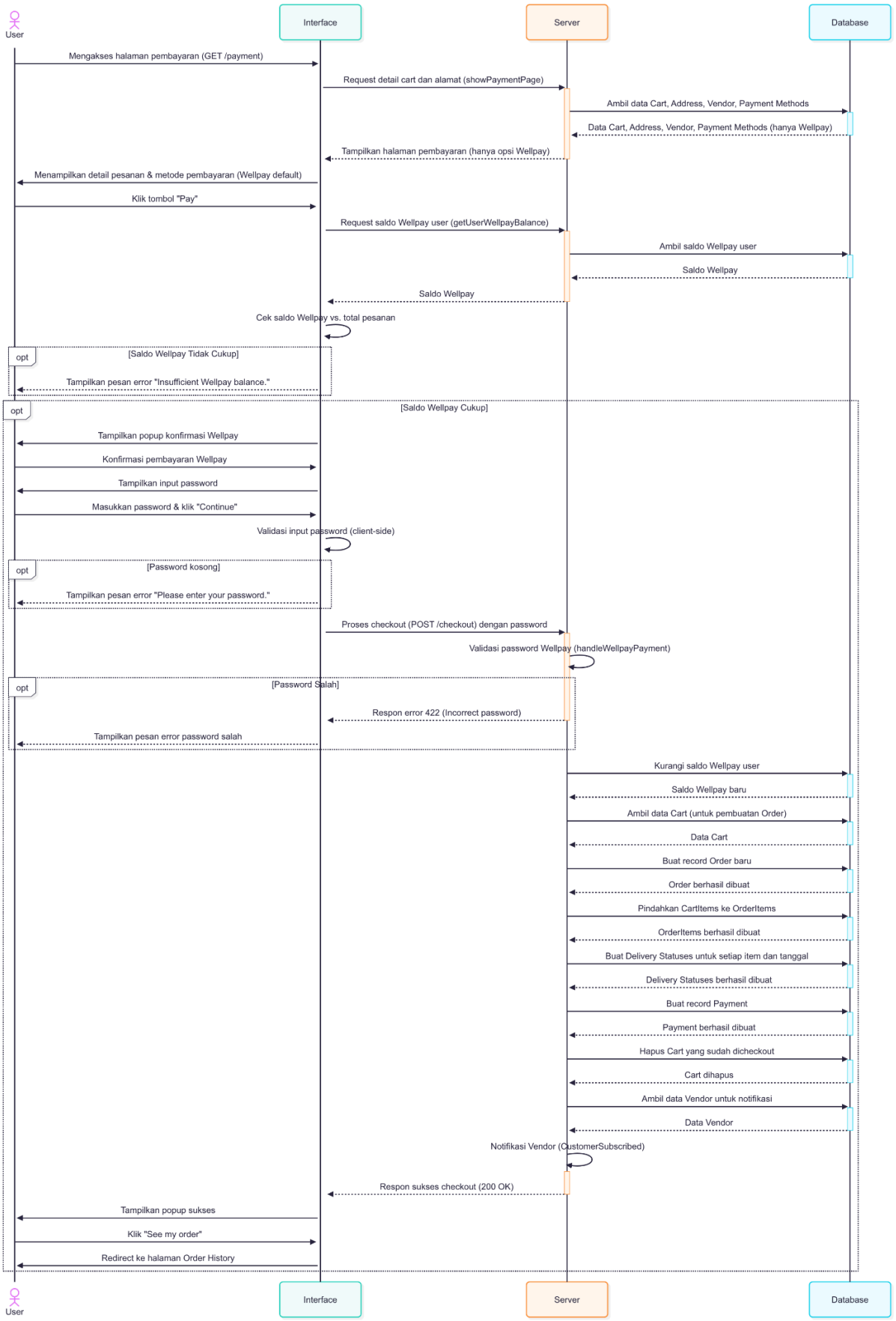
1. Vendor Update Status Delivery



2. Search Catering



3. Checkout Cart



BAB III FEATURES & IMPLEMENTATION

A. Login

User dengan peran Customer, Vendor, dan Admin dapat login ke website dengan menggunakan email dan password yang telah didaftarkan. Bila email mereka belum terverifikasi, maka mereka akan diarahkan ke laman verifikasi email menggunakan OTP. Terdapat 2 validasi utama pada login, yaitu email harus terdaftar dan password diisi dengan benar. Setelah berhasil login, customer langsung diarahkan ke halaman Home (beranda) milik customer, vendor diarahkan ke catering home page, dan admin diarahkan ke admin dashboard. Berikut adalah implementasi dari backend login yang dibuat:

```php
public function store(SessionRequest $request)
    {
        $attrs = $request->validated();

        $remember = request()->has('remember');
        $user = User::where('email', $attrs['email'])->first();
        Session(['remember' => $remember]);
        Session(['email' => $attrs['email']]);
        if(!$user){
            loginLog($request->email, ' Login Failed : Error, user not found');
            throw ValidationException::withMessages([
                'email' => 'Credentials do not match',
                'password' => 'Credentials do not match'
            ]);
        }

        if(!($user->email === $attrs['email'] && Hash::check($attrs['password'], $user->password))){
            loginLog($request->email, ' Login Failed : Error, credentials do not match');
            throw ValidationException::withMessages([
                'email' => 'Credentials do not match',
                'password' => 'Credentials do not match'
            ]);
        }

        if(!$user->email_verified_at || $user->enabled_2fa){
            $otp = rand(100000, 999999);
            $user->update([
                'otp' => $otp,
                'otp_expires_at' => Carbon::now()->addMinutes(3),
            ]);

            $email = $user->email;
            Mail::send('emails.otp', ['otp' => $otp], function ($message) use ($email){
                $message->to($email)->subject('Your OTP');
            });

            return redirect()->route('auth.verify');
        }

        Auth::login($user, $remember);
        loginLog($request->email, 'Successfully');
        return redirect()->route('home');
    }
```

Terdapat pengecekan bila email dari user tidak ditemukan (belum terdaftar) dan pengecekan kesamaan password untuk email yang diinput. Sebelumnya, input yang diterima disaring (validasi melalui Request Form) dengan validasi berikut:

```php
public function rules()
    {
        return [
            'email' => ['required', 'email', 'exists:users'],
            'password' => ['required'],
        ];
    }
```

Validasi ini sudah disesuaikan juga dengan validasi ketika customer maupun vendor register.

B. Change Language

Fitur ini memungkinkan user untuk mengubah bahasa aplikasi (antara Bahasa Indonesia atau Inggris). Perubahan bahasa disimpan di database jika user sudah login, dan di session jika user belum login.

1. LanguageRequest.php:

Digunakan untuk memvalidasi data bahasa yang dikirim dari form atau dropdown pemilihan bahasa.

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

class LanguageRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'lang' => [
                'required',
                Rule::in([
                    'en', 'id'
                ]),
            ],
        ];
    }
}
```

Atribut yang diterima:

- lang: kode bahasa yang dipilih oleh pengguna.

Validasi:

- lang:
    - Wajib diisi (required)
    - Harus bernilai 'en' (English) atau 'id' (Indonesian)
    - Validasi menggunakan Rule::in(['en', 'id'])

2. LanguageController.php:

Mengelola logika perubahan bahasa berdasarkan status login pengguna.

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\LanguageRequest;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\App;
use Illuminate\Support\Facades\Session;
use Illuminate\Support\Facades\Auth;
use App\Models\User;

class LanguageController extends Controller
{
    /**
```

```
     * Handle the incoming request.
     */
    public function __invoke(LanguageRequest $request)
    {
        $lang = $request->validated()['lang'];

        $user = Auth::user();
        if($user)
        {
            $userId = $user->userId;
            $user = User::find($userId);
            $user->locale = $lang;
            $user->save();
        }
        else
        {
            Session::put('lang', $lang);
        }

        return redirect()->back();
    }
}
```

Alur:

- ➢ Ambil data lang hasil validasi dari LanguageRequest.
- ➢ Cek apakah user sedang login:
  - Jika login: Ambil user ID dan update kolom locale milik user di database.
  - Jika belum login: Simpan data bahasa ke dalam Session menggunakan Session::put('lang', $lang).
- ➢ Redirect kembali ke halaman sebelumnya (redirect()->back()).

3. Integrasi Middleware: SetLocale

Untuk memastikan aplikasi menggunakan bahasa yang dipilih, middleware SetLocale digunakan. Middleware ini memeriksa apakah user login. Jika ya, ambil locale dari database user. Jika tidak login, ambil lang dari session. Kemudian, set bahasa aplikasi dengan: App::setLocale($lang). Middleware ini didaftarkan di AppServiceProvider atau bootstrap/app.php:

```
<?php

use App\Http\Middleware\AccountSetup\EnsureAddressExists;
use App\Http\Middleware\AccountSetup\EnsureNoAddressExist;
use App\Http\Middleware\RoleMiddleware;
use App\Http\Middleware\SetLocale;
use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;
use App\Http\Middleware\LogUserActivity;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function (Middleware $middleware) {
        $middleware->web(append: [
            SetLocale::class,
        ]);
        $middleware->alias([
            'role' => RoleMiddleware::class,
            'ensureAddress' => EnsureAddressExists::class,
            'ensureNoAddress' => EnsureNoAddressExist::class,
        ]);
    })

    ->withExceptions(function (Exceptions $exceptions) {
        //
    })->create();
```

C. Customer

a. Update Cart

Fitur ini memungkinkan pengguna untuk memperbarui isi keranjang mereka berdasarkan pilihan paket makanan dan jumlah porsi (sarapan, makan siang, makan malam). Jika tidak ada paket yang dipilih, maka seluruh item dalam keranjang akan dihapus. Sistem akan secara otomatis menambahkan, memperbarui, atau menghapus item di database sesuai input pengguna. Total harga dan jumlah item akan dihitung ulang secara real-time, dan keranjang akan dihapus sepenuhnya jika tidak ada lagi item tersisa.

1. LoadCartRequest.php:

Digunakan untuk memvalidasi permintaan saat pengguna membuka atau memuat isi keranjang.

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;

class LoadCartRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check();
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'vendor_id' => 'required|integer|exists:vendors,vendorId',
        ];
    }

    /**
     * Prepare the data for validation.
     */
    protected function prepareForValidation()
    {
        $this->merge([
            'user_id' => Auth::id(),
            'session_id' => session()->getId(),
        ]);
    }
}
```

Atribut yang diterima:

- vendor_id: ID vendor yang ingin dimuat keranjangnya.
- user_id: diisi otomatis dari user yang sedang login.
- session_id: ID sesi pengguna.

Validasi:

- vendor_id wajib berupa integer dan harus exist di tabel vendors.

2. UpdateCartRequest.php:

Digunakan untuk memvalidasi data yang dikirim frontend saat pengguna memperbarui isi keranjang.

```php
<?php
```

```php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;

class UpdateCartRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check();
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'packages' => 'required|array',
            'packages.*.id' => 'required|integer|exists:packages,packageId',
            'packages.*.items' => 'required|array',
            'packages.*.items.breakfast' => 'nullable|integer|min:0',
            'packages.*.items.lunch' => 'nullable|integer|min:0',
            'packages.*.items.dinner' => 'nullable|integer|min:0',
            'vendor_id' => 'required|integer|exists:vendors,vendorId',
        ];
    }

    /**
     * Prepare the data for validation.
     */
    protected function prepareForValidation()
    {
        $packages = $this->input('packages', []);
        $formattedPackages = [];

        foreach ($packages as $packageId => $packageData) {
            $formattedPackages[$packageId] = [
                'id' => $packageData['id'] ?? $packageId,
                'items' => [
                    'breakfast' => (int) ($packageData['items']['breakfast'] ?? 0),
                    'lunch' => (int) ($packageData['items']['lunch'] ?? 0),
                    'dinner' => (int) ($packageData['items']['dinner'] ?? 0),
                ],
            ];
        }

        $this->merge([
            'packages' => $formattedPackages,
            'user_id' => Auth::id(),
            'session_id' => session()->getId(),
        ]);
    }
}
```

Atribut yang diterima:

- packages: array berisi daftar paket makanan yang dipilih.
- packages.*.id: ID dari paket makanan.
- packages.*.items.breakfast: jumlah porsi sarapan.
- packages.*.items.lunch: jumlah porsi makan siang.
- packages.*.items.dinner: jumlah porsi makan malam.
- vendor_id: ID vendor dari paket yang dipilih.
- user_id: diisi otomatis dari user yang sedang login (tidak dikirim manual).
- session_id: ID sesi pengguna, juga otomatis.

Validasi:

- packages harus berupa array dan wajib ada.
- packages.*.id harus berupa integer dan exist di tabel packages.
- packages.*.items wajib berupa array.
- breakfast, lunch, dan dinner boleh kosong (nullable), tapi jika ada harus integer minimal 0.
- vendor_id wajib berupa integer dan exist di tabel vendors.

3. CartController.php:

- updateOrderSummary() digunakan untuk memperbarui isi keranjang berdasarkan input pengguna.
- loadCart() digunakan untuk memuat isi keranjang saat pengguna mengakses halaman cart.

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\LoadCartRequest;
use App\Http\Requests\UpdateCartRequest;
use App\Models\Cart;
use App\Models\CartItem;
use App\Models\Package;
use Illuminate\Support\Facades\Log;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class CartController extends Controller
{
    public function updateOrderSummary(UpdateCartRequest $request)
    {
        $selectedPackages = $request->input('packages', []);
        $userId = $request->input('user_id');
        $vendorId = $request->input('vendor_id');

        $cart = Cart::firstOrCreate(
            ['userId' => $userId, 'vendorId' => $vendorId],
            ['totalPrice' => 0]
        );

        $totalItems = 0;
        $totalPrice = 0;

        $packageIdsInRequest = array_keys($selectedPackages);

        if (empty($selectedPackages)) {
            $deletedAllCount = $cart->cartItems()->delete();

            $totalItems = 0;
            $totalPrice = 0;
        } else {
            $actualPackageIdsWithItems = [];

            $deletedCountInitial = $cart->cartItems()->whereNotIn('packageId', $packageIdsInRequest)->delete();

            foreach ($selectedPackages as $packageId => $packageData) {
                if (is_array($packageData) && isset($packageData['items']) && is_array($packageData['items'])) {
                    $itemsData = $packageData['items'];

                    $breakfastQty = (int) ($itemsData['breakfast'] ?? 0);
                    $lunchQty = (int) ($itemsData['lunch'] ?? 0);
                    $dinnerQty = (int) ($itemsData['dinner'] ?? 0);

                    if ($breakfastQty === 0 && $lunchQty === 0 && $dinnerQty === 0) {
                        $deletedCountZeroQty = CartItem::where('cartId', $cart->cartId)
                                        ->where('packageId', $packageId)
                                        ->delete();
                        continue;
                    }

                    $actualPackageIdsWithItems[] = $packageId;
```

```
                    $cartItem = CartItem::updateOrCreate(
                        ['cartId' => $cart->cartId, 'packageId' => $packageId],
                        [
                            'breakfastQty' => $breakfastQty,
                            'lunchQty' => $lunchQty,
                            'dinnerQty' => $dinnerQty,
                        ]
                    );

                    $package = Package::find($packageId);
                    if ($package) {
                        $totalItems += $breakfastQty + $lunchQty + $dinnerQty;
                        $totalPrice += ($breakfastQty * ($package->breakfastPrice ?? 0)) +
                                       ($lunchQty * ($package->lunchPrice ?? 0)) +
                                       ($dinnerQty * ($package->dinnerPrice ?? 0));
                    } else {
                        Log::warning("Package with ID {$packageId} not found in database.");
                    }

                } else {
                      Log::warning('Package data received with unexpected structure for packageId: ' . $packageId,
['packageData' => $packageData]);
                }
            }
        }

        $cart->update(['totalPrice' => $totalPrice]);

        $currentCartItemCount = $cart->cartItems()->count();

        if ($currentCartItemCount === 0) {
            $cart->delete();
        } else {
               Log::info('Main Cart ' . $cart->cartId . ' NOT deleted, still has ' . $currentCartItemCount . '
items.');
        }

        return response()->json([
            'totalItems' => $totalItems,
            'totalPrice' => $totalPrice,
        ]);
    }

    public function loadCart(LoadCartRequest $request)
    {
        $userId = $request->input('user_id');
        $vendorId = $request->input('vendor_id');

        if (!$userId || !$vendorId) {
            return redirect()->route('landingPage');
        }

        $cart = Cart::with('cartItems.package')
            ->where('userId', $userId)
            ->where('vendorId', $vendorId)
            ->first();

        $initialPackages = [];
        $initialTotalItems = 0;
        $initialTotalPrice = 0;

        if ($cart) {
            foreach ($cart->cartItems as $cartItem) {
                $package = $cartItem->package;
                if ($package) {
                    $initialPackages[$package->packageId] = [
                        'id' => $package->packageId,
                        'items' => [
                            'breakfast' => $cartItem->breakfastQty,
                            'lunch' => $cartItem->lunchQty,
                            'dinner' => $cartItem->dinnerQty,
                        ],
                    ];
                    $initialTotalItems += $cartItem->breakfastQty + $cartItem->lunchQty + $cartItem->dinnerQty;
                    $initialTotalPrice += ($cartItem->breakfastQty * ($package->breakfastPrice ?? 0)) +
```

```
                        ($cartItem->lunchQty * ($package->lunchPrice ?? 0)) +
                        ($cartItem->dinnerQty * ($package->dinnerPrice ?? 0));
                }
            }
        }

        return response()->json([
            'packages' => $initialPackages,
            'totalItems' => $initialTotalItems,
            'totalPrice' => $initialTotalPrice,
        ]);
    }
}
```

- updateOrderSummary(UpdateCartRequest $request):
  - ➢ Menerima data paket dari frontend.
  - ➢ Mencari atau membuat cart baru berdasarkan user_id dan vendor_id.
  - ➢ Jika paket kosong, semua item dihapus dari keranjang.
  - ➢ Jika ada paket:
    - ○ Menghapus item yang tidak ada lagi di input frontend.
    - ○ Menambah atau memperbarui item berdasarkan quantity breakfast/lunch/dinner.
    - ○ Menghapus item dengan kuantitas nol (sebagai *safety net*).
    - ○ Menghitung total harga dan total item.
  - ➢ Jika setelah diproses isi keranjang kosong, maka keranjang utama dihapus dari database.
  - ➢ Mengembalikan total item dan total harga sebagai respons JSON.
- loadCart(LoadCartRequest $request):
  - ➢ Mengambil user_id dan vendor_id dari request.
  - ➢ Mencari keranjang aktif dari user dan vendor tersebut.
  - ➢ Meload item-item yang ada di dalamnya (termasuk data paket).
  - ➢ Menghitung total item dan total harga.
  - ➢ Mengembalikan data keranjang sebagai respons JSON ke frontend.

b. Search dan Filter Catering

Pengguna dapat memfilter catering berdasarkan kategori paket, rating, dan harga. Diimplementasikan menggunakan query builder Laravel yang dikirim via GET request dengan inputan filter. Pengguna juga dapat mencari catering berdasarkan nama atau deskripsi. Diimplementasikan dengan fungsi pencarian full-text menggunakan query LIKE di Laravel. Berikut adalah backend yang menghandle filter dan search catering yang diinput user:

```
public function search(VendorSearchRequest $request)
    {
        $user = Auth::user();

        // Get address from session if available
        $mainAddress = null;
        $addressId = session('address_id');

        if ($addressId && $user) {
            $mainAddress = Address::find($addressId);

            // Validate that the address belongs to the user
            if (!$mainAddress || $mainAddress->userId !== $user->userId) {
                $mainAddress = null;
            }
        }

        // Fallback: use user's default address if session address is missing or invalid
        if (!$mainAddress && $user) {
```

```
            if (method_exists($user, 'defaultAddress')) {
                $mainAddress = $user->defaultAddress;
            } else {
                $mainAddress = Address::where('userId', $user->userId)
                    ->where('is_default', 1)
                    ->first();
            }

            // Store it in session if found
            if ($mainAddress) {
                session(['address_id' => $mainAddress->id]);
            }
        }

        $validated = $request->validated();

        // Use validated input data
        $query = $validated['query'] ?? null;
        $minPrice = $validated['min_price'] ?? 0;
        $maxPrice = $validated['max_price'] ?? 999999999;
        $rating = $validated['rating'] ?? null;
        $categories = $validated['category'] ?? [];

        $all_categories = PackageCategory::all();

        $province = $mainAddress?->provinsi;

        $vendors = Vendor::query()
            // Add a province match priority column
            ->selectRaw("vendors.*,
            CASE WHEN vendors.provinsi = ? THEN 1 ELSE 0 END AS province_priority", [$province])

            ->when($query, function ($q) use ($query) {
                $q->where(function ($q) use ($query) {
                    $q->where('name', 'like', "%{$query}%")
                        ->orWhereHas('packages', function ($q2) use ($query) {
                            $q2->where('name', 'like', "%{$query}%")
                                ->orWhereHas('category', function ($q3) use ($query) {
                                    $q3->where('categoryName', 'like', "%{$query}%");
                                });
                        });
                });
            })

            ->when($rating, function ($q) use ($rating) {
                $q->where('rating', '>=', $rating);
            })

            ->when($categories, function ($q) use ($categories) {
                $q->whereHas('packages.category', function ($q2) use ($categories) {
                    $q2->whereIn('categoryName', (array) $categories);
                });
            })

            ->when($minPrice || $maxPrice, function ($q) use ($minPrice, $maxPrice) {
                $q->whereHas('packages', function ($q2) use ($minPrice, $maxPrice) {
                    $q2->where(function ($q3) use ($minPrice, $maxPrice) {
                        if ($minPrice) {
                            $q3->where(function ($q4) use ($minPrice) {
                                $q4->where('breakfastPrice', '>=', $minPrice)
                                    ->orWhere('lunchPrice', '>=', $minPrice)
                                    ->orWhere('dinnerPrice', '>=', $minPrice);
                            });
                        }
                        if ($maxPrice) {
                            $q3->where(function ($q4) use ($maxPrice) {
                                $q4->where('breakfastPrice', '<=', $maxPrice)
                                    ->orWhere('lunchPrice', '<=', $maxPrice)
                                    ->orWhere('dinnerPrice', '<=', $maxPrice);
                            });
                        }
                    });
                });
            })
```

```
            // Sort by province priority first, then by name or any other logic
            ->orderByDesc('province_priority')
            ->orderBy('name')

            ->with(['packages.category', 'packages.cuisineTypes'])

            ->distinct()
            ->paginate(9)
            ->appends($request->query());

        // Pass paginated vendors to the view
        logActivity('Successfully', 'Visited', "Vendor Search Page and Searched for: {$query}");
        return view('customer.search', compact('vendors', 'all_categories', 'user', 'mainAddress'));
    }
```

Satu function search ini bertugas untuk mencari dan memfilter catering berdasarkan inputan user. Pada query builder yang dibuat, ditambahkan case untuk mendapatkan catering yang berada satu provinsi terlebih dahulu baru sisanya catering yang berbeda provinsi. Ini ditujukan agar catering yang berbeda provinsi tidak bisa dipesan oleh customer karena jaraknya terlalu jauh karena belum mengimplementasikan jarak dari google maps. Adapun validasi terhadap input nya sebagai berikut:

```
public function rules(): array
    {
        return [
            'query' => [
                'nullable',
                'string',
                'max:255',
            ],
            'min_price' => [
                'nullable',
                'integer',
            ],
            'max_price' => [
                'nullable',
                'integer',
            ],
            'rating' => [
                'nullable',
                'numeric',
                'min:1',
                'max:5'
            ],
            'category' => [
                'nullable',
                'array',
            ],
            'category.*' => [
                'string'
            ],
        ];
    }
```

Atribut yang diterima dan validasinya:

- query:
  - Opsional (boleh kosong)
  - Harus berupa teks
  - Maksimal 255 karakter
  - Biasanya digunakan sebagai kata kunci pencarian
- min_price dan max_price
  - Opsional
  - Harus berupa angka bulat (integer)
  - Digunakan untuk memfilter rentang harga minimum dan maksimum
- rating
  - Opsional
  - Harus berupa angka desimal

- Nilai minimal: 1 dan maksimal: 5
- Digunakan untuk filter berdasarkan rating vendor

- category
  - Opsional
  - Harus berupa array (daftar beberapa kategori)
  - Setiap elemen di dalam array harus berupa teks (string

c. Pay Order

Fitur Pay Order bertanggung jawab untuk memproses pembayaran pesanan pengguna berdasarkan isi keranjang (Cart) mereka. Proses mencakup validasi metode pembayaran (termasuk WellPay), pemindahan data keranjang menjadi order tetap, pembuatan item pesanan dan status pengiriman, serta notifikasi ke vendor.

1. ProcessCheckoutRequest.php:

Digunakan untuk memvalidasi data saat pengguna melakukan checkout, termasuk metode pembayaran, tanggal pemesanan, dan password (khusus untuk WellPay).

```php
<?php

namespace App\Http\Requests;

use App\Models\PaymentMethod;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\Rule;

class ProcessCheckoutRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check();
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'payment_method_id' => 'required|exists:payment_methods,methodId',
            'start_date' => 'required|date_format:Y-m-d',
            'end_date' => 'required|date_format:Y-m-d|after_or_equal:start_date',
            'password' => [
                Rule::requiredIf(function () {
                                return $this->input('payment_method_id') == PaymentMethod::where('name', 'like',
'WellPay')->first()->methodId;
                }),
                'string',
            ],
        ];
    }

        public function messages(): array
    {
        return [
            'payment_method_id.required' => 'Payment method is required.',
            'payment_method_id.exists' => 'The selected payment method is invalid.',
            'start_date.required' => 'Start date is required.',
            'start_date.date_format' => 'Start date must be in YYYY-MM-DD format.',
            'end_date.required' => 'End date is required.',
            'end_date.date_format' => 'End date must be in YYYY-MM-DD format.',
            'end_date.after_or_equal' => 'End date must be on or after start date.',
            'password.required_if' => 'Password is required for this payment method.',
        ];
```

```
    }

    protected function prepareForValidation(): void
    {
        $this->merge([
            'vendor_id' => (int) $this->input('vendor_id'),
            'payment_method_id' => (int) $this->input('payment_method_id'),
        ]);
    }
}
```

Atribut yang diterima:

- payment_method_id (integer): ID metode pembayaran yang dipilih user.
- start_date (string, format: Y-m-d): Tanggal awal langganan.
- end_date (string, format: Y-m-d): Tanggal akhir langganan.
- password (string, opsional): Diperlukan jika metode pembayaran adalah WellPay.
- vendor_id (integer): Diatur otomatis di prepareForValidation, digunakan saat pengecekan saldo WellPay.

Validasi:

- payment_method_id:
  - required
  - exists:payment_methods,methodId
- start_date:
  - required
  - date_format:Y-m-d
- end_date:
  - required
  - date_format:Y-m-d
  - after_or_equal:start_date
- password:
  - required_if:payment_method_id,<id wellpay>
  - string

Selain itu, terdapat juga custom error message yang memberikan pesan error yang informatif.

2. OrderController.php:

```
    public function getUserWellpayBalance()
    {
        $user = Auth::user();
        if (!$user) {
            return response()->json(['message' => 'User not authenticated.'], 401);
        }
        return response()->json(['wellpay' => $user->wellpay]);
    }

    public function processCheckout(ProcessCheckoutRequest $request)
    {
        /** @var \App\Models\User $user */
        $user = Auth::user();
        $userId = $user->userId;

        $validatedData = $request->validated();

        $vendorId = session('selected_vendor_id');
        if(!$vendorId) {
            return response()->json(['message' => 'No vendor selected.'], 400);
        }
        $paymentMethodId = $validatedData['payment_method_id'];
        $startDate = $validatedData['start_date'];
        $endDate = $validatedData['end_date'];
        $password = $validatedData['password'] ?? null;
        $addressId = session('address_id');
        $address = Address::find($addressId);
```

```
        if(!$address){
            return response()->json(['message' => 'No address selected.'], 400);
        }
        $orderAddressData = $this->extractOrderAddressData($address);
        $notes = $validatedData['notes'] ?? null;

        try {
            DB::beginTransaction();

            $cart = Cart::with('cartItems.package')
                ->where('userId', $userId)
                ->where('vendorId', $vendorId)
                ->first();

            if (!$cart || $cart->cartItems->isEmpty()) {
                DB::rollBack();
                return response()->json(['message' => 'Your cart is empty or expired.'], 400);
            }

            $orderTotalPrice = $cart->totalPrice;

            $wellpayMethod = PaymentMethod::where('name', 'WellPay')->first();
            $wellpayMethodId = $wellpayMethod ? $wellpayMethod->methodId : null;

            if (is_null($wellpayMethodId)) {
                DB::rollBack();
                      return response()->json(['message' => 'Payment method configuration error. Please try again
later.'], 500);
            }

            // Handle Wellpay payment logic
            $this->handleWellpayPayment($user, $paymentMethodId, $orderTotalPrice, $password, $wellpayMethodId);

            // Create Order
                       $order = $this->createOrder($userId, $vendorId, $orderTotalPrice, $startDate, $endDate,
$orderAddressData, $notes);

            // Move CartItems to OrderItems and generate Delivery Statuses
            $this->processOrderItemsAndDeliveryStatuses($order, $cart);

            // Create Payment entry
            $this->createPaymentEntry($order->orderId, $paymentMethodId);

            // Delete Cart
            $cart->delete();

            // Notify vendor
            $this->notifyVendor($vendorId, $order);

            DB::commit();

            logActivity('Successfully', 'Processed', 'Checkout for Order ID: ' . $order->orderId);

            return response()->json(['message' => 'Checkout successful!', 'orderId' => $order->orderId], 200);
        } catch (ValidationException $e) {
            DB::rollBack();
            logActivity('Failed', 'Process', 'Checkout');
            return response()->json(['message' => 'Validation Error', 'errors' => $e->errors()], 422);
        } catch (\Exception $e) {
            DB::rollBack();
            logActivity('Failed', 'Processed', 'Checkout');
                      return response()->json(['message' => 'Checkout failed. Please try again.', 'error' =>
$e->getMessage()], 500);
        }
    }

    private function extractOrderAddressData(Address $address): array
    {
        return [
            'provinsi' => $address->provinsi,
            'kota' => $address->kota,
            'kecamatan' => $address->kecamatan,
            'kelurahan' => $address->kelurahan,
            'kode_pos' => $address->kode_pos,
            'jalan' => $address->jalan,
```

```
            'recipient_name' => $address->recipient_name,
            'recipient_phone' => $address->recipient_phone,
        ];
    }

    /**
     *
     * @param User $user
     * @param int $selectedPaymentMethodId
     * @param float $orderTotalPrice
     * @param string|null $password
     * @param int|null $wellpayMethodId
     * @throws \Illuminate\Validation\ValidationException
     * @throws \Exception
     */
    private function handleWellpayPayment(User $user, int $selectedPaymentMethodId, float $orderTotalPrice, ?string
$password, ?int $wellpayMethodId): void
    {
        if ($selectedPaymentMethodId === $wellpayMethodId) {
            if (!Hash::check($password, $user->getAuthPassword())) {
                logActivity('Failed', 'Processed', 'Checkout due to incorrect Wellpay password');
                throw ValidationException::withMessages([
                    'password' => ['Incorrect password.'],
                ]);
            }

            if ($user->wellpay < $orderTotalPrice) {
                logActivity('Failed', 'Processed', 'Checkout due to insufficient Wellpay balance');
                throw new \Exception('Insufficient Wellpay balance. Please top up.', 402);
            }

            $user->wellpay -= $orderTotalPrice;
            $user->save();
        }
    }

    private function createOrder(
        int $userId,
        int $vendorId,
        float $totalPrice,
        string $startDate,
        string $endDate,
        array $addressData,
        ?string $notes
    ): Order {
        $order = Order::create(array_merge([
            'userId' => $userId,
            'vendorId' => $vendorId,
            'totalPrice' => $totalPrice,
            'startDate' => Carbon::parse($startDate)->startOfDay(),
            'endDate' => Carbon::parse($endDate)->endOfDay(),
            'isCancelled' => false,
            'notes' => $notes,
        ], $addressData));
        return $order;
    }

    private function processOrderItemsAndDeliveryStatuses(Order $order, Cart $cart): void
    {
        $selectedTimeSlots = [];
        foreach ($cart->cartItems as $cartItem) {
            $package = $cartItem->package;
            if ($package) {
                if ($cartItem->breakfastQty > 0) {
                    OrderItem::create([
                        'orderId' => $order->orderId,
                        'packageId' => $package->packageId,
                        'packageTimeSlot' => 'Morning',
                        'price' => ($cartItem->breakfastQty * ($package->breakfastPrice ?? 0)),
                        'quantity' => $cartItem->breakfastQty,
                    ]);
                    $selectedTimeSlots['Morning'] = true;
                }
                if ($cartItem->lunchQty > 0) {
                    OrderItem::create([
```

```
                    'orderId' => $order->orderId,
                    'packageId' => $package->packageId,
                    'packageTimeSlot' => 'Afternoon',
                    'price' => ($cartItem->lunchQty * ($package->lunchPrice ?? 0)),
                    'quantity' => $cartItem->lunchQty,
                ]);
                $selectedTimeSlots['Afternoon'] = true;
            }
            if ($cartItem->dinnerQty > 0) {
                OrderItem::create([
                    'orderId' => $order->orderId,
                    'packageId' => $package->packageId,
                    'packageTimeSlot' => 'Evening',
                    'price' => ($cartItem->dinnerQty * ($package->dinnerPrice ?? 0)),
                    'quantity' => $cartItem->dinnerQty,
                ]);
                $selectedTimeSlots['Evening'] = true;
            }
        }
    }

    $this->generateDeliveryStatuses($order, $selectedTimeSlots);
}

private function generateDeliveryStatuses(Order $order, array $selectedTimeSlots): void
{
    $countDeliveryStatuses = 0;

    foreach (array_keys($selectedTimeSlots) as $slot) {
        $currentDeliveryDate = Carbon::parse($order->startDate);
        for ($i = 0; $i < 7; $i++) {
            DeliveryStatus::create([
                'orderId' => $order->orderId,
                'deliveryDate' => $currentDeliveryDate->toDateString(),
                'slot' => $slot,
                'status' => 'Prepared',
            ]);
            $currentDeliveryDate->addDay();
            $countDeliveryStatuses++;
        }
    }
}

private function createPaymentEntry(int $orderId, int $paymentMethodId): void
{
    Payment::create([
        'methodId' => $paymentMethodId,
        'orderId' => $orderId,
        'paid_at' => Carbon::now(),
    ]);
}

private function notifyVendor(int $vendorId, Order $order): void
{
    $vendor = Vendor::find($vendorId);
    $vendorUserId = $vendor->userId;
    $vendorUser = User::find($vendorUserId);

    if ($vendorUser) {
        $vendorUser->notify(new CustomerSubscribed($order));
    }
}
```

- getUserWellpayBalance():
  Fungsi: Mengembalikan sisa saldo WellPay milik pengguna.
  Output:
  - 200 OK dengan { wellpay: int }
  - 401 Unauthorized jika belum login

- processCheckout(ProcessCheckoutRequest $request):

Fungsi: Melakukan proses checkout dan pembayaran. Semua data dari Cart akan dipindahkan ke Order, OrderItems, dan DeliveryStatuses.
Proses utama:

- Validasi input melalui ProcessCheckoutRequest.
- Ambil data user, vendor, alamat pengiriman, dan cart.
- Validasi:
  - Keranjang harus tidak kosong.
  - Metode pembayaran valid.
  - Jika WellPay, cek password & saldo.
- Jalankan transaksi DB:
  - Buat entri Order.
  - Buat OrderItems berdasarkan waktu makan (breakfast/lunch/dinner).
  - Buat entri DeliveryStatus untuk setiap slot waktu selama 7 hari.
  - Buat entri Payment.
  - Hapus Cart.
  - Kirim notifikasi ke vendor.
- Output:
  - 200 OK → sukses checkout.
  - 422 Unprocessable Entity → password salah / validasi gagal.
  - 500 Internal Server Error → gagal umum.

- Fungsi Pendukung Private:
  - extractOrderAddressData(Address $address): array
    - Ekstraksi data alamat dari model Address menjadi array standar untuk disimpan di tabel orders.
  - handleWellpayPayment(...)
    - Validasi dan proses potongan saldo WellPay jika metode pembayaran yang dipilih adalah WellPay. Validasinya adalah password harus cocok dan saldo mencukupi.
  - createOrder(...)
    - Membuat entri baru di tabel orders berdasarkan data user, vendor, harga, tanggal, dan alamat.
  - processOrderItemsAndDeliveryStatuses(...)
    - Memproses isi Cart menjadi OrderItems, dikelompokkan berdasarkan slot waktu (pagi/siang/malam). Juga membuat 7 entri DeliveryStatus untuk tiap slot waktu yang dipilih.
  - generateDeliveryStatuses(...)
    - Generate entri DeliveryStatus untuk 7 hari ke depan mulai dari hari Senin berikutnya, berdasarkan slot waktu (pagi/siang/malam) yang dipilih.
  - createPaymentEntry(...)
    - Membuat entri payments setelah order berhasil.
  - notifyVendor(...)
    - Kirim notifikasi ke user milik vendor melalui notifikasi Laravel.

d. Favorite Catering

Pengguna bisa menandai catering favorit untuk akses cepat. Data disimpan di tabel pivot user-catering dan ditampilkan di dashboard pengguna. Berikut adalah backend yang menghandle favorit dan unfavorite catering:

```
public function favorite(String $id)
    {
        /**
         * @var User | Authenticatable $user
         */
        $user = Auth::user();
        // Ensure relation exists and not duplicated
        if (!$user->favoriteVendors()->where('vendors.vendorId', '=', $id)->exists()) {
            $user->favoriteVendors()->attach($id);
        }

        $vendor = Vendor::findOrFail($id);

        logActivity('Successfully', 'Favorited', 'Catering : ' . $vendor->name);
        return response()->json(['favorited' => true]);
    }

    public function unfavorite(String $id)
    {
        /**
         * @var User | Authenticatable $user
         */
        $user = Auth::user();
        $user->favoriteVendors()->detach($id);

        $vendor = Vendor::findOrFail($id);

        logActivity('Successfully', 'Unfavorited', 'Catering : ' . $vendor->name);
        return response()->json(['favorited' => false]);
    }
```

Kedua function ini memanfaatkan pivot table favorite_catering yang dipanggil melalui sintaks AJAX javascript untuk memberikan update yang real time kepada usernya.

```
let favButtons = document.querySelectorAll('.btn-fav');
favButtons.forEach((favbutton) => {
    favbutton.addEventListener('click', function(e) {
        e.preventDefault();
        e.stopPropagation();
        const vendorId = favbutton.dataset.vendorId;
        const isFavorited = favbutton.classList.contains('favorited');
        fetch(isFavorited ? `/unfavorite/${vendorId}` : `/favorite/${vendorId}`, {
            method: 'POST',
            headers: {
                'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content,
                'Accept': 'application/json'
            }
        })
        .then(res => res.json())
        .then(data => {
            favbutton.classList.toggle('favorited', data.favorited);
        });
    })
});
```

e. Checkout Cart

Menampilkan halaman pembayaran berdasarkan isi Cart, Vendor, dan Alamat pengguna.

OrderController.php:

```
    public function showPaymentPage()
    {
        $user = Auth::user();
        $userId = $user->userId;
        $vendorId = session('selected_vendor_id');

        if(!$vendorId) {
            return redirect()->back();
        }
```

```
        $vendor = Vendor::find($vendorId);

        if(!$vendor) {
            return redirect()->back();
        }

        $cart = Cart::with(['cartItems.package'])
            ->where('userId', $userId)
            ->where('vendorId', $vendor->vendorId)
            ->first();

        if (!$cart || $cart->cartItems->isEmpty()) {
              return redirect()->back()->with('warning', 'Your cart is empty. Please add items before proceeding to
payment.');
        }

        $orderDateTime = Carbon::now();
        $startDate = $orderDateTime->copy()->next(Carbon::MONDAY)->toDateString();
        $endDate = Carbon::parse($startDate)->copy()->next(Carbon::SUNDAY)->toDateString();

        $cartDetails = [];
        $totalOrderPrice = 0;

        foreach ($cart->cartItems as $item) {
            $package = $item->package;
            if ($package) {
                $itemPrice = ($item->breakfastQty * ($package->breakfastPrice ?? 0)) +
                    ($item->lunchQty * ($package->lunchPrice ?? 0)) +
                    ($item->dinnerQty * ($package->dinnerPrice ?? 0));

                $cartDetails[] = [
                    'package_id' => $package->packageId,
                    'package_name' => $package->name,
                    'breakfast_qty' => $item->breakfastQty,
                    'lunch_qty' => $item->lunchQty,
                    'dinner_qty' => $item->dinnerQty,
                    'breakfast_price' => $item->breakfastQty * ($package->breakfastPrice ?? 0),
                    'lunch_price' => $item->lunchQty * ($package->lunchPrice ?? 0),
                    'dinner_price' => $item->dinnerQty * ($package->dinnerPrice ?? 0),
                    'item_total_price' => $itemPrice,
                ];
                $totalOrderPrice += $itemPrice;
            }
        }

        $selectedAddressId = session('address_id');
        $selectedAddress = null;

        if ($selectedAddressId) {
            $selectedAddress = Address::find($selectedAddressId);
            if ($selectedAddress && $userId && $selectedAddress->userId !== $userId) {
                $selectedAddress = null;
                logActivity('Failed', '', 'Payment with invalid address');
                return redirect()->back()->with('error', 'The selected address does not belong to your account.');
            }

            if($selectedAddress->provinsi != $vendor->provinsi) {
                logActivity('Failed', '', 'Payment, Catering is too far from customer');
                return redirect()->back()->with('error', 'Catering is too far from you.');
            }
        } else {
            if (method_exists($user, 'defaultAddress')) {
                $selectedAddress = $user->defaultAddress;
            } else {
                $selectedAddress = Address::where('userId', $user->userId)
                    ->where('is_default', 1)
                    ->first();
            }
        }

        if (!$selectedAddress) {
            logActivity('Failed', '', 'Payment with no address selected');
            return redirect()->back()->with('error', 'Alamat pengiriman tidak valid atau tidak dipilih.');
        }
```

```
        $paymentMethod = PaymentMethod::all();

         return view('payment', compact('vendor', 'cart', 'cartDetails', 'totalOrderPrice', 'startDate', 'endDate',
'selectedAddress', 'paymentMethod'));
    }
```

Alur:

- ➢ Ambil user login dan vendor yang dipilih (dari session).
- ➢ Cek keberadaan vendor dan keranjang (beserta isinya).
- ➢ Hitung total harga dari seluruh item di keranjang.
- ➢ Validasi alamat yang dipilih:
  - ○ Harus dimiliki oleh user saat ini.
  - ○ Harus dalam jangkauan provinsi vendor.
- ➢ Ambil semua metode pembayaran yang tersedia.
- ➢ Return view payment.blade.php dengan:
  - ○ vendor, cart, cartDetails, totalOrderPrice, startDate, endDate, selectedAddress, paymentMethod.

f. Rate & Review

Pengguna bisa memberikan rating dan ulasan setelah pesanan selesai. Data disimpan dan dihitung untuk menghasilkan rating rata-rata tiap catering yang langsung diupdate ketika sebuah rating masuk. Berikut adalah backend yang mengatur rating yang dibuat user:

```
public function store(CustomerRatingStoreRequest $request, $orderId)
    {

        $validated = $request->validated();

        $order = Order::findOrFail($orderId);

        if($order->isCancelled == 0 && Carbon::now()->lessThan($order->endDate)) {
            return response()->json(['message' => 'You can only rate finished orders.'], 403);
        }

        $userId = Auth::user()->userId;

        // Prevent duplicate reviews per user/order
        $existing = VendorReview::where('orderId', $orderId)
            ->where('userId', $userId)
            ->first();

        if ($existing) {
            return response()->json(['message' => 'You have already reviewed this order.'], 409);
        }

        $review = VendorReview::create([
            'vendorId' => $order->vendorId,
            'userId' => $userId,
            'orderId' => $orderId,
            'rating' => $request->rating,
            'review' => $request->review,
        ]);

        logActivity('Successfully', 'Upload', 'Review');
        return response()->json(['success' => true, 'review' => $review]);
    }
```

Dilakukan pengecekan bila order tersebut belum selesai atau sudah di rating. Inputan diberikan lewat modal kepada user lalu divalidasi terlebih dahulu dengan validasi berikut:

```
public function authorize(): bool
    {
        return Auth::check() && Auth::user()->role === UserRole::Customer;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
```

```
     */
    public function rules(): array
    {
        return [
            'rating' => [
                'required',
                'integer',
                'min:1',
                'max:5',
            ],
            'review' => [
                'nullable',
                'string',
                'max:1000',
            ],
        ];
    }
```

di mana atribut yang diterima dan syaratnya adalah:

- rating:
  - Wajib diisi
  - Harus berupa angka bulat (integer)
  - Nilai minimal: 1 dan maksimal: 5
  - Menunjukkan penilaian pengguna terhadap vendor
- review:
  - Opsional (boleh dikosongkan)
  - Harus berupa teks (string)
  - Maksimal 1000 karakter
  - Digunakan untuk menuliskan ulasan/komentar pengguna

g. Change Address

Pengguna dapat mengubah alamat utama pengiriman sebelum checkout. Diimplementasikan dengan validasi form dan update pada tabel alamat pengguna. Customer dapat mengganti alamat sebelum memesan di halaman search dan backend yang menghandlenya:

```
public function setAddress(SetAddressRequest $request)
    {
        $validated = $request->validated();
        $user = Auth::user();
        $addressId = $validated['address_id'];

        $address = Address::find($addressId);

        // Validate the address belongs to the logged-in user
        if ($address && $user && $address->userId === $user->userId) {
            session(['address_id' => $address->addressId]);
        }

        return redirect()->route('search');
    }
```

Jika input tervalidasi benar, maka address yang dipilih akan disimpan idnya di session. Validasi inputannya ada:

```
public function authorize(): bool
    {
        return Auth::check();
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
```

```
        return [
            'address_id' => [
                'required',
                'integer',
                'exists:addresses,addressId',
            ],
        ];
    }
```

address_id nya dicek harus berupa integer dan exist dalam database.

h. Cancel Upcoming Order

Pengguna dapat membatalkan pesanan yang belum diproses oleh vendor atau statusnya akan datang (upcoming). Status pesanan diubah ke “Canceled”, dan saldo dikembalikan otomatis ke WellPay jika sudah dibayar dengan WellPay. Berikut adalah backend yang mengaturnya:

```
public function cancelOrder(string $id)
    {
        /**
         * @var User | Authenticatable $user
         */
        $user = Auth::user();
        $order = Order::findOrFail($id);

        if($order->userId != $user->userId) {
            return redirect()->back()->with('error', 'Invalid order to cancel!');
        }

        if($order->payment->paymentMethod->name === 'WellPay') {
            $user->wellpay += $order->totalPrice;
            $user->save();
        }

        $order->isCancelled = true;
        $order->save();

        logActivity('Successfullyy', 'Cancelled', "Order ". $order->orderId);
        return redirect()->back()->with('message', 'Success cancelling order!');
    }
```

Id order yang mau di cancel akan diotorisasi dahulu, jika itu bukan order dari customer tersebut maka akan dikembalikan error message, jika benar maka dicek metode pembayarannya dan jika WellPay, maka WellPay customer akan dikembalikan. Kemudian status pesanan dijadikan cancel.

i. Top-Up WellPay

Fitur ini memungkinkan pengguna untuk mengisi saldo dompet digital mereka (WellPay) dengan jumlah tertentu, setelah memverifikasi kata sandi akun. Proses ini melibatkan validasi jumlah top-up, autentikasi pengguna, serta pembatasan saldo maksimum.

1. TopUpWellPayRequest.php:

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;

class TopUpWellPayRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check();
    }

    /**
```

```
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'amount' => 'required|integer|min:1000|max:20000000',
            'password' => 'required|string',
        ];
    }

    public function messages(): array
    {
        return [
            'amount.required' => 'Top-up amount is required.',
            'amount.integer' => 'Top-up amount must be a number.',
            'amount.min' => 'The minimum top-up amount is Rp 1.000.',
            'amount.max' => 'The maximum top-up amount is Rp 20.000.000.',
            'password.required' => 'Please enter your password.',
        ];
    }
}
```

Atribut yang diterima:

- amount (int): Jumlah saldo yang ingin di-top-up.
- password (string): Password akun pengguna, digunakan untuk verifikasi.

Validasi:

- amount:
  - required: Wajib diisi.
  - integer: Harus berupa bilangan bulat.
  - min:1000: Minimal Rp 1.000.
  - max:20000000: Maksimal Rp 20.000.000.
- password:
  - required: Wajib diisi.
  - string: Harus berupa teks.

2. UserController.php:

```
    public function topUpWellPay(TopUpWellPayRequest $request)
    {
        if (!Auth::check()) {
            return response()->json(['message' => 'Unauthorized. Please login first.'], 401);
        }

        $user = User::find(Auth::id());

        try {
            $amount = $request->input('amount');
            $password = $request->input('password');

            if (!Hash::check($password, $user->password)) {
                throw ValidationException::withMessages([
                    'password' => ['Incorrect password.'],
                ]);
            }

            $newBalance = $user->wellpay + $amount;
            $maxAllowedBalance = 1000000000;

            if ($newBalance > $maxAllowedBalance) {
                                logActivity('Failed', 'top-up', 'WellPay, Error : Balance cannot exceed Rp ' .
number_format($maxAllowedBalance, 0, ',', '.') . '.');
                                return response()->json(['message' => 'Your balance cannot exceed Rp ' .
number_format($maxAllowedBalance, 0, ',', '.') . '.'], 400);
            }

            $user->wellpay = $newBalance;
            $user->save();
```

```
            logActivity('successfully', 'top-up', 'WellPay');

            $locale = App::getLocale();
            $prefix = $locale === 'id' ? 'Isi saldo Rp ' : 'Top-up of Rp ';
            $sufix = $locale === 'id' ? ' berhasil' : ' success';

            return response()->json([
                'message' => $prefix . number_format($amount, 0, ',', '.') . $sufix . '!',
                'new_balance' => $newBalance,
            ], 200);
        } catch (ValidationException $e) {
            logActivity('Failed', 'top-up', 'WellPay, Error : ' . $e->getMessage());
            return response()->json([
                'message' => 'Validation Error',
                'errors' => $e->errors()
            ], 422);
        } catch (\Exception $e) {
            logActivity('Failed', 'top-up', 'WellPay, Error : ' . $e->getMessage());
            return response()->json(['message' => 'An error occurred: ' . $e->getMessage()], 500);
        }
    }
```

- topUpWellPay(TopUpWellPayRequest $request):
  - ➢ Cek Autentikasi
    - Memastikan pengguna sudah login.
    - Jika belum: return JSON 401 Unauthorized.
  - ➢ Ambil Data User
    - Mengambil data user berdasarkan ID login aktif.
  - ➢ Ambil dan Verifikasi Input
    - Ambil amount dan password dari request.
    - Cek apakah password cocok dengan password akun di database. Jika tidak cocok, throw ValidationException.
  - ➢ Cek Batas Maksimum Saldo
    - Maksimum saldo WellPay yang diperbolehkan: Rp 1.000.000.000.
    - Jika hasil top-up melebihi batas: Log aktivitas gagal, Return JSON 400 Bad Request.
  - ➢ Update Saldo dan Simpan
    - Tambahkan jumlah top-up ke saldo saat ini.
    - Simpan ke database.
  - ➢ Log Aktivitas Berhasil
    - Catat log: status successfully, tipe top-up, keterangan WellPay.
  - ➢ Respon Berhasil
    - Format pesan berdasarkan bahasa lokal (ID/EN).
    - Kirim JSON 200 OK dengan: Pesan top-up sukses dan nilai saldo terbaru.
  - ➢ Penanganan Error
    - Jika ValidationException: Log aktivitas gagal dan return JSON 422 dengan pesan dan daftar error.
    - Jika exception lain: Log aktivitas gagal dan return JSON 500 dengan pesan error umum.

j. Manage Address

Pengguna dapat menambah, mengedit, dan menghapus beberapa alamat pengiriman. Pengguna juga dapat menentukan satu alamat sebagai default, dan melihat daftar alamat melalui tampilan ManageAddress.

1. **AddressDefaultRequest.php:**

```
<?php
```

```php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\Rule;

class AddressDefaultRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check();
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        $user = Auth::user();
        $loggedInUserId = $user->userId;

        return [
            'address_id' => [
                'required',
                'numeric',
                Rule::exists('addresses', 'addressId')->where(function ($query) use ($loggedInUserId) {
                    $query->where('userId', $loggedInUserId);
                }),
            ],
        ];
    }
}
```

Atribut yang diterima:

- address_id (int): ID alamat yang ingin dijadikan default.

Validasi:

- address_id:
  - required, numeric
  - exists di tabel addresses dengan syarat userId sesuai user login untuk menjamin user hanya bisa set alamat miliknya.

2. AddressStoreRequest.php:

```php
<?php

namespace App\Http\Requests;

use Illuminate\Contracts\Validation\Validator;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Http\Exceptions\HttpResponseException;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\App;


class AddressStoreRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check();
    }

    /**
```

```
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'provinsi_name' => [
                'required',
                'string',
                'max:255',
            ],
            'kota_name' => [
                'required',
                'string',
                'max:255',
            ],
            'kecamatan_name' => [
                'required',
                'string',
                'max:255',
            ],
            'kelurahan_name' => [
                'required',
                'string',
                'max:255',
            ],
            'jalan' => [
                'required',
                'string',
                'max:255',
            ],
            'kode_pos' => [
                'required',
                'string',
                'digits:5',
            ],
            'notes' => [
                'nullable',
                'string',
                'max:255',
            ],
            'recipient_name' => [
                'required',
                'string',
                'max:100',
            ],
            'recipient_phone' => [
                'required',
                'string',
                'min:10',
                'max:15',
                'regex:/^[0-9]+$/',
            ],
        ];
    }
    public function messages()
{
    $locale = App::getLocale();

    return [
        // provinsi_name
        'provinsi_name.required' => $locale === 'id' ? 'Nama provinsi wajib diisi.' : 'Province name is required.',
         'provinsi_name.string' => $locale === 'id' ? 'Nama provinsi harus berupa teks.' : 'Province name must be a
string.',
         'provinsi_name.max' => $locale === 'id' ? 'Nama provinsi tidak boleh lebih dari 255 karakter.' : 'Province
name must not be more than 255 characters.',

        // kota_name
        'kota_name.required' => $locale === 'id' ? 'Nama kota wajib diisi.' : 'City name is required.',
        'kota_name.string' => $locale === 'id' ? 'Nama kota harus berupa teks.' : 'City name must be a string.',
         'kota_name.max' => $locale === 'id' ? 'Nama kota tidak boleh lebih dari 255 karakter.' : 'City name must
not be more than 255 characters.',

        // kecamatan_name
```

```
            'kecamatan_name.required' => $locale === 'id' ? 'Nama kecamatan wajib diisi.' : 'District name is required.',
            'kecamatan_name.string' => $locale === 'id' ? 'Nama kecamatan harus berupa teks.' : 'District name must be a string.',
            'kecamatan_name.max' => $locale === 'id' ? 'Nama kecamatan tidak boleh lebih dari 255 karakter.' : 'District name must not be more than 255 characters.',

            // kelurahan_name
            'kelurahan_name.required' => $locale === 'id' ? 'Nama kelurahan wajib diisi.' : 'Sub-district name is required.',
            'kelurahan_name.string' => $locale === 'id' ? 'Nama kelurahan harus berupa teks.' : 'Sub-district name must be a string.',
            'kelurahan_name.max' => $locale === 'id' ? 'Nama kelurahan tidak boleh lebih dari 255 karakter.' : 'Sub-district name must not be more than 255 characters.',

            // jalan
            'jalan.required' => $locale === 'id' ? 'Nama jalan wajib diisi.' : 'Street is required.',
            'jalan.string' => $locale === 'id' ? 'Nama jalan harus berupa teks.' : 'Street must be a string.',
            'jalan.max' => $locale === 'id' ? 'Nama jalan tidak boleh lebih dari 255 karakter.' : 'Street must not be more than 255 characters.',

            // kode_pos
            'kode_pos.required' => $locale === 'id' ? 'Kode pos wajib diisi.' : 'Postal code is required.',
            'kode_pos.string' => $locale === 'id' ? 'Kode pos harus berupa teks.' : 'Postal code must be a string.',
            'kode_pos.digits' => $locale === 'id' ? 'Kode pos harus terdiri dari 5 digit.' : 'Postal code must be 5 digits.',

            // notes
            'notes.string' => $locale === 'id' ? 'Catatan harus berupa teks.' : 'Notes must be a string.',
            'notes.max' => $locale === 'id' ? 'Catatan tidak boleh lebih dari 255 karakter.' : 'Notes must not be more than 255 characters.',

            // recipient_name
            'recipient_name.required' => $locale === 'id' ? 'Nama penerima wajib diisi.' : 'Recipient name is required.',
            'recipient_name.string' => $locale === 'id' ? 'Nama penerima harus berupa teks.' : 'Recipient name must be a string.',
            'recipient_name.max' => $locale === 'id' ? 'Nama penerima tidak boleh lebih dari 100 karakter.' : 'Recipient name must not be more than 100 characters.',

            // recipient_phone
            'recipient_phone.required' => $locale === 'id' ? 'Nomor telepon penerima wajib diisi.' : 'Recipient phone is required.',
            'recipient_phone.string' => $locale === 'id' ? 'Nomor telepon penerima harus berupa teks.' : 'Recipient phone must be a string.',
            'recipient_phone.min' => $locale === 'id' ? 'Nomor telepon penerima minimal 10 digit.' : 'Recipient phone must be at least 10 digits.',
            'recipient_phone.max' => $locale === 'id' ? 'Nomor telepon penerima maksimal 15 digit.' : 'Recipient phone must not be more than 15 digits.',
            'recipient_phone.regex' => $locale === 'id' ? 'Nomor telepon penerima hanya boleh berisi angka.' : 'Recipient phone must contain only numbers.',
        ];
    }


    protected function failedValidation(Validator $validator)
    {
        logActivity('Failed', 'Add', "Address due to validation errors : " . implode($validator->errors()->all()));

        throw new HttpResponseException(
            redirect()->back()
                ->withErrors($validator)
                ->withInput()
        );
    }
}
```

Atribut yang diterima:

- provinsi_name, kota_name, kecamatan_name, kelurahan_name: Nama wilayah (ditampilkan di form, tapi disimpan berdasarkan ID).
- jalan, kode_pos, notes: detail alamat.
- recipient_name, recipient_phone: informasi penerima.

Validasi:

- Semua atribut *_name: required, string, max:255
- jalan: required, string, max:255
- kode_pos: required, string, digits:5
- notes: nullable, string, max:255
- recipient_name: required, string, max:100
- recipient_phone: required, string, min:10, max:15, hanya angka (regex:/^[0-9]+$/)

Terdapat juga custom messages yang mendukung bahasa Indonesia dan Inggris, dan setiap rule memiliki pesan lokal berdasarkan App::getLocale(). Selain itu, terdapat juga behavior jika validasi gagal, yaitu redirect kembali dengan input sebelumnya dan log aktivitas: "Failed Add Address due to validation errors...".

3. AddressController.php:

```
    public function setDefaultAddress(AddressDefaultRequest $request)
    {
        $user = Auth::user();
        $loggedInUserId = $user->userId;

        $requestedAddressId = $request->input('address_id');

        try {
            Address::where('userId', $loggedInUserId)
                ->where('is_default', true)
                ->update(['is_default' => false]);

            $newDefaultAddress = Address::where('userId', $loggedInUserId)
                ->where('addressId', $requestedAddressId)
                ->firstOrFail();
            $newDefaultAddress->is_default = true;
            $newDefaultAddress->save();

            return response()->json(['success' => true, 'message' => 'Alamat utama berhasil diatur.']);
        } catch (\Illuminate\Database\Eloquent\ModelNotFoundException $e) {
                return response()->json(['success' => false, 'message' => 'Alamat tidak ditemukan atau bukan milik
Anda.'], 404);
        } catch (\Exception $e) {
                    return response()->json(['success' => false, 'message' => 'Terjadi kesalahan internal: ' .
$e->getMessage()], 500);
        }
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(AddressStoreRequest $request)
    {
        $newAddress = new Address();
        $newAddress->userId = Auth::id();

        $newAddress->provinsi = Province::find($request->provinsi_id)->name;
        $newAddress->kota = City::find($request->kota_id)->name;
        $newAddress->kecamatan = District::find($request->kecamatan_id)->name;
        $newAddress->kelurahan = Village::find($request->kelurahan_id)->name;

        $newAddress->jalan = $request->jalan;
        $newAddress->kode_pos = $request->kode_pos;
        $newAddress->notes = $request->notes;
        $newAddress->recipient_name = $request->recipient_name;
        $newAddress->recipient_phone = $request->recipient_phone;
        $newAddress->is_default = 0;

        $newAddress->save();

        logActivity('Successfully', 'Added', "New Address");
        return redirect('/manage-address')->with('success', 'Alamat berhasil ditambahkan.');
    }

    /**
     * Show the form for editing the specified resource.
```

```
     */
    public function edit(Address $address)
    {
        if (Auth::id() !== $address->userId) {
            return redirect()->route('manage-address')->with('error', 'Unauthorized action.');
        }

        return view('editAddress', compact('address'));
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(AddressStoreRequest $request, Address $address)
    {
        if (Auth::id() !== $address->userId) {
            abort(403, 'Unauthorized action.');
        }

        $address->provinsi = Province::find($request->provinsi_id)->name;
        $address->kota = City::find($request->kota_id)->name;
        $address->kecamatan = District::find($request->kecamatan_id)->name;
        $address->kelurahan = Village::find($request->kelurahan_id)->name;

        $address->jalan = $request->jalan;
        $address->kode_pos = $request->kode_pos;
        $address->notes = $request->notes;
        $address->recipient_name = $request->recipient_name;
        $address->recipient_phone = $request->recipient_phone;

        $address->save();

        return redirect('/manage-address')->with('update_success', 'Alamat berhasil diperbarui');
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Address $address)
    {
        if (Auth::id() !== $address->userId) {
            abort(403, 'Unauthorized action.');
        }

        if ($address->is_default) {
                return redirect('/manage-address')->with('error', 'Tidak dapat menghapus alamat utama. Silakan atur
alamat lain sebagai utama terlebih dahulu.');
        }

        $address->delete();

        return redirect('/manage-address')->with('delete_success', 'Alamat berhasil dihapus.');
    }
```

- setDefaultAddress(AddressDefaultRequest $request):
  - ➢ Ambil ID user & ID alamat yang ingin dijadikan default.
  - ➢ Langkah:
    - Set semua alamat user is_default = false.
    - Cari alamat berdasarkan addressId & userId. Jika tidak ditemukan, return 404.
    - Set is_default = true dan simpan.
  - ➢ Return JSON sukses atau error sesuai hasil.
- store(AddressStoreRequest $request):
  - ➢ Simpan alamat baru ke database:
    - Ambil data provinsi/kota/kecamatan/kelurahan dari ID dan simpan nama-nya.
  - ➢ Set is_default = 0 (alamat bukan default saat ditambahkan).
  - ➢ Simpan alamat dan log aktivitas sukses.
  - ➢ Redirect ke manage-address dengan notifikasi sukses.

- edit(Address $address):
    - ➢ Cek apakah alamat milik user login:
        - ○ Jika tidak, redirect dengan pesan error.
    - ➢ Tampilkan halaman editAddress dengan data alamat.
- update(AddressStoreRequest $request, Address $address):
    - ➢ Validasi ulang.
    - ➢ Cek kepemilikan alamat. Jika bukan milik user, return 403.
    - ➢ Update seluruh field alamat dengan data baru.
    - ➢ Simpan & redirect ke manage-address dengan notifikasi sukses.
- destory(Address $address):
    - ➢ Cek kepemilikan alamat. Jika bukan milik user, return 403.
    - ➢ Jika alamat adalah default, maka gagal hapus dan minta user ubah default terlebih dahulu.
    - ➢ Jika bukan default, hapus alamat dan redirect dengan notifikasi berhasil.

k. Register

Pengguna baru dapat mendaftar dengan email dan password. Data disimpan setelah validasi, dan pengguna akan diarahkan ke halaman OTP untuk memverifikasi email mereka. Berikut adalah backend yang mengatur register:

```
public function store(RegisteredUserStoreRequest $request, String $role)
    {
        $attrs = $request->validated();

        $attrs['role'] = Str::ucfirst($role);
        $user = User::create($attrs);

        $user->locale = App::currentLocale();
        $user->save();


        $otp = rand(100000, 999999);
        $email = $attrs['email'];

        $user = User::where('email', $email)->first();
        $user->update([
            'otp' => $otp,
            'otp_expires_at' => Carbon::now()->addMinutes(3),
        ]);

        Mail::send('emails.otp', ['otp' => $otp], function ($message) use ($email){
            $message->to($email)->subject('Your OTP');
        });

        session(['email' => $user->email, 'remember' => false]);
        return redirect()->route('auth.verify');
    }
```

Pada fungsi ini, digenerate OTP lalu dikirimkan ke email ayng diinput user. Sebelumnya dipastikan email dan password telah divalidasi dahulu:

```
public function rules(): array
    {
        return [
            'name' => ['required'],
            'email' => ['required', 'email', 'unique:users'],
            'password' => ['required', Password::min(8)->letters()->numbers()->uncompromised(), 'confirmed'],
            'password_confirmation' => ['required']
        ];
    }
```

input yang diterima dan validasinya:

- name: wajib diisi
- email: wajib diisi, harus email dan unique

- password: wajib diiisi, Password minimal 8 huruf dengan kombinasi alfanumerik serta tidak boleh mudah ditebak (uncompromised) serta password konfirmasi harus sama.
- password_confirmation: harus diisi dan sama dengan password.

l. Manage Account Profile

Pengguna bisa mengubah nama, foto, dan detail lainnya. Data disimpan setelah validasi dan pembaruan dilakukan secara real-time. Berikut adalah backend yang mengatur update profile:

```php
public function updateProfile(ProfileRequest $request)
    {
        $user = Auth::user();
        $userId = $user->userId;

        $updated_user = User::find($userId);



        $updated_user->name = $request->nameInput;

        if($user->name != $request->nameInput) {
            // logActivity('Successfully', 'Updated', "Profile to {$updated_user->name}");
            logActivity('Successfully', 'Updated', "Profile to :  {$updated_user->name}");
        }

        if ($request->filled('dateOfBirth')) {
            $updated_user->dateOfBirth = $request->input('dateOfBirth');
        }

        if ($request->hasFile('profilePicInput')) {
            $file = $request->file('profilePicInput');
            $filename = time() . '.' . $file->getClientOriginalExtension();
            $file->move(public_path('storage/profiles'), $filename);
            $updated_user->profilePath = 'storage/profiles/'.$filename;
            logActivity('Successfully', 'Updated', "Profile picture {$updated_user->name}");
        }


        $updated_user->genderMale = ($request->gender === 'male') ? 1 : 0;

        $updated_user->save();



        return redirect()->route('manage-profile')->with('success', 'Profile updated successfully!');
    }
```

input yang diberikan juga divalidasi dahulu seperti berikut:

```php
public function authorize(): bool
    {
        return Auth::check();
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'nameInput' => [
                'required',
                'string',
                'max:255',
                'not_regex:/<[^>]*>/'
            ],

            'dateOfBirth' => 'nullable|date|before:today',
            'gender' => 'required|in:male,female',
            'profilePicInput' => 'nullable|image|mimes:jpg,jpeg,png',
            'nameInput.not_regex' => 'Name must not contain HTML or script tags.',
```

```
        ];
    }
```

Berikut adalah inputan yang diterima dan validasinya:

- nameInput:
  - Wajib diisi
  - Harus berupa teks (string)
  - Maksimal 255 karakter
  - Tidak boleh mengandung tag HTML atau script (< >)
  - Pesan error khusus: "Name must not contain HTML or script tags."
- dateOfBirth
  - Opsional (boleh kosong)
  - Harus berupa tanggal yang valid
  - Tanggal harus lebih awal dari hari ini (tidak boleh tanggal sekarang atau masa depan)
- gender
  - Wajib diisi
  - Harus bernilai male atau female saja
- profilePicInput
  - Opsional
  - Jika diisi, harus berupa gambar dengan format jpg, jpeg, atau png

Tidak hanya mengubah profil, pengguna juga dapat mengaktifkan 2FA jika diinginkan untuk meningkatkan keamanan akunnya. Berikut adalah backend yang mengatur aktivasi ini:

```
public function index()
    {
        $userId = Auth::user()->userId;
        $user = User::find($userId);
        $current_2fa_status = $user->enabled_2fa;
        $user->enabled_2fa = !$current_2fa_status;
        $user->save();

        return redirect()->back();
    }
```

D. Catering Vendor

a. Register

Fitur Register Vendor bertujuan untuk memungkinkan pengguna baru mendaftar sebagai vendor secara aman dan valid dengan melakukan verifikasi data melalui form, menyimpan informasi ke dalam database, serta mengirimkan OTP ke email sebagai langkah autentikasi, sehingga hanya vendor yang terverifikasi yang dapat mengakses dan menggunakan sistem.

RegisterUserController.php

```
public function store(RegisteredUserStoreRequest $request, String $role)
    {
        $attrs = $request->validated();

        $attrs['role'] = Str::ucfirst($role);
        $user = User::create($attrs);

        $user->locale = App::currentLocale();
        $user->save();



        $otp = rand(100000, 999999);
        $email = $attrs['email'];

        $user = User::where('email', $email)->first();
        $user->update([
```

```
            'otp' => $otp,
            'otp_expires_at' => Carbon::now()->addMinutes(3),
        ]);

        Mail::send('emails.otp', ['otp' => $otp], function ($message) use ($email){
            $message->to($email)->subject('Your OTP');
        });

        session(['email' => $user->email, 'remember' => false]);
        return redirect()->route('auth.verify');
    }

}
```

Alur Register Vendor

1. Akses Halaman Register
   - Endpoint: GET /register/vendor
   - Method create($role) menampilkan view auth.vendorRegister jika $role == 'vendor'.
2. Submit Form Registrasi
   - Endpoint: POST /register/vendor
   - Method store(Request $request, $role) menangani data yang dikirim dari form.
3. Validasi Data Form
   - name harus diisi.
   - email harus diisi, format email yang valid, dan unik di tabel users.
   - password harus diisi, minimal 8 karakter, dan cocok dengan password_confirmation.
   - password_confirmation harus diisi.
4. Membuat Akun User Baru
   - Role ditambahkan berdasarkan $role dari URL dan dikapitalisasi (Vendor).
   - Data user disimpan ke tabel users.
   - Locale disimpan berdasarkan bahasa saat ini (App::currentLocale()).
5. Generate OTP
   - OTP 6 digit di-generate secara random.
   - OTP disimpan ke kolom otp di tabel users.
   - Batas waktu OTP (otp_expires_at) adalah 3 menit dari sekarang.
6. Kirim OTP via Email
   - Email dikirim ke alamat email yang didaftarkan menggunakan view emails.otp.
7. Logging Aktivitas
   - Fungsi logActivity() mencatat aktivitas registrasi user sebagai log audit.
8. Redirect ke Halaman Verifikasi OTP
   - Data email dan status "remember" disimpan ke session.
   - Pengguna diarahkan ke route('auth.verify').

b. Manage Account Profile

Fitur ini bertujuan untuk memungkinkan vendor mengelola dan memperbarui profil mereka secara mandiri, termasuk informasi seperti nama, nomor telepon, jam operasional pengantaran (sarapan, makan siang, dan makan malam), serta unggahan foto profil atau logo. Dengan fitur ini, vendor dapat menjaga akurasi data yang ditampilkan kepada pelanggan, meningkatkan profesionalitas, serta memberikan fleksibilitas dalam menyesuaikan layanan yang mereka tawarkan. Selain itu, sistem juga mencatat setiap perubahan sebagai log aktivitas untuk keperluan audit dan pemantauan internal.

VendorController.php

```
public function manageProfile()
    {
        $user = Auth::user();
        $vendor = Vendor::where('userId', $user->userId)->first();
```

```
        $breakfast_start = $breakfast_end = null;
        $lunch_start = $lunch_end = null;
        $dinner_start = $dinner_end = null;

        if ($vendor->breakfast_delivery) {
            [$breakfast_start, $breakfast_end] = explode('-', $vendor->breakfast_delivery);
        }

        if ($vendor->lunch_delivery) {
            [$lunch_start, $lunch_end] = explode('-', $vendor->lunch_delivery);
        }

        if ($vendor->dinner_delivery) {
            [$dinner_start, $dinner_end] = explode('-', $vendor->dinner_delivery);
        }

        $address = $vendor->provinsi . ' ' . $vendor->kota . ' ' . $vendor->kecamatan . ' ' . $vendor->kelurahan .
' ' . $vendor->jalan . ' ' . $vendor->kode_pos;
        // logActivity('Successfully', 'Visited', 'Manage Profile Vendor Page');

        return view('manage-profile-vendors', compact(
            'user',
            'vendor',
            'breakfast_start',
            'breakfast_end',
            'lunch_start',
            'lunch_end',
            'dinner_start',
            'dinner_end',
            'address',
        ));
    }


    public function updateProfile(UpdateProfileVendorRequest $request)
    {

        // dd($request);
        $user = Auth::user();
        $userId = $user->userId;
        $vendor = Vendor::where('userId', $userId)->first();

        if ($vendor->name != $request->nameInput)
        {
            logActivity('Successfully', 'Updated', 'Vendor Name to ' . $request->nameInput);
        }

        $vendor->name = $request->nameInput;
        $vendor->phone_number = $request->telpInput;

        $vendor->breakfast_delivery = $request->breakfast_time_start . '-' . $request->breakfast_time_end;
        $vendor->lunch_delivery = $request->lunch_time_start . '-' . $request->lunch_time_end;
        $vendor->dinner_delivery = $request->dinner_time_start . '-' . $request->dinner_time_end;


        if ($request->hasFile('profilePicInput')) {
            $file = $request->file('profilePicInput');
            $filename = time() . '.' . $file->getClientOriginalExtension();
            $file->move(public_path('asset/vendorLogo'), $filename);
            $vendor->logo = $filename;
            logActivity('Successfully', 'Added', 'Profile pict inManage Profile Vendor Page');
        }

        $vendor->save();

        logActivity('Successfully', 'Updated', 'Manage Profile Vendor Page');
        return redirect()->route('manage-profile-vendor')->with('success', 'Profile updated successfully!');
    }
```

Alur:

1. Akses Halaman Manage Profile
    - Saat user membuka halaman profil vendor (manageProfile()), sistem akan:

- Mengambil data user yang sedang login.
- Mengambil data vendor yang terhubung dengan user.
- Mengekstrak waktu pengantaran untuk sarapan, makan siang, dan makan malam dari field breakfast_delivery, lunch_delivery, dan dinner_delivery dalam format start-end.
- Menggabungkan alamat lengkap dari beberapa field: provinsi, kota, kecamatan, kelurahan, jalan, dan kode_pos.
- Melempar semua data ke view manage-profile-vendors untuk ditampilkan di form edit profil.

2. Update Profile
   - Saat user menyimpan perubahan di form, maka fungsi updateProfile() akan dijalankan:
   - Validasi dilakukan melalui UpdateProfileVendorRequest.

```php
<?php

namespace App\Http\Requests;

use App\Enums\UserRole;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Contracts\Validation\Validator;
use Illuminate\Http\Exceptions\HttpResponseException;
use Illuminate\Support\Facades\App;

class UpdateProfileVendorRequest extends FormRequest
{
    public function authorize(): bool
    {
        return Auth::check();
    }

    public function rules(): array
    {
        $vendorId = optional(Auth::user()->vendor)->vendorId;

        return [
            'nameInput' => [
                'bail',
                'required',
                'string',
                'max:255',
                'unique:vendors,name,' . $vendorId . ',vendorId',
                'not_regex:/<[^>]*script.*?>.*?<\/[^>]*script.*?>/i',
                'not_regex:/<[^>]+>/i',
            ],
            'telpInput' => 'bail|required|string|max:15|min:10|regex:/^[0-9]+$/',
            'profilePicInput' => 'nullable|image|mimes:jpg,jpeg,png',

            // Time fields
            'breakfast_time_start' => ['nullable', 'date_format:H:i'],
            'breakfast_time_end'   => ['nullable', 'date_format:H:i', 'after:breakfast_time_start'],

            'lunch_time_start' => ['nullable', 'date_format:H:i'],
            'lunch_time_end'   => ['nullable', 'date_format:H:i', 'after:lunch_time_start'],

            'dinner_time_start' => ['nullable', 'date_format:H:i'],
            'dinner_time_end'   => ['nullable', 'date_format:H:i', 'after:dinner_time_start'],
        ];
    }

    public function messages(): array
    {
        $locale = App::getLocale();
        return [

            'nameInput.required' => $locale === 'id' ? 'Nama vendor harus diisi!' : 'Vendor name must be
filled!',
            'nameInput.max' => $locale === 'id' ? 'Nama vendor tidak boleh lebih dari 255 karakter.' :
'Vendor name may not be greater than 255 characters.',
```

```
            'nameInput.unique' => $locale === 'id' ? 'Nama vendor sudah digunakan!' : 'Vendor name is
already taken!',
            'nameInput.not_regex' => $locale === 'id' ? 'Tag HTML atau script tidak diperbolehkan dalam
nama vendor.' : 'HTML or script tags are not allowed in the vendor name.',

            'telpInput.required' => $locale === 'id' ? 'Nomor telepon harus diisi!' : 'Telp number must
be filled!',
            'telpInput.max' => $locale === 'id' ? 'Nomor telepon tidak boleh lebih dari 15 karakter.' :
'Telp number may not be greater than 15 characters.',
            'telpInput.min' => $locale === 'id' ? 'Nomor telepon minimal terdiri dari 10 karakter.' :
'Telp number must be at least 10 characters.',
            'telpInput.regex' => $locale === 'id' ? 'Nomor telepon harus berupa angka 0 - 9.' : 'Telp
number must be number between 0 - 9',

            'profilePicInput.image' => $locale === 'id' ? 'Foto profil harus berupa gambar.' : 'Profile
picture must be an image.',
            'profilePicInput.mimes' => $locale === 'id' ? 'Foto profil harus bertipe: jpg, jpeg, png.' :
'Profile picture must be a file of type: jpg, jpeg, png.',

            'breakfast_time_start.date_format' => $locale === 'id' ? 'Waktu mulai sarapan harus dalam
format HH:MM.' : 'Breakfast start time must be in the format HH:MM.',
            'breakfast_time_end.date_format' => $locale === 'id' ? 'Waktu selesai sarapan harus dalam
format HH:MM.' : 'Breakfast end time must be in the format HH:MM.',
            'breakfast_time_end.after' => $locale === 'id' ? 'Waktu selesai sarapan harus setelah waktu
mulai.' : 'Breakfast end time must be after the start time.',

            'lunch_time_start.date_format' => $locale === 'id' ? 'Waktu mulai makan siang harus dalam
format HH:MM.' : 'Lunch start time must be in the format HH:MM.',
            'lunch_time_end.date_format' => $locale === 'id' ? 'Waktu selesai makan siang harus dalam
format HH:MM.' : 'Lunch end time must be in the format HH:MM.',
            'lunch_time_end.after' => $locale === 'id' ? 'Waktu selesai makan siang harus setelah waktu
mulai.' : 'Lunch end time must be after the start time.',

            'dinner_time_start.date_format' => $locale === 'id' ? 'Waktu mulai makan malam harus dalam
format HH:MM.' : 'Dinner start time must be in the format HH:MM.',
            'dinner_time_end.date_format' => $locale === 'id' ? 'Waktu selesai makan malam harus dalam
format HH:MM.' : 'Dinner end time must be in the format HH:MM.',
            'dinner_time_end.after' => $locale === 'id' ? 'Waktu selesai makan malam harus setelah waktu
mulai.' : 'Dinner end time must be after the start time.',


        ];
    }

    protected function withValidator(\Illuminate\Validation\Validator $validator)
    {
        $validator->after(function ($validator) {
            // Breakfast time must be between 07:00 - 10:00
            if ($this->breakfast_time_start && ($this->breakfast_time_start < '07:00' ||
$this->breakfast_time_start > '10:00')) {
                $validator->errors()->add('breakfast_time_start', 'Breakfast start time must be between
07:00 and 10:00.');
            }
            if ($this->breakfast_time_end && ($this->breakfast_time_end < '07:00' ||
$this->breakfast_time_end > '10:00')) {
                $validator->errors()->add('breakfast_time_end', 'Breakfast end time must be between 07:00
and 10:00.');
            }

            // Lunch time must be between 10:00 - 13:00
            if ($this->lunch_time_start && ($this->lunch_time_start < '10:00' || $this->lunch_time_start
> '13:00')) {
                $validator->errors()->add('lunch_time_start', 'Lunch start time must be between 10:00 and
13:00.');
            }
            if ($this->lunch_time_end && ($this->lunch_time_end < '10:00' || $this->lunch_time_end >
'13:00')) {
                $validator->errors()->add('lunch_time_end', 'Lunch end time must be between 10:00 and
13:00.');
            }

            // Dinner time must be between 14:00 - 21:00
            if ($this->dinner_time_start && ($this->dinner_time_start < '14:00' ||
$this->dinner_time_start > '21:00')) {
```

```
                $validator->errors()->add('dinner_time_start', 'Dinner start time must be between 14:00
and 21:00.');
            }
            if ($this->dinner_time_end && ($this->dinner_time_end < '14:00' || $this->dinner_time_end >
'21:00')) {
                $validator->errors()->add('dinner_time_end', 'Dinner end time must be between 14:00 and
21:00.');
            }
        });
    }

    protected function failedValidation(Validator $validator)
    {
        logActivity('Failed', 'Updated', "Vendor Profile, Validation Errors: " . implode(' | ',
$validator->errors()->all()));

        throw new HttpResponseException(
            redirect()->back()
                ->withErrors($validator)
                ->withInput()
        );
    }
}
```

- Sistem mengambil ulang data user dan vendor berdasarkan userId.
- Jika ada perubahan pada nama vendor, maka akan dicatat di log aktivitas.
- Semua data form akan dimasukkan ke field yang sesuai:
    - Nama vendor (nameInput)
    - Nomor telepon (telpInput)
    - Jam pengantaran (diformat sebagai start-end)
- Jika user mengupload gambar baru:
    - Gambar akan disimpan ke direktori publik asset/vendorLogo dengan nama file timestamp.
    - Field logo diupdate dengan nama file baru.
    - Dicatat juga di log aktivitas.
- Semua perubahan disimpan dan diarahkan kembali ke halaman profil dengan notifikasi sukses.

Validasi Form Update Profile Vendor

1. nameInput
    - Wajib diisi (required)
    - Harus berupa string
    - Maksimal 255 karakter
2. telpInput
    - Wajib diisi
    - Harus berupa angka
    - Format sesuai nomor HP Indonesia, misalnya diawali dengan 08 (bisa gunakan regex)
3. breakfast_time_start dan breakfast_time_end
    - Boleh kosong (nullable)
    - Format waktu harus valid (H:i)
    - Jika salah satu diisi, maka keduanya harus diisi
    - breakfast_time_end harus lebih besar dari breakfast_time_start
4. lunch_time_start dan lunch_time_end
    - Boleh kosong
    - Format waktu harus valid (H:i)
    - Jika salah satu diisi, maka keduanya harus diisi
    - lunch_time_end harus lebih besar dari lunch_time_start
5. dinner_time_start dan dinner_time_end

- Boleh kosong
- Format waktu harus valid (H:i)
- Jika salah satu diisi, maka keduanya harus diisi
- dinner_time_end harus lebih besar dari dinner_time_start

6. profilePicInput
   - Boleh tidak diisi (optional)
   - Jika diisi, harus berupa file gambar (jpeg, jpg, png, atau webp)
   - Ukuran maksimal 2MB

c. Generate Statistics

Membantu vendor menganalisis performa penjualan mingguan secara visual dengan tampilan grafik garis. Statistik ini berguna untuk mengevaluasi pendapatan mingguan, mengidentifikasi tren naik/turun dalam bulan aktif, dan melihat efektivitas promosi/paket dalam periode tertentu.

Implementasi :

1. Frontend (Visualisasi Grafik Penjualan)

- Library: Chart.js digunakan untuk menampilkan grafik penjualan mingguan.
- Bentuk Grafik: Line chart (grafik garis).
- Responsif: Ya, grafik menyesuaikan ukuran tampilan perangkat.
- Warna titik & garis: Disesuaikan agar sesuai dengan tema brand (point hijau, garis hitam).

Struktur Data:

chartData di-inject dari backend (Blade template Laravel) dan berisi array penjualan dari minggu 1–4.

```
new Chart(ctx, {

  type: 'line',

  data: {

      labels: [

          locale.week_1,

          locale.week_2,

          locale.week_3,

          locale.week_4,

      ],

      datasets: [{

          data: chartData,

          borderColor: '#000',

          backgroundColor: 'transparent',

          pointBackgroundColor: 'rgba(0,128,0,1)',

          pointRadius: 6,

      }]
```

```
    },
    options: {
        animation: {
            duration: 1500,
            easing: 'easeOutQuart'
        },
        scales: {
            y: {
                beginAtZero: true,
                grid: {
                    display: false
                },
                ticks: {
                    color: 'rgba(0,128,0,.7)',
                    callback: v => 'Rp' + v.toLocaleString('id-ID')
                }
            },
            x: {
                grid: {
                    display: false
                },
                ticks: {
                    color: 'rgba(0,128,0,.7)'
                }
            }
        },
        plugins: {
            legend: {
                display: false
            }
        },
        responsive: true,
        maintainAspectRatio: false
```

```
        }

    });
```

2. Backend (OrderVendorController)

Fungsi utama:

Mengambil data total penjualan vendor untuk masing-masing minggu dan mengirimkannya ke view.

Validasi:

- Autentikasi vendor: hanya vendor yang login dapat mengakses statistik mereka.
- Validasi query order: hanya order dengan status 'completed' dan dalam rentang waktu minggu tertentu yang dihitung.
- Perhitungan otomatis per minggu: menggunakan Carbon (PHP DateTime Library) untuk menentukan range tanggal setiap minggu.

```
public function totalOrder()

    {

        $vendor = Auth::user()->vendor ?? Vendor::find(39);

        $vendorId = Auth::user()->vendor->vendorId ?? 39;

        $today = Carbon::today('Asia/Jakarta');



        $orders = Order::with([

            'orderItems.package',

            'deliveryStatuses' => fn($q) =>

                $q->whereDate('deliveryDate', $today),

        ])

            ->where('vendorId', $vendorId)

            ->where('isCancelled', 0)

            ->whereDate('startDate', '<=', $today)

            ->whereDate('endDate', '>=', $today)

            ->get();



        /* ---- bangun $slotCounts persis seperti kode lama ---- */

        $slotEnumToLabel = [

            TimeSlot::Morning->value => 'breakfast',

            TimeSlot::Afternoon->value => 'lunch',

            TimeSlot::Evening->value => 'dinner',

        ];
```

```
        $slotCounts = ['breakfast' => [], 'lunch' => [], 'dinner' => []];



        foreach ($orders as $order) {

            foreach ($order->orderItems as $item) {

                $slotLabel = $slotEnumToLabel[$item->packageTimeSlot] ?? null;

                if (!$slotLabel)

                    continue;



                $pkg = $item->package->name ?? 'Package';

                $slotCounts[$slotLabel][$pkg] =

                    ($slotCounts[$slotLabel][$pkg] ?? 0) + $item->quantity;

            }

        }

        foreach ($slotCounts as &$pkgs)

            ksort($pkgs);



        /* ----- Net-sales bulan berjalan ----- */

        $today = now('Asia/Jakarta');   // sekali saja

        $stats = Payment::join('orders', 'orders.orderId', '=', 'payments.orderId')

            ->where('orders.vendorId', $vendorId)

            ->whereYear('payments.paid_at', $today->year)

            ->whereMonth('payments.paid_at', $today->month)

            ->selectRaw("( WEEK(payments.paid_at,3) - WEEK(DATE_SUB(payments.paid_at, INTERVAL
DAYOFMONTH(payments.paid_at)-1 DAY),3) + 1 ) AS wk,

                    SUM(orders.totalPrice * 0.95) AS nett")

            ->groupBy('wk')

            ->pluck('nett', 'wk')

            ->toArray();



        /* normalisasi 4 minggu – pakai closure agar aman PHP 7.3 */

        $salesData = array_map(function ($w) use ($stats) {

            return (float) ($stats[$w] ?? 0);

        }, range(1, 4));
```

```
        $salesMonth = $today->format('F Y');

        $endMonth = $today->lastOfMonth();



        return view('cateringHomePage', compact('slotCounts', 'vendor', 'salesData', 'salesMonth',
'endMonth'));

    }
```

1. Validasi dan Default pada Vendor
    - Fungsi ini memeriksa apakah user yang sedang login memiliki entitas vendor.
    - Jika tidak, maka digunakan default fallback Vendor::find(39) dan vendorId = 39.
    - Tujuan validasi ini adalah untuk mencegah error ketika pengguna belum memiliki relasi vendor (misal admin atau testing account), sehingga data masih dapat ditampilkan untuk vendor default.
    - Jenis validasi: Fallback/null-check manual.
2. Validasi Tanggal Hari Ini
    - Validasi zona waktu dilakukan eksplisit untuk memastikan hasil yang konsisten pada seluruh perhitungan waktu (misal deliveryDate, startDate, endDate).
    - Menghindari bug yang disebabkan oleh server yang berada di zona waktu berbeda (misal UTC).
3. Validasi Status Pengiriman
    - Validasi ini memastikan hanya status pengiriman yang relevan dengan hari ini yang diambil.
    - Menghindari ambiguitas status dari hari-hari sebelumnya atau setelahnya.
4. Validasi Status Order
    - Ini adalah validasi untuk memastikan:
        - Order yang diambil adalah aktif (tidak dibatalkan),
        - dan masih berlaku (tanggal mulai sampai selesai mencakup hari ini).
    - Jenis validasi: Logika bisnis berbasis tanggal dan status.
5. Validasi dan Normalisasi Data Slot Waktu
    - Validasi melalui pemetaan konversi enum ke label.
    - Jika data packageTimeSlot tidak valid (misalnya null atau tidak cocok dengan enum), maka dilewati:
    - Ini mencegah error akibat data packageTimeSlot yang corrupt/tidak valid.
    - Jenis validasi: Pengamanan terhadap data input tak terduga.
6. Validasi Nama Paket
    - Jika package->name tidak tersedia, maka fallback ke 'Package'.
    - Mencegah error jika relasi package tidak ditemukan (misalnya data rusak).
    - Jenis validasi: Fallback property data relasi.
7. Validasi Perhitungan dan Format Bulan
    - Validasi waktu saat ini dilakukan ulang (meskipun bisa reuse dari sebelumnya).
    - Tujuannya agar semua proses berdasarkan waktu (mingguan & bulanan) berada pada zona waktu yang sama.
8. Validasi dan Transformasi Mingguan Statistik Penjualan

- Validasi dilakukan untuk membatasi statistik hanya pada bulan berjalan.
- Juga disertai perhitungan WEEK(...,3) agar minggu dimulai dari Senin (ISO-8601), serta penyesuaian minggu ke-1 dengan DATE_SUB(...).

9. Normalisasi Mingguan (4 Minggu)
    - Ini adalah bentuk validasi indeks array untuk memastikan 4 minggu selalu tersedia.
    - Jika minggu tertentu tidak memiliki data (null), akan dikonversi ke 0 (float(0.0)).
    - Mencegah error saat melakukan visualisasi grafik statistik.

d. View Today's Order

Fitur View Today's Catering Orders dirancang untuk membantu vendor katering melihat seluruh pesanan makanan yang dijadwalkan untuk hari ini secara real-time. Dengan fitur ini, vendor dapat secara langsung mengetahui jumlah pesanan, menu yang dipesan, jumlah porsi per waktu, waktu pengantaran, dan informasi pelanggan, sehingga proses produksi dan distribusi katering menjadi lebih efisien, tepat waktu, dan minim kesalahan.

Route

Route::get('/cateringHomePage', [OrderVendorController::class, 'totalOrder'])->name('vendor.home');

Controller: OrderVendorController.php

```
public function totalOrder()
{
    $vendor = Auth::user()->vendor ?? Vendor::find(39);
    $vendorId = Auth::user()->vendor->vendorId ?? 39;
    $today = Carbon::today('Asia/Jakarta');

    $orders = Order::with([
        'orderItems.package',
        'deliveryStatuses' => fn($q) =>
            $q->whereDate('deliveryDate', $today),
    ])
        ->where('vendorId', $vendorId)
        ->where('isCancelled', 0)
        ->whereDate('startDate', '<=', $today)
        ->whereDate('endDate', '>=', $today)
        ->get();

    $slotEnumToLabel = [
        TimeSlot::Morning->value => 'breakfast',
        TimeSlot::Afternoon->value => 'lunch',
        TimeSlot::Evening->value => 'dinner',
    ];

    $slotCounts = ['breakfast' => [], 'lunch' => [], 'dinner' => []];

    foreach ($orders as $order) {
        foreach ($order->orderItems as $item) {
            $slotLabel = $slotEnumToLabel[$item->packageTimeSlot] ?? null;
            if (!$slotLabel)
                continue;

            $pkg = $item->package->name ?? 'Package';
            $slotCounts[$slotLabel][$pkg] =
                ($slotCounts[$slotLabel][$pkg] ?? 0) + $item->quantity;
        }
    }
    foreach ($slotCounts as &$pkgs)
        ksort($pkgs);

    /* ----- Net-sales bulan berjalan ----- */
    $today = now('Asia/Jakarta');
    $stats = Payment::join('orders', 'orders.orderId', '=', 'payments.orderId')
        ->where('orders.vendorId', $vendorId)
```

```
            ->whereYear('payments.paid_at', $today->year)
            ->whereMonth('payments.paid_at', $today->month)
            ->selectRaw("( WEEK(payments.paid_at,3) - WEEK(DATE_SUB(payments.paid_at, INTERVAL
DAYOFMONTH(payments.paid_at)-1 DAY),3) + 1 ) AS wk,
                    SUM(orders.totalPrice * 0.95) AS nett")
            ->groupBy('wk')
            ->pluck('nett', 'wk')
            ->toArray();

        $salesData = array_map(function ($w) use ($stats) {
            return (float) ($stats[$w] ?? 0);
        }, range(1, 4));

        $salesMonth = $today->format('F Y');
        $endMonth = $today->lastOfMonth();

        return view('cateringHomePage', compact('slotCounts', 'vendor', 'salesData', 'salesMonth', 'endMonth'));
    }
```

Penjelasan:

- Carbon::today() digunakan untuk menetapkan filter tanggal hari ini.
- Menggunakan Auth::user()->vendorId agar hanya pesanan milik vendor terkait yang ditampilkan.
- Menggunakan with(['orderDetails.menuPackage']) untuk melakukan eager loading relasi agar efisien dan menghindari N+1 problem.
- groupBy('menuPackage.packageName') digunakan untuk mengelompokkan pesanan berdasarkan nama paket dan menjumlahkan quantity-nya.
- Semua hasil ini dikirim ke view catering.cateringhomePage.

Validasi dilakukan pada beberapa aspek penting:

1. Validasi Autentikasi Vendor:
    - Sistem memastikan bahwa hanya pengguna dengan status login dan role sebagai vendor yang dapat mengakses route ini.
    - Jika vendor belum login, akan langsung diarahkan ke halaman login disertai dengan pesan error.
2. Validasi Tanggal Order:
   Sistem hanya mengambil data pesanan yang memiliki tanggal pemesanan sama dengan tanggal saat ini. Menggunakan Carbon::now()->toDateString() untuk menghindari error zona waktu dan memastikan keakuratan tanggal.
3. Validasi Akses Data:
    - Hanya data pesanan yang berhubungan dengan vendor yang sedang login (berdasarkan vendorId) yang akan ditampilkan.
    - Tidak ada kemungkinan vendor melihat data vendor lain, menjaga privasi dan integritas data.

e. Export Sales Report

Fitur ini memungkinkan vendor untuk mengunduh laporan penjualan mereka dalam format Excel (.xlsx), berdasarkan rentang tanggal tertentu, agar bisa digunakan untuk keperluan laporan internal, audit, maupun pengambilan keputusan bisnis.

Endpoint Utama: SalesController::export_sales()

```
public function export_sales(FilterSalesRequest $request)
    {
        $user = Auth::user();
        $vendorId = $user->vendor->vendorId;
        $validated = $request->validated();

        $startDate = $validated['startDate'] ?? null;
        $endDate = $validated['endDate'] ?? null;

        // Call helper method to fetch filtered data
```

```
        [$orders, $totalSales] = $this->getVendorSalesRanged($vendorId, $startDate, $endDate);

        $start = Carbon::parse($startDate)->format('d M Y') ?? null;
        $end = Carbon::parse($endDate)->format('d M Y') ?? null;

        return Excel::download(new SalesExport($orders, $totalSales, $start, $end), "laporan_penjualan.xlsx");
       }
```

Fungsi ini:

- Menerima input rentang tanggal (startDate, endDate).
- Melakukan validasi menggunakan FilterSalesRequest.
- Mengambil data penjualan vendor (yang sudah dibayar) selama rentang waktu tersebut.
- Menyiapkan file Excel menggunakan class SalesExport.
- Mengunduh file dengan nama laporan_penjualan.xlsx.

```
<?php



namespace App\Http\Requests;



use App\Enums\UserRole;

use Illuminate\Foundation\Http\FormRequest;

use Illuminate\Support\Facades\Auth;

use Illuminate\Validation\Rule;



class FilterOrderHistoryRequest extends FormRequest

{

    /**

     * Determine if the user is authorized to make this request.

     */

    public function authorize(): bool

    {

        return Auth::check() && Auth::user()->role === UserRole::Customer;

    }



    /**

     * Get the validation rules that apply to the request.

     *

     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>

     */

    public function rules(): array
```

```
    {
        return [
            'status' => [
                'nullable',
                'string',
                Rule::in(['active', 'upcoming', 'cancelled', 'finished', 'all']),
            ],
            'query' => [
                'nullable',
                'string',
                'max:255',
            ]
        ];
    }
}
```

Validasi:

1. date
    - Tipe: nullable
        - Artinya, field ini opsional. Jika tidak dikirim oleh user, tidak akan menimbulkan error.
    - Validasi: date
        - Laravel akan memverifikasi bahwa input adalah format tanggal valid (Y-m-d, ISO8601, dll).
    - Validasi: before_or_equal:today
        - Memastikan bahwa tanggal yang dipilih tidak melewati hari ini.
        - Ini penting untuk menjaga agar vendor tidak bisa meminta data di masa depan (yang belum terjadi).
2. slot
    - Tipe: nullable
        - Slot juga opsional. Misalnya user hanya ingin lihat total semua slot, bisa dikosongkan.
    - Validasi: string
        - Data yang dikirim harus berupa string. Misalnya "lunch", "dinner", dll.
    - Validasi: Rule::in(['breakfast', 'lunch', 'dinner'])
        - Field ini hanya menerima nilai tertentu yang sudah ditentukan sistem.
        - Tujuannya adalah membatasi input agar tidak sembarangan, dan memastikan sistem backend hanya memproses slot yang dikenali.

Validasi Otorisasi

- Request ini hanya akan dijalankan jika user sudah login dan memiliki relasi vendor.
- Hal ini mencegah akses dari user yang tidak berhak (misalnya customer biasa) untuk melihat atau menghitung total order vendor.

Data penjualan diambil dari method privat:

```
private function getVendorSalesRanged($vendorId, $startDate = null, $endDate = null)

    {

        $orders = Order::where('vendorId', $vendorId)

            ->whereHas('payment', function ($query) use ($startDate, $endDate) {

                $query->whereNotNull('paid_at');



                if ($startDate) {

                    $query->whereDate('paid_at', '>=', $startDate);

                }



                if ($endDate) {

                    $query->whereDate('paid_at', '<=', $endDate);

                }

            })

            ->get();



        $orders->load(['user', 'orderItems.package', 'payment']);



        $totalSales = $orders->sum('totalPrice');



        foreach ($orders as $order) {

            $grouped = $order->orderItems

                ->groupBy(function ($item) {

                    return $item->packageId . '|' . strtolower($item->packageTimeSlot); // group by both packageId
& timeSlot

                })

                ->map(function ($items) {

                    $first = $items->first();

                    return [

                        'packageName' => $first->package->name,

                        'timeSlots' => ucfirst(strtolower($first->packageTimeSlot)),

                        'quantity' => $items->sum('quantity'),
```

```
                ];

            });


        $order->groupedPackages = $grouped->values();

    }




    return [$orders, $totalSales];

}
```

Data yang diambil:

- Orders dengan vendorId yang sesuai.
- Hanya order yang sudah dibayar (payment.paid_at tidak null).
- Filter berdasarkan rentang tanggal paid_at antara startDate dan endDate.
- Setiap order dimuat beserta:
    - Nama user.
    - Items dan nama paket.
    - Slot waktu paket.
    - Kuantitas masing-masing paket.

Struktur Output untuk Excel:

Class SalesExport menerima data berikut:

```
new SalesExport($orders, $totalSales, $start, $end)
```

Dengan:

- orders: Daftar lengkap order dalam rentang waktu.
- totalSales: Jumlah keseluruhan penjualan (dalam rupiah).
- start, end: Label rentang tanggal format d M Y.

File yang dihasilkan:

- Format: .xlsx
- Nama file: laporan_penjualan.xlsx
- Dapat langsung dibuka di Excel atau Google Sheets.

Alur Penggunaan:

1. Vendor memilih tanggal mulai dan tanggal akhir dari filter form.
2. Submit form (dengan validasi otomatis).
3. Backend mengambil data sesuai rentang, hanya order yang sudah dibayar.
4. Data dikemas ke dalam Excel via SalesExport.
5. File langsung terunduh ke browser.

f. Manage Package/Menu

Fitur *Manage Catering Package* dirancang untuk memudahkan vendor dalam mengelola daftar paket catering mereka yang tersedia di sistem. Melalui fitur ini, vendor dapat menambahkan paket baru (*store*), memperbarui informasi paket yang sudah ada

(*update*), menghapus paket yang tidak lagi disediakan (*destroy*), serta mengimpor banyak data paket sekaligus melalui file Excel (*import*). Semua proses ini berperan penting dalam menjaga ketersediaan informasi menu dan memastikan bahwa customer hanya melihat paket-paket terkini yang valid dan aktif.

PackageController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\ImportPackageRequest;
use App\Http\Requests\StorePackageRequest;
use App\Http\Requests\UpdatePackageRequest;
use Illuminate\Http\Request;
use App\Models\Package;
use App\Models\CuisineType;
use App\Models\PackageCuisine;
use Maatwebsite\Excel\Facades\Excel;
use App\Imports\PackagesImport;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\ValidationException;

class PackageController extends Controller
{
    // Menampilkan semua package
    public function index()
    {
        $vendorId = Auth::user()->vendor->vendorId;

        $packages = Package::with('cuisineTypes', 'category')
            ->where('vendorId', $vendorId)
            ->get();

        $cuisines = CuisineType::all();
        return view('manageCateringPackage', compact('packages', 'cuisines', 'vendorId'));
```

```
}



// Menyimpan data package baru

public function store(StorePackageRequest $request)

{

    $validated = $request->validated();



    $venAcc = Auth::user();

    $validated['vendorId'] = $venAcc->vendor->vendorId;



        // Upload file PDF

        if ($request->hasFile('menuPDFPath')) {

            $menuFile = $request->file('menuPDFPath');

            $menuFileName = 'menu_' . time() . '.' . $menuFile->getClientOriginalExtension();

            $menuFile->move(public_path('asset/menus'), $menuFileName);

            $validated['menuPDFPath'] = $menuFileName;

        }



        // Upload gambar

        if ($request->hasFile('imgPath')) {

            $imgFile = $request->file('imgPath');

            $imgFileName = 'img_' . time() . '.' . $imgFile->getClientOriginalExtension();

            $imgFile->move(public_path('asset/menus'), $imgFileName);

            $validated['imgPath'] = $imgFileName;

        }



        // // Ambil cuisine_types terpisah

    // $cuisineTypes = $validated['cuisine_types'] ?? [];



    // // Hapus dari validated array

    // unset($validated['cuisine_types']);



    // Simpan ke database (tanpa cuisine_types)
```

```
        logActivity('Successfully', 'Added', 'Catering Package with Name : ' . $validated['name']);

        $newpackage = Package::create($validated);



        // if (!empty($cuisineTypes)) {

        //     $newpackage->cuisineTypes()->sync($cuisineTypes);

        // }



        return redirect(route('manageCateringPackage'));

    }



    // Menghapus package berdasarkan ID

    public function destroy($id)

    {

        $user = Auth::user();

        $vendor = $user->vendor;

        $package = Package::findOrFail($id);



        if (!$vendor || $package->vendorId !== $vendor->vendorId) {

            abort(403, 'Unauthorized');

        }



        $package->delete();



        logActivity('Successfully', 'Deleted', 'Package : ' . $package->name);



        return response()->json([

            'success' => true,

            'message' => 'Package deleted successfully'

        ]);

    }




    public function update(UpdatePackageRequest $request, $id)
```

```
{
    $user = Auth::user();
    $vendorId = $user->vendor->vendorId;
    $package = Package::findOrFail($id);

    // Ownership check
    if ($vendorId != $package->vendorId) {
        abort(403, 'Unauthorized. You cannot edit a package that is not yours.');
    }

    $validated = $request->validated();

    // Upload file PDF ke public/asset/menus
    if ($request->hasFile('menuPDFPath')) {
        $menuFile = $request->file('menuPDFPath');
        $menuFileName = 'menu_' . time() . '.' . $menuFile->getClientOriginalExtension();
        $menuFile->move(public_path('asset/menus'), $menuFileName);
        $validated['menuPDFPath'] = $menuFileName;
    }

    // Upload gambar ke public/asset/menus
    if ($request->hasFile('imgPath')) {
        $imgFile = $request->file('imgPath');
        $imgFileName = 'img_' . time() . '.' . $imgFile->getClientOriginalExtension();
        $imgFile->move(public_path('asset/menus'), $imgFileName);
        $validated['imgPath'] = $imgFileName;
    }

    $package->update($validated);
    logActivity('Successfully', 'Updated', 'Catering Package');
    return redirect()->back()->with('success', 'Berhasil mengubah paket!');
}
```

```
    public function import(ImportPackageRequest $request)

    {

        Excel::import(new PackagesImport, $request->file('excel_file'));



        return redirect()->back()->with('success', 'Packages imported successfully!');

    }



    public function manage_profile()

    {



    }
}
```

1. [GET] Menampilkan Semua Paket
   Function: index()
   Alur:
   - Ambil vendorId dari user yang sedang login.
   - Ambil semua paket (Package) milik vendor tersebut, termasuk relasi:
     a. cuisineTypes (jenis masakan)
     b. category (kategori paket)
   - Ambil seluruh data CuisineType untuk dropdown pada form.
   - Tampilkan view manageCateringPackage.blade.php dengan semua data.

   Tujuan: Untuk menampilkan daftar paket milik vendor tertentu dan menyediakan data dropdown untuk membuat atau mengedit paket.
2. [POST] Menyimpan Paket Baru
   Function: store(StorePackageRequest $request)
   Alur:
   1. Validasi otomatis dijalankan via StorePackageRequest.

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Contracts\Validation\Validator;
use Illuminate\Http\Exceptions\HttpResponseException;
use Illuminate\Support\Facades\Log;

class StorePackageRequest extends FormRequest
{
    public function authorize(): bool
    {
        return true;
    }

    public function rules(): array
    {
        return [
            'categoryId' => 'required|integer|exists:package_categories,categoryId',
            'name' => 'required|string|max:255',

            'averageCalories' => [
                'required',
```

```
                    'filled',
                    'numeric',
                    function ($attribute, $value, $fail) {
                        if ((float) $value < 1) {
                            $fail('Average Calories harus minimal 1.');
                        }
                    }
                ],
                'breakfastPrice' => 'nullable|numeric|gt:0',
                'lunchPrice' => 'nullable|numeric|gt:0',
                'dinnerPrice' => 'nullable|numeric|gt:0',

                'menuPDFPath' => 'nullable|file|mimes:pdf,csv',
                'imgPath' => 'nullable|image|mimes:jpeg,png,jpg',
            ];
        }

        public function messages(): array
        {
            return [
                'categoryId.required' => 'Package category is required',
                'categoryId.exists' => 'Selected category does not exist in the database',
                'name.required' => 'Package name required',
            ];
        }

        protected function failedValidation(Validator $validator)
        {
            logActivity('Failed', 'Add', 'Package due to validation errors : ' .
implode($validator->errors()->all()));

            throw new HttpResponseException(
                redirect()->back()
                    ->withErrors($validator)
                    ->withInput()
            );
        }
    }
```

2. Ambil vendorId dari user login.
3. Simpan file PDF menu jika diunggah → ke folder public/asset/menus.
4. Simpan gambar jika diunggah → ke folder public/asset/menus.
5. Buat data baru ke tabel packages dengan field terisi validasi dan file path.
6. Log aktivitas "Added".
7. Redirect ke halaman list (manageCateringPackage).
8. Validasi (StorePackageRequest):
    - categoryId: required, integer, harus ada di tabel package_categories
    - name: required, max 255 karakter
    - averageCalories: required, minimal nilai 1
    - breakfastPrice, lunchPrice, dinnerPrice: nullable, harus > 0 jika diisi
    - menuPDFPath: optional, hanya PDF/CSV
    - imgPath: optional, hanya JPG/JPEG/PNG

3. [PUT] Memperbarui Paket
   Function: update(UpdatePackageRequest $request, $id)
   Alur:
    1. Ambil vendorId dari user login.
    2. Cari Package berdasarkan ID, dan pastikan vendorId cocok.
    3. Validasi otomatis dijalankan via UpdatePackageRequest.

```
use Illuminate\Contracts\Validation\Validator;

use Illuminate\Foundation\Http\FormRequest;

use Illuminate\Http\Exceptions\HttpResponseException;
```

```
class UpdatePackageRequest extends FormRequest

{

    public function authorize(): bool

    {

        return true;

    }



    public function rules(): array

    {

        return [

            'name' => 'required|string|max:255',

            'categoryId' => 'required|integer|exists:package_categories,categoryId',



            'breakfastPrice' => 'nullable|decimal:0,2|gte:0',

            'lunchPrice' => 'nullable|decimal:0,2|gte:0',

            'dinnerPrice' => 'nullable|decimal:0,2|gte:0',

            'averageCalories' => 'nullable|decimal:0,2|gte:0',



            'menuPDFPath' => 'nullable|file|mimes:pdf',

            'imgPath' => 'nullable|image|mimes:jpeg,png,jpg',



            'cuisine_types' => 'nullable|array',

            'cuisine_types.*' => 'exists:cuisine_types,cuisineId',

        ];

    }



    public function messages(): array

    {

        return [

            'categoryId.required' => 'Package category is required',

            'categoryId.exists' => 'Selected category is invalid or not found',
```

```
            'name.required' => 'Package name required',
            'name.max' => 'Package name maximal 255 characters',

            'breakfastPrice.decimal' => 'Breakfast price must be numeric',
            'breakfastPrice.gte' => 'Breakfast price must be >= 0',

            'lunchPrice.decimal' => 'Lunch price must be numeric',
            'lunchPrice.gte' => 'Lunch price must be >= 0',

            'dinnerPrice.decimal' => 'Dinner price must be numeric',
            'dinnerPrice.gte' => 'Dinner price must be >= 0',
        ];
    }

    protected function failedValidation(Validator $validator)
    {
        logActivity('Failed', 'Updated', "Packages due to validation errors : " .
    implode($validator->errors()->all()));

        throw new HttpResponseException(
            redirect()->back()
                ->withErrors($validator)
                ->withInput()
        );
    }
}
```

4. Jika ada file baru (PDF/gambar), simpan dan update path-nya.
5. Update data di database.
6. Log aktivitas "Updated".
7. Redirect kembali dengan pesan sukses.
8. Validasi (UpdatePackageRequest):
9. Sama seperti store, dengan penyesuaian:
10. Nilai averageCalories tidak wajib.
11. Validasi tambahan cuisine_types sebagai array, jika digunakan.

4. [DELETE] Menghapus Paket
   Function: destroy($id)
   Alur:

1. Ambil user login dan vendorId.
2. Cari Package berdasarkan ID.
3. Cek apakah vendor yang sedang login memiliki paket itu.
4. Jika tidak cocok, lempar 403 Unauthorized.
5. Jika cocok, hapus data dari tabel packages.
6. Log aktivitas "Deleted".
7. Return response JSON success.

5. [POST] Impor Paket via Excel
   Function: import(ImportPackageRequest $request)
   Alur:
   1. Validasi file dilakukan via ImportPackageRequest.

```php
class ImportPackageRequest extends FormRequest
{
    public function authorize(): bool
    {
        return true; // ubah ke false jika mau batasi user
    }

    public function rules(): array
    {
        return [
            'excel_file' => 'required|file|mimes:xlsx,csv|max:2048',
        ];
    }

    public function messages(): array
    {
        return [
            'excel_file.required' => __('validation.required', ['attribute' => 'file']),
            'excel_file.file'     => __('validation.file', ['attribute' => 'file']),
            'excel_file.mimes'    => __('validation.mimes', ['attribute' => 'file']),
            'excel_file.max'      => __('validation.max.file', ['attribute' => 'file', 'max' => 2048]),
        ];
    }
}
```

   2. File dikirim menggunakan method Excel::import().
   3. Proses parsing dilakukan oleh class PackagesImport.
   4. Redirect kembali ke halaman sebelumnya dengan notifikasi sukses.
   5. Validasi (ImportPackageRequest):
      - File wajib diunggah.
      - Harus bertipe xlsx atau csv.
      - Ukuran maksimal 2MB.

g. Manage Preview Vendor

Fitur ini memungkinkan vendor untuk mengelola galeri gambar paket catering mereka. Gambar-gambar ini digunakan sebagai preview visual di halaman paket yang dilihat pelanggan. Sistem ini mendukung hingga maksimal 5 preview per vendor, dan disarankan menggunakan gambar landscape dengan kualitas baik.

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\UpdateVendorPreviewRequest;
use App\Http\Requests\UploadVendorPreviewRequest;
use App\Http\Requests\VendorPreviewIndexRequest;
use App\Models\Vendor;
use App\Models\VendorPreview;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
```

```php
class VendorPreviewController extends Controller
{
    public function index(VendorPreviewIndexRequest $request)
    {
        $vendorId = $request->query('vendorId');
        if (!$vendorId) {
            return response()->json(['status' => 'error', 'message' => 'vendorId is required'], 400);
        }

        $previews = VendorPreview::where('vendorId', $vendorId)->get(['vendorPreviewId', 'previewPicturePath']);
        return response()->json([
            'status' => 'success',
            'previews' => $previews,
        ]);
    }

    public function destroy($id)
    {
        $preview = VendorPreview::findOrFail($id);

        // Hapus file dari storage
        $filePath = public_path($preview->previewPicturePath);
        if (file_exists($filePath)) {
            unlink($filePath);
        }

        // Hapus database
        $preview->delete();

        return response()->json(['status' => 'success']);
    }

    public function upload(UploadVendorPreviewRequest $request)
    {

        $file = $request->file('image');
        $fileName = uniqid() . '.' . $file->getClientOriginalExtension();
        $destinationPath = public_path('asset/catering-preview');

        // Pastikan folder ada
        if (!file_exists($destinationPath)) {
            mkdir($destinationPath, 0755, true);
        }

        // Simpan file ke folder public/asset/catering-preview
        $file->move($destinationPath, $fileName);

        $relativePath = 'asset/catering-preview/' . $fileName;

        $vendorPreview = VendorPreview::create([
            'vendorId' => $request->vendorId,
            'previewPicturePath' => $relativePath,
        ]);

        return response()->json([
            'status' => 'success',
            'message' => 'Vendor preview berhasil ditambahkan.',
            'preview' => [
                'id' => $vendorPreview->vendorPreviewId,
                'previewPicturePath' => $vendorPreview->previewPicturePath,
                'image_url' => asset($vendorPreview->previewPicturePath),
            ],
        ]);
    }


    public function update($id, UpdateVendorPreviewRequest $request)
    {
        $preview = VendorPreview::findOrFail($id);


        // Hapus file lama kalau ada
        if ($preview->previewPicturePath && file_exists(public_path($preview->previewPicturePath))) {
            unlink(public_path($preview->previewPicturePath));
        }
```

```
        // Upload file baru
        $file = $request->file('image');
        $fileName = uniqid() . '.' . $file->getClientOriginalExtension();
        $destinationPath = public_path('asset/catering-preview');

        if (!file_exists($destinationPath)) {
            mkdir($destinationPath, 0755, true);
        }

        $file->move($destinationPath, $fileName);
        $relativePath = 'asset/catering-preview/' . $fileName;

        // Update path di database
        $preview->previewPicturePath = $relativePath;
        $preview->save();

        return response()->json([
            'status' => 'success',
            'message' => 'Vendor preview berhasil diperbarui.',
            'preview' => [
                'id' => $preview->vendorPreviewId,
                'previewPicturePath' => $relativePath,
                'image_url' => asset($relativePath),
            ]
        ]);
    }

    public function showVendorDetail($vendorId)
    {
        $vendor = Vendor::findOrFail($vendorId);
        return view('manageCateringPackage', compact('vendor'));
    }
}
```

1. Menampilkan Daftar Preview Vendor

   Alur:

   1. Frontend mengirim GET /vendor-previews?vendorId={id}.
   2. Method index pada VendorPreviewController menerima dan memproses request melalui VendorPreviewIndexRequest.

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class VendorPreviewIndexRequest extends FormRequest
{
    public function authorize(): bool
    {
        return true;
    }

    public function rules(): array
    {
        return [
            'vendorId' => 'required',
        ];
    }

    public function messages(): array
    {
        return [
            'vendorId.required' => 'vendorId is required',
        ];
    }

    public function validationData()
    {
        // agar bisa validasi query param
        return $this->query();
    }
}
```

3. Validasi dilakukan pada query string (vendorId harus ada).
4. Jika valid, maka data gambar (vendorPreviewId, previewPicturePath) diambil dari tabel vendor_previews sesuai vendorId.
5. Hasil dikembalikan dalam format JSON.

Validasi:

Validasi dilakukan pada query() (bukan body) menggunakan method validationData().

2. Menambahkan Gambar Baru (Upload)

Alur:

1. Frontend mengirim POST dengan form-data: image (file), vendorId (UUID)
2. UploadVendorPreviewRequest memproses dan memvalidasi input.

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UploadVendorPreviewRequest extends FormRequest
{
    public function rules()
    {
        return [
            'image' => 'required|image|mimes:jpg,jpeg,png|max:2048',
            'vendorId' => 'required|exists:vendors,vendorId',
        ];
    }

    public function messages()
    {
        return [
            'image.required' => 'Image is required',
            'image.image' => 'File must be an image',
            'image.mimes' => 'Image must be JPG, JPEG, or PNG',
            'vendorId.required' => 'vendorId is required',
            'vendorId.exists' => 'Vendor not found',
        ];
    }
}
```

3. File gambar disimpan ke folder public/asset/catering-preview/
4. Folder otomatis dibuat jika belum ada (mkdir(..., 0755, true)).
5. Nama file digenerate secara unik menggunakan uniqid().
6. Path file disimpan ke database melalui VendorPreview::create().

Validasi:

1. Cek format file (harus gambar)
2. Batas ukuran maksimal 2MB
3. vendorId harus sesuai dengan yang ada di tabel vendors

3. Memperbarui Gambar (Update)

Alur:

1. Frontend mengirim PUT dengan form-data berisi gambar baru.
2. Controller:
   a. Mencari data preview berdasarkan ID (findOrFail)
   b. Menghapus file gambar lama (unlink)
   c. Mengunggah file baru dan menyimpannya di direktori publik
   d. Update previewPicturePath di database
   e. File disimpan di asset/catering-preview/ dan response dikembalikan dengan URL gambar baru.

Validasi:

1. Validasi image dilakukan oleh UpdateVendorPreviewRequest (meskipun tidak disertakan di sini, diasumsikan validasi serupa seperti UploadVendorPreviewRequest).
2. Jika file lama tidak ada, proses lanjut tanpa error.

4. Menghapus Gambar Preview

Alur:

1. Frontend mengirim DELETE ke endpoint dengan ID gambar.
2. Controller:
   a. Cari gambar berdasarkan ID (findOrFail)
   b. Jika file ada di storage (file_exists()), maka dihapus dengan unlink()
   c. Data gambar dihapus dari database dengan $preview->delete()

Validasi:

1. Jika ID tidak ditemukan → otomatis throw 404 (karena findOrFail)
2. Aman untuk dijalankan berkali-kali karena file_exists() dicek sebelum unlink

h. Update Delivery Status

Tujuan Fitur Update Delivery Status bertujuan untuk memungkinkan vendor memperbarui status pengiriman pesanan pelanggan secara real-time, seperti menandai pesanan sebagai "sudah dikirim" atau "sudah diterima". Hal ini penting untuk memastikan transparansi proses logistik, memungkinkan monitoring oleh admin atau sistem, serta menjadi syarat untuk fitur lanjutan seperti review pelanggan. Dengan adanya pembaruan status ini, sistem dapat menyajikan data statistik yang akurat, menjaga akuntabilitas vendor, dan meningkatkan pengalaman pengguna secara keseluruhan.

```
private function weekBounds(string $which = 'current'): array
  {
      $monday = Carbon::now('Asia/Jakarta')->startOfWeek(Carbon::MONDAY); // Senin

      return $which === 'next'
          ? [$monday->copy()->addWeek(), $monday->copy()->endOfWeek()->addWeek()]
          : [$monday, $monday->copy()->endOfWeek()];
  }

  /* ------------------------------------------------------------
   *  Daftar order (this week / next week lewat query ?week=)
   * ------------------------------------------------------------ */
  public function index(DeliveryOrderVendorIndexRequest $request)
  {
      $validated = $request->validated();

      $vendor = Auth::user()->vendor;
      $vendorId = $vendor->vendorId;

      $whichWeek = $request->query('week', 'current');       // current | next
      [$from, $to] = $this->weekBounds($whichWeek);

      $orders = Order::with([
          'user.defaultAddress',
          'orderItems.package',
          'deliveryStatuses',
      ])
          ->where('vendorId', $vendorId)
          ->where('isCancelled', 0)
          ->whereDate('startDate', '>=', $from)    // ← pakai startDate
          ->whereDate('startDate', '<=', $to)
          ->get()
          ->map(function ($order) {
              // $addr = $order->user->defaultAddress;

              return [
                  'id' => $order->orderId,
                  'startDate' => Carbon::parse($order->startDate)->format('Y-m-d'),
```

```
                'user' => [
                    'name' => $order->recipient_name,
                    'phone' => $order->recipient_phone,
                    'address' => $order->jalan,
                    'notes' => $order->notes,
                ],

                'order_items' => $order->orderItems->map(fn($it) => [
                    'package' => ['name' => $it->package->name ?? 'Paket'],
                    'quantity' => $it->quantity,
                    'package_time_slot' => $it->packageTimeSlot,
                ]),

                'delivery_statuses' => $order->deliveryStatuses
                    ->filter(fn($ds) => $ds->deliveryDate->isToday())
                    ->map(fn($ds) => [
                        'slot' => strtolower($ds->slot),
                        'status' => $ds->status,
                        'delivery_date' => $ds->deliveryDate->toDateString(),
                    ])->values(),
            ];
        });

    $packages = $orders
        ->flatMap(fn($o) => $o['order_items']->pluck('package.name'))
        ->unique()
        ->values();

    return view('manageOrder', compact('orders', 'packages', 'vendor'));
}
```

1. **Menampilkan Daftar Order Mingguan (Index View)**

Alur:

1. Vendor mengakses halaman "Manage Orders" untuk melihat daftar pesanan.
2. Request dikirimkan ke controller OrderVendorController@index.
3. Validasi dilakukan oleh DeliveryOrderVendorIndexRequest.

```
<?php

namespace App\Http\Requests;

use App\Enums\UserRole;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\Rule;

class DeliveryOrderVendorIndexRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check() && Auth::user()->role === UserRole::Vendor;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'week' => [
                'nullable',
                'string',
                Rule::in(['current', 'next']),
            ],
        ];
```

```
    }
}
```

week

- nullable: Boleh tidak dikirim dalam request.
- string: Jika dikirim, nilainya harus berupa string.
- in: ['current', 'next']: Nilainya hanya boleh 'current' (minggu ini) atau 'next' (minggu depan).Misalnya week=last atau week=random akan ditolak karena tidak masuk daftar nilai yang diperbolehkan.

4. Backend ambil semua order:
   a. Berdasarkan vendorId vendor yang sedang login.
   b. Dengan startDate dalam minggu ini atau minggu depan (current/next).
   c. Yang belum dibatalkan (isCancelled = 0).
5. Data dikumpulkan dan ditransformasikan:
   a. Info user dan alamat pengiriman.
   b. Item pesanan & slot waktunya (pagi/siang/malam).
   c. Status pengantaran untuk hari ini saja.
6. Ditampilkan ke view manageOrder.blade.php.

2. **Update Status Pengantaran per Slot (Delivery Status)**

```php
public function updateStatus(DeliveryStatusUpdateOrderVendorRequest $request, $orderId, $slot)
    {
        $validated = $request->validated();

        $ds = DeliveryStatus::where('orderId', $orderId)
            ->where('slot', $slot)
            ->whereDate('deliveryDate', today())
            ->first();

        if (!$ds) {
            return response()->json([
                'success' => false,
                'message' => 'Delivery status not found for this order and slot.'
            ], 404);
        }
        $ds->status = $validated['status'];
        $ds->save();

        #Note: Get the order, user and status
        $order = Order::find($orderId);
        $userId = $order->userId;
        $user = User::find($userId);
        $orderStatus = $request->status;

        #Note: generating appropriate notification to be sent based on the status;
        $toBeNotified = null;
        if($orderStatus === 'Prepared')
        {
           $toBeNotified = new OrderPrepared($order);
        }
        else if($orderStatus === 'Delivered')
        {
            $toBeNotified = new OrderDelivered($order);
        }
        else if($orderStatus === 'Arrived')
        {
            $toBeNotified = new OrderArrived($order);
        }

        if($toBeNotified !== null)
        {
            #Note: only send when notification is successfuly created;
            $user->notify($toBeNotified);
        }

        logActivity('Successfully', 'Updated', 'Order Delivery Status to ' . $ds->status->value, $request);
        return response()->json([
```

```
            'success' => true,
            'message' => 'Status updated successfully.',
            'updated_status' => $ds
        ]);
    }
```

Alur:

1. Vendor mengklik tombol update status di UI (misalnya dari "Prepared" ke "Delivered").
2. Frontend mengirimkan PUT request dengan orderId dan slot (morning/afternoon/evening).
3. Validasi request oleh DeliveryStatusUpdateOrderVendorRequest, yaitu:

```php
<?php

namespace App\Http\Requests;

use App\Enums\UserRole;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\Rule;

class DeliveryStatusUpdateOrderVendorRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check() && Auth::user()->role === UserRole::Vendor;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'status' => [
                'required',
                Rule::in(['Prepared', 'Delivered','Arrived']),
            ]
        ];
    }

    public function messages()
    {
        return [
            'status.required' => 'Status is required.',
            'status.in' => 'Status must be one of: Prepared, Delivered, or Arrived.',
        ];
    }

    /**
     * Force JSON validation error response.
     */
    // protected function failedValidation(Validator $validator)
    // {
    //     throw new HttpResponseException(response()->json([
    //         'success' => false,
    //         'message' => 'Validation failed.',
    //         'errors' => $validator->errors()
    //     ], 422));
    // }
}
```

- orderId
  Harus ada (required).
  Harus berupa UUID yang valid (uuid).
  Dan harus ada di database pada kolom orderId dari tabel orders (exists:orders,orderId).
- deliveryStatus

Harus ada (required).
Harus bernilai salah satu dari enum DeliveryStatus, misalnya PROCESSING, DELIVERED, CANCELLED, dll. Ini dicek menggunakan aturan Enum(DeliveryStatus::class).

4. Backend cari data DeliveryStatus berdasarkan:
   a. orderId,
   b. slot,
   c. dan deliveryDate = today().
5. Jika tidak ditemukan → 404.
6. Jika ditemukan → update status.
7. Kirim notifikasi ke user:
   a. Prepared → notifikasi persiapan
   b. Delivered → notifikasi pengiriman
   c. Arrived → notifikasi pesanan sampai
8. Catat log aktivitas.
9. Kirim response JSON sukses.

i. Cancel Upcoming Order

Fitur Cancel Upcoming Order bertujuan untuk memberikan fleksibilitas kepada vendor dalam membatalkan pesanan yang akan datang sebelum proses pengiriman dimulai. Fitur ini penting untuk mengelola situasi tak terduga seperti keterlambatan bahan baku, kendala operasional, atau force majeure. Dengan membatalkan pesanan lebih awal, sistem dapat menginformasikan pelanggan secara otomatis, mencegah persiapan yang sia-sia, dan menjaga reputasi vendor dengan memberikan alasan pembatalan yang jelas. Selain itu, fitur ini juga membantu menjaga integritas data transaksi dan menyederhanakan proses rekapitulasi penjualan serta logistik.

OrderVendorController.php

```
public function cancel(Order $order)
    {
        // pastikan vendor yang login memang pemilik pesanan
        if ($order->vendorId !== (Auth::user()->vendor->vendorId)) {
            abort(403);
        }

        if ($order->isCancelled) {
            return response()->json(['success' => false, 'message' => 'Order already canceled.'], 400);
        }

        if ($order->endDate <= today()) {
            return response()->json(['success' => false, 'message' => 'Completed orders cannot be canceled.'],
400);
        }

        if ($order->startDate <= today()) {
            return response()->json(['success' => false, 'message' => 'Only upcoming orders can be canceled.'],
400);
        }


        $order->isCancelled = 1;
        $order->save();

        return response()->json(['success' => true], 200);
    }
```

Alur:

1. Vendor menekan tombol cancel pada pesanan.
2. Request masuk ke OrderVendorController@cancel.

3. Backend validasi kondisi:
   a. Apakah vendor memang pemilik pesanan?
   b. Apakah order sudah dibatalkan sebelumnya?
   c. Apakah tanggal endDate sudah lewat (tidak bisa dibatalkan)?
   d. Apakah sudah berjalan (startDate <= today()) → tidak bisa dibatalkan?
4. Jika valid → update isCancelled = 1.
5. Response JSON sukses.

Validasi (manual di dalam controller):

1. vendorId pemilik harus cocok.
2. Hanya boleh membatalkan order yang:
   a. Belum berakhir (endDate > today()),
   b. Belum dimulai (startDate > today()),
   c. Belum pernah dibatalkan (isCancelled == 0).

j. View Customer's Review

Fitur View Review memungkinkan vendor untuk melihat ulasan yang diberikan oleh pelanggan terhadap produk atau layanan yang mereka jual. Setiap ulasan disertai dengan nama (tercensorship sebagian demi privasi), kutipan isi review, tanggal pemesanan, serta rating dalam bentuk bintang. Di bagian atas, jumlah total penjualan vendor juga ditampilkan dalam bentuk badge.

VendorController.php

```
public function review(Vendor $vendor)
    {

        $vendorReviews = VendorReview::where('vendorId', $vendor->vendorId)
            ->with(['user', 'order']) // Load user dan order
            ->orderBy('created_at', 'desc')
            ->get();

        $numSold = Order::where('vendorId', $vendor->vendorId)->count();

        return view('ratingAndReview', compact('vendor', 'vendorReviews', 'numSold'));
    }

    public function reviewVendor()
    {
        // Ambil vendor yang sedang login
        $vendor = Auth::user()->vendor; // Pastikan user login adalah vendor

        // Ambil review dari vendor yang sedang login
        $vendorReviews = VendorReview::where('vendorId', $vendor->vendorId)
            ->with(['user', 'order']) // Load relasi user dan order
            ->orderBy('created_at', 'desc')
            ->get();

        // Hitung jumlah order yang dijual oleh vendor
        $numSold = Order::where('vendorId', $vendor->vendorId)->count();

        return view('ratingAndReviewVendor', compact('vendor', 'vendorReviews', 'numSold'));
    }
```

1. Akses Fitur
   - Vendor membuka halaman review melalui menu "Rating & Review".
   - Sistem memanggil method reviewVendor() untuk menampilkan semua review terhadap vendor tersebut.
2. Autentikasi dan Otorisasi
   - Sistem memverifikasi apakah pengguna yang login memiliki role vendor.
   - Jika bukan vendor, akses ditolak dan sistem akan melakukan redirect atau menampilkan halaman error.

3. Pengambilan Data
    - Sistem mengambil data vendorId dari akun vendor yang sedang login.
    - Sistem melakukan query ke tabel VendorReview untuk mengambil semua review yang sesuai.
    - Sistem juga mengambil data relasi user dan order menggunakan with(['user', 'order']).
    - Sistem mengambil jumlah total order vendor melalui Order::where('vendorId', $vendor->vendorId)->count().
4. Pengolahan dan Penyajian
    - Data review diurutkan berdasarkan waktu (created_at DESC).
    - Untuk setiap review:
    - Nama user disensor sebagian.
    - Isi review ditampilkan sebagian (summary atau kutipan).
    - Tanggal pemesanan diambil dari objek order.
    - Rating disimbolkan dengan bintang dan nilai numerik.
    - Total penjualan vendor ditampilkan dalam bentuk badge “X Sold”.
5. Tampilan Akhir
    - Data ditampilkan dalam view ratingAndReviewVendor.blade.php (atau ratingAndReview.blade.php untuk admin).
    - Vendor dapat melihat semua ulasan beserta rating, tanggal order, dan identitas terbatas pengguna.

k. View vendor sales

Fitur Sales Report memungkinkan vendor untuk melihat rekapitulasi penjualan berdasarkan rentang tanggal tertentu. Laporan ini dapat difilter berdasarkan periode waktu dan dapat diekspor menjadi file eksternal untuk kebutuhan dokumentasi dan analisis.

```php
public function index(FilterSalesRequest $request)
    {
        $user = Auth::user();
        $vendor = $user->vendor;
        $vendorId = $vendor->vendorId;
        $validated = $request->validated();

        $startDate = $validated['startDate'] ?? null;
        $endDate = $validated['endDate'] ?? null;

        // Call helper method to fetch filtered data
        [$orders, $totalSales] = $this->getVendorSalesRanged($vendorId, $startDate, $endDate);

        return view('catering.vendorSales', compact('orders', 'totalSales', 'vendor', 'startDate', 'endDate'));
    }

    public function export_sales(FilterSalesRequest $request)
    {
        $user = Auth::user();
        $vendorId = $user->vendor->vendorId;
        $validated = $request->validated();

        $startDate = $validated['startDate'] ?? null;
        $endDate = $validated['endDate'] ?? null;

        // Call helper method to fetch filtered data
        [$orders, $totalSales] = $this->getVendorSalesRanged($vendorId, $startDate, $endDate);

        $start = Carbon::parse($startDate)->format('d M Y') ?? null;
        $end = Carbon::parse($endDate)->format('d M Y') ?? null;

        return Excel::download(new SalesExport($orders, $totalSales, $start, $end), "laporan_penjualan.xlsx");
    }
```

Alur:

1. User Login

Hanya user dengan role Vendor atau Admin yang dapat mengakses fitur ini.

2. Form Filter Ditampilkan
   User (vendor) dapat memilih tanggal Start Date dan End Date melalui form.
3. Request Dikirim & Diverifikasi
   - Permintaan difilter menggunakan FilterSalesRequest untuk:
     - Memastikan user yang mengakses valid dan memiliki hak akses.
     - Memastikan input tanggal valid dan endDate setelah startDate.

```php
<?php

namespace App\Http\Requests;

use App\Enums\UserRole;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\App;
use Illuminate\Support\Facades\Auth;

class FilterSalesRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return Auth::check() && (Auth::user()->role === UserRole::Vendor || Auth::user()->role ===
UserRole::Admin);
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'startDate' => [
                'nullable',
                'date',
            ],
            'endDate' => [
                'nullable',
                'date',
                'after:startDate',
            ],
        ];
    }

    public function messages()
    {
        $locale = App::getLocale();

        return [
            'endDate.after' => $locale === 'id' ? 'Tanggal akhir harus sesudah tanggal mulai' : 'Invalid date
range',
        ];
    }
}
```

1. Validasi Autentikasi dan Otorisasi Pengguna
   Sistem melakukan verifikasi untuk memastikan bahwa pengguna yang mengirimkan permintaan telah terautentikasi (login) dan memiliki hak akses yang sesuai. Hanya pengguna dengan peran Vendor atau Admin yang diizinkan untuk mengakses fitur laporan penjualan ini. Permintaan dari pengguna yang tidak memenuhi kriteria ini akan ditolak.
2. Validasi Field startDate (Tanggal Mulai)
   - Field ini bersifat opsional (nullable).

- Jika diisi, nilainya harus dalam format tanggal yang valid sesuai standar ISO (YYYY-MM-DD).
- Field ini digunakan sebagai batas awal periode waktu dalam proses penyaringan data penjualan.

3. Validasi Field endDate (Tanggal Akhir)
   - Field ini juga bersifat opsional (nullable).
   - Jika diisi, nilainya harus dalam format tanggal yang valid.
   - Selain itu, nilai endDate harus lebih besar dari startDate jika keduanya diisi, untuk memastikan rentang waktu yang logis.
4. Penanganan Pesan Validasi (Custom Error Message)
   Apabila validasi terhadap endDate gagal (misalnya nilainya lebih awal dari startDate), maka sistem akan menampilkan pesan kesalahan yang disesuaikan dengan bahasa yang sedang digunakan (lokalisasi).

4. Ambil Data Vendor dan Parameter Tanggal
   Sistem mengambil data vendor dari user yang login, lalu menangkap parameter filter dari form.
5. Ambil Data Penjualan
   Sistem menjalankan method getVendorSalesRanged() untuk mengambil data penjualan berdasarkan vendorId dan rentang tanggal yang difilter.
6. Kembalikan ke View
   Data penjualan ditampilkan dalam bentuk tabel, termasuk detail pesanan dan total sales.
7. Opsi Ekspor
   Jika ada hasil penjualan, tombol Export akan aktif dan dapat digunakan untuk mengunduh laporan.

E. Admin

a. Admin dashboard

Menampilkan ringkasan bulanan berupa:

- Profit: total keuntungan bulan ini dan persentase kenaikannya dibanding bulan sebelumnya
- Total sales: total penjualan bulan ini dan persentase kenaikan dibanding bulan sebelumnya

Grafik penjualan per bulan sepanjang tahun
Recent logs: daftar 10 aktivitas terakhir pengguna yang tercatat
Berikut adalah backend yang mengatur tampilan admin dashboard:

```
public function index()
    {
        $orders = Order::whereMonth('created_at', Carbon::now()->month)
            ->whereYear('created_at', Carbon::now()->year)
            ->get();

        $totalPrice = $orders->sum(function ($order) {
            return (float) $order->totalPrice;
        });

        $lastMonthOrder = Order::whereMonth('created_at', Carbon::now()->subMonth()->month())
            ->whereYear('created_at', Carbon::now()->year)
            ->get();

        $lastMonthSale = $lastMonthOrder->sum(function ($lastMonthOrder) {
            return (float) $lastMonthOrder->totalPrice;
        });
```

```
        $increment = $totalPrice - $lastMonthSale;

        $percentage = 0;
        if ($lastMonthSale == 0) {
            $percentage = 100;
        } else {
            $percentage = ($increment / $lastMonthSale) * 100;
        }

        $lmprofit = 0.05 * $lastMonthSale;

        $profit = 0.05 * $totalPrice;

        $lmprofit = 10000;


        $percentageprofit = 0;
        if ($lmprofit == 0) {
            $percentageprofit = 100;
        } else
        {
            $percentageprofit = (($profit - $lmprofit) / $lmprofit) * 100;
        }


        $profit = number_format($profit, 0, ',', '.');

        $totalPrice = number_format($totalPrice, 0, ',', '.');

        $monthlySales = Order::selectRaw('MONTH(created_at) as month, SUM(totalPrice) as total')
            ->whereYear('created_at', Carbon::now()->year)
            ->groupBy(DB::raw('MONTH(created_at)'))
            ->orderBy(DB::raw('MONTH(created_at)'))
            ->pluck('total', 'month');

        $allMonths = collect(range(1, 12))->map(function ($month) use ($monthlySales) {
            return $monthlySales->get($month, 0);
        });

        $chartData = $allMonths->values()->toArray();

        $labels = collect(range(1, 12))->map(function ($month) {
            return Carbon::create()->month($month)->locale('id')->translatedFormat('F');
        })->toArray();

        $logs = UserActivity::query()->orderBy('accessed_at', 'desc')->limit(10)->get();

        logActivity('Successfully', 'Visited', 'Admin Dashboard Page');

        return view('adminDashboard', compact('totalPrice', 'percentage', 'profit', 'increment',
'percentageprofit', 'chartData', 'labels', 'logs'));
    }
```

b. View all transactions

Menampilkan seluruh data transaksi (pembayaran) yang terjadi dalam sistem dengan bentuk tabel yang terdiri dari ID pembayaran, metode pembayaran, nama pelanggan, nama penjual, total harga, dan waktu pembayaran. Berikut adalah backend yang menampilkan semua transaksi yang ada:

```
public function view_all_transactions()
    {

        // $payments = Payment::all()->paginate(10);
        $payments = Payment::orderBy('paid_at', 'asc')->paginate(25);

        logActivity('Successfully', 'Visited', 'View all transaction');
        return view('view-all-transactions', compact('payments'));
    }
```

c. Manage Package Category

Menampilkan daftar kategori makanan yang tersedia dalam bentuk tabel dengan kolom nama kategori, jumlah paket terkait, waktu dibuatnya kategori tersebut, dan aksi yang dapat dilakukan. Admin dapat:

- Tambah kategori baru dengan syarat nama tidak boleh sama dengan yang sudah ada. Berikut adalah backend yang mengatur penambahan kategori:

```php
public function store(CategoryStoreRequest $request)
    {
        $validated = $request->validated();

        PackageCategory::create([
            'categoryName' => $validated['categoryName'],
            'created_at' => now(),
            'updated_at' => now(),
        ]);

        return redirect()->back()->with('success', __('admin/package_category.store_success'));
    }
```

input yang diberikan divalidasi dahulu nama yang diberikan harus berupa string dan unik.

```php
public function authorize(): bool
    {
        return Auth::check() && Auth::user()->role === UserRole::Admin;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'categoryName' => [
                'required',
                'string',
                Rule::unique('package_categories', 'categoryName')
            ]
        ];
    }
```

- Hapus kategori dengan syarat tidak digunakan oleh paket manapun. Jika admin mencoba hapus kategori yang terasosiasi dengan paket manapun, maka akan ada peringatan terlebih dahulu. Berikut adalah backend yang mengurusnya:

```php
public function destroy(string $id)
    {
        $category = PackageCategory::findOrFail($id);

        if ($category->packages()->exists()) {
            return redirect()->back()->with('error',  __('admin/package_category.delete_failed'));
        }

        $category->delete(); // Soft delete

        return redirect()->back()->with('success', __('admin/package_category.delete_success'));
    }
```

Dicek juga jika kategori tertaut pada paket apapun, maka penghapusan gagal, jika tidak ada paket tertaut, maka kategori akan dihapus. Karena sistem tidak menyediakan fungsi untuk restore yang telah dihapus, namun memakai soft delete dan terdapat constraint harus unik, maka diatur di observer agar namanya diganti tiap kali data kategori di soft delete:

```php
public function deleted(PackageCategory $packageCategory): void
    {
        if ($packageCategory->isForceDeleting()) {
            return;
        }
```

```
        $packageCategory->categoryName = $packageCategory->categoryName . '_' . now()->format('Ymd_His');
        $packageCategory->saveQuietly();
    }
```

d. Search Vendors

Admin dapat melihat semua profil singkat vendor yang terdaftar dan dilengkapi fitur pencarian vendor berdasarkan namanya.

```
public function search(AdminSearchRequest $request)
    {
        $validated = $request->validated();
        $name = $validated['name'];

        // Cari vendor berdasarkan nama
        $vendors = Vendor::where('name', 'like', '%' . $name . '%')->get();

        // Ambil total penjualan (sales) berdasarkan vendor
        $sales = DB::table('orders')
            ->select('vendorId', DB::raw('SUM(totalPrice) as totalSales'))
            ->groupBy('vendorId')
            ->pluck('totalSales', 'vendorId');

        // Catat aktivitas pencarian
        logActivity('Successfully', 'Searched', 'Vendor by Name');

        // Kirim data ke view
        return view('viewAllVendor', compact('vendors', 'sales'));
    }
```

Input divalidasi di form request agar namanya yang dicari berupa string dan maksimal panjangnya 255 karakter:

```
public function rules(): array
    {
        return [
            'name' => 'required|string|max:255',
        ];
    }
```

e. View User Logs

Menampilkan semua log aktivitas user yang terjadi dalam sistem termasuk peran, url, dan deskripsi dari aksi sang user yang diurutkan dari yang terbaru dan dipaginasi per 25 logs.

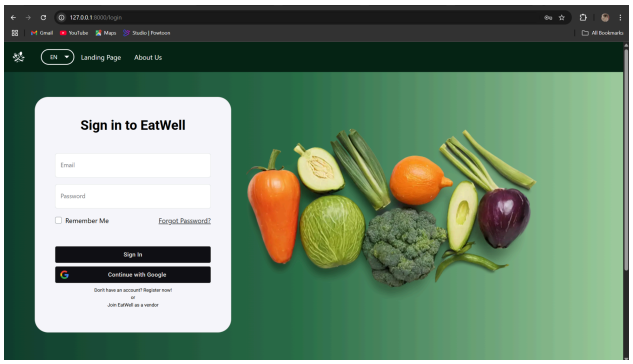
```
public function view_all_logs()
    {
        $all_logs = UserActivity::orderBy('accessed_at', 'desc')->paginate(25);

        return view('view-all-logs', compact('all_logs'));
    }
```

f. Filter & Export Order History

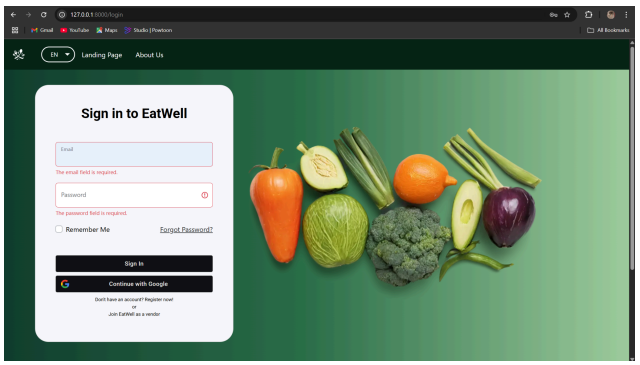
Admin dapat melihat seluruh riwayat pemesanan yang tercatat dengan informasi ID pesanan, nama vendor, nama pelanggan, paket yang dipesan, periode pesanan, dan status pesanan. Admin juga dapat memfilter pesanan berdasarkan tanggal dibuatnya pesanan. Kemudian admin juga dapat mengunduh laporan riwayat pesanan secara keseluruhan atau yang telah difilter nya. Berikut adalah backend yang mengatur filter dan export order history:

BAB IV USER INTERFACE

A. Login



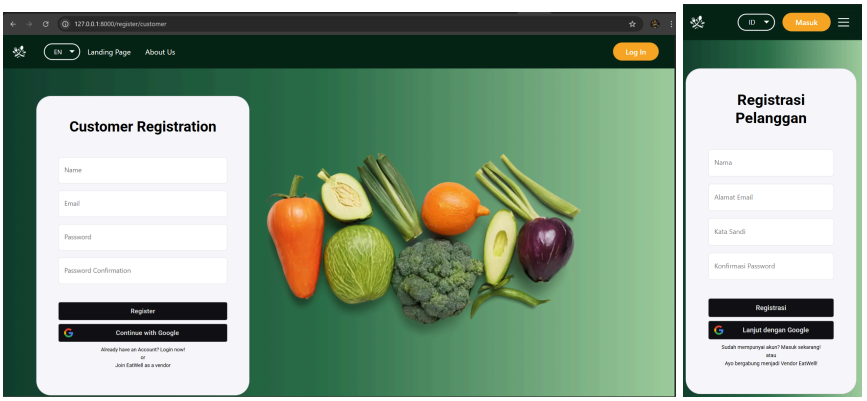
- Jika user click “Sign In” dan tidak lolos validasi:



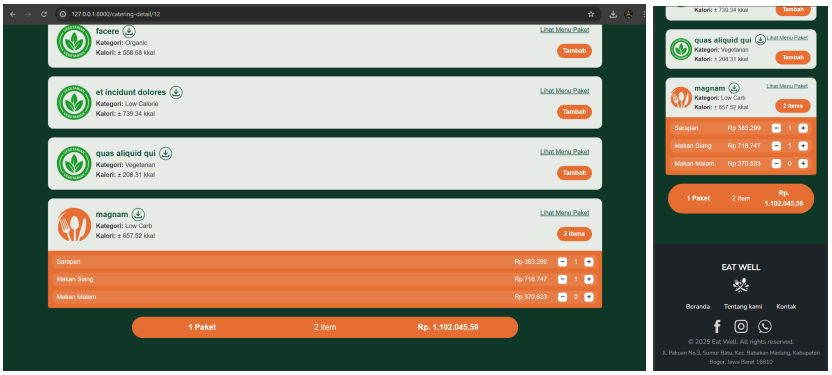
- Jika user click “Sign In” lolos validasi, maka akan diarahkan ke page Home.

B. Customer

a. Register

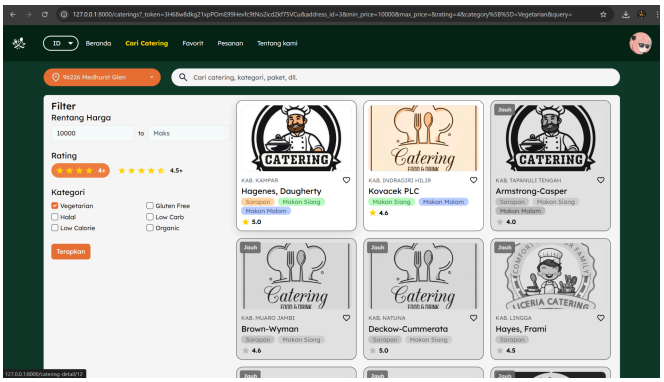


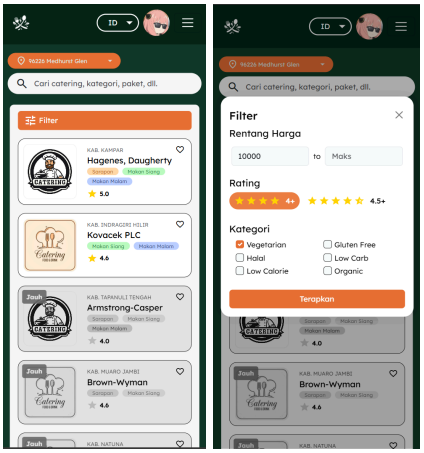
b. Update Cart



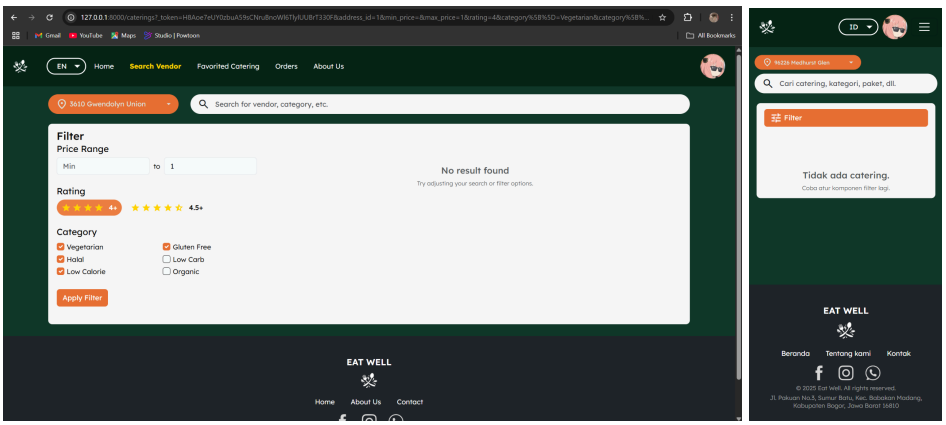
c. Filter Catering

- Jika hasil filter ada:

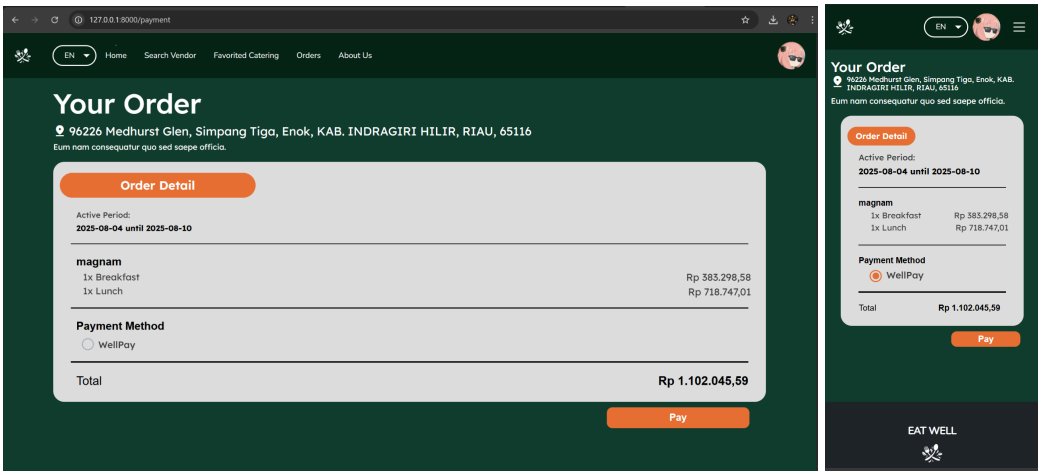




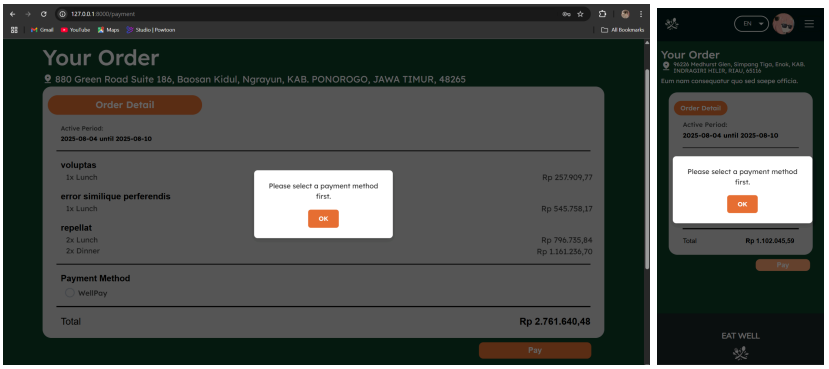
- Jika hasil filter tidak ada:



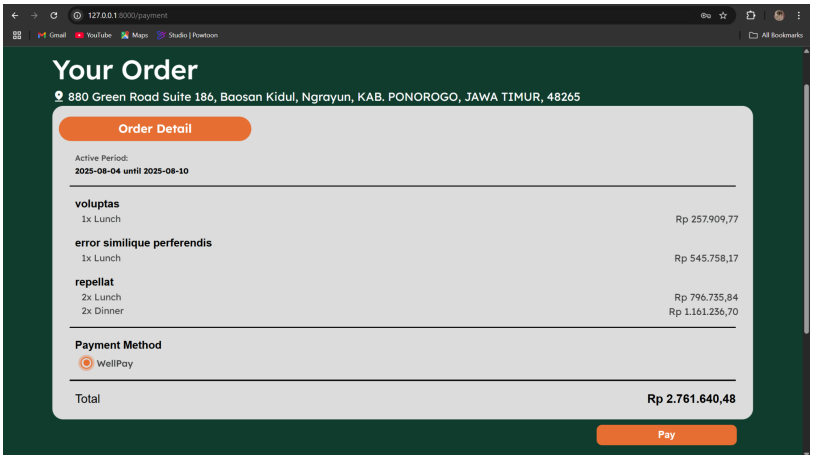
d. Pay Order

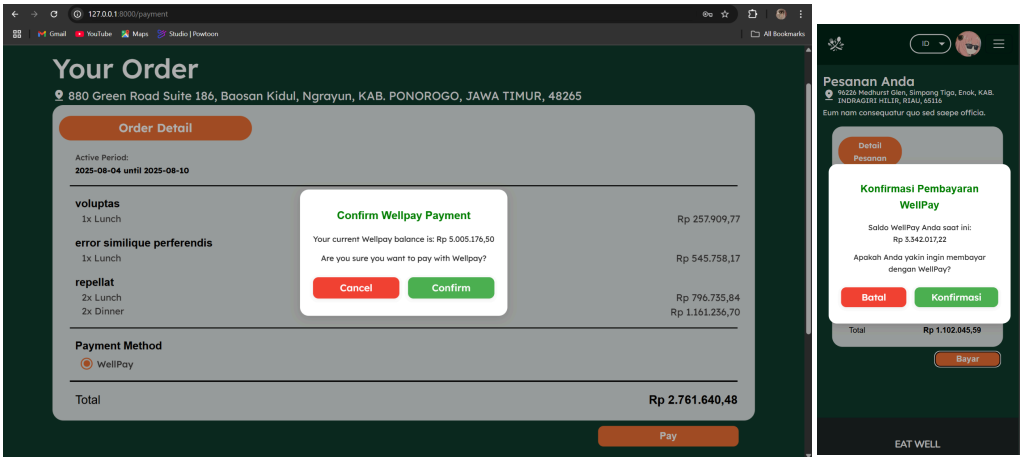


- Jika tombol “Pay” diclick tanpa memilih payment method:

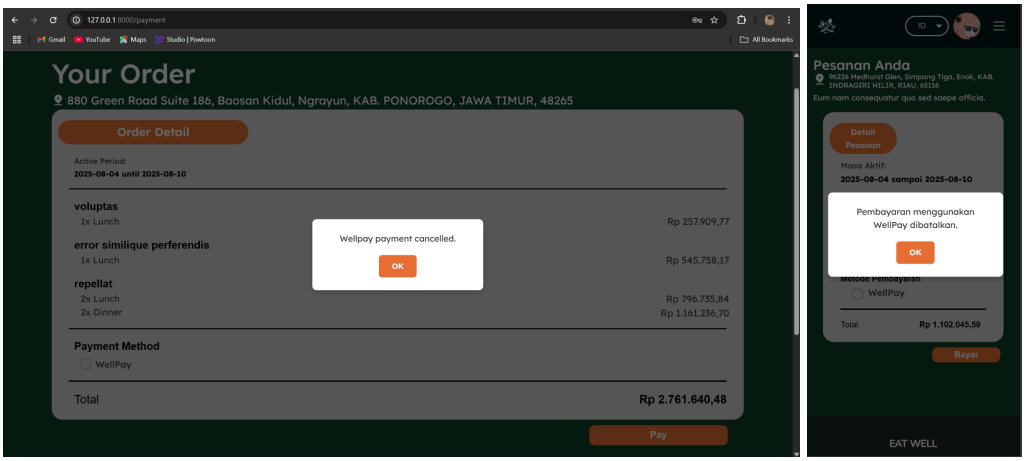


- Ketika click “OK” dan memilih WellPay method:

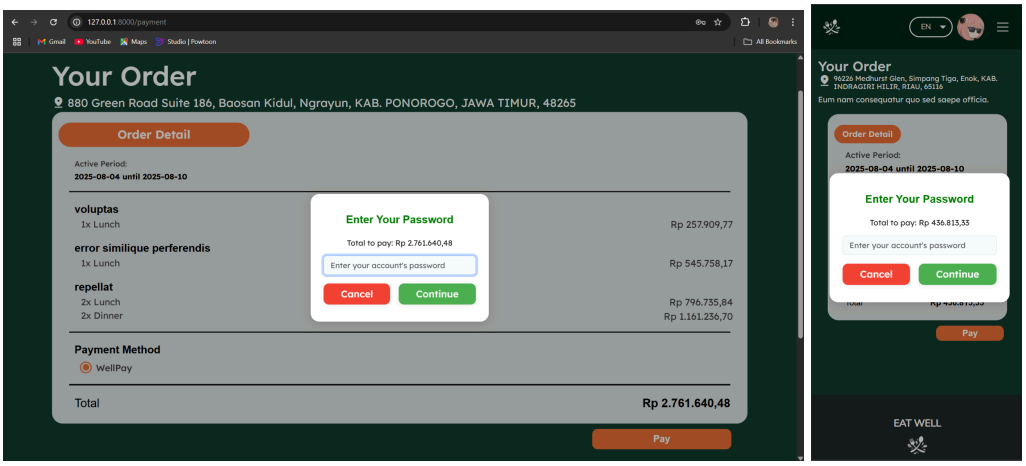




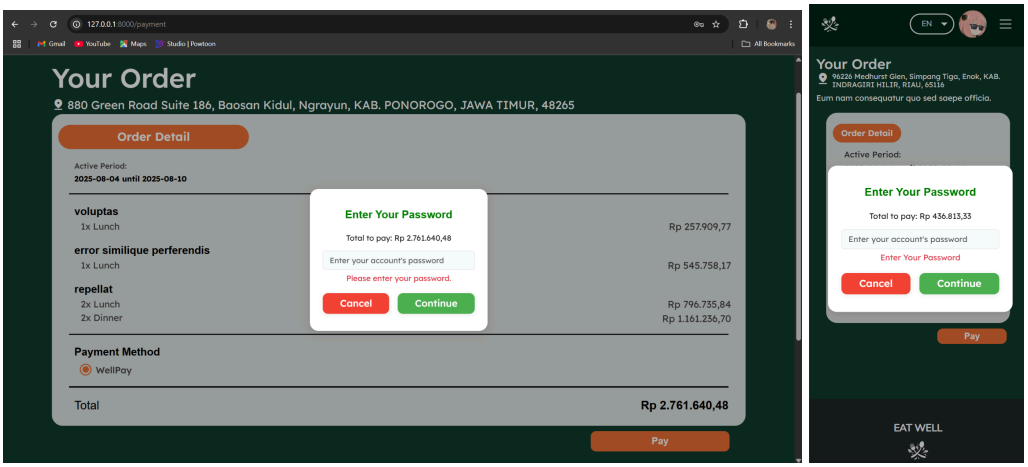
- Jika user click “cancel”:



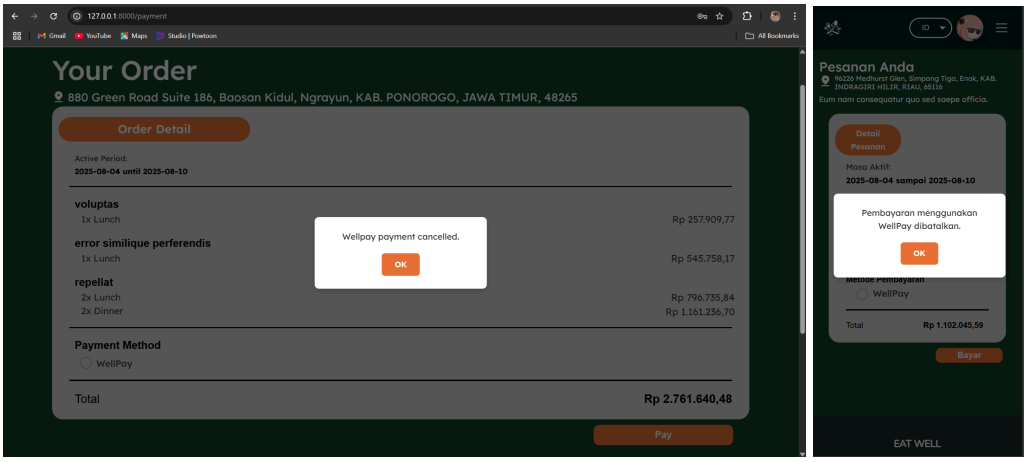
- Jika user click “Confirm”:



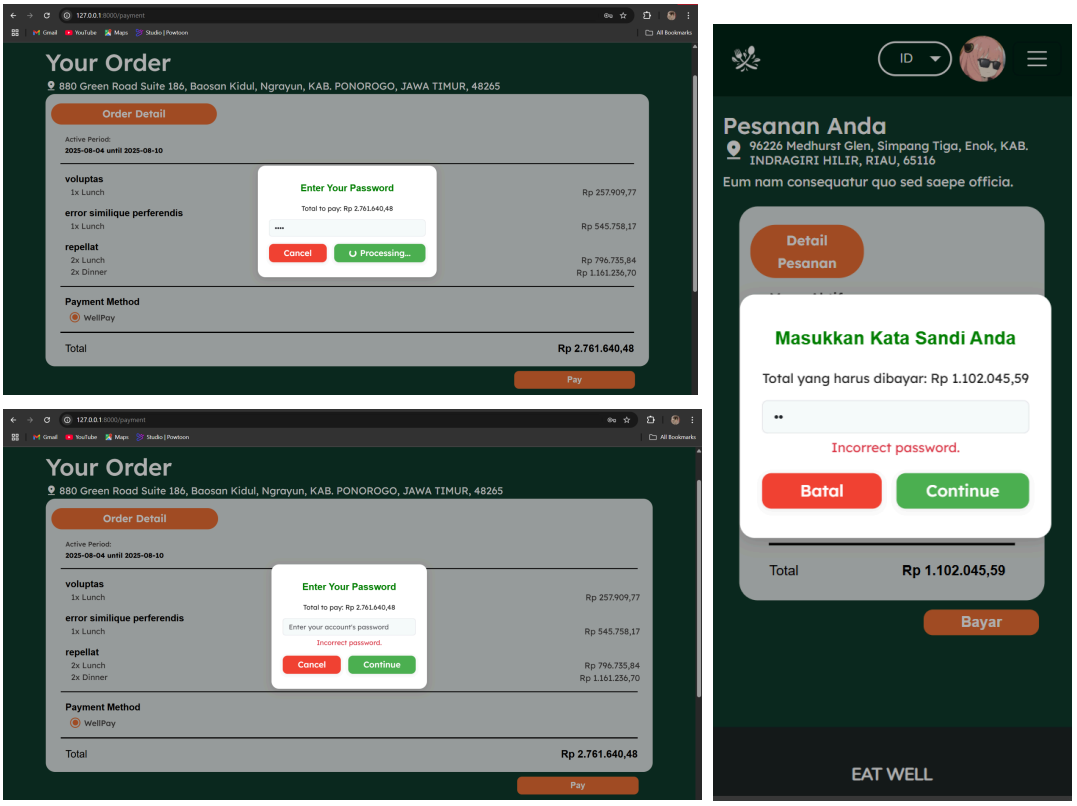
- Jika user click “Continue” dan password kosong, akan muncul pesan error:



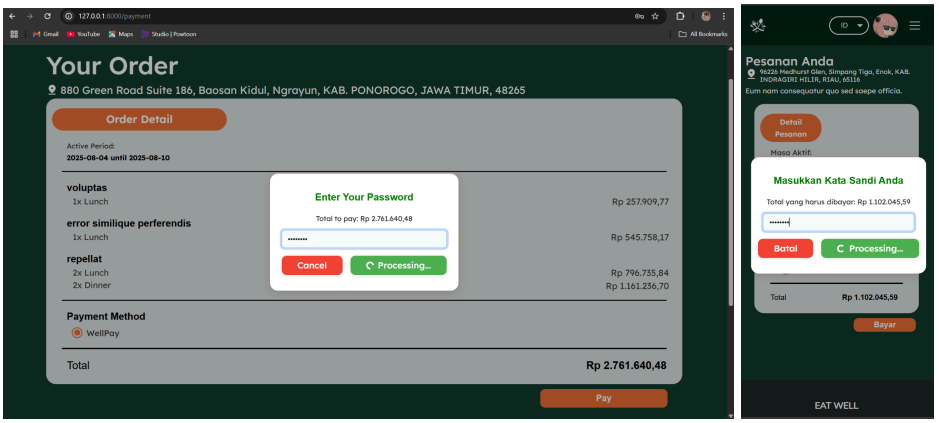
- Jika user click “Cancel”:

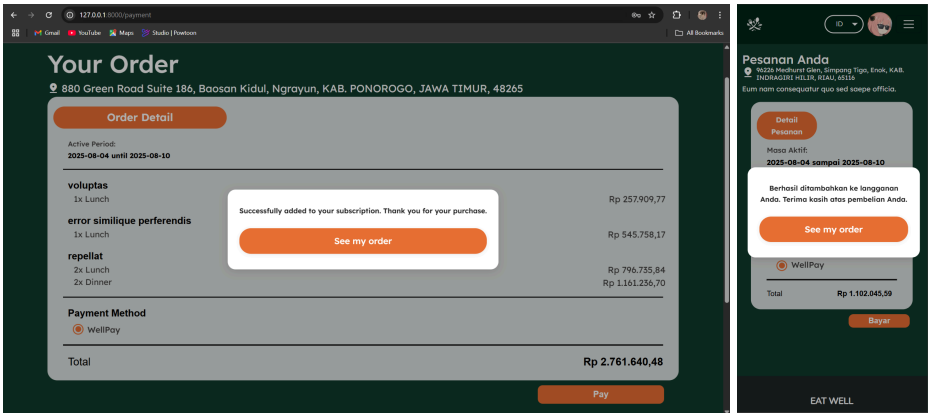


- Jika user click “Continue” dan password salah:

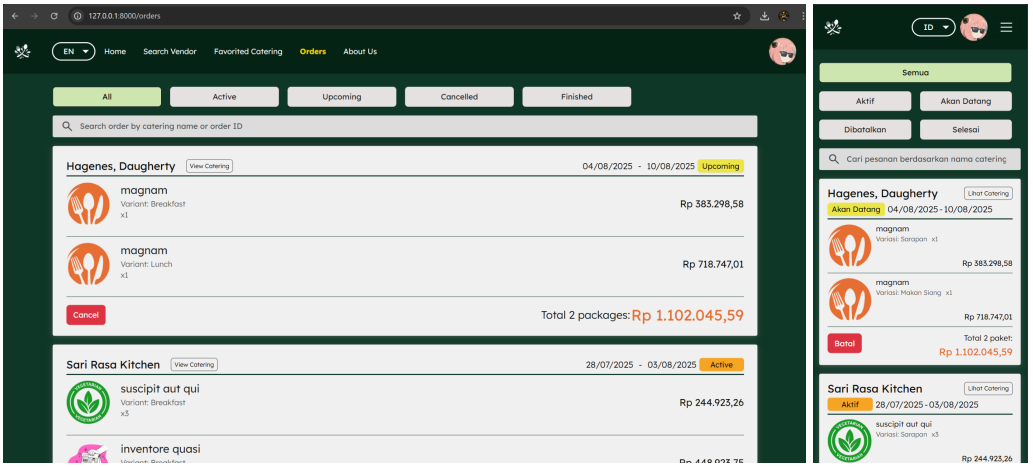


- Jika user click “Continue” dan password benar:

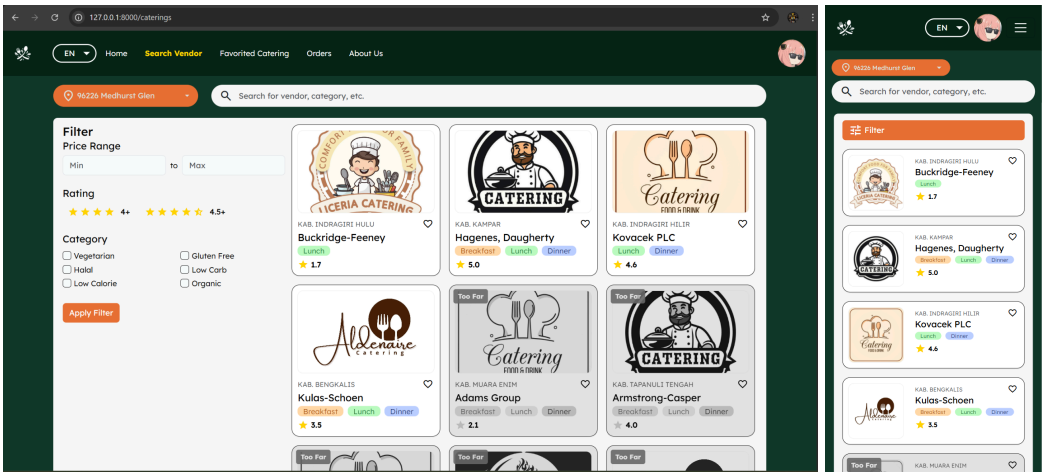




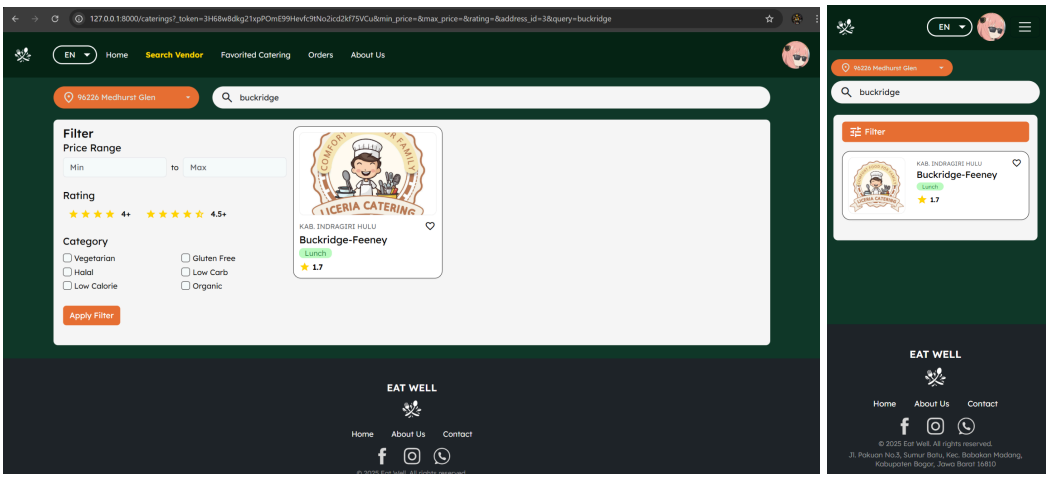
- Jika user click “See my order”:



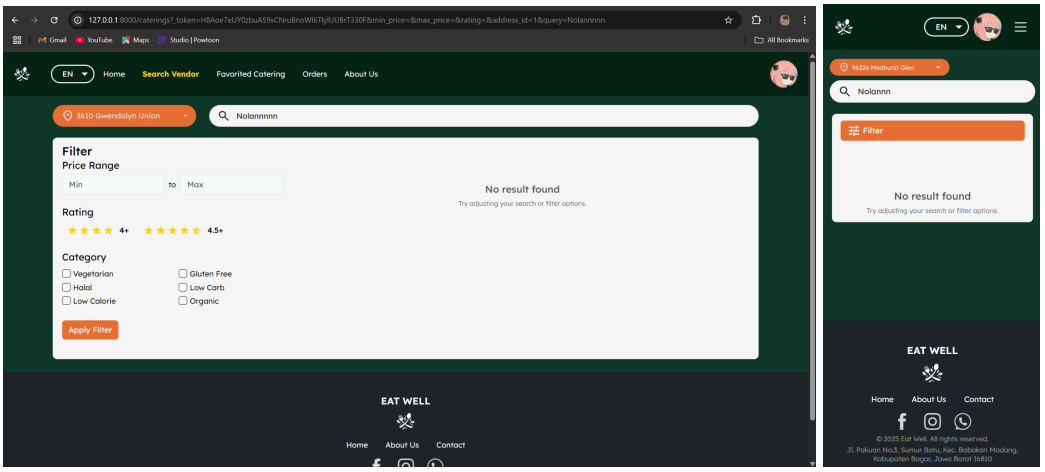
e. Search Catering



- Jika hasil search muncul:

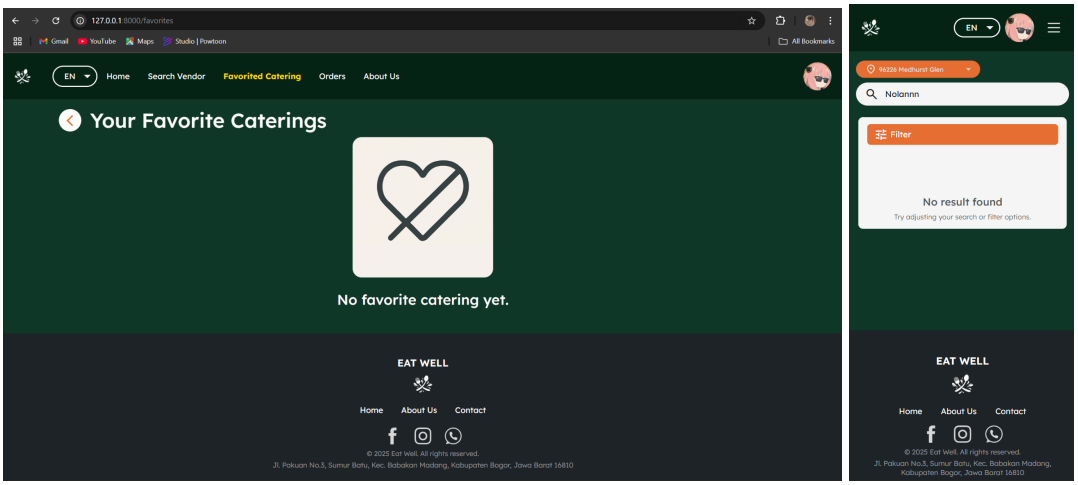


- Jika hasil search tidak muncul:

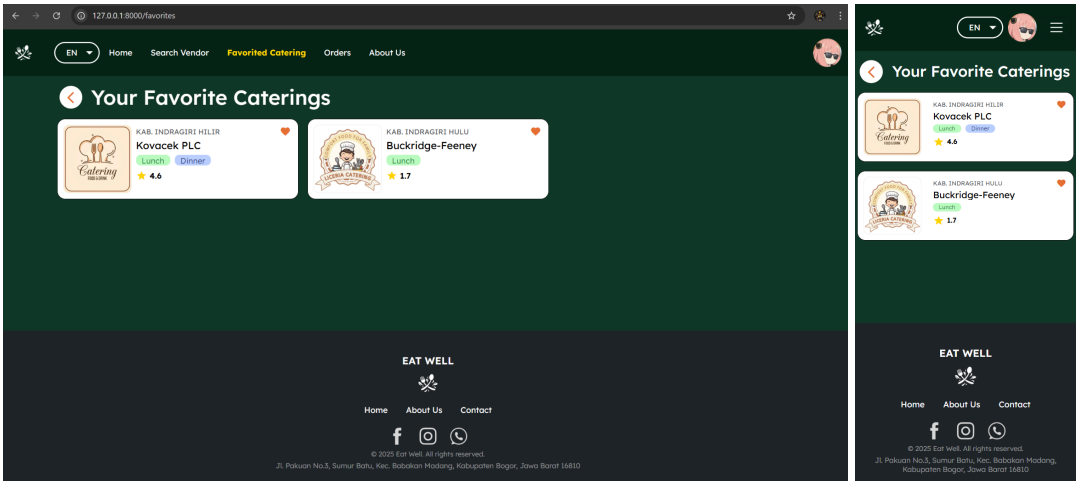


f. Favorite Catering

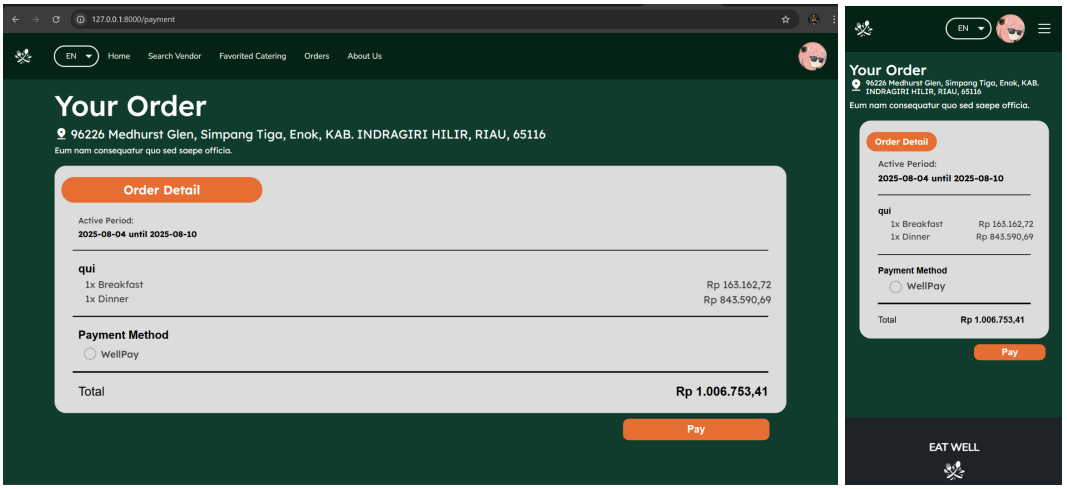
- Jika belum ada favorite catering:



- Jika ada favorite catering:

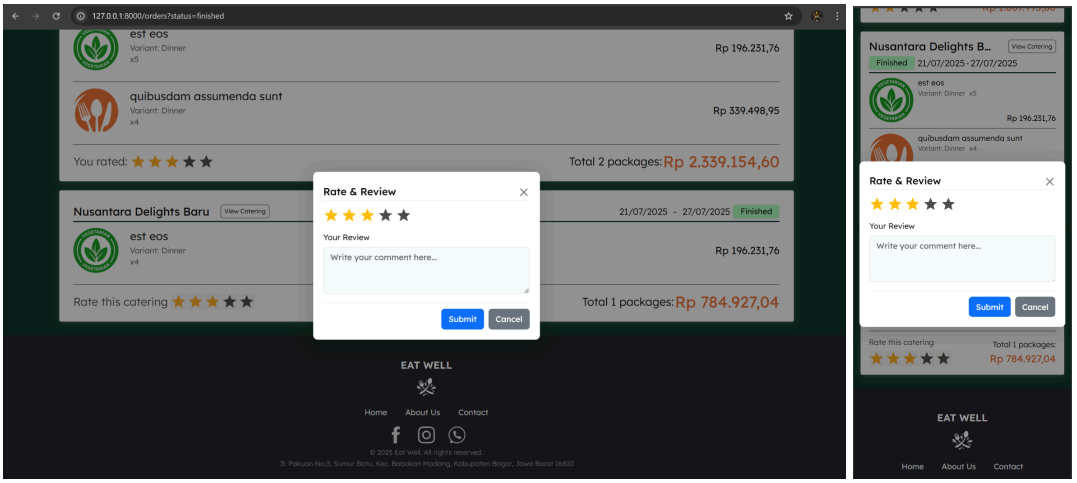


g. Checkout Cart

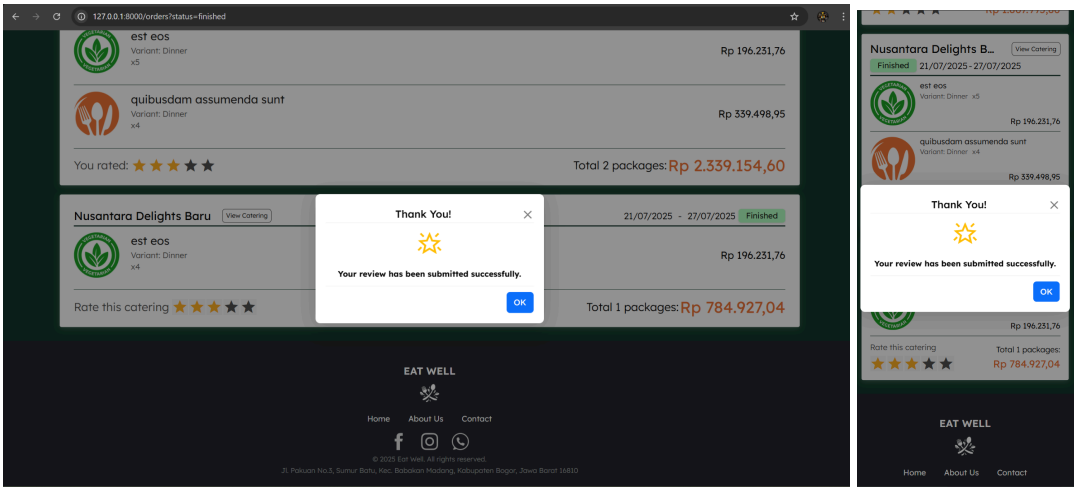


h. Rate & Review

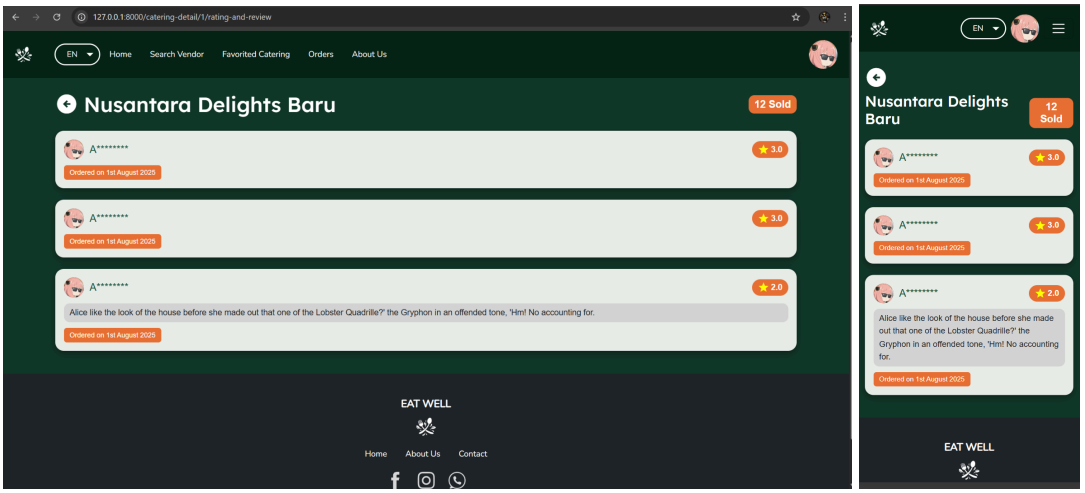
- User hanya bisa review order yang sudah finished.



- Jika user click “Cancel”, pop up nya akan tertutup. Jika user click “Submit”:

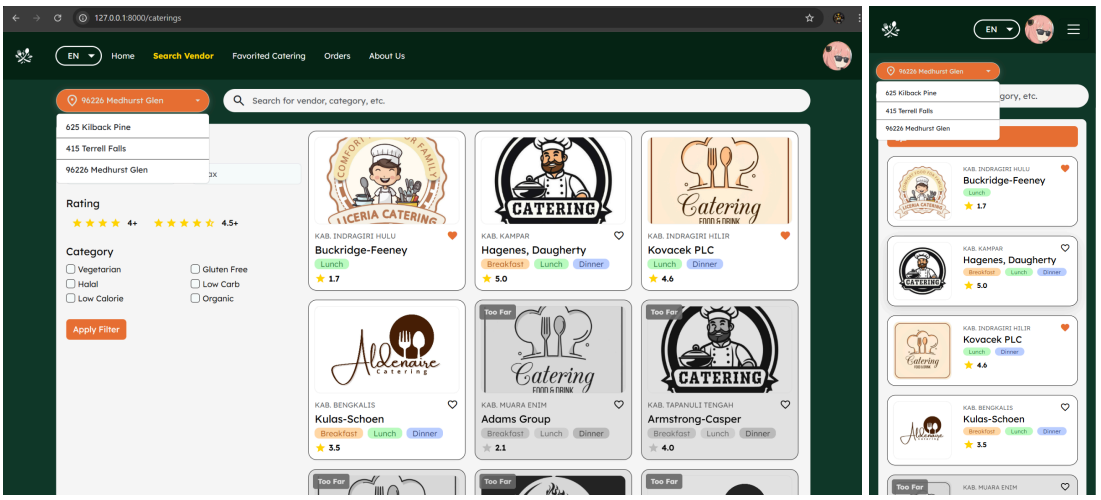


- Setelah di submit dan berhasil, review akan muncul di page rating & review catering terkait:

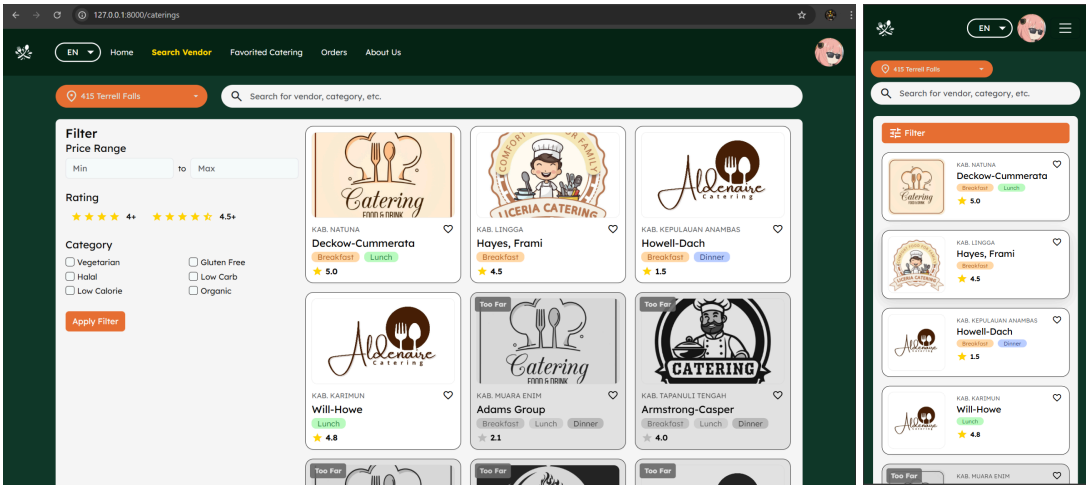


i. Change Address

- Jika dropdown untuk mengganti alamat (di samping search field) dipilih:

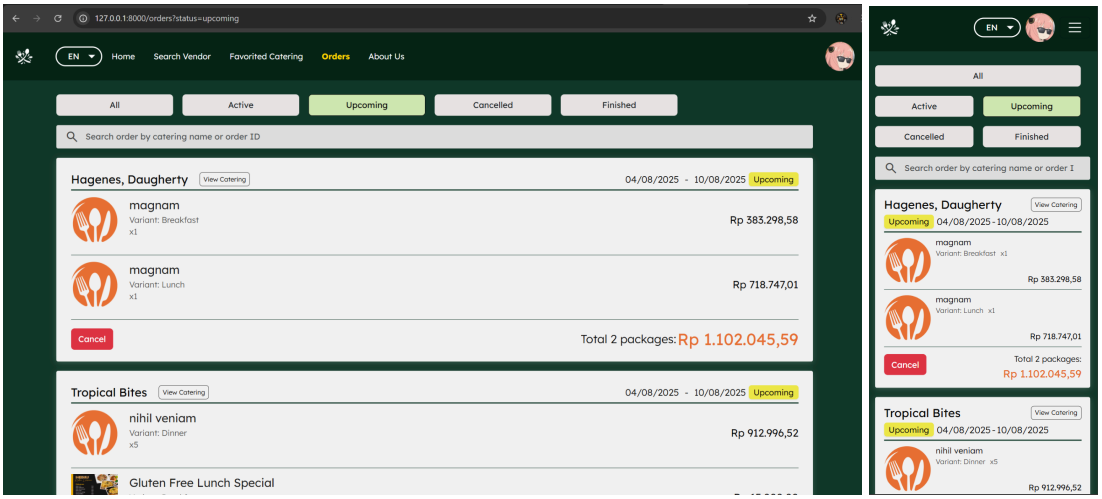


- Jika alamat lain dipilih:

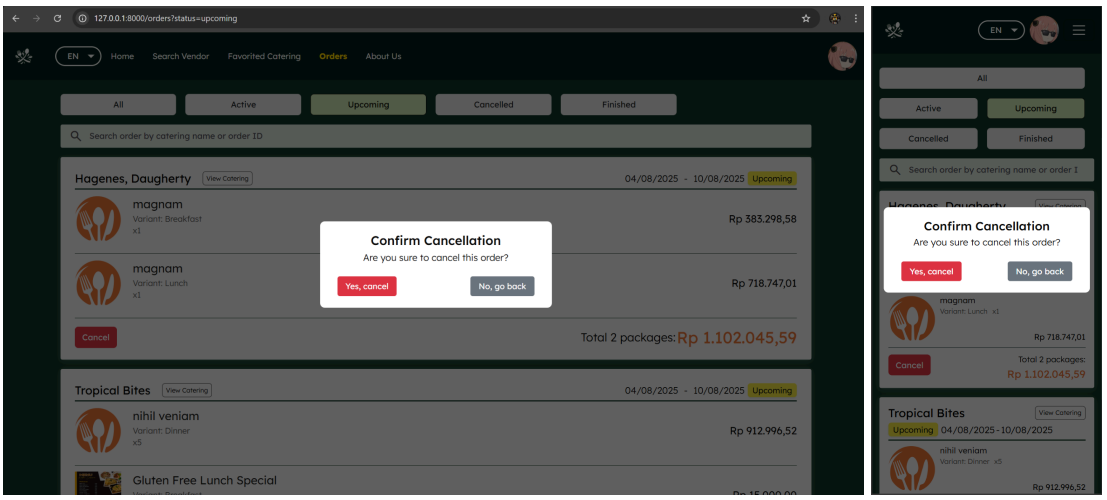


j. Cancel Active Order

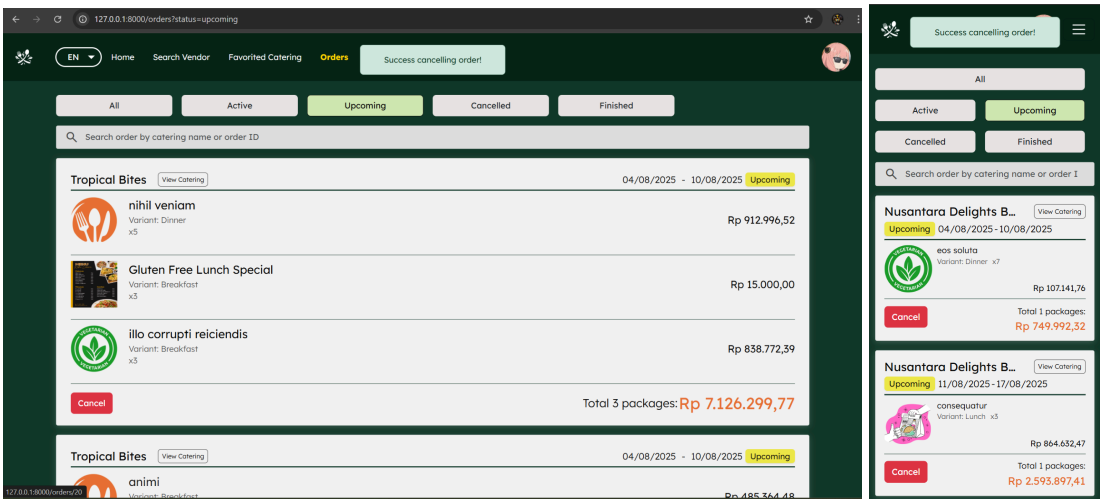
- User hanya bisa cancel upcoming order.



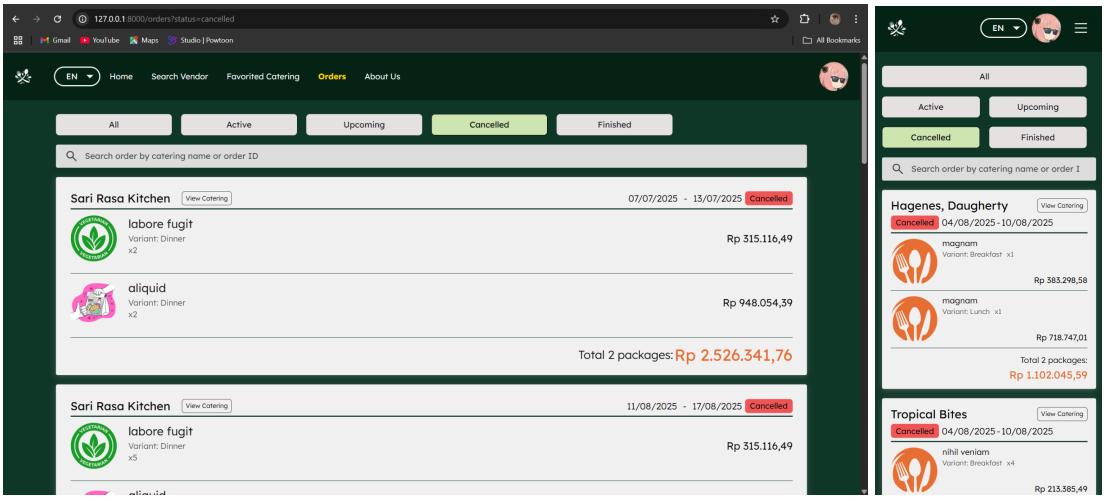
- Jika user click “Cancel”:



- Jika user click “No, go back”, pop up akan tertutup. Jika user click “Yes, cancel”, akan muncul pesan success di bagian atas:

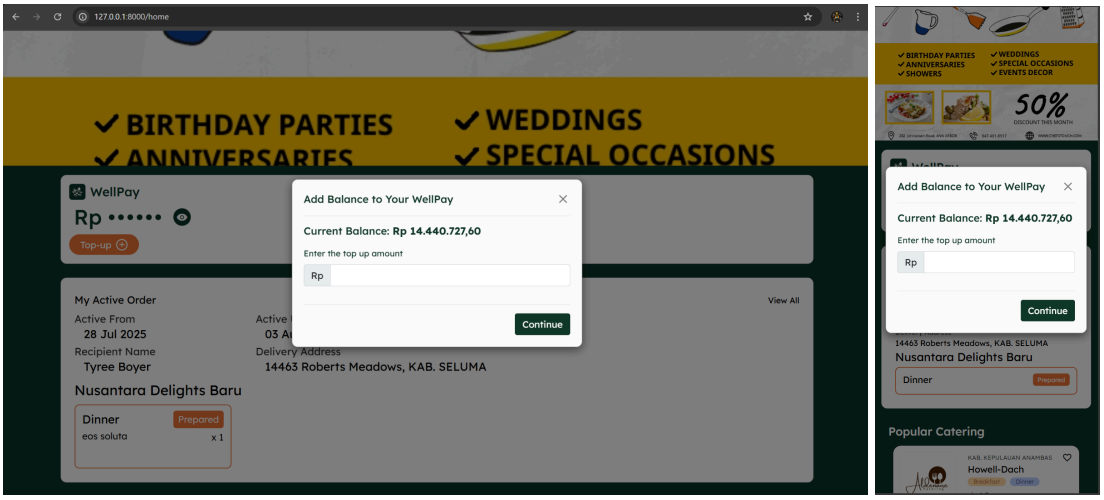


- Order yang di cancel akan muncul di tab “Cancelled”:

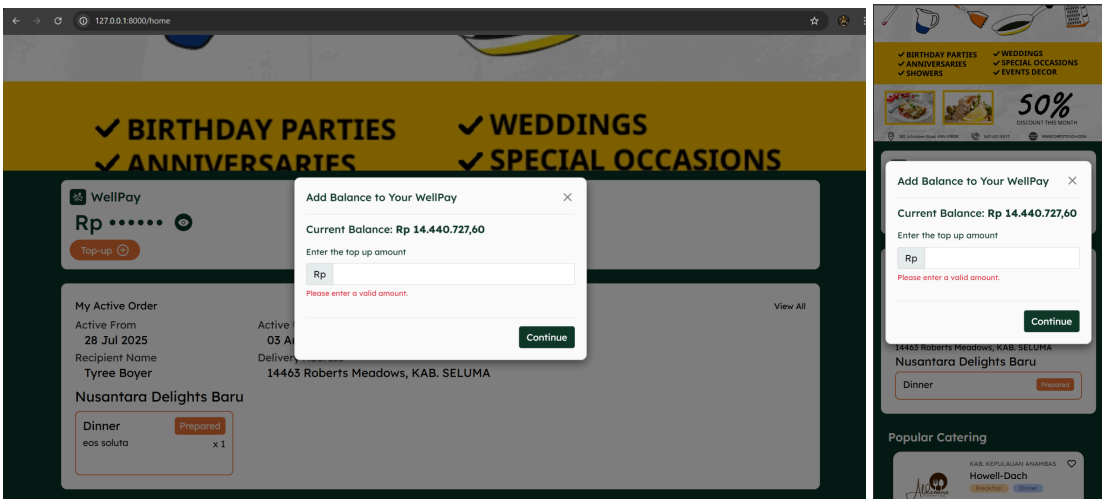


k. Top-Up WellPay

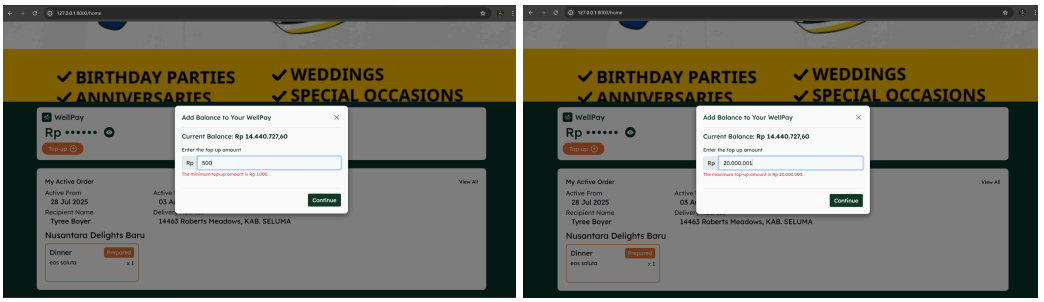
- Jika user click “Top up”:



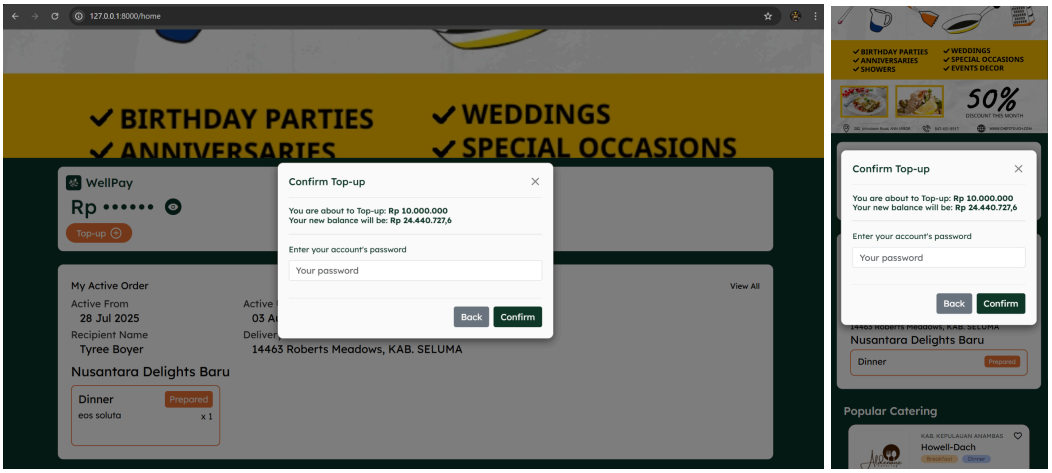
- Jika user click “Continue” tanpa mengisi field:



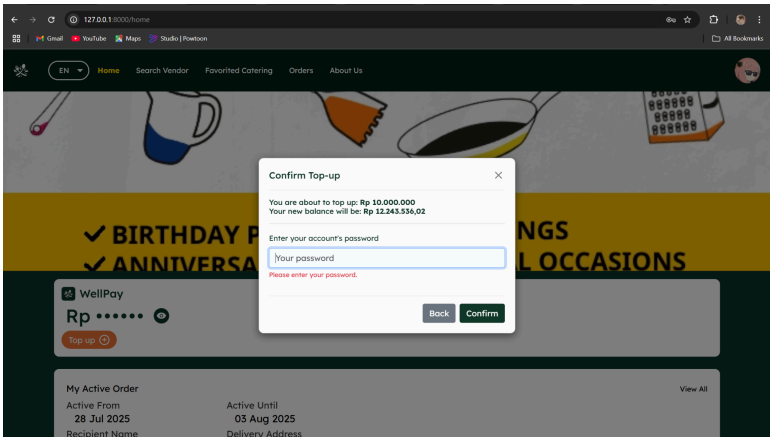
- Validasi saat top up:



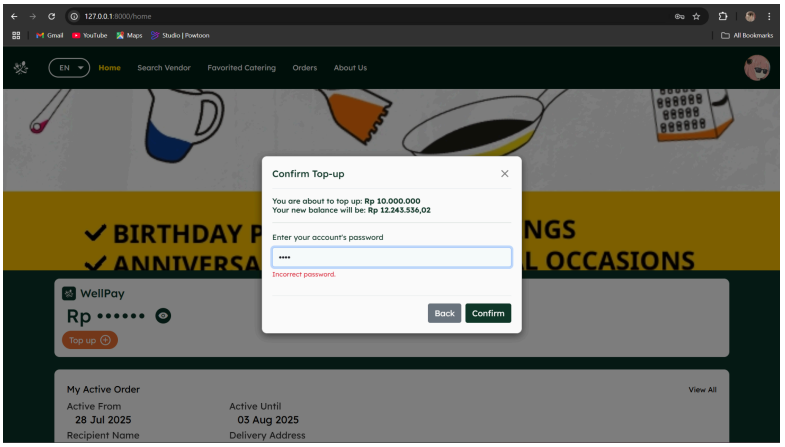
- Jika user top up dengan saldo >= 1.000 dan <= 20.000.000 (misal: 10.000.000):



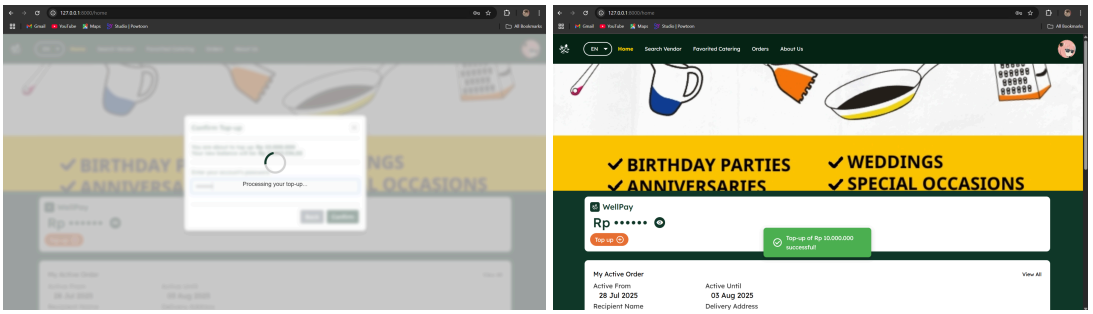
- Jika user tidak memasukkan password dan click “Confirm”:



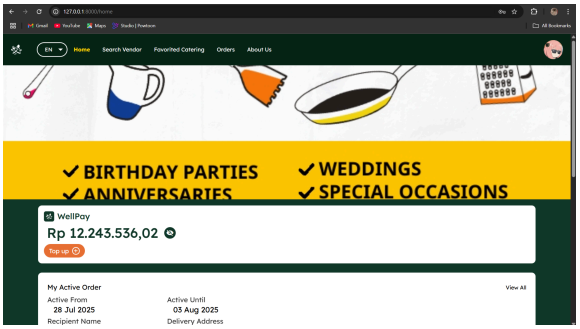
- Jika user memasukkan password yang salah:



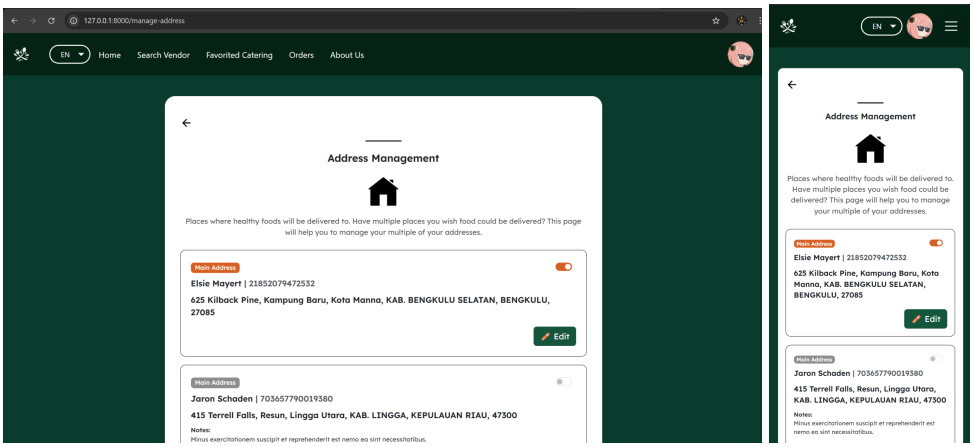
- Jika user memasukkan password yang benar:



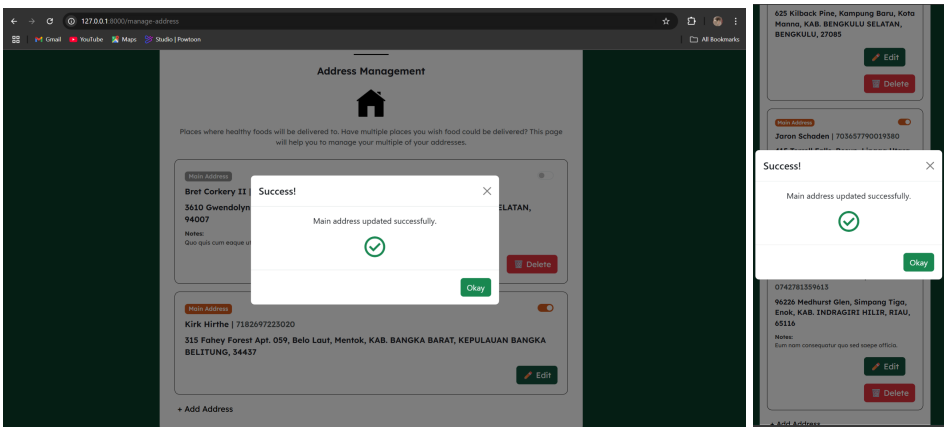
- Jika user ingin melihat saldo (click ikon mata):



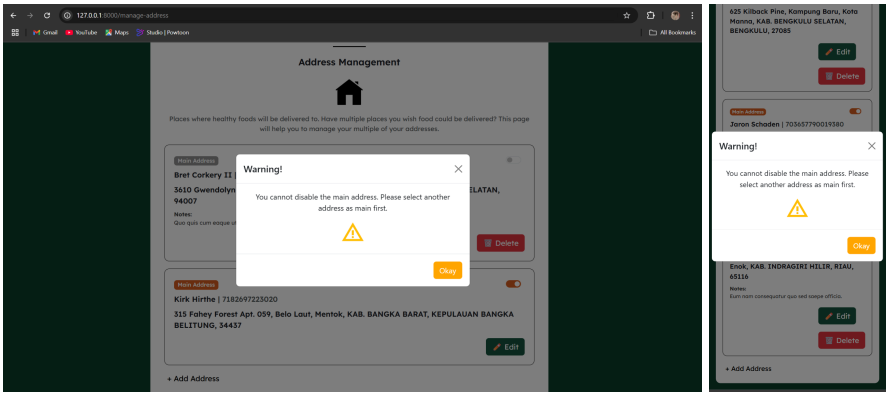
l. Manage Address



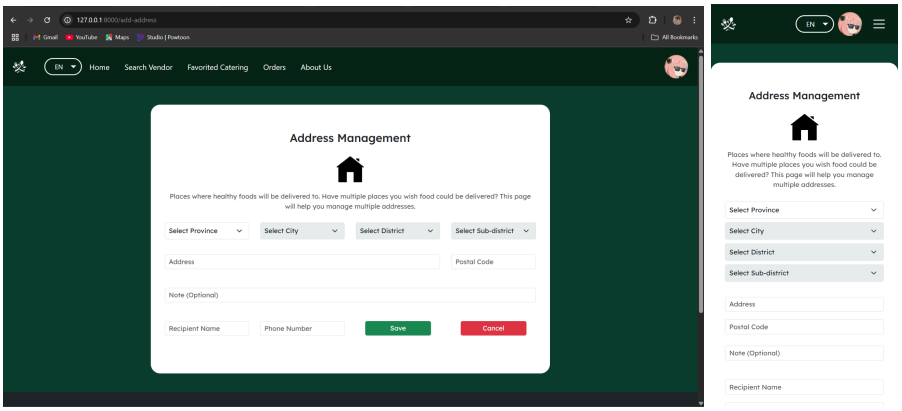
- Jika user ingin mengganti main addressnya dengan toggle bagian kanan atas alamat yang ingin dijadikan main address:



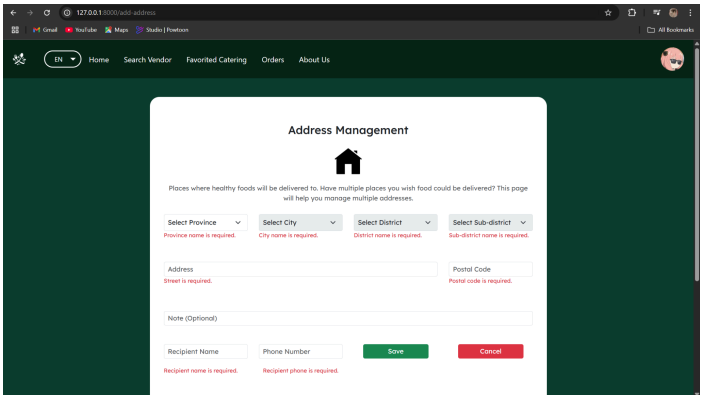
- Jika user ingin menonaktifkan main addressnya:

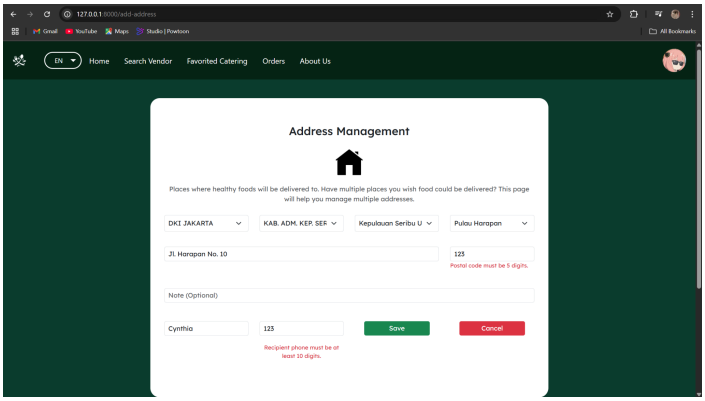


- Jika user click “Add Address”:

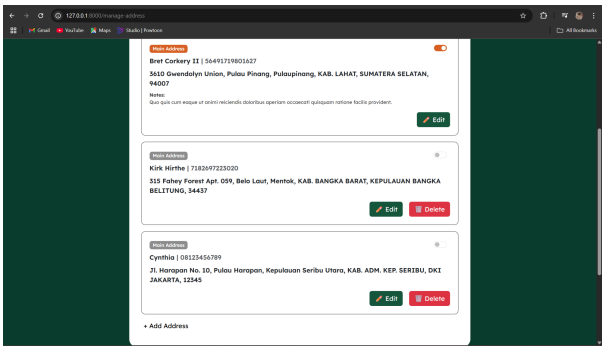


- Jika user click “Save” dan tidak lolos validasi:

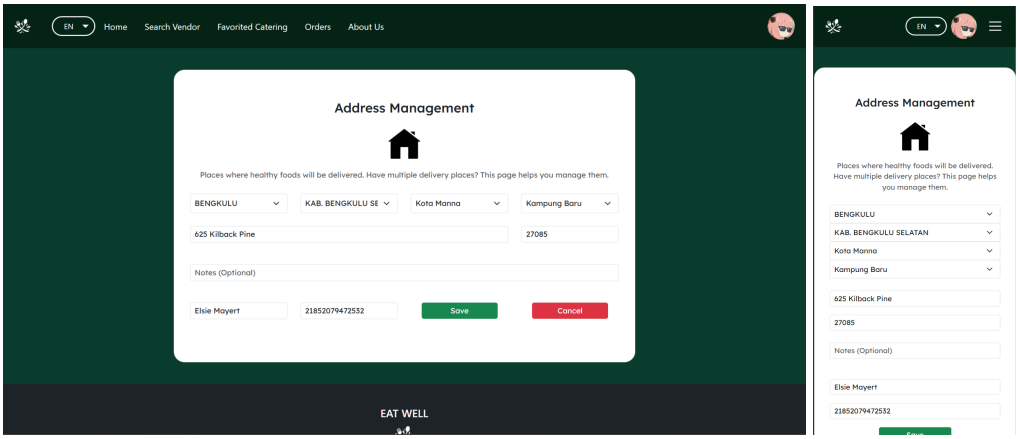




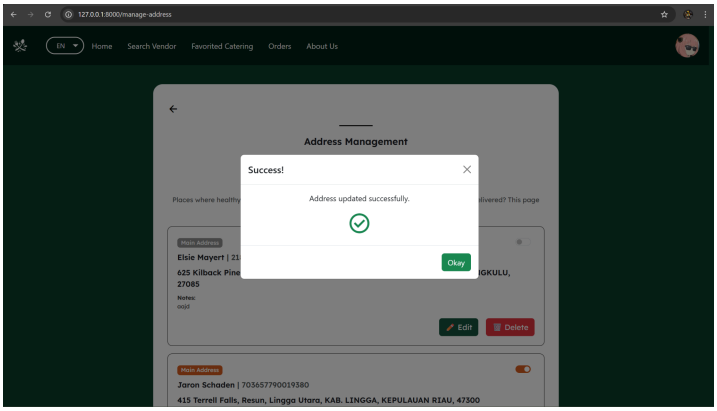
- Jika user click “Save” dan lolos validasi, maka address baru akan muncul di manage address:



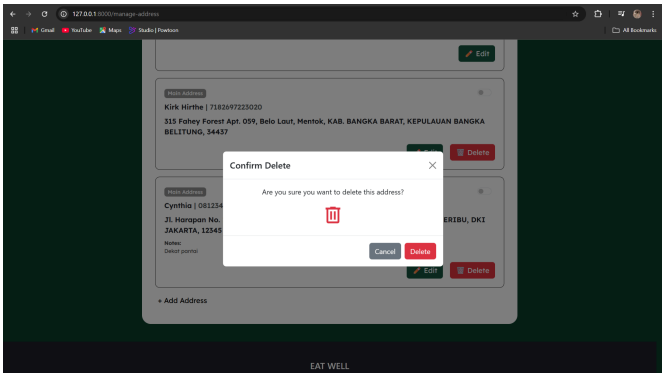
- Jika user click “Edit”:



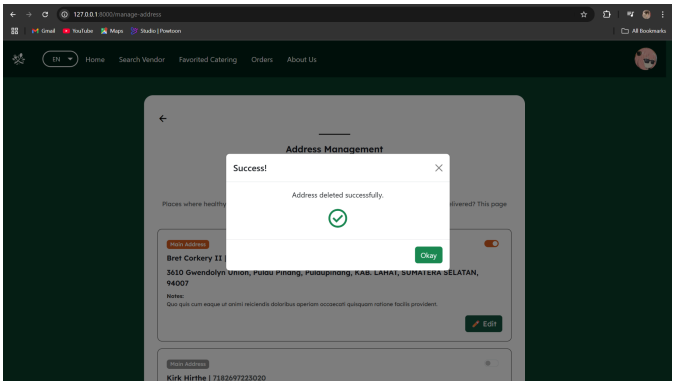
- Jika user click “Save”, akan muncul popup dan alamat diperbarui:

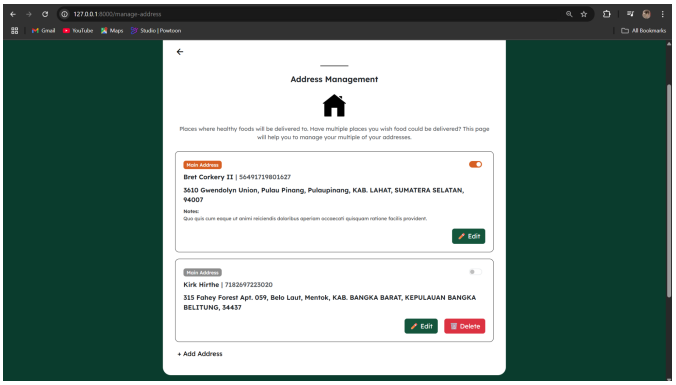


- Jika user click “Delete”:



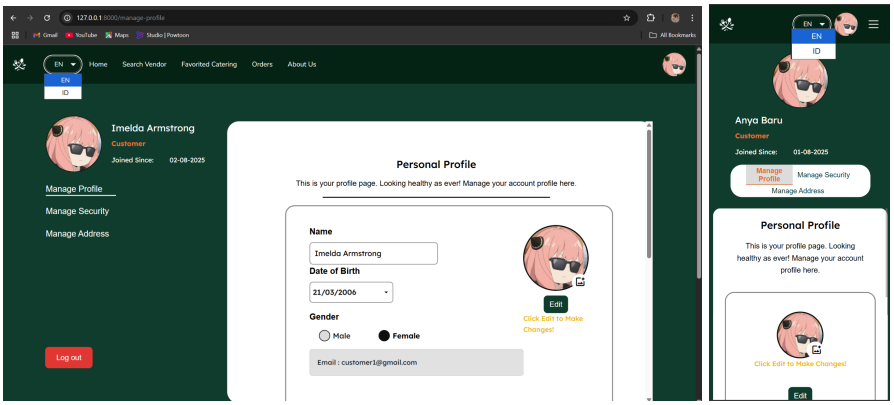
- Jika user click “Cancel”, maka pop up akan hilang. Jika user click “Delete”, pop up success akan muncul dan alamat yang dihapus akan hilang dari page tersebut:



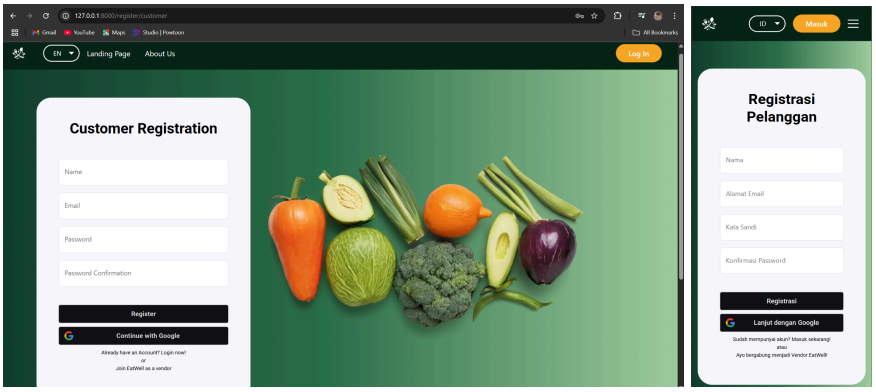


m. Change Language

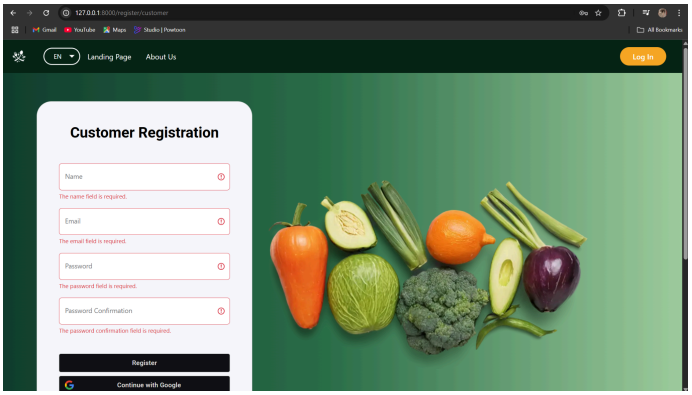
- Dropdown untuk mengganti bahasa terdapat pada navbar di bagian kiri:



n. Register

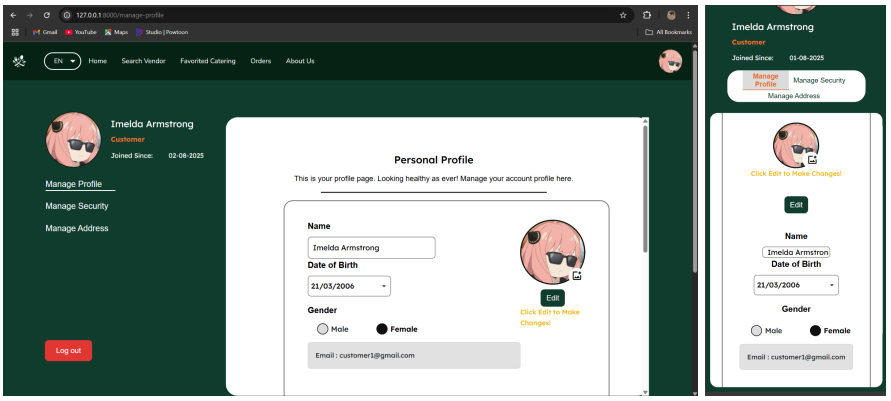


- Jika user click “Register” dan tidak lolos validasi:

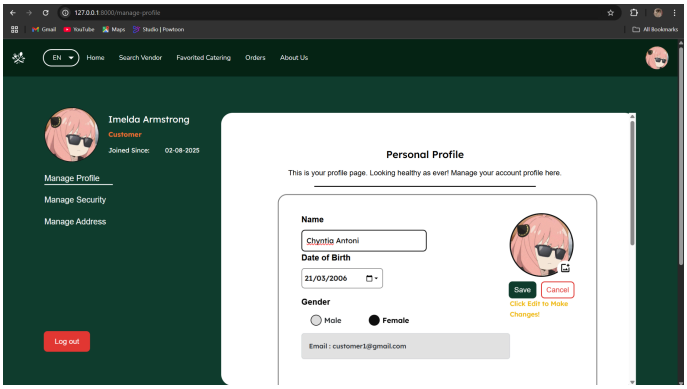


- Jika user click “Register” dan lolos validasi, maka akan diarahkan ke page Home

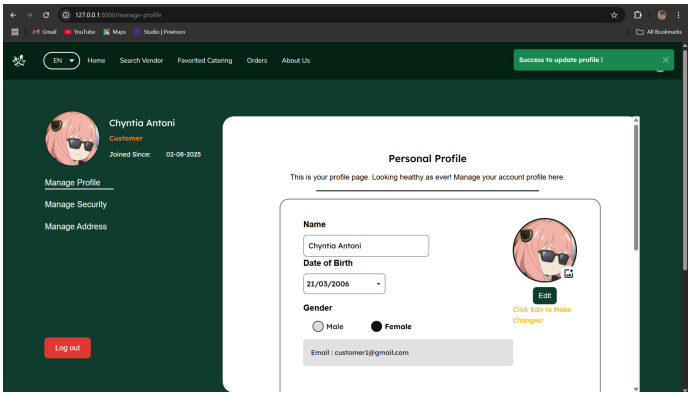
o. Manage Account Profile



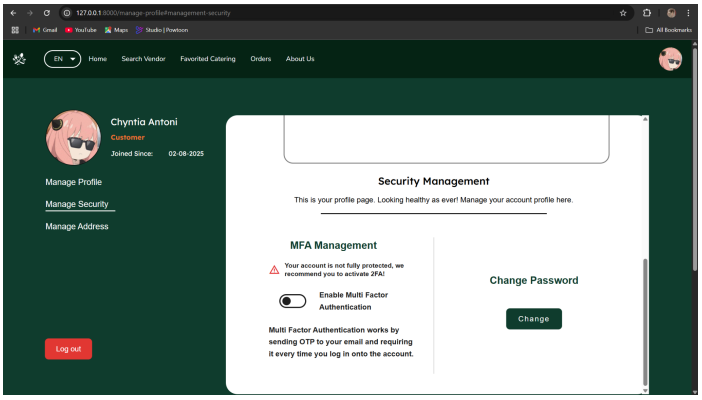
- Jika user click “Edit” dan mengubah namanya:



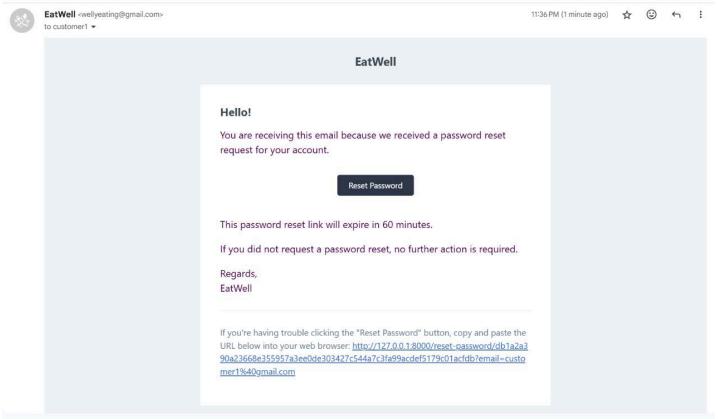
- Jika user click “Save”, nama yang diedit akan tersimpan dan muncul pesan success di bagian kanan atas:



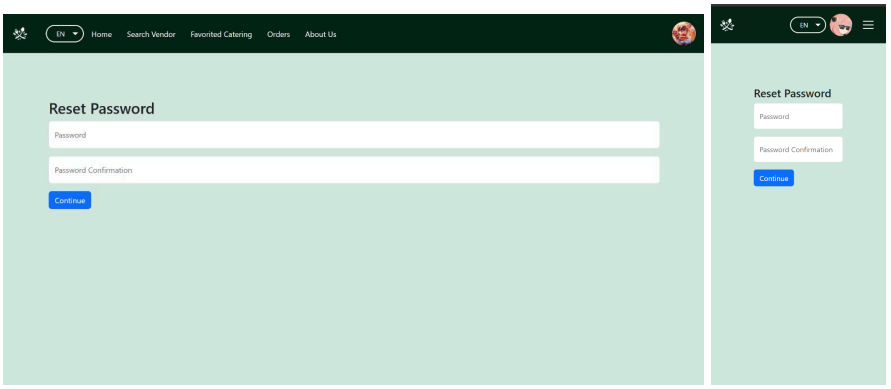
- Jika user click tab “Manage Security”:



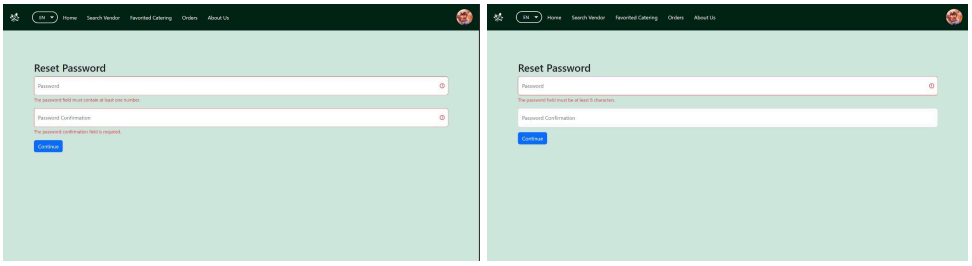
- Jika user click “Change” pada bagian Change Password, maka akan ada email yang dikirim kepada user:



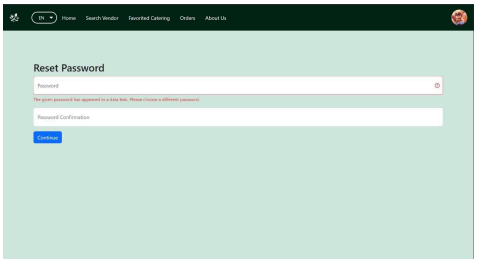
  - Jika user click “Reset Password”:



- Jika user click “Continue” dengan validasi yang salah:



- Jika user click “Continue” dengan password yang lemah:

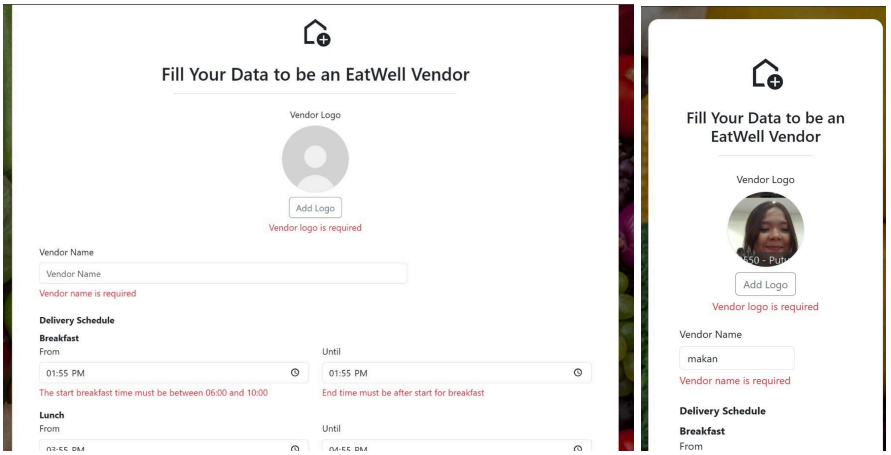


- Jika user click “Continue” dan lolos validasi, maka user akan diarahkan ke page Home.

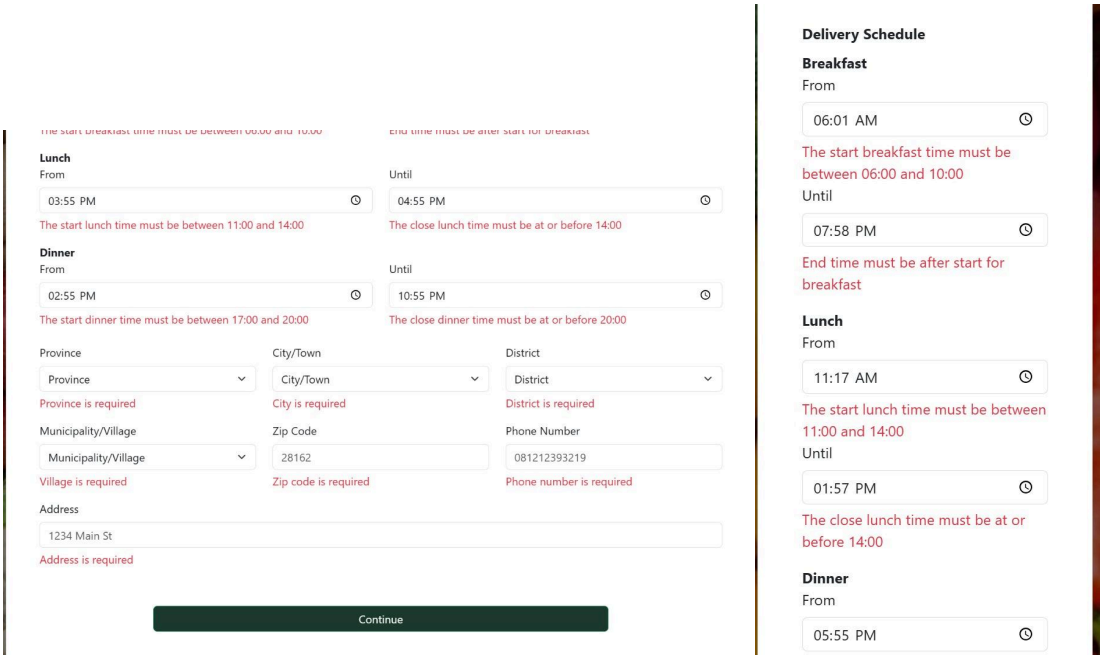
C. Catering Vendor

a. Register

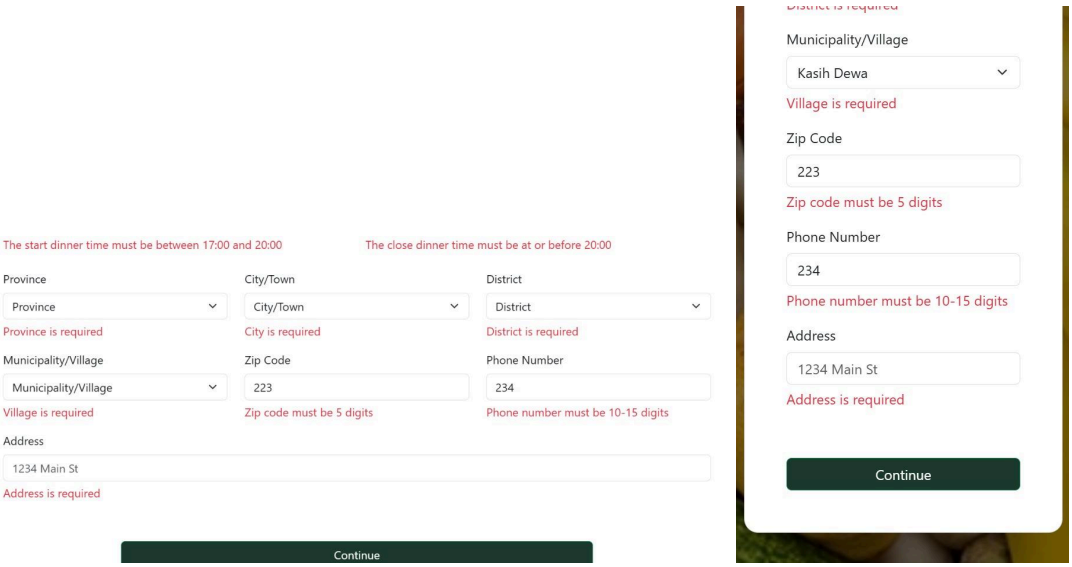
- Vendor ketika mengisi data untuk mendaftar sebagai vendor dengan validasi tiap fill



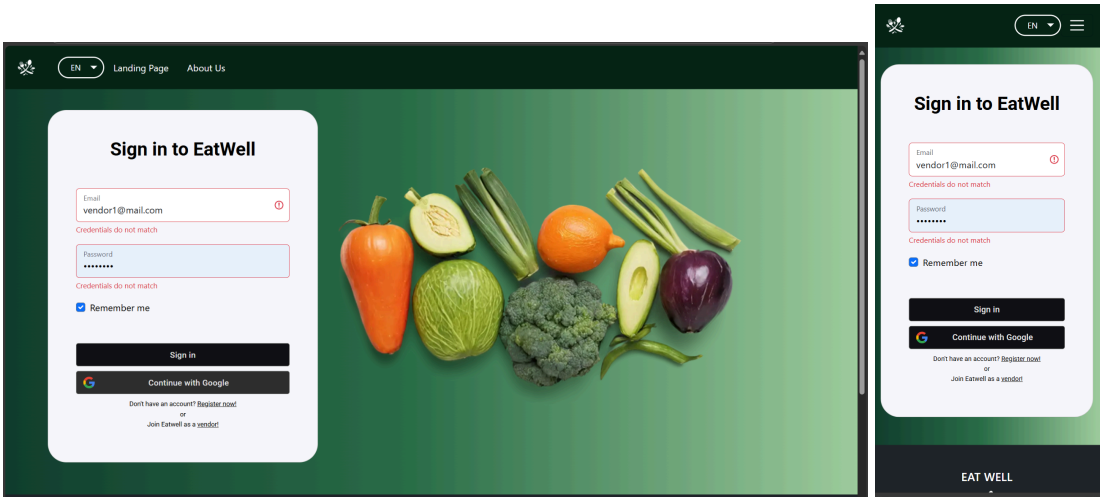
- Validasi waktu start dan end Breakfast, Lunch, and Dinner



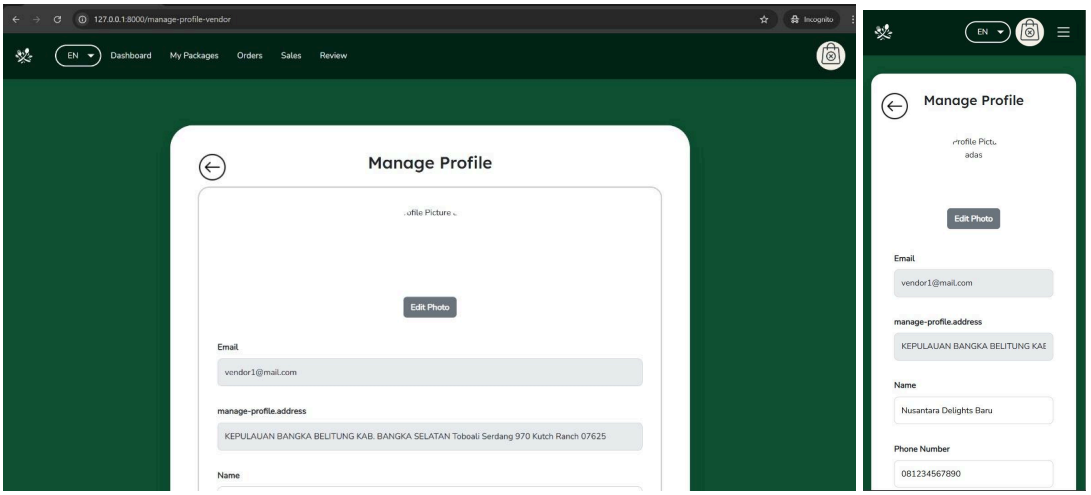
- Validasi untuk lokasi alamat vendor



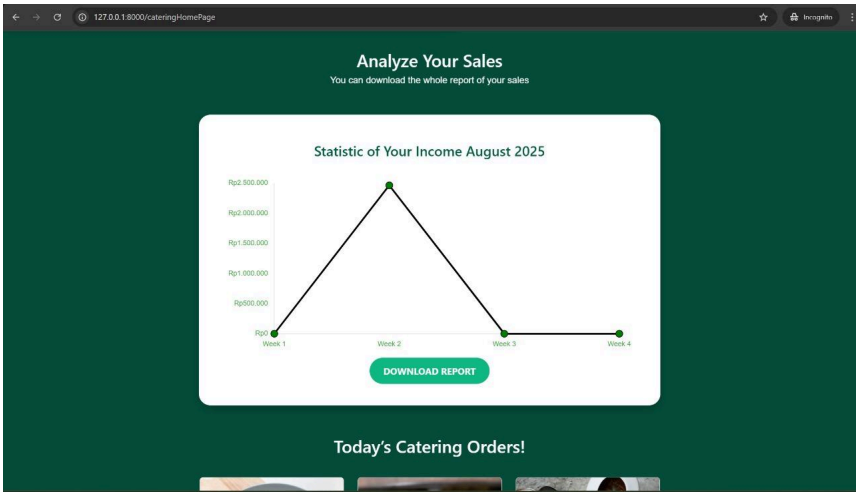
b. Login

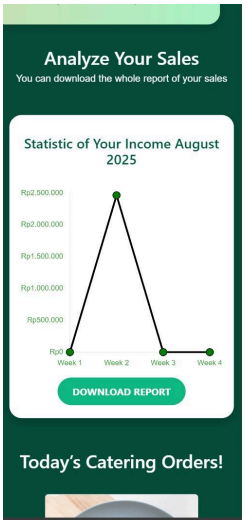


c. Manage Account Profile

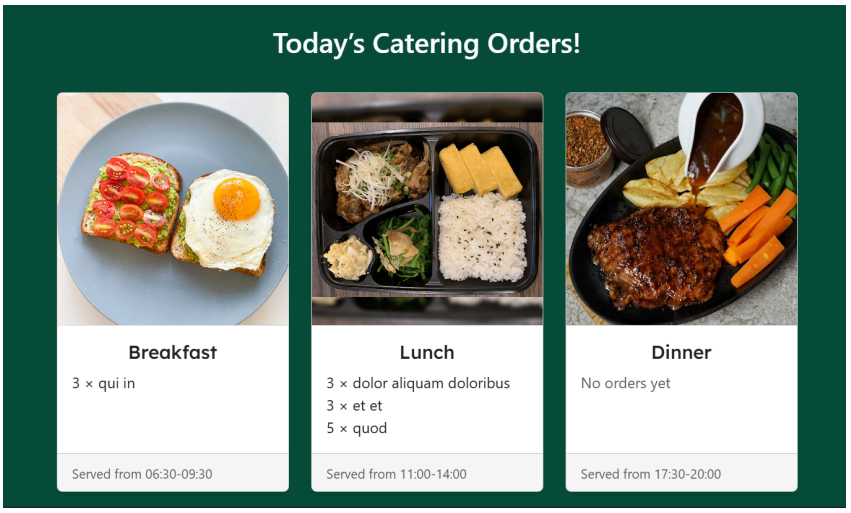


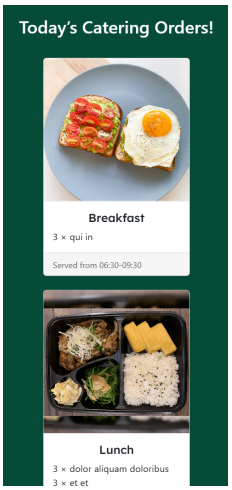
d. Generate Statistics





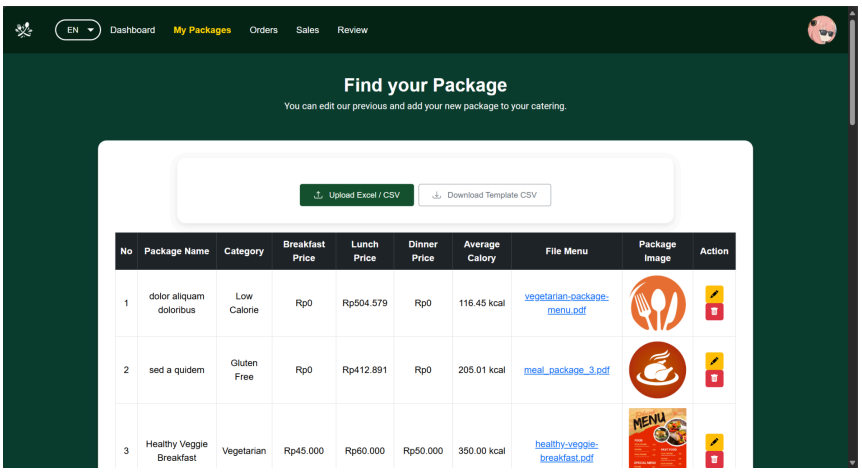
e. View Today's Order

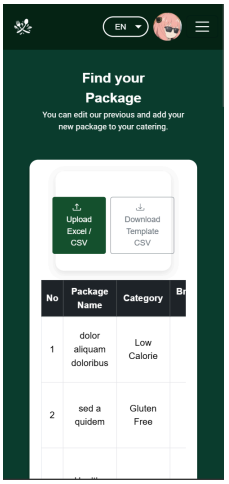




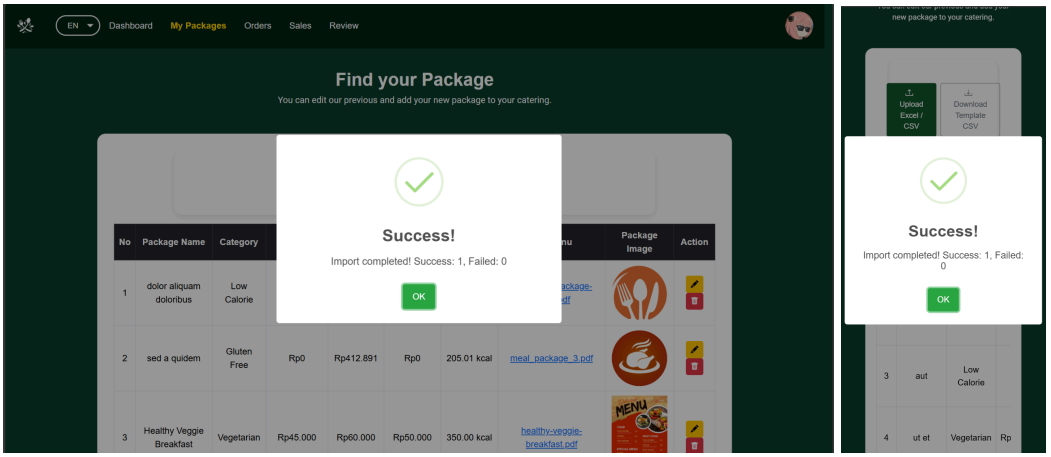
f. Manage Package/Menu

- Vendor Package Page

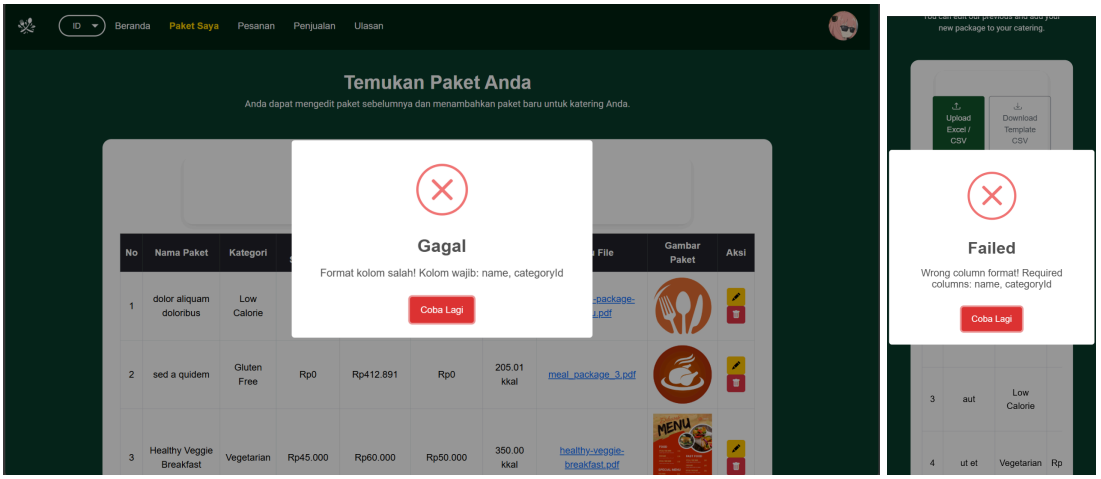




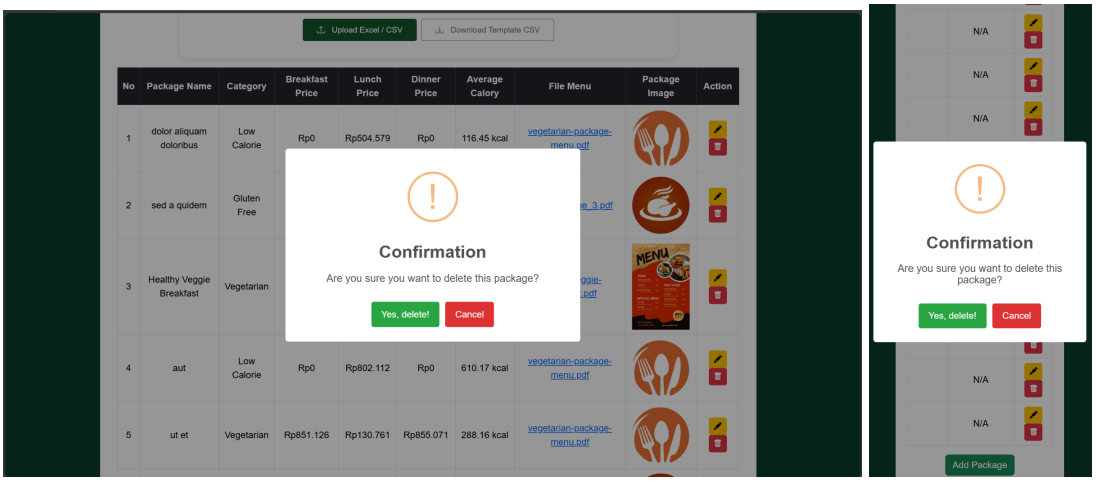
- Success Pop Up Notification ketika Upload Excel/CSV



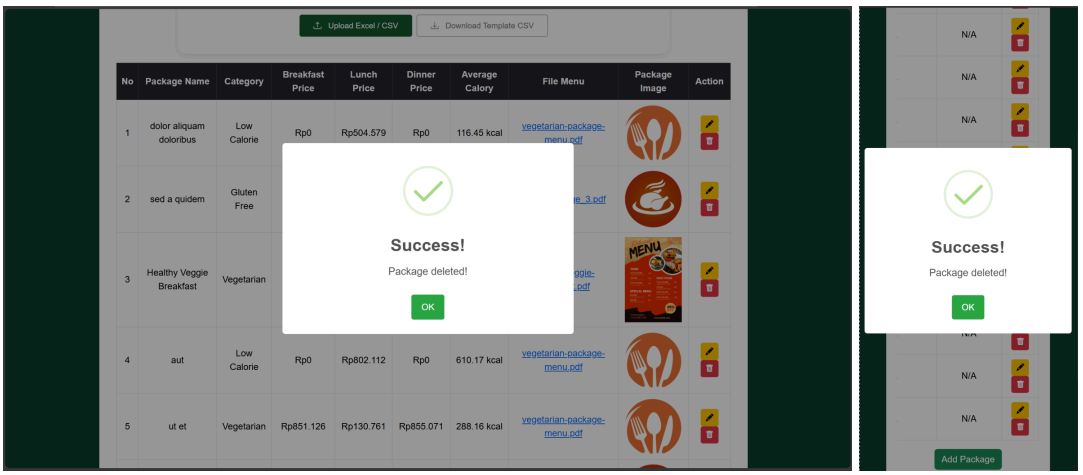
- Failed Pop Up Notification ketika Upload Excel/CSV



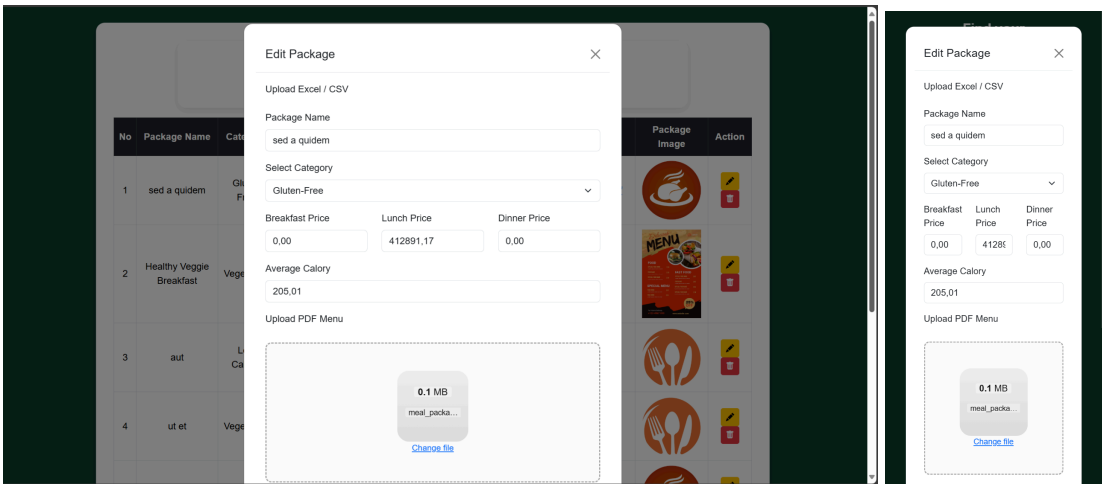
- Confirmation Pop Up Notification ketika Menghapus Paket



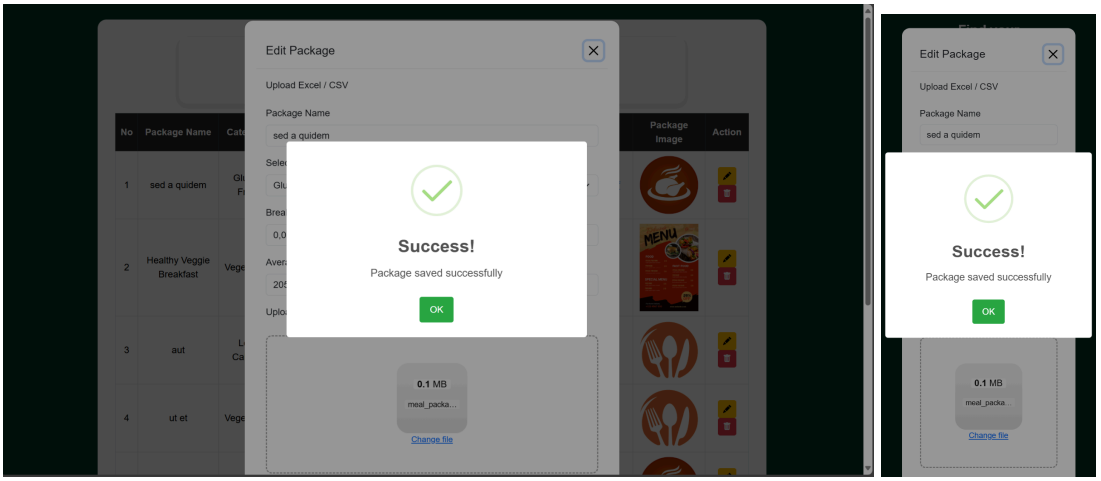
- Success Pop Up Notification ketika Paket Dihapus



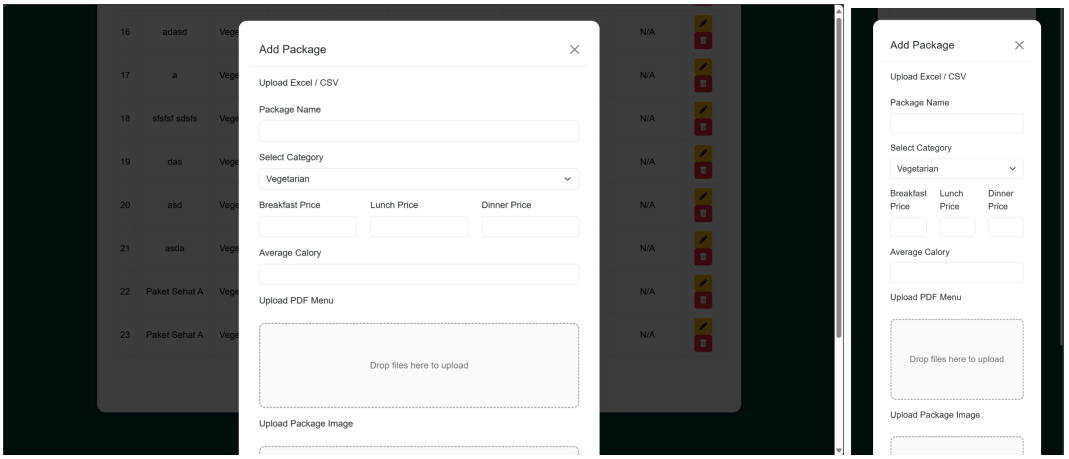
- Modal Form Edit Paket



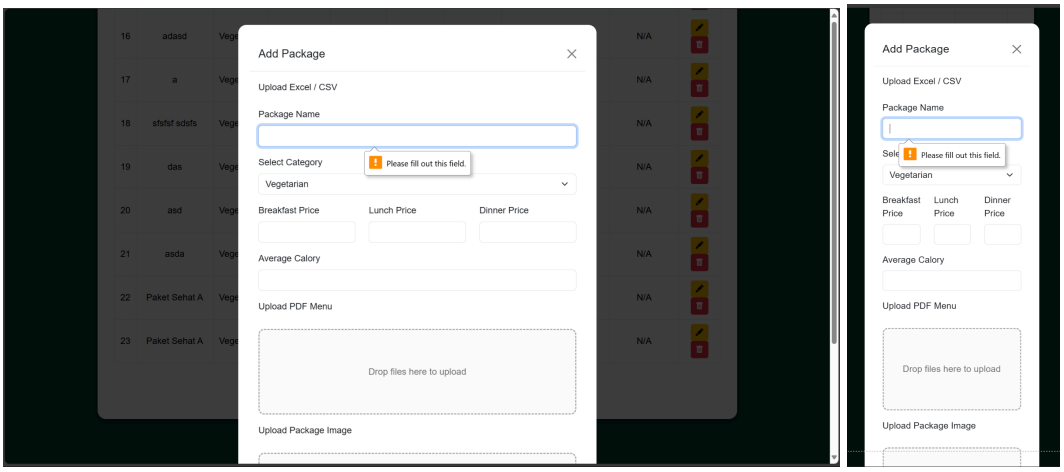
- Success Pop Up Notification ketika Paket berhasil di Update



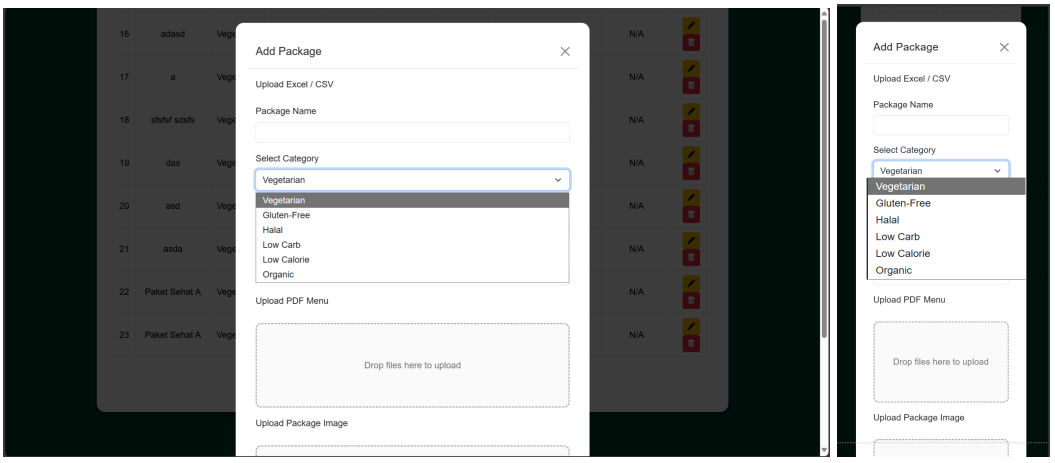
- Modal Form Add Package

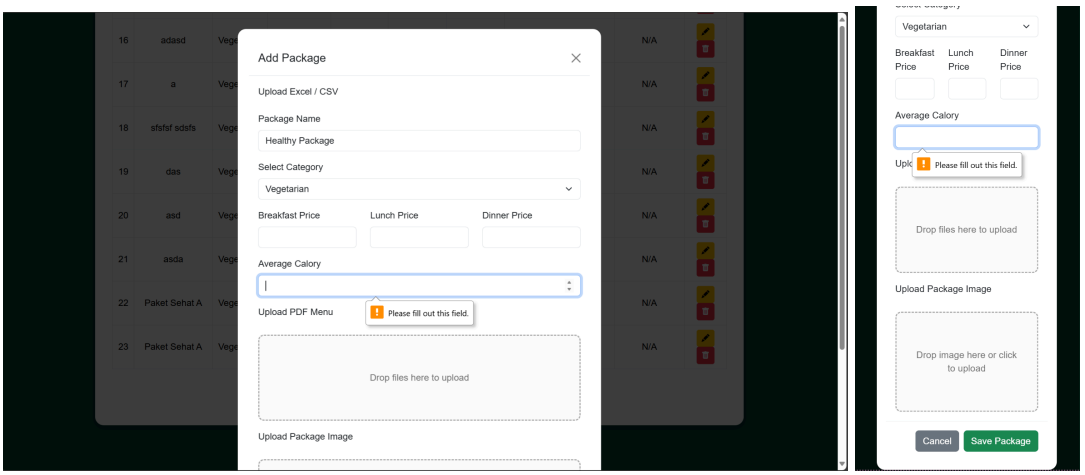


- Validasi “Nama Paket” pada Add Package Modal Form

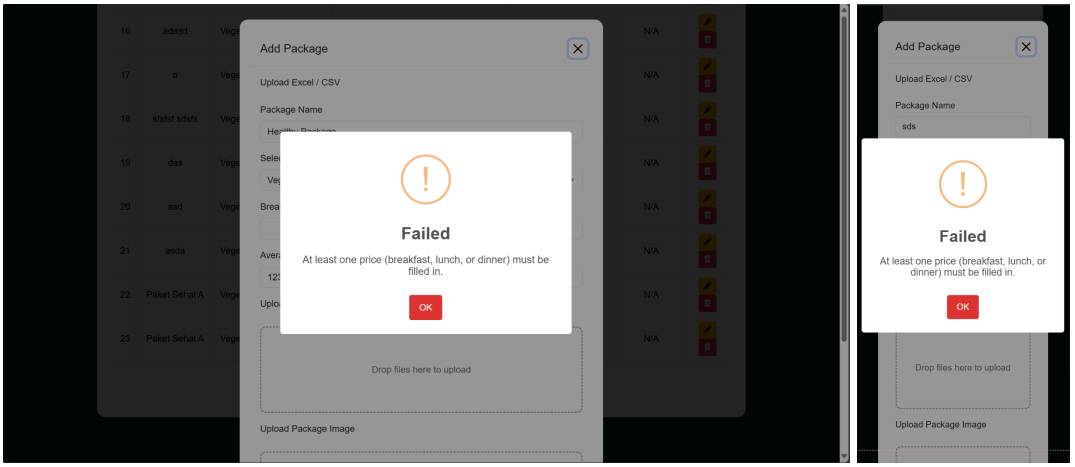


- Dropdown Select Categories

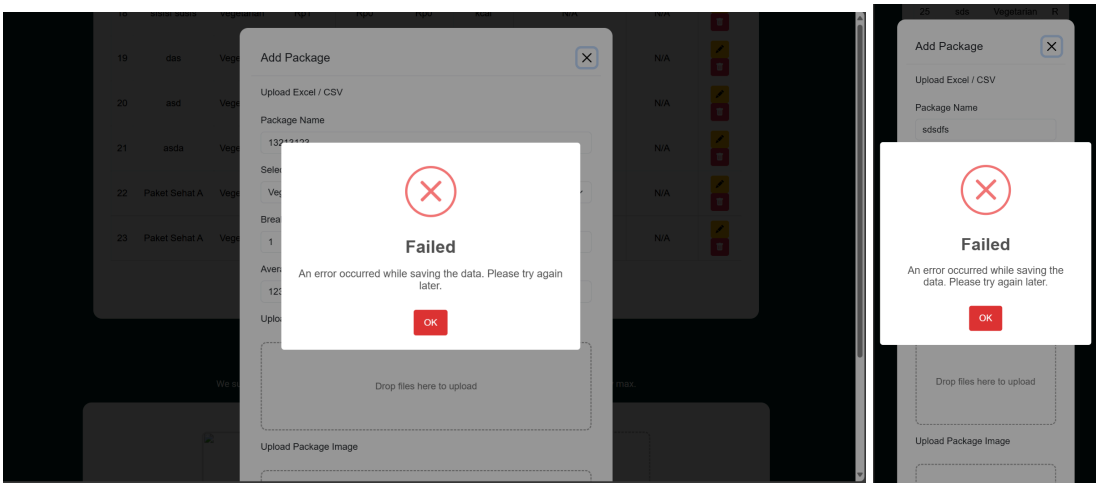




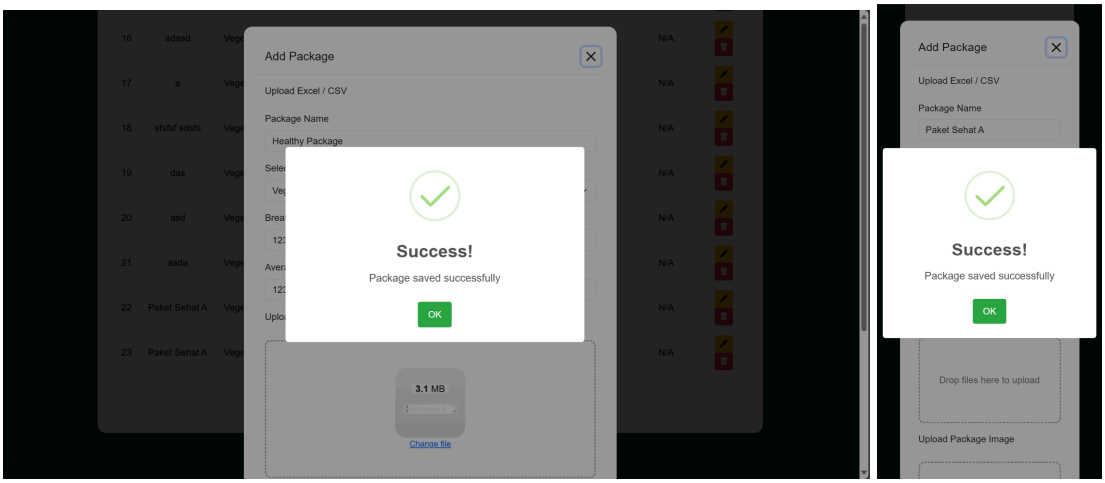
- Warning Failed karena mewajibkan untuk mengisi atleast 1 antara Breakfast Price atau Lunch Price atau Dinner Price ketika Add Package Modal Form, memastikan kalau ada paket.



- Failed Pop Up Notification ketika Paket Ditambah

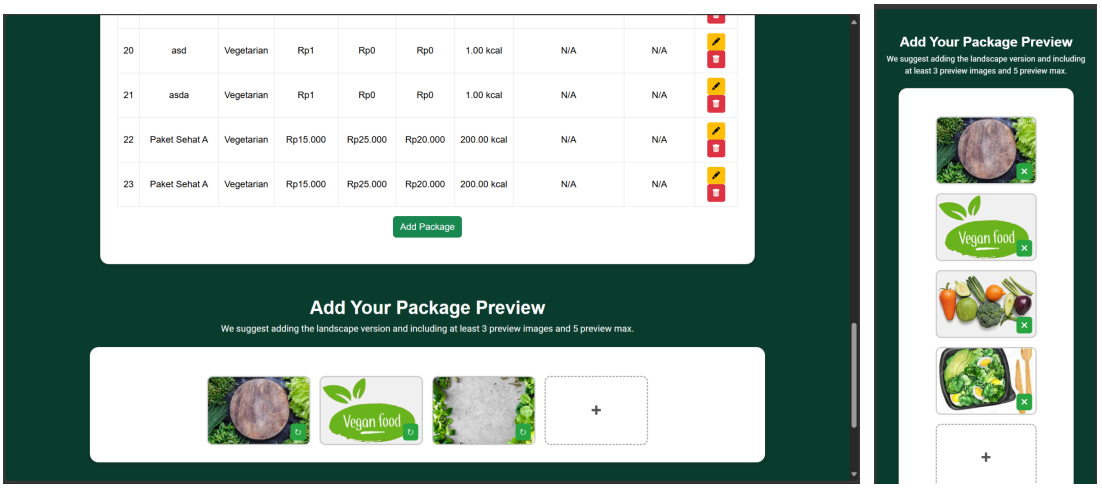


- Success Pop Up Notification ketika Paket Ditambah

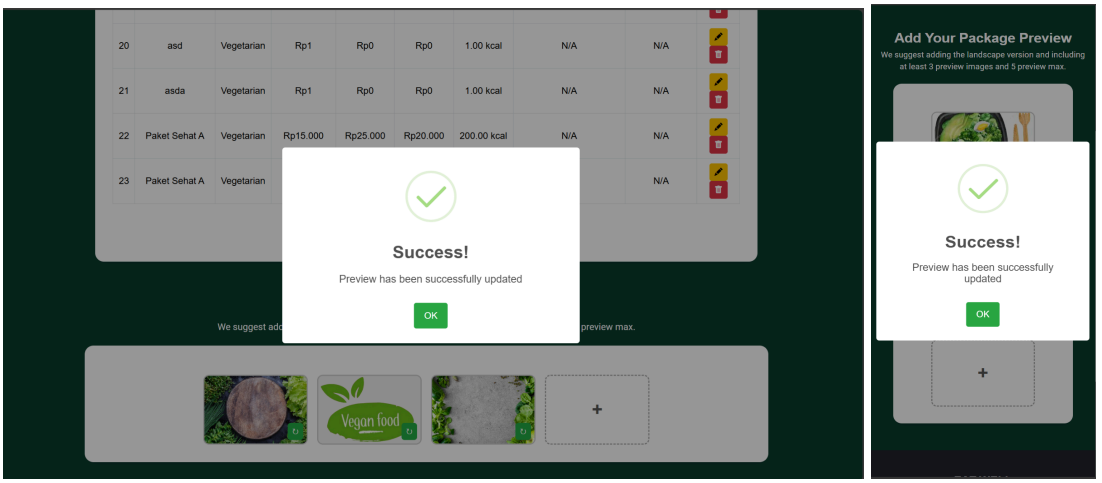


g. Manage Preview Vendor

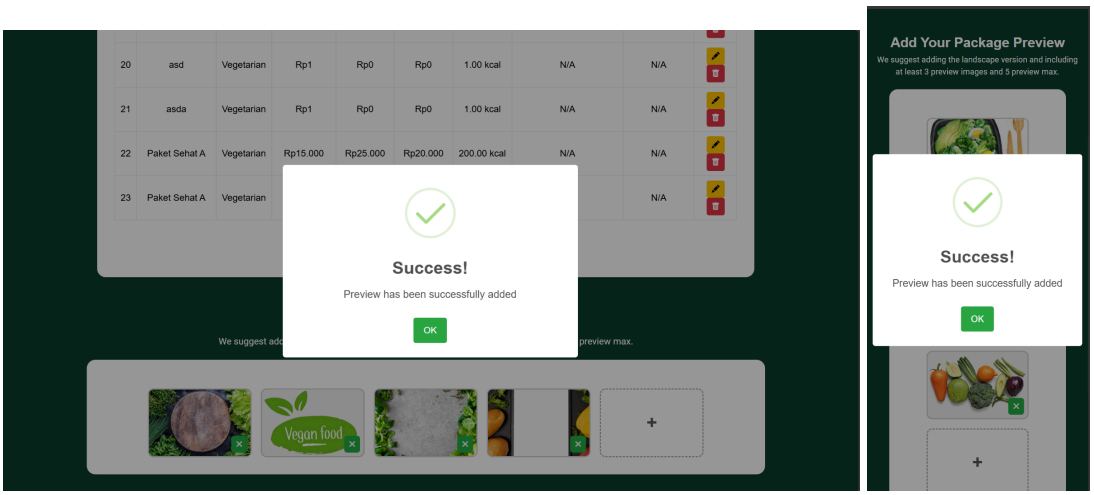
- Preview Vendor Ketika 3 images (tombol yang muncul adalah tombol replace, karena mewajibkan ada 3 minimal gambar)



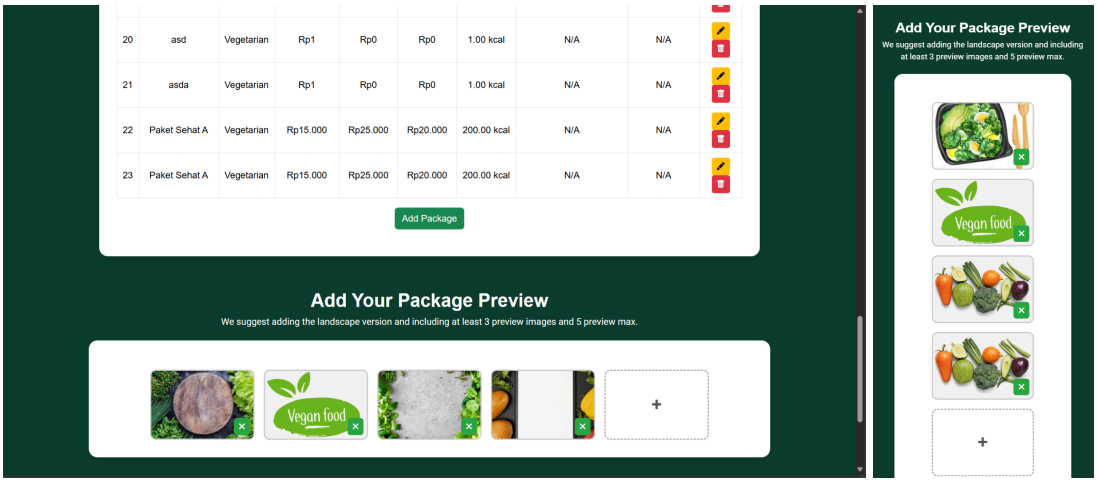
- Success Pop Up Notification ketika Preview di Replace



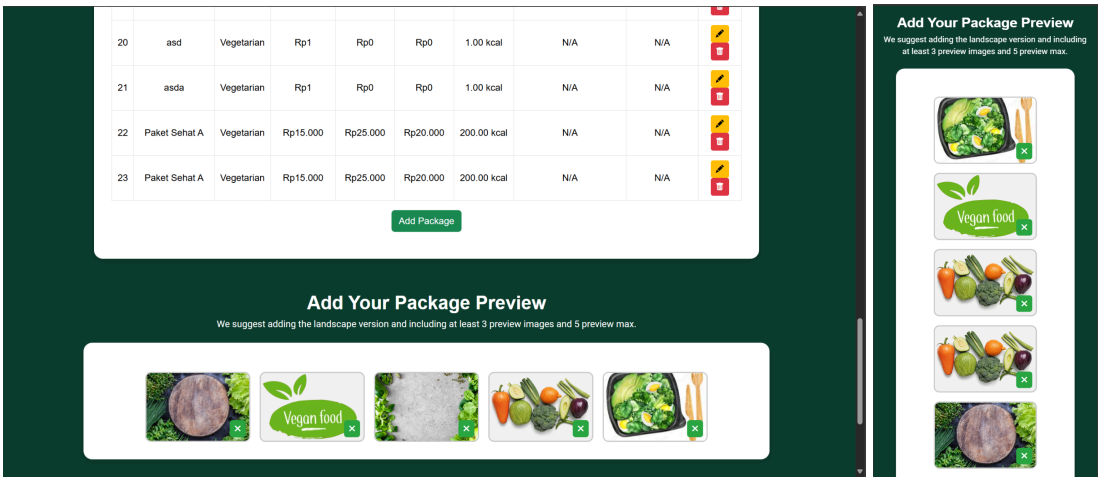
- Success Pop Up Notification ketika Preview di add



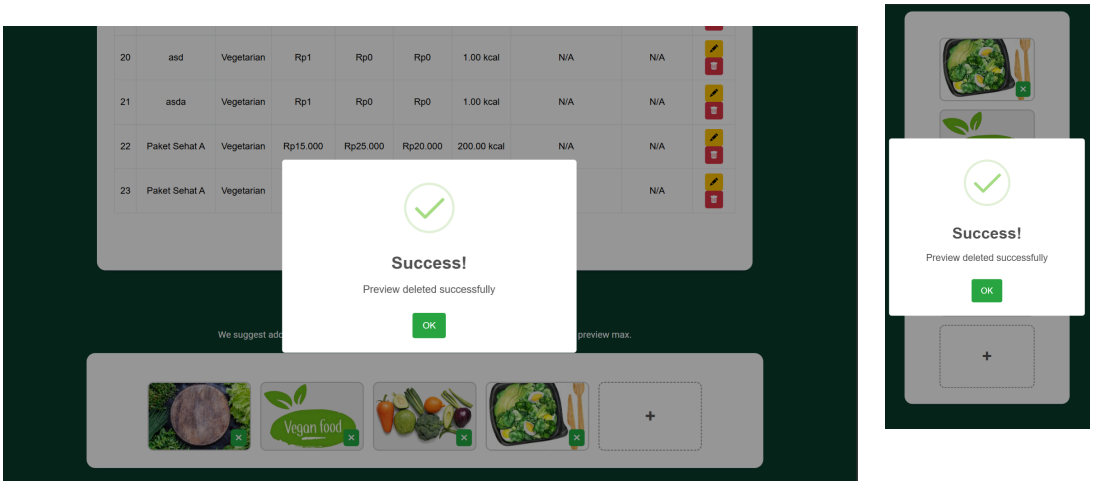
- Preview Vendor Ketika 4 images (tombol yang muncul adalah tombol delete pada tiap photo dan tombol tambah untuk tambah photo)



- Preview Vendor Ketika 5 images ((tombol yang muncul adalah tombol delete pada tiap photo dan tombol tambah menghilang, karena maksimal hanya 5 photo saja)

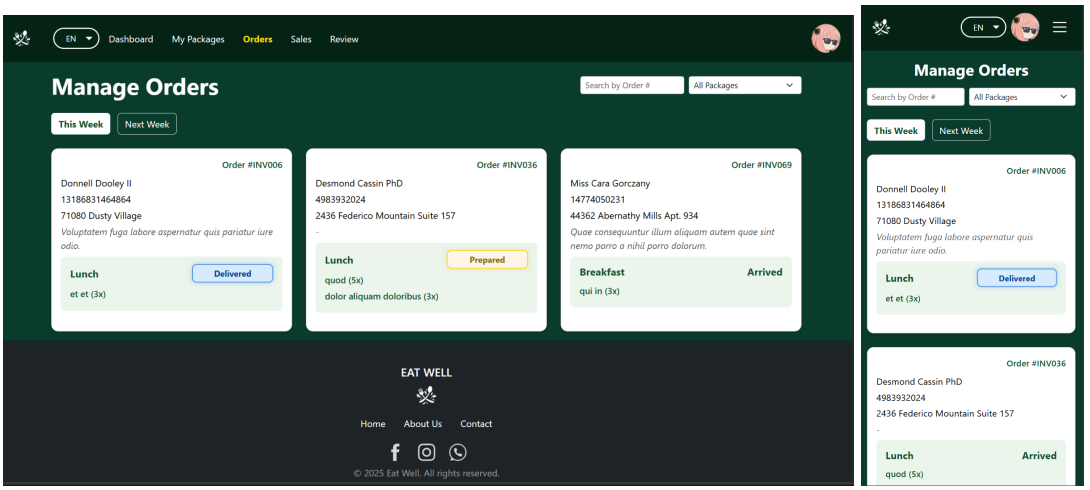


- Success Pop Up Notification ketika Preview di delete

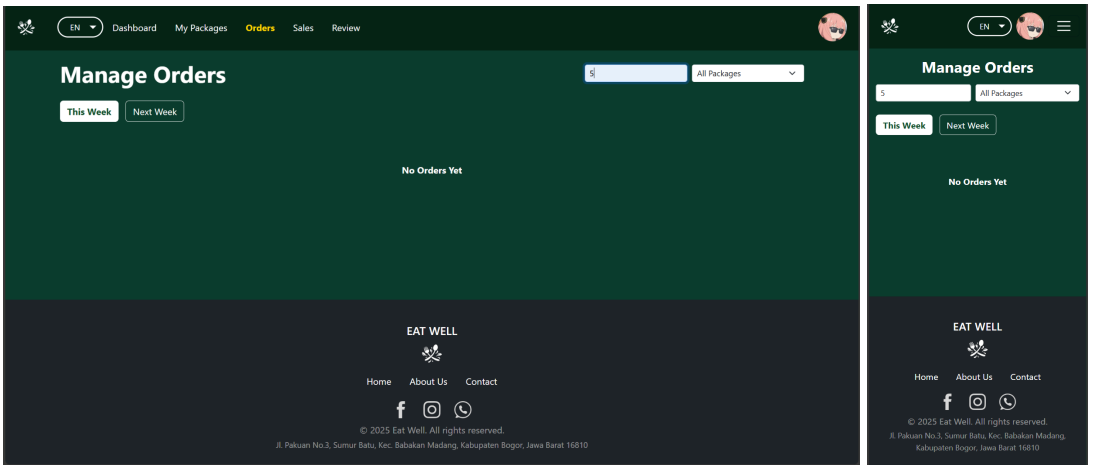


h. Update Delivery Status

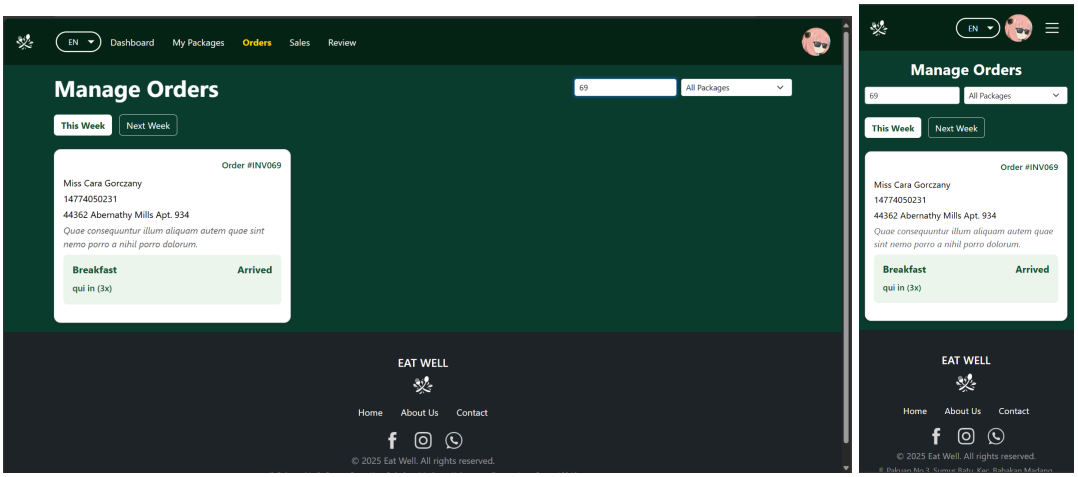
- Manage Order Secara Keseluruhan, terdapat tombol filter berdasarkan nama paket dan search berdasarkan id, serta tab this week dan next week, dan paket



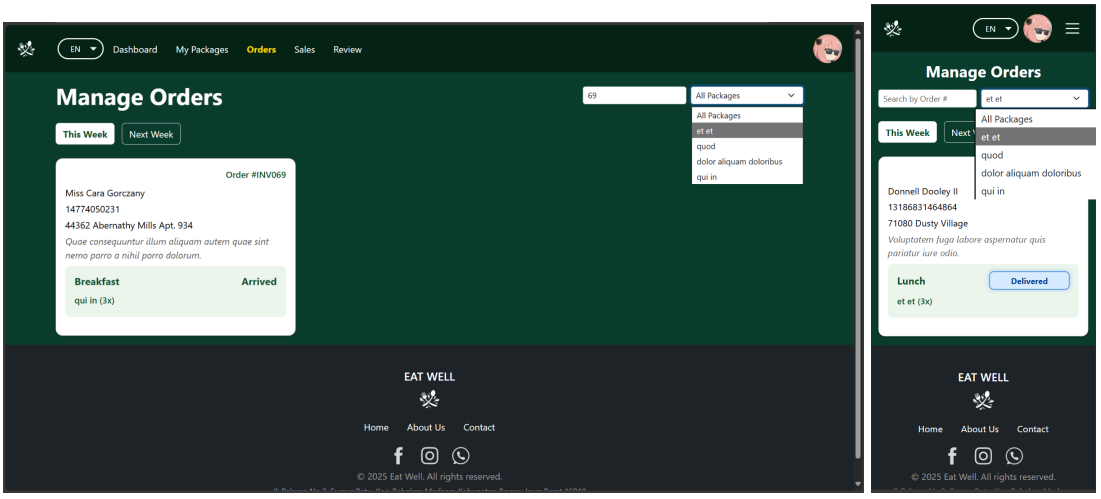
- Search id yang tidak ada di orderan



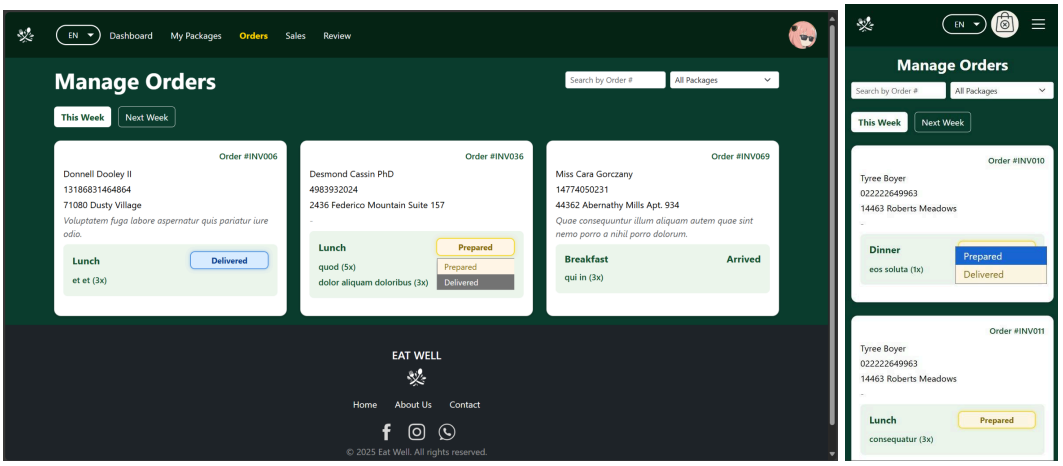
- Search id yang ada di orderan



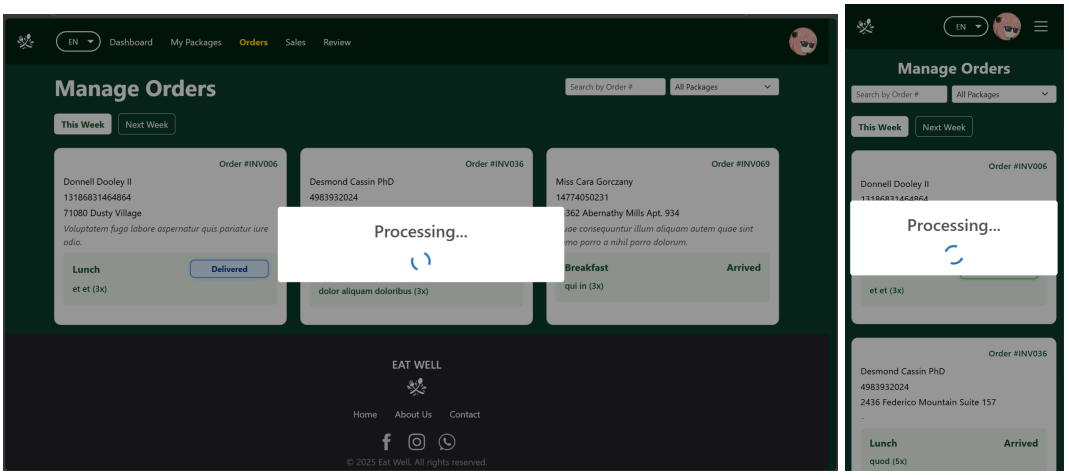
- Filter berdasarkan paket



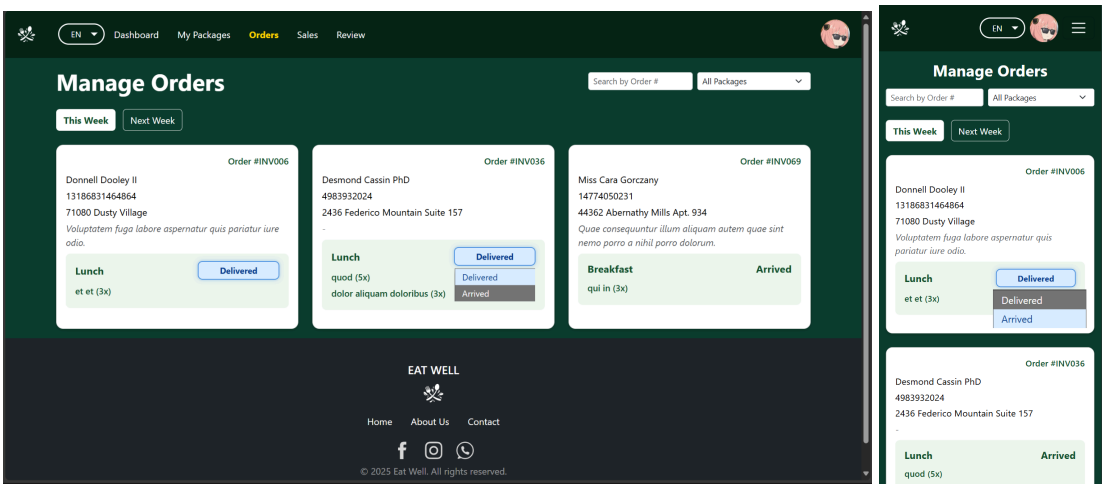
- Tampilan ketika status masih prepared dan dropdownnya



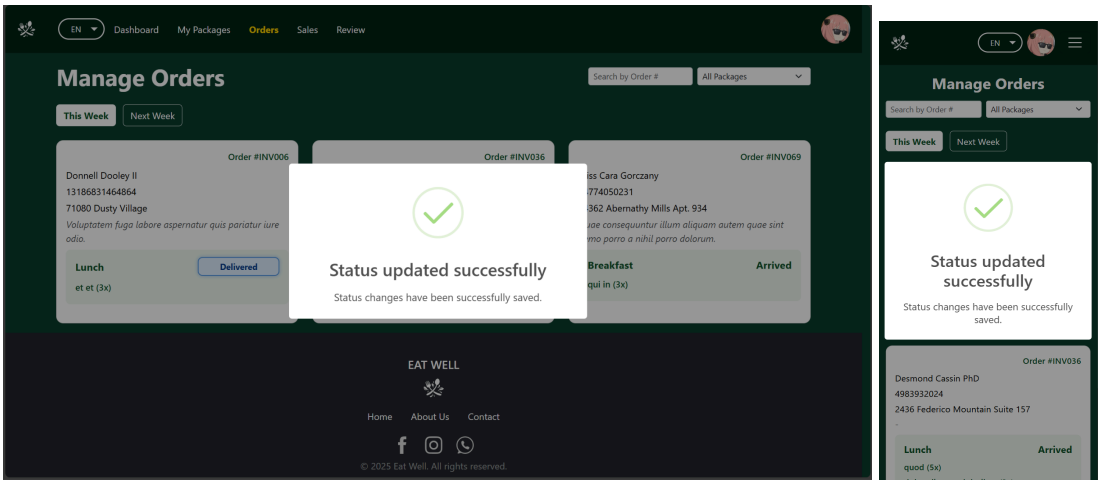
- Ketika mengubah status, muncul Processing Loading



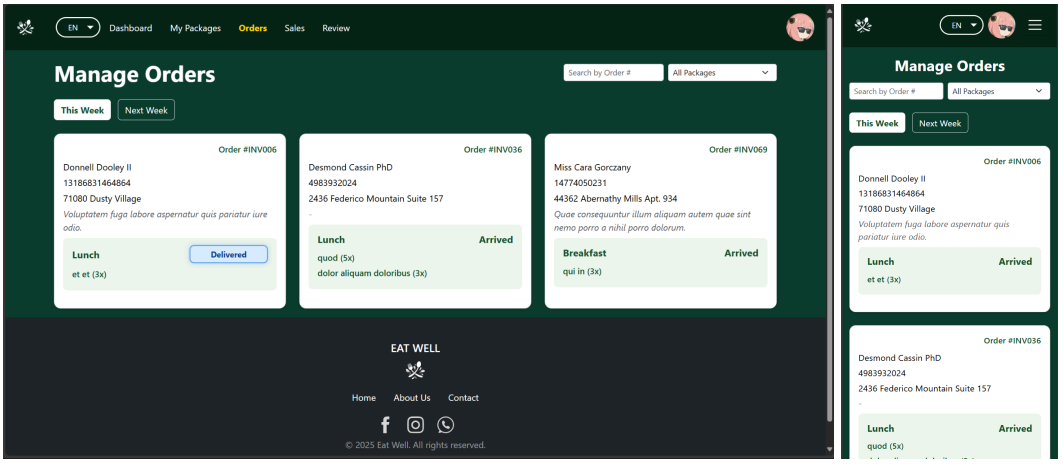
- Tampilan ketika status masih delivered dan dropdownnya



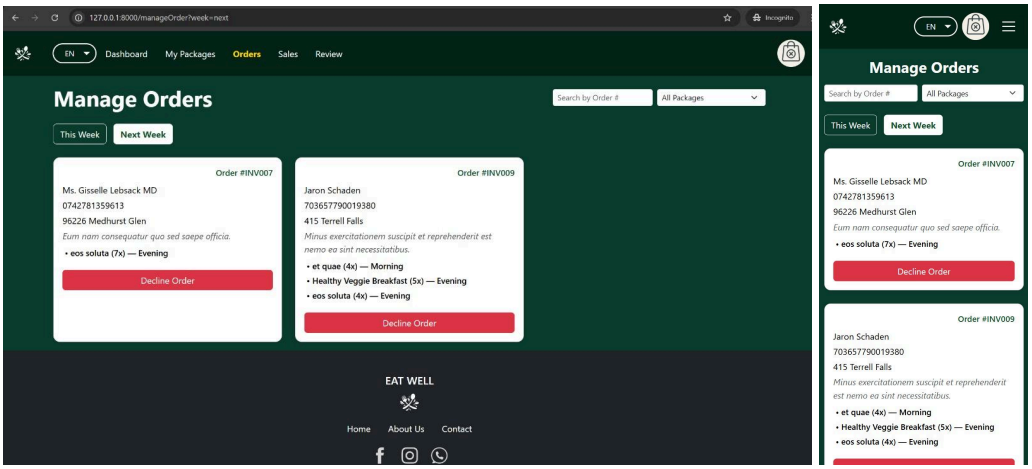
- Success Pop Up Notification ketika Status Berhasil di Update



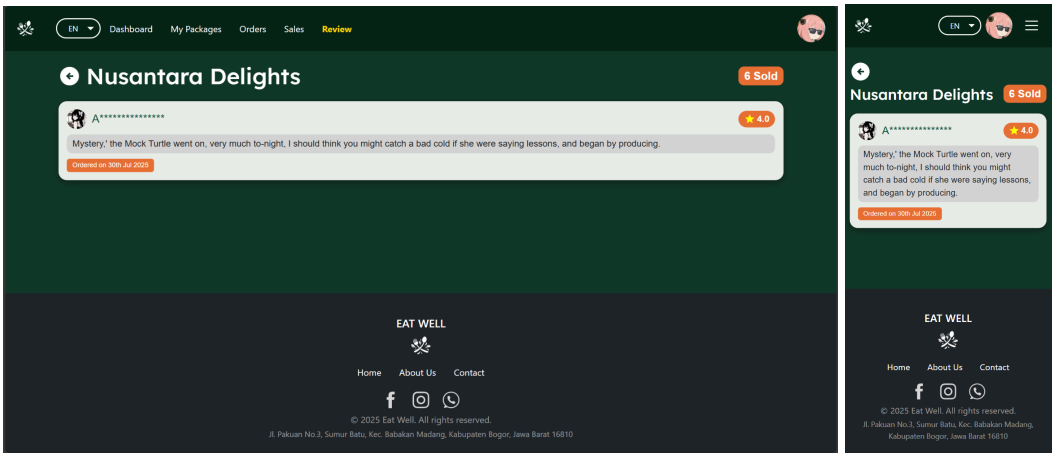
- Tampilan ketika arrived



i. Cancel Upcoming Order

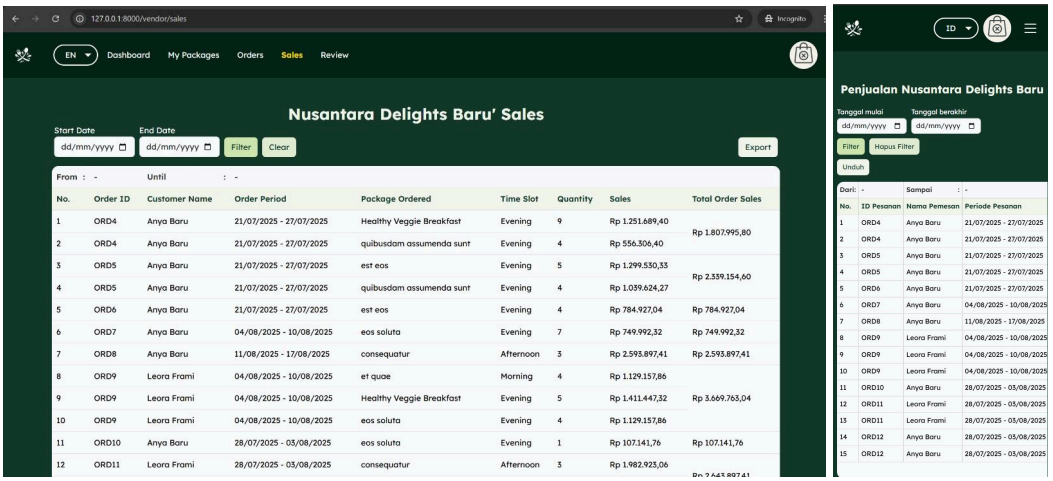


j. View Customer's Review

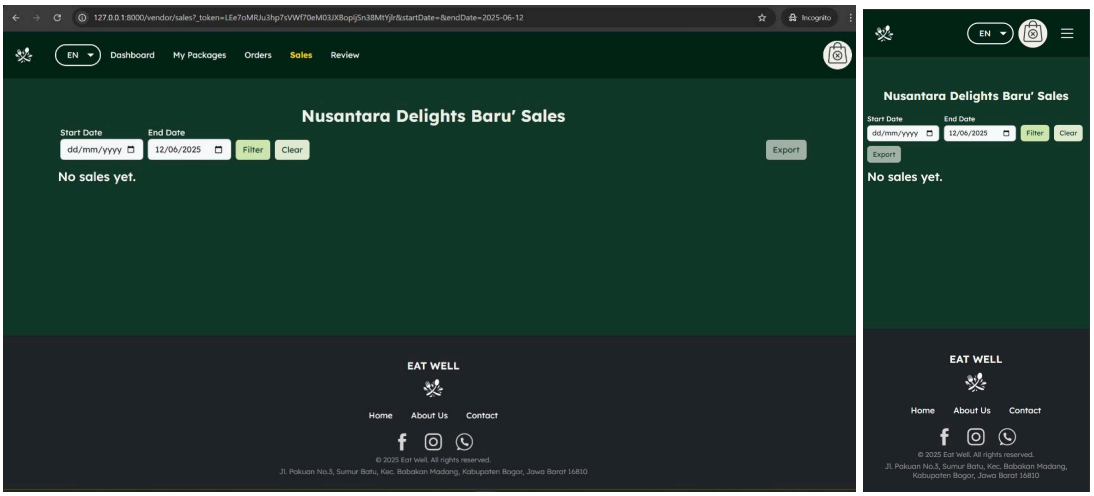


k. View vendor sales

- Tampilan ketika sales vendor secara keseluruhan



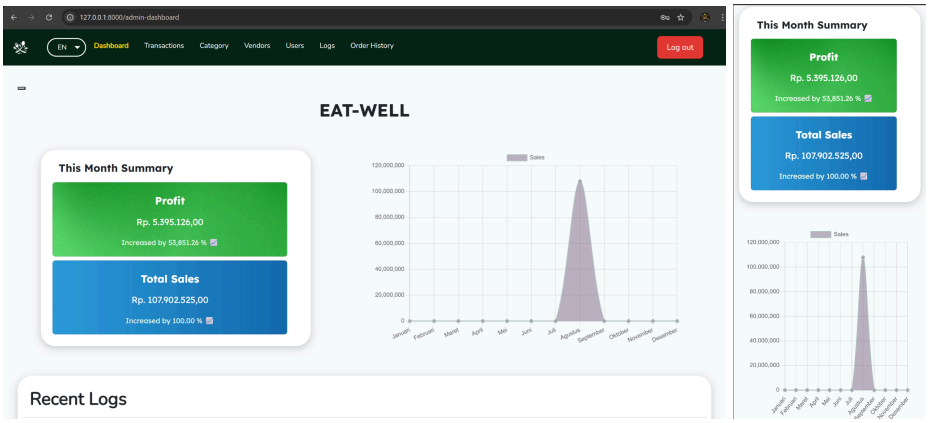
- Filter tanggal



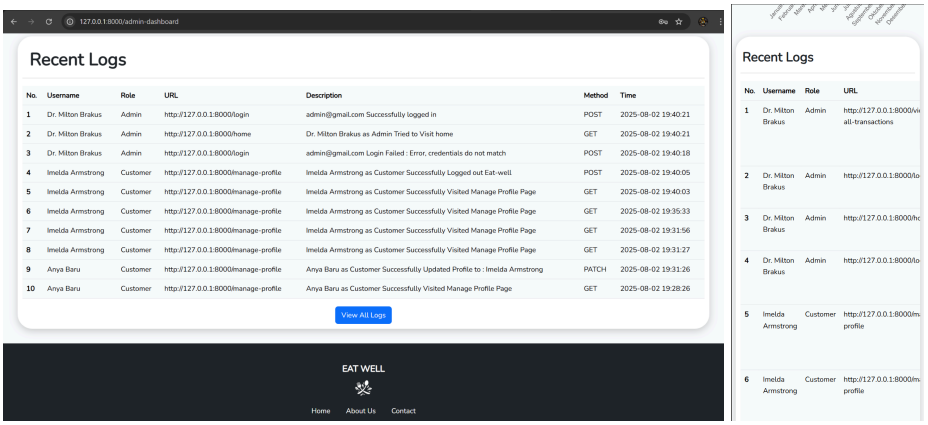
D. Admin

a. Admin dashboard

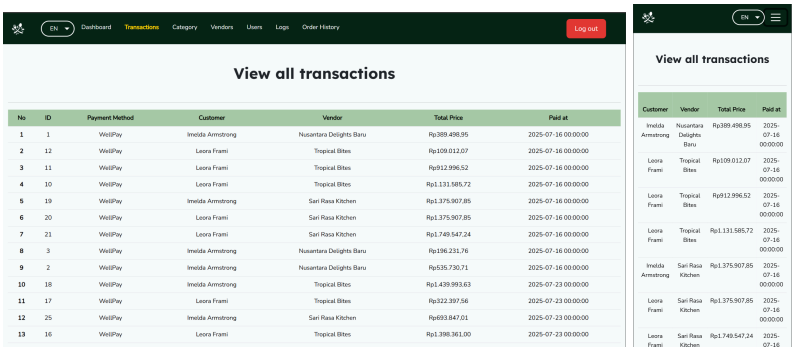
- Profit dan Total sales summary, grafik penjualan bulanan



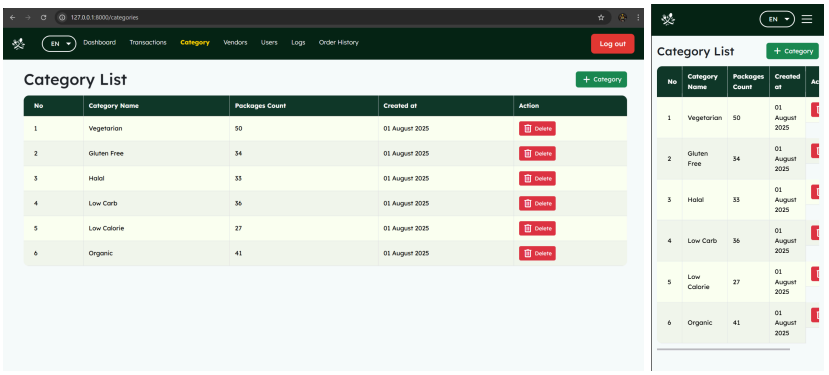
- Recent logs berisikan 10 log aktivitas user terbaru



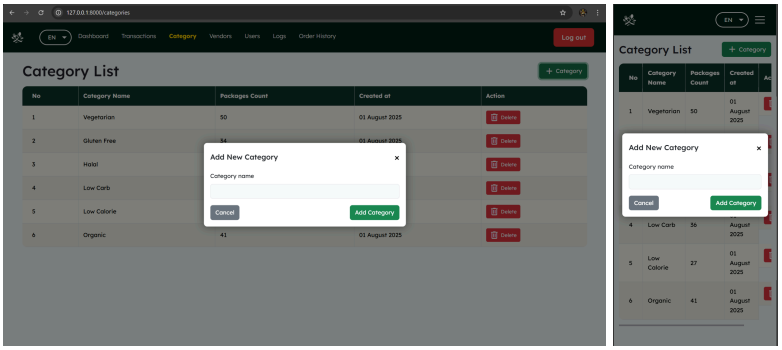
b. View all transactions

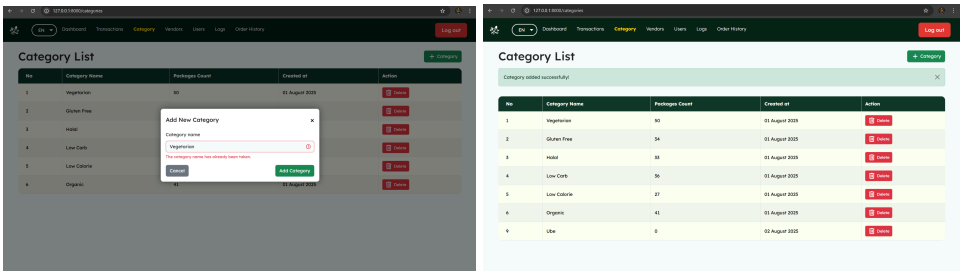


c. Manage Package Category

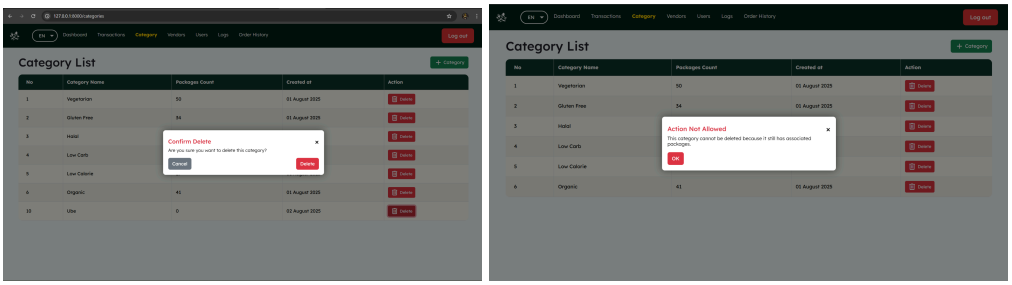


- Admin add category

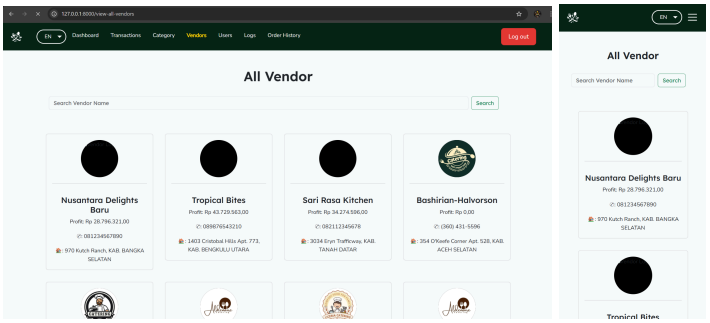




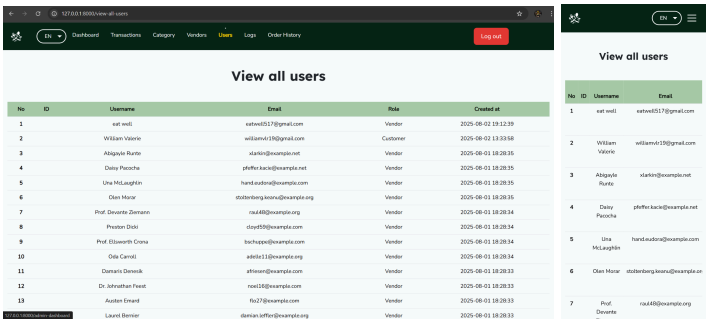
- Admin delete category



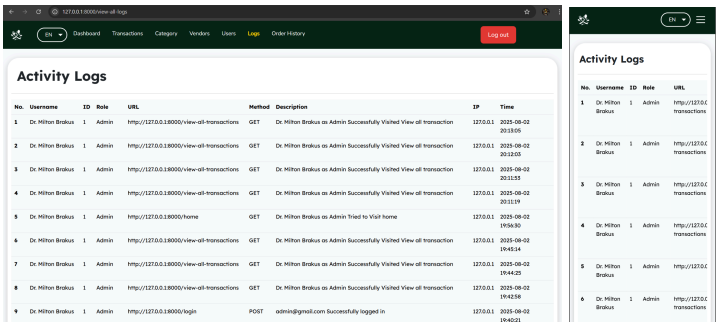
d. Search Vendors



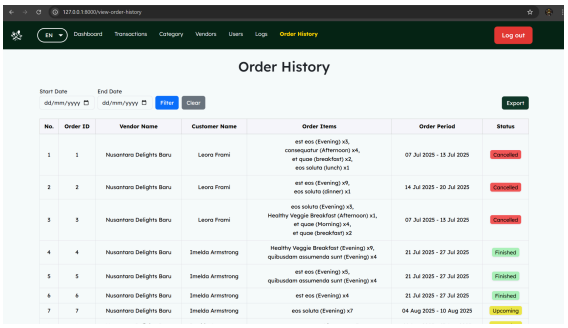
e. View All Users



f. View User Logs



g. Filter & Export Order History



BAB V SETUP INSTRUCTIONS

PREREQUISITES

Pastikan tools dan dependensi berikut telah terinstal pada komputer kalian:

- PHP 8.4
- Composer
- MySQL 8 or XAMPP 8.2.12
- Node.js (>= 18.x) & npm
- Git

LOCAL SETUP

1. **Clone repositori github EatWell**
   git clone https://github.com/WilliamVlr/eat-well.git
   cd eat-well

2. **Install dependensi PHP**
   Uncommand extension=gd pada file php.ini dari xampp kalian, lalu jalankan:
   composer install
   Atau bisa langsung dengan menjalankan ini (recommended):
   composer install --ignore-platform-req=ext-gd

3. **Install Laravel UI and Bootstrap**
   php artisan ui bootstrap
   npm install && npm run build

4. **Copy .env file**
   cp .env.example .env

5. **Configure symbolic link for storage**
   php artisan storage:link

6. **Set up environment**
   Adapun yang harus diubah pada environment sebagai berikut:
   - Set up database yang digunakan, di sini kami menggunakan mysql. Pastikan nama database (DB_DATABASE) sudah dibuat dan ada pada phpMyAdmin kalian.

```
23  DB_CONNECTION=mysql
24  DB_HOST=127.0.0.1
25  DB_PORT=3306
26  DB_DATABASE=eat_well
27  DB_USERNAME=root
28  DB_PASSWORD=
29  DB_COLLATION=utf8mb4_unicode_ci
```

   - Ubah konfigurasi mailer seperti berikut untuk dapat menjalankan fitur kirim OTP dan notifikasi dengan benar.

```
51  MAIL_MAILER=smtp
52  MAIL_HOST=smtp.gmail.com
53  MAIL_PORT=465
54  MAIL_USERNAME=wellyeating@gmail.com
55  MAIL_PASSWORD=rhfxeykklgqzcvtb
56  MAIL_ENCRYPTION=tls
57  MAIL_FROM_ADDRESS=wellyeating@gmail.com
58  MAIL_FROM_NAME="${APP_NAME}"
```

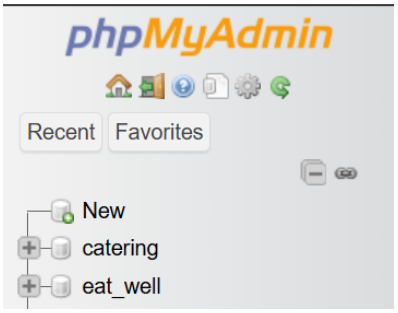
- Perlu menambahkan variabel google client id dan secret pada environment untuk dapat menggunakan fitur login/register with google

```
67  GOOGLE_CLIENT_ID=
68  GOOGLE_CLIENT_SECRET=
```

- Buat akun binderbyte dan aktifkan langganan gratis API Wilayah Indonesia (gratis). Ikuti panduannya lalu cek API key pada halaman profil. Copy API key tersebut ke environment:

```
70  BINDER_BYTE_API_KEY=
```

7. **Generate application key**
   php artisan key:generate
8. **Set up kredensial database di .env lalu jalankan migration**
   Pastikan database yang digunakan telah dituliskan dalam env dengan benar seperti berikut:




```
23  DB_CONNECTION=mysql
24  DB_HOST=127.0.0.1
25  DB_PORT=3306
26  DB_DATABASE=eat_well
27  DB_USERNAME=root
28  DB_PASSWORD=
29  DB_COLLATION=utf8mb4_unicode_ci
```

   Lalu jalankan migration:
   php artisan migrate:fresh --seed
9. **Serve aplikasinya**
   php artisan serve

LAMPIRAN

- Link GitHub: https://github.com/WilliamVlr/eat-well.git
- Link Dokumentasi testing: https://docs.google.com/spreadsheets/d/1zHeU7alBg2N_iuG4L2dCb9D27SvgFp-_ljCuA086LqE/edit?usp=sharing
- Link Trello: https://trello.com/b/Ry7WqIHq/ppti-17-agile-webprog-kelompok-4-eatwell