BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Posisi magang yang dilaksanakan adalah sebagai Pengembang Web pada divisi *Strategic Sourcing* yang merupakan bagian dari *Business Support & General Affairs.*

Dikarenakan pandemi Covid-19, koordinasi dilakukan secara daring (*online*) menggunakan aplikasi Microsoft Teams. Setiap minggu selama masa pelaksanaan kerja magang, penulis mengabari *progress* pekerjaan yang telah dilakukan, serta menerima arahan baru dari pembimbing lapangan.

3.2 Tugas yang Dilakukan

Tugas utama yang diberikan oleh pembimbing lapangan selama kerja magang adalah mengembangkan Vendor Data Management System (VDMS). VDMS adalah sebuah sistem berbasis web yang digunakan oleh karyawan divisi Strategic Sourcing untuk mengelola data vendor Garuda Indonesia. VDMS dikembangkan karena sistem pengelolaan data yang ada saat ini (menggunakan Microsoft Excel) kurang cocok untuk mengakomodasi kebutuhan data vendor yang sering berubah-ubah. Tugas yang dilakukan selama pelaksanaan kerja magang adalah sebagai berikut:

1. Pengarahan awal dan perkenalan perusahaan Garuda Indonesia.
2. Mempelajari Framework Laravel 8.

3. Membuat front-end dan back-end Edit Vendor Contact Person.
4. Memperbaiki fitur search Vendor.
5. Membangun front-end sistem SSO (Single Sign On).
6. Membangun back-end sistem SSO (Single Sign On).
7. Mengembangkan sistem CRUD (Create, Read, Update, Delete) User.
8. Menyesuaikan privilese pada role masing-masing User.
9. Menempatkan blokir pada fitur dan laman tertentu.
10. Modifikasi environment sehingga akun Superadmin dapat dibuat secara aman.
11. Melakukan bug-fixing dan memperbaiki tampilan.
12. Finalisasi hasil kerja, membuat file delta, serah terima hasil kerja pada hari Sabtu, 1 Mei 2021.

3.3 Uraian Pelaksanaan Kerja Magang

3.3.1 Pengarahan Awal dan Perkenalan

Pada tahap ini, dilakukan *briefing awal* bersama dengan Mba Ongga dari divisi magang Garuda Indonesia. Terdapat beberapa poin penting hasil dari *briefing* yang dilakukan, yaitu sebagai berikut:

- Posisi magang yang dijalankan berada di bawah unit *procurement* yaitu divisi *strategic sourcing.*
- Penjelasan dan tanya jawab mengenai tata tertib dan peraturan magang.

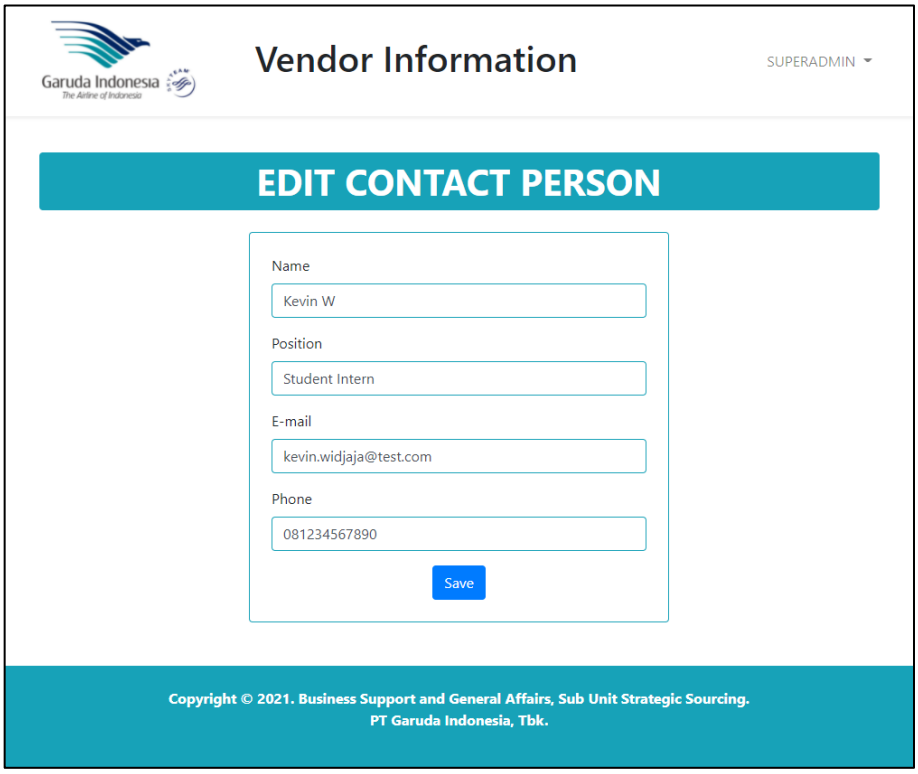
- Penjelasan mengenai tugas utama magang, yaitu pengembangan sistem VDMS Garuda Indonesia.
- Penjelasan mengenai jam kerja dan durasi kerja selama pandemi Covid-19, yaitu Senin-Jumat secara WFH.

3.3.2 Mempelajari Framework Laravel 8

Pada tahap ini, Framework Laravel 8 dipelajari secara otodidak dan dengan bantuan dari salah satu staf IT dari Garuda Indonesia, yaitu pak Bayu. Sumber pembelajaran Laravel 8 diperoleh dari internet, terutama situs Youtube dan dokumentasi resmi Laravel 8.

3.3.3 Membuat front-end dan back-end Edit Vendor Contact Person

Tugas pertama yang diberikan oleh pembimbing lapangan adalah membuat front-end dan back-end untuk menyunting informasi mengenai *contact person* masing-masing vendor.



Gambar 3.1 Laman formulir "Edit Contact Person" pada VDMS

Pada Gambar 3.1, terlihat hasil dari front-end edit contact person. Tampilan front-end tersebut dibuat menggunakan bahasa pemrograman Blade, yang serupa dengan PHP.

Sedangkan dari sisi back-end, terdapat 2 bagian yang dibuat yaitu kode yang digunakan untuk menampilkan laman "Edit Contact Person" seperti pada Gambar 3.1, dan kode untuk menerima perubahan pengguna dan melakukan *update* ke basis data. Berikut tampilan potongan kode masing-masing fungsi tersebut.

```php
// Menampilkan tampilan Edit Contact Person
public function editCp($id)
{
    // Ambil data contact person dari DB
    $contactPerson = ContactPerson::where('id',$id)->first();
    // Tampilkan laman edit contact person beserta data contact person
    return view('mng_vendor.contact_person.edit',compact('contactPerson'));
}
```

Gambar 3.2 Kode untuk menampilkan laman Edit Contact Person

Pada Gambar 3.2, terlihat potongan kode untuk menampilkan laman *edit contact person*. Sebelum laman tersebut ditampilkan, terlebih dahulu data mengenai *contact person* tersebut diambil dari basis data. Tujuannya adalah untuk mengisi terlebih dahulu formulir *edit contact person*, dengan data lama dari *contact person* tersebut.

```php
// Melakukan Update ke tabel contact_person
public function updateCp($id)
{
    // Ambil ContactPerson yang hendak disunting dan simpan di dalam variabel
    $contactPerson = ContactPerson::where('id',$id)->first();
    // Validasi Data Input
    $data = request()->validate([
        'name' => 'required',
        'position' => 'required',
        'email' => ['required','email'],
        'phone' => 'numeric',
    ]);

    // Update ke Database
    $contactPerson->update($data);
    $contactPerson->save();

    // Kembali ke laman tampilan vendor
    return redirect("/show/vendor-" . $contactPerson->vendor_id);
}
```

Gambar 3.3 Kode untuk melakukan *update* ke tabel contact_person

Pada Gambar 3.3, terlihat kode yang digunakan untuk melakukan fungsi *update* ke tabel contact_person, yang menyimpan data mengenai *contact person*. Sebelum dilakukan update, terlebih dahulu dilakukan validasi data input. Tujuan dilakukannya validasi input adalah untuk mencegah data tidak valid untuk masuk

ke basis data. Untuk contact person, nama, posisi, dan email wajib ada, tetapi nomor telepon tidak wajib. Email wajib memiliki format email pada umumnya. Kemudian nomor telepon wajib bersifat angka.

3.3.4 Memperbaiki Fitur Search Vendor.

Pada tahap ini, pembimbing magang memberi tugas untuk memperbaiki fitur pencarian Vendor yang sudah ada sebelumnya. Terdapat masalah pada saat menampilkan hasil pencarian, di mana hasil yang ditampilkan tidak terurut secara abjad, sehingga menyulitkan bagi pengguna untuk menemukan vendor yang diinginkan. Untuk mengatasi masalah tersebut, ditambahkan potongan kode untuk melakukan pengurutan secara abjad.

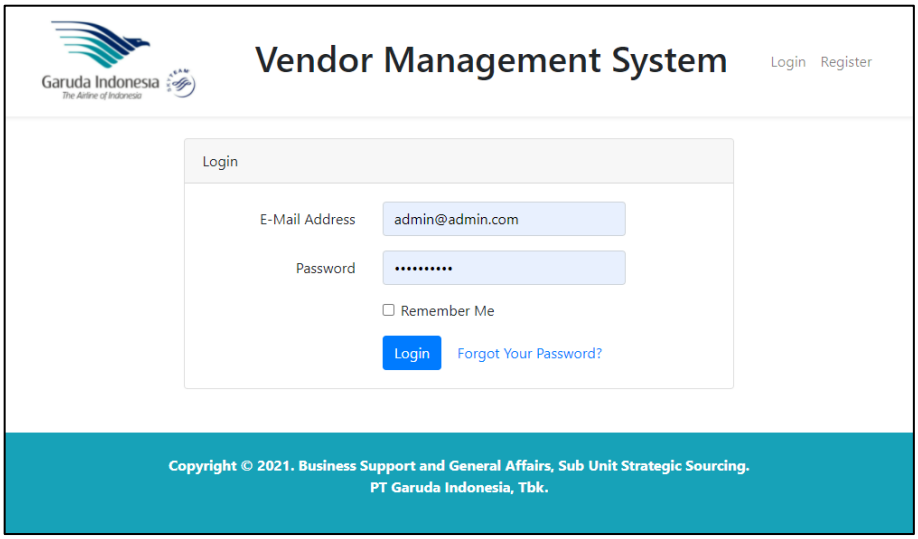
```
if(isset($_POST["search"])){
    // User melakukan search, maka tampilkan vendor yang sesuai dengan kata kunci
    $search = $_POST["search"];
    $data = \App\Models\Vendor::select('id', 'name', 'reg_code', 'flag_code', 'forgive')
        ->where('name', 'like', '%' . $_POST["vendor_name"] . '%')
        ->where('status', '=', 1)
        ->where('flag_code', '<>', 3)
        ->where('avl', '<>', 0)
        ->orderBy('name', 'ASC')        Kevin Widjaja, 7 months ago • Update code with newe
        ->get();

}else{
    // User tidak melakukan search, maka tampilkan semua vendor yang memenuhi syarat.
    $search = null;
    $data = \App\Models\Vendor::select('id', 'name', 'reg_code', 'flag_code', 'forgive')
        ->where('status', '=', 1)
        ->where('flag_code', '<>', 3)
        ->where('avl', '<>', 0)
        ->orderBy('name', 'ASC')        Kevin Widjaja, 7 months ago • Update code with newe
        ->paginate(30);
}
```

Gambar 3.4 Kode untuk mengurutkan hasil pencarian secara abjad

Pada Gambar 3.4, terlihat potongan kode yang ditambahkan untuk mengurutkan nama vendor secara abjad. ‘ASC’ merupakan kependekan dari *ASCENDING* yang berarti mengurutkan dari A ke Z.

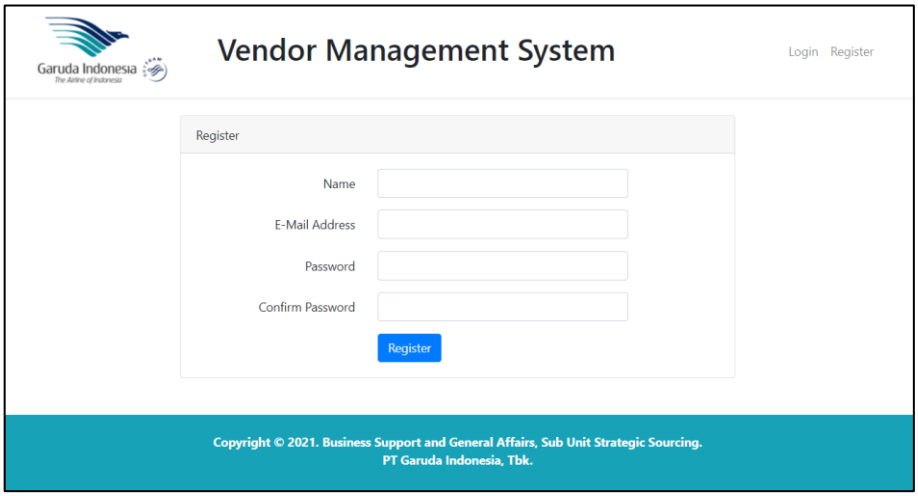
3.3.5 Membangun Front-end Sistem SSO

Di tahap ini, pembimbing lapangan memberikan tugas untuk membangun front-end sistem SSO (Single Sign On) yang terdiri dari fungsi login, register akun baru, log-out, dan role user.



Gambar 3.5 Laman login pada VDMS Garuda Indonesia

Pada Gambar 3.5, terlihat tampilan laman login pada VDMS Garuda Indonesia. Untuk melakukan login, pengguna menggunakan email dan password yang sudah di-register sebelumnya.



Gambar 3.6 Laman register pada VDMS Garuda Indonesia

Pada Gambar 3.6, terlihat laman register yang digunakan untuk membuat akun baru. Untuk membuat akun baru dibutuhkan nama, email, dan kata sandi.

3.3.6 Membangun Back-end Sistem SSO

Sistem SSO (Single Sign On) merupakan sistem yang penting dan vital untuk menjaga keamanan sistem VDMS. Menerapkan fitur SSO dalam aplikasi web secara manual dapat menjadi upaya yang sangat kompleks dan sangat berisiko. Oleh karena itu, back-end sistem SSO dibuat menggunakan *best practices* yang disediakan oleh Laravel. *Best practices* ini sudah termasuk teknologi Laravel seperti "guards", dan "providers" (Authentication Laravel, 2021). Pembuatan back-end sistem SSO menggunakan library *laravel/ui*.

Tabel 3.1 menampilkan daftar rute yang digunakan oleh sistem back-end SSO.

Tabel 3.1 Rute yang digunakan sistem back-end SSO VDMS

Method	URI	Action
GET	/login	Show login page
POST	/login	Log user in

Method	URI	Action
GET	/register	Show registration page
POST	/register	Register new user
POST	/logout	Log user out

3.3.7 Mengembangkan Sistem CRUD User

Setelah proses pembuatan front-end dan back-end sistem SSO sudah berjalan, pembimbing lapangan memberikan tugas berikutnya yaitu mengembangkan sistem CRUD User. CRUD adalah singkatan dari Create, Read, Update, Delete. Tujuan dari fungsi ini adalah agar admin dapat mengelola pengguna sistem VDMS, yang adalah karyawan Garuda Indonesia di 2 divisi, yaitu divisi IBS dan IBP. Sistem CRUD ini terpisah dari sistem SSO di atas karena digunakan untuk mengelola semua pengguna VDMS dan hanya diperuntukkan untuk pengguna yang memiliki *role* admin.

Sistem CRUD User terbagi menjadi 2 bagian yaitu front-end dan back-end. Pertama-tama akan dijabarkan bagian back-end terlebih dahulu.

```php
class UserController extends Controller
{
    public function __construct()
    {
        // Hanya admin yang bisa akses
        $this->middleware(['auth', 'role:' . User::ADMIN]);
    }
```

Gambar 3.7 Kode agar UserController hanya dapat diakses admin

Pada Gambar 3.7, terlihat potongan kode yang bertujuan untuk membatasi agar fungsi-fungsi pada UserController hanya dapat diakses oleh pengguna yang memiliki role admin.

```php
// Laman untuk buat user baru
public function create()
{
    return view('mng_user.create');
}

// Validation rules user
private static function validate_user(Request $request)
{
    return $request->validate([
        'name' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
        'role' => ['required', 'string', Rule::in([
            'ADMIN',
            'IBS',
            'IBP'
        ])]
    ]);
}

// Simpan user baru ke DB
public function store()
{
    // Validate data
    $data = UserController::validate_user(request());

    // Store data
    User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'password' => Hash::make($data['password']),
        'role' => $data['role'],
    ]);

    // Redirect to User Management Page
    return redirect('/user');
}
```

Gambar 3.8 Kode Create User

Gambar 3.8 menampilkan fungsi “C” dari “CRUD” yaitu “Create”. Pada Laravel, implementasi fungsi tersebut terbagi menjadi 2 bagian yaitu fungsi “Create” untuk menampilkan laman formulir pembuatan user baru, dan fungsi “Store” untuk menyimpan data yang dikumpulkan dari formulir ke dalam basis data.

```php
// Tampilkan semua user yang terdaftar
public function index()
{
    $users = User::all();
    return view('mng_user.index', compact('users'));
}
```

Gambar 3.9 Kode Read User

Gambar 3.9 menampilkan fungsi “R” dari “CRUD” yaitu “Read”. Pada Laravel, implementasi fungsi read adalah index dan show, tetapi karena menyesuaikan kebutuhan dari pembimbing lapangan, maka fungsi show tidak diimplementasi. Fungsi index digunakan untuk menampilkan semua user yang terdaftar.

```php
// Laman untuk edit user
public function edit(User $user)
{
    return view('mng_user.edit', compact('user'));
}

// Update user database
public function update(User $user)
{
    // Validate data
    $data = request()->validate([
        'name' => ['nullable', 'string', 'max:255'],
        'email' => ['nullable', 'string', 'email', 'max:255'],
        'password' => ['nullable', 'string', 'min:8', 'confirmed'],
        'role' => ['nullable', 'string', Rule::in([
            'ADMIN',
            'IBS',
            'IBP'
        ])]
    ]);

    // Hash password bila ada isinya
    $data['password'] != null ? $data['password'] = Hash::make($data['password']) : null;

    // Update data
    $user->update(array_filter($data));

    // Redirect to User Management Dashboard Page
    return redirect('/user');
}
```

Gambar 3.10 Kode Update User

Gambar 3.10 menampilkan fungsi “U” dari “CRUD” yaitu “Update”. Pada Laravel, implementasi fungsi update dipecah menjadi 2 yaitu fungsi edit untuk menampilkan laman edit user, dan fungsi update untuk menyimpan perubahan data user ke dalam basis data.

Pada fungsi edit, terlihat potongan kode jalan pintas untuk memperoleh data user yang sedang disunting. Cukup dengan meletakkan objek User pada parameter dan menambahkan “compact(‘user’)” pada bagian akhir, Laravel secara otomatis mengambil data user tersebut dari basis data.

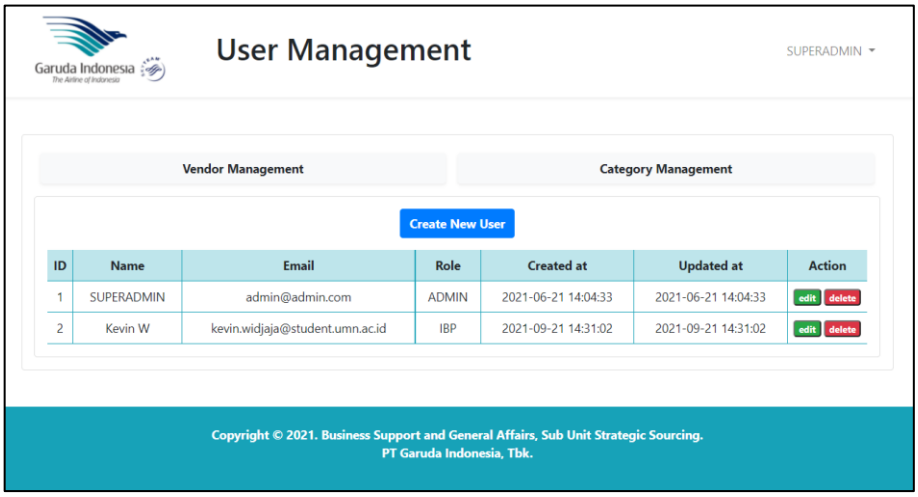
Pada fungsi update, sebelum data dimasukkan ke dalam basis data, data terlebih dahulu dilakukan validasi. Validasi pada update berbeda dengan validasi pada store, karena admin bisa saja hanya hendak mengubah sebagian data pengguna, misalnya hanya mengubah role. Oleh karena itu, validasi “nullable” ditambahkan, yang memperbolehkan admin menginput data blank. Bila server menerima data null, maka data lama tidak berubah. Setelah dilakukan validasi, program akan cek apakah admin mengubah password, bila ada maka dilakukan hash. Langkah terakhir adalah melakukan update user kemudian redirect user kembali ke laman dasbor user.

```php
// Delete user
public function destroy(User $user)
{
    // Delete user from DB
    $user->delete();
    // Redirect to User Management Page
    return redirect('/user');
}
```

Gambar 3.11 Kode Delete User

Pada Gambar 3.11, terlihat potongan kode yang digunakan untuk fungsi “D” pada “CRUD” yaitu “Delete”. Pada Laravel, implementasi fungsi delete adalah fungsi destroy. Fungsi destroy pada Laravel sangat sederhana, yaitu memperoleh objek user, kemudian dihapus dari basis data. Langkah terakhir adalah mengembalikan user ke laman dasbor manajemen user.

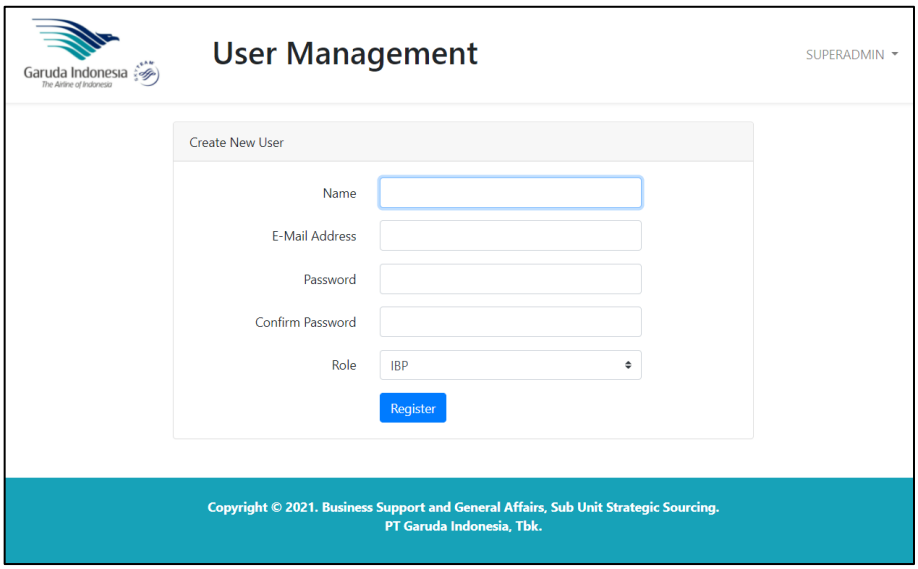
Kemudian, berikut penjabaran untuk bagian front-end sistem CRUD User.



Gambar 3.12 Laman dasbor manajemen User

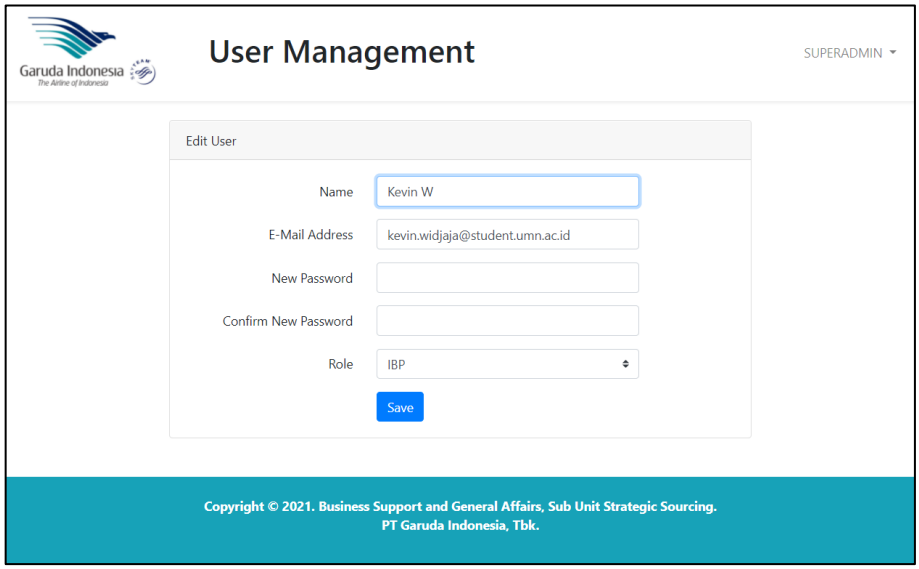
Pada Gambar 3.12, terlihat tampilan laman dasbor manajemen user. Pada laman ini, admin dapat melihat daftar seluruh pengguna yang terdaftar. Selain itu,

admin juga bisa klik tombol “create new user” untuk membuat user baru, “edit” untuk menyunting data user, dan “delete” untuk menghapus user tersebut.



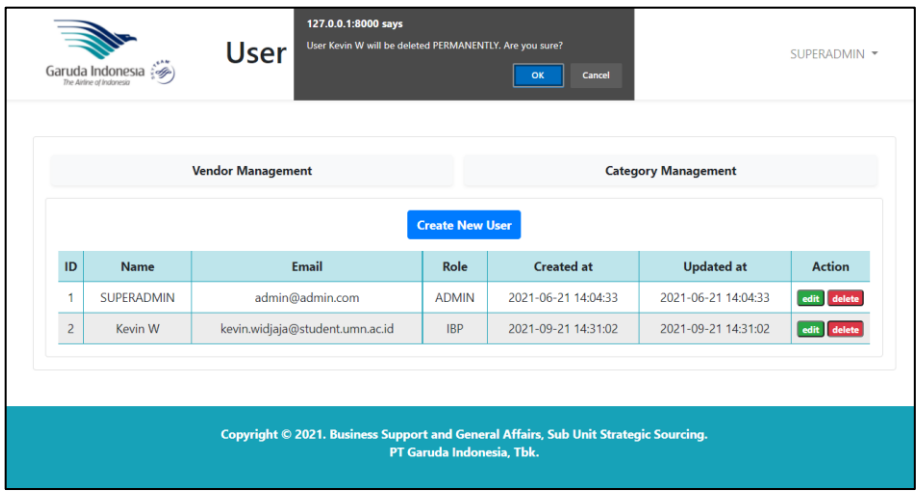
Gambar 3.13 Laman formulir buat user baru

Gambar 3.13 menampilkan laman formulir untuk membuat user baru. Laman ini diakses saat admin klik tombol “create new user”. Terlihat terdapat 4 buah *text field* yang dapat diisi oleh admin, yaitu name, email address, password, dan confirm password. Selain itu, juga terdapat 1 buah dropdown untuk menginput role user. Pilihan menu dropdown adalah IBP (default), IBS, dan Admin.



Gambar 3.14 Laman formulir edit user

Gambar 3.14 merupakan tampilan laman formulir edit user, yang dapat diakses oleh admin dengan klik tombol "edit" pada masing-masing user. Saat admin klik tombol "edit", text field yang tersedia otomatis terisi dengan data yang sudah ada saat ini. Data password tidak ditampilkan karena bersifat sensitif, akan tetapi, admin memiliki hak untuk mengubah kata sandi menjadi kata sandi yang baru. Admin juga bisa mengubah role seorang user.



Gambar 3.15 Pop-up konfirmasi delete user

Gambar 3.15 menampilkan pop-up yang akan muncul saat admin klik tombol "delete" pada masing-masing user. Bila admin klik OK, maka pengguna tersebut akan terhapus secara permanen dan datanya tidak tampil lagi di dasbor manajemen user.

3.3.8 Menyesuaikan Privilese pada Role masing-masing User

Di tahap ini, pembimbing lapangan memberikan penjelasan mengenai role dan privilese masing-masing user. Terdapat 3 role pada sistem VDMS yaitu IBP, IBS, dan ADMIN. IBS merupakan kode dari divisi *Strategic Sourcing*, sedangkan IBP merupakan kode dari divisi *Procurement Management*. Kedua divisi tersebut berada di bawah naungan unit IB atau *Business Support*. Privilese masing-masing role dapat dilihat pada Tabel 3.2.

Tabel 3.2 Tabel privilese role user

Fungsi	Role		
	IBP	IBS	ADMIN
Index	✔	✔	✔
Create		✔	✔
Store		✔	✔
Show	✔	✔	✔
Edit		✔	✔
Update		✔	✔
Destroy		✔	✔
Manage User			✔

Berikut masing-masing penjelasan fungsi yang terdapat pada VDMS:

- Index : menampilkan daftar semua vendor.
- Create : mengakses laman untuk menambahkan data vendor baru.
- Store : menyimpan data vendor baru ke basis data.

- Show : menampilkan detail data per vendor.
- Edit : mengakses laman untuk menyunting data vendor.
- Update : menyimpan perubahan hasil edit data vendor ke basis data.
- Destroy : menghapus data vendor.
- Manage User : mengelola pengguna lain, termasuk mengubah role pengguna lain.

3.3.9 Menempatkan Blokir pada Fitur dan Laman Tertentu

Dengan diterimanya informasi mengenai privilese masing-masing jenis pengguna, maka pembimbing lapangan memberikan tugas untuk menerapkan privilese tersebut secara teknis, yaitu dengan menempatkan blokir bagi user yang tidak berwenang untuk mengakses fitur dan laman tertentu. Pada Laravel 8, blokir dapat dilakukan dengan menggunakan fitur *middleware*. *Middleware* pada Laravel bertindak seperti jembatan antara *request* dari klien dan respons dari server. Dengan kata lain, *middleware* adalah sejenis mekanisme penyaringan, yang pada kasus ini, dapat digunakan untuk "menyaring" pengguna berdasarkan *role* mereka.

```php
class Role
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @param  \String  $roles
     * @return mixed
     */
    public function handle(Request $request, Closure $next, ... $roles)
    {
        // Ambil user yang sedang login dan disimpan di dalam variabel $user
        $user = Auth::user();

        // Cek apakah role user yang login adalah ADMIN,
        // bila true maka langsung izinkan user untuk akses laman
        if ($user->isAdmin()) {
            return $next($request);
        // Cek apakah role user sesuai dengan syarat yang disediakan pada parameter $roles
        } else {
            foreach($roles as $role) {
                if($user->hasRole($role)) return $next($request);
            }
        }
        abort(403, "Cannot access to restricted page");
    }
}
```

Gambar 3.16 Kode middleware role

Pada Gambar 3.16, terlihat tampilan potongan kode *middleware role* yang terdapat pada VDMS. Bila user memiliki *role* admin, maka program akan langsung memperbolehkan user tersebut untuk mengakses laman selanjutnya. Hal ini disebabkan karena admin memiliki kuasa penuh atas VDMS. Sedangkan untuk *role* lainnya, *middleware role* akan cek daftar *role* yang diperbolehkan untuk laman tersebut. Implementasi middleware dapat dilihat pada Gambar 3.17.

```php
// Hanya role ADMIN dapat akses
    Route::group(['middleware' => 'role:'.User::ADMIN], function () {
        // Routes untuk User Management
        Route::get('/user', [UserController::class, 'index']) -> name("user.index");
        Route::get('/user/create', [UserController::class, 'create']) -> name("user.create");
        Route::post('/user', [UserController::class, 'store']) -> name("user.store");
        Route::get('/user/{user}/edit', [UserController::class, 'edit']) -> name("user.edit");
        Route::patch('/user/{user}', [UserController::class, 'update']) -> name("user.update");
        Route::delete('/user/{user}', [UserController::class, 'destroy']) -> name("user.destroy");
    });
```

Gambar 3.17 Kode implementasi middleware role Admin

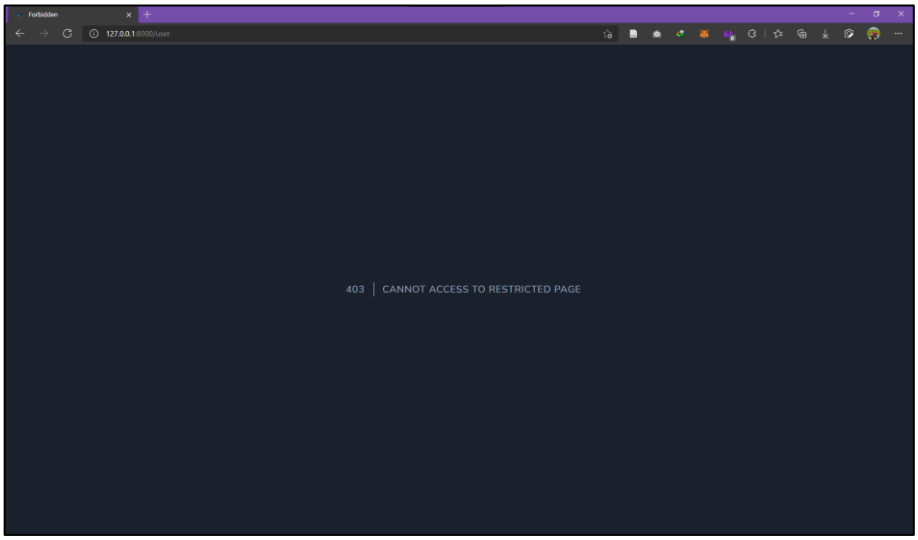
Pada Gambar 3.17, terlihat potongan kode implementasi fungsi *middleware role* pada *file* web.php. Potongan kode tersebut menggunakan fungsi middleware

role untuk membuat pengguna dengan role admin saja yang dapat mengakses rute-rute tersebut, dalam hal ini rute untuk manajemen user.

```
// Role IBS dan ADMIN dapat akses
Route::group(['middleware' => 'role:'.User::ADMIN.','.User::IBS], function () {
    Route::get('/vendor/export/spreadsheet', [VendorController::class, 'exportXls']);
    Route::get('/new-vendor', [VendorController::class, 'create']);
    Route::get('/forgive/{id}', [VendorController::class, 'forgive']);
    Route::get('/vendor/restore/{id}', [VendorController::class, 'restore']);
    Route::get('/vendor/contact/{id}/edit', [VendorController::class, 'editCp']);
    Route::get('/{id}/confirm/delete', [VendorController::class, 'confirmDelete']);
    Route::get('/vendor/contract/{id}/end', [VendorController::class, 'endContract']);
    Route::get('/vendor/regis/{id}/delete', [VendorController::class, 'deleteRegis']);
    Route::get('/vendor/contact/{id}/delete', [VendorController::class, 'deleteCp']);
    Route::get('/vendor/election/{id}/delete', [VendorController::class, 'deleteElection']);
    Route::post('/vendor/store', [VendorController::class, 'store']);
    Route::post('/vendor/delete', [VendorController::class, 'delete']);
    Route::post('/vendor/store/contact', [VendorController::class, 'storeCp']);
    Route::post('/vendor/store/election', [VendorController::class, 'storeElection']);
    Route::post('/vendor/info/{id}/update', [VendorController::class, 'update']);
    Route::patch('/vendor/contact/{id}', [VendorController::class, 'updateCp']);
    Route::post('/vendor/store/registration', [VendorController::class, 'storeRegistration']);

    // Create new category and sub-category
    Route::post("/store", [CategoryController::class, 'store']);
    // Delete and restore existing category and sub-category
    Route::post("/cat/mng", [CategoryController::class, 'catMng']);
    Route::post("/sub/mng", [CategoryController::class, 'subMng']);
    // Update Category and Sub-Category Name
    Route::post("/update-category", [CategoryController::class, 'updateCategory']);
    Route::post("/update-subcategory", [CategoryController::class, 'updateSubCategory']);
});
```

Gambar 3.18 Kode implementasi middleware role IBS dan Admin

Pada Gambar 3.18, terlihat potongan kode implementasi fungsi *middleware role* untuk hanya memperbolehkan user dengan role admin dan IBS mengakses laman tersebut. Dengan kata lain, hanya role IBP yang tidak diperbolehkan mengakses fitur-fitur tersebut. Ini mencakup fungsi Create, Store, Edit, Update, dan Destroy, beserta rute-rute lain yang mendukung fungsi tersebut.



Gambar 3.19 Tampilan saat mengakses laman dasbor user bagi pengguna yang tidak diizinkan

Implementasi *middleware role* terlihat pada Gambar 3.19, di mana pengguna yang tidak memiliki privilese untuk mengakses suatu laman, tidak akan bisa mengakses laman tersebut. Dengan kata lain, saat user yang tidak berwenang mengakses suatu laman, user tersebut akan menerima respons *403 Forbidden*.

3.3.10 Modifikasi file environment sehingga akun Superadmin dapat dibuat secara aman.

Pada tahap ini, pembimbing lapangan memberikan tugas yaitu untuk menyediakan sebuah akun pengguna “Superadmin”, yaitu akun admin pertama, yang setelahnya dapat digunakan untuk mengakses sistem VDMS dengan role admin. Data username, email, password, dan role dari admin tersimpan secara aman di dalam file environment, namun karena alasan keamanan, baik username, email dan password tidak ditampilkan pada laporan magang ini. Kemudian, data pada file environment tersebut diakses melalui sintaks “config(‘admin.item’)”.

```
// Buat default akun superadmin saat seed database
        if (config('admin.admin_name')) {
            User::create([
                'name' => config('admin.admin_name'),
                'email' => config('admin.admin_email'),
                'password' => Hash::make(config('admin.admin_password')),
                'role' => config('admin.admin_role'),
            ]);
        }
```

Gambar 3.20 Kode untuk membuat user Superadmin

Gambar 3.20 menampilkan kode yang dijalankan saat *seeding* database untuk membuat akun superadmin. Terlihat baik nama, email, password, dan role, tidak langsung ditulis di dalam file tersebut, melainkan diakses dari file environment yang sudah dimodifikasi.

3.3.11 Melakukan Bug Fixing dan Memperbaiki Tampilan

Pada tahap ini, pembimbing lapangan memberikan tugas terakhir untuk memperbaiki hal-hal minor seperti kesalahan pada tampilan dan bug yang ditemukan. Berikut daftar hal-hal yang diperbaiki:

1. Memperbaiki tampilan dasbor manajemen user (*user management dashboard*).

Pada dasbor manajemen user, terdapat kekurangan pada tampilan di mana tampilannya masih sangat standar dan kurang bagus, oleh karena itu, ditambahkan beberapa kelas *Bootstrap* untuk memperbaiki tampilan.

```
@foreach ($users as $user)
<tr>
    <td class="border-bottom border-info p-2">{{ $user->id }}</td>
    <td class="border-bottom border-left border-right border-info p-2">{{ $user->name }}</td>
    <td class="border-bottom border-left border-right border-info p-2">{{ $user->email }}</td>
    <td class="border-bottom border-left border-right border-info p-2">{{ $user->role }}</td>
    <td class="border-bottom border-left border-right border-info p-2">{{ $user->created_at }}</td>
    <td class="border-bottom border-left border-right border-info p-2">{{ $user->updated_at }}</td>
    <td class="border-bottom border-info p-2" style="width: 1%;">
        <div class="d-flex justify-content-between w-100">
            <a href="{{ url("/user/$user->id/edit") }}">
                <button class="badge badge-success">edit</button>
            </a>
            <form action="{{ url("/user/$user->id") }}" method="POST" id="deleteUser">
                @method("DELETE")
                @csrf
                <button id="deleteButton" type="submit"
                    onclick="return confirm('User {{ $user->name }} will be deleted PERMANENTLY. Are you sure?')"
                    class="badge badge-danger ml-1">delete</button>
            </form>
        </div>
    </td>
</tr>
@endforeach
```

Gambar 3.21 Kode blade.php untuk menampilkan user

Pada Gambar 3.21, terlihat tambahan kelas *Bootstrap* pada masing-masing elemen *td* dan *button*, seperti *border-bottom*, *border-left*, *border-right*, dan *border-info*. Begitu juga terdapat kelas *badge-danger* pada tombol delete yang membuat tombol berwarna merah, dan kelas *badge-success* pada tombol edit yang membuat tombol memiliki warna hijau.

2. Memperbaiki bug pada fitur *export* PDF.

Pada tahap ini, pembimbing lapangan memberi tugas untuk memperbaiki fitur *export* PDF. Fitur ini merupakan fitur yang sudah ada sebelumnya, bukan merupakan fitur yang dibuat oleh penulis. Bug yang terjadi adalah fitur export PDF melakukan export terhadap semua data vendor, sedangkan yang diperlukan hanyalah data vendor yang aktif. Untuk memperbaiki masalah tersebut, ditambahkan sebuah filter pada fungsi export PDF, seperti yang terlihat pada Gambar 3.22.

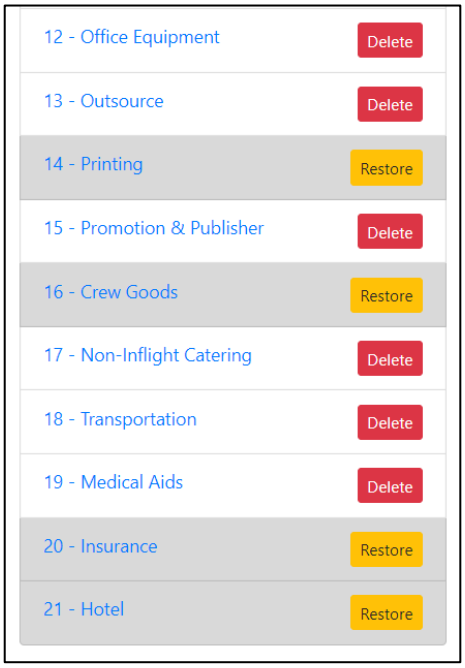
```php
public function exportDetailPdf()
{
    $mpdf = new \Mpdf\Mpdf();
    $mpdf->h2bookmarks = array('H1'=>0, 'H2'=>1, 'H3'=>2);

    $vendors = \App\Models\Vendor::
        select(
            'id',
            'reg_code',
            'name',
            'address',
            'phone',
            'email',
            'description',
            )
        ->where('avl', 1)
        ->orderBy('name', 'ASC')
        ->get();

        $vendor_style = 'style="font-size: 16px; font-family: Arial, Helvetica, sans-serif;"';
        $general_style = 'style="font-size: 12px; font-family: Arial, Helvetica, sans-serif;"';
        $table_style = 'style="vertical-align: top;"';
```

Gambar 3.22 Potongan kode fungsi *export* PDF

Pada Gambar 3.22, terlihat potongan kode dari fungsi untuk melakukan *export* data ke PDF. Pada VDMS, vendor aktif ditandai dengan nilai avl sama dengan 1. Karena hanya vendor yang aktif saja yang masuk ke dalam PDF, ditambahkan kode "->where('avl',1)" yang maksudnya adalah untuk melakukan penyaringan di mana hanya vendor dengan nilai avl = 1 yang diambil datanya.

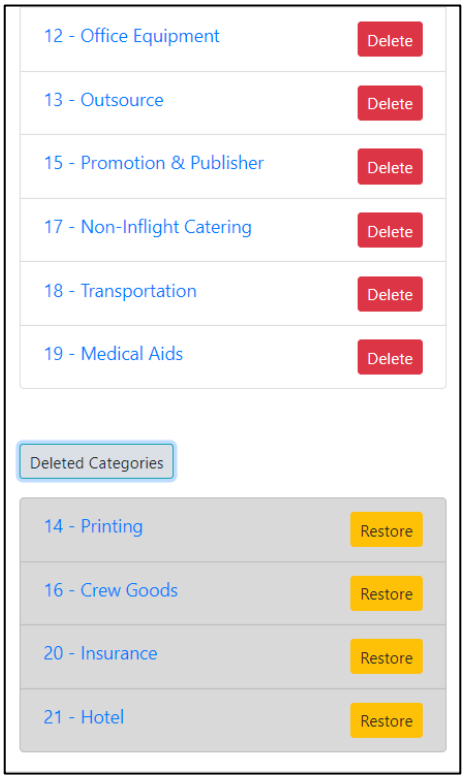
3. Memperbaiki tampilan daftar *category* yang sudah dihapus.

Pada menu *Category Management,* terdapat daftar kategori yang tersedia, di mana pengguna dapat menambahkan dan menghapus kategori yang ada. Pada tampilan yang sudah ada, saat pengguna menghapus sebuah kategori, kategori tersebut tidak hilang dari daftar kategori, seperti yang terlihat pada Gambar 3.23.



Gambar 3.23 Tampilan daftar kategori sebelum diperbaiki

Pada Gambar 3.23, kategori “Printing”, “Crew Goods”, “Insurance”, dan “Hotel” sudah dihapus, namun masih berada pada daftar kategori aktif, dan hanya dibedakan dengan warna dan tombol “Restore” di sampingnya. Hal ini menyebabkan kebingungan karena kategori aktif dan tidak aktif tercampur dalam satu daftar yang sama. Pembimbing lapangan memberi tugas untuk memisahkan kategori yang sudah dihapus dari daftar kategori yang masih aktif. Hasil perbaikan dapat dilihat pada Gambar 3.24.

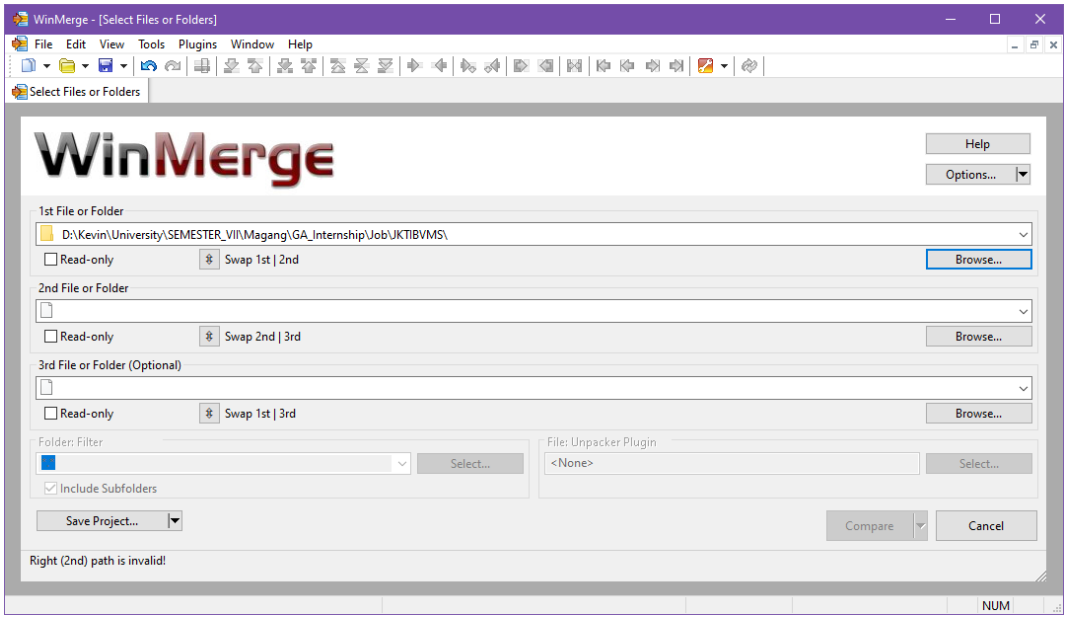


Gambar 3.24 Tampilan daftar kategori setelah diperbaiki

Pada Gambar 3.24, untuk memisahkan kategori yang sudah dihapus dari yang masih aktif, penulis membuat sebuah daftar tambahan yang terletak di bawah daftar kategori yang masih aktif. Daftar kategori yang terhapus ini bisa ditampilkan dan dihilangkan menggunakan *toggle button Deleted Categories*.

3.3.12 Finalisasi dan Serah Terima Hasil Pekerjaan

Pada tahap akhir, pembimbing lapangan meminta untuk dibuatkan *file* delta, yaitu sebuah *file* .zip yang berisi semua perubahan, baik penambahan dan pengurangan yang dibuat selama proses magang berjalan. Tujuan dari *file* delta ini adalah untuk memudahkan proses *deployment* ke server Garuda Indonesia. Proses pembuatan *file* delta menggunakan program WinMerge.



Gambar 3.25 Tampilan program WinMerge

Dengan adanya *file* delta tersebut, staf IT pada Garuda Indonesia cukup melakukan *copy-paste* ke *source code* yang dimiliki, dan semua perubahan dan penambahan yang dibuat selama proses magang otomatis akan masuk. Dengan begitu, staf IT tidak perlu menghapus ulang kode yang sudah ada di server. Serah terima *file* delta dilakukan pada tanggal 1 Mei 2021 melalui e-mail, dan menandakan akhir dari proses kerja magang selama 3 bulan.

3.4 Kendala yang Dihadapi

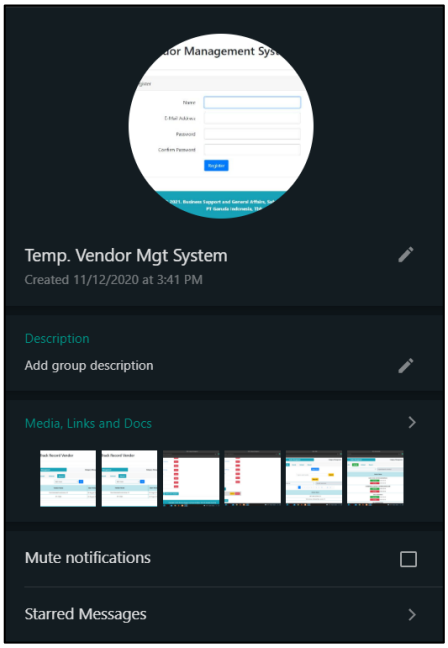
Kendala terbesar yang dihadapi selama kerja magang adalah kendala berkomunikasi. Karena pandemi Covid-19, sebagian besar komunikasi hanya dapat dilakukan secara daring. Ditambah gangguan jaringan yang kadang terjadi, komunikasi yang efektif di masa pandemi ini menjadi sangat menantang. Tantangan ini menjadi sumber pengalaman dan pembelajaran yang baik dalam berkomunikasi,

berinteraksi, dan bekerja sama dengan orang-orang yang sebelumnya belum pernah ditemui secara tatap muka.

Selain itu, terdapat juga beberapa kendala teknis pada proses kerja magang. Kendala teknis tersebut berkaitan dengan *framework* yang digunakan yaitu Laravel 8, beserta bahasa pemrograman PHP. Salah satu contoh kendala teknis yang dihadapi adalah perbedaan lingkungan program (*environment*) pada laptop yang digunakan untuk mengembangkan VDMS, dengan lingkungan program yang terdapat pada server Garuda Indonesia.

3.5 Solusi atas Kendala

Untuk mengatasi kendala berkomunikasi, digunakan aplikasi Whatsapp untuk membuat grup “Temp. Vendor Mgt System”. Grup Whatsapp tersebut beranggotakan Kevin Widjaja sebagai peserta magang, Ibu Raniyah sebagai pembimbing lapangan, Pak Bayu sebagai IT staf Garuda, dan Ibu Rischa sebagai asisten Ibu Raniyah.



Gambar 3.26 Informasi mengenai grup Whatsapp "Temp. Vendor Mgt System"

Informasi mengenai grup Whatsapp yang dibuat dapat dilihat pada Gambar 3.26. Grup Whatsapp tersebut digunakan untuk koordinasi mengenai pengerjaan magang, sekaligus menjadi tempat bertanya bila terdapat kendala. Selain itu, digunakan juga program Microsoft Teams untuk melakukan video call pada saat perkenalan dan penjelasan awal.

Sebagian besar kendala teknis yang dihadapi dapat terselesaikan dengan baik dengan cara mencari solusi atas kendala tersebut di Internet, dengan bantuan situs seperti Google, Stack Overflow, Youtube, dan Laracasts. Untuk kendala lingkungan pemrograman yang berbeda, pada komputer tempat dikerjakannya tugas magang ini, di-instal berbagai perangkat lunak seperti Composer, npm, dan artisan untuk membangun lingkungan pemrograman yang sesuai.