BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Dalam perusahaan Begawan Avaloka Indonesia Konsultama penulis di tempatkan di divisi Junior *Back End Programmer*. Tugas yang di berikan adalah membuat atau mengembangkan sebuah aplikasi atau sistem sesuai permintaan klien. Bapak Fauzi Hanif HarditYo selaku *supervisor* yang menjabat sebagai divisi *Project Manager*. Beliau bertugas untuk mengatur proyek setiap divisi.

3.2 Tugas yang Dilakukan

Penulis bekerja sebagai *Back End Programmer* dan memiliki tanggung jawab selama magang berlangsung. Tugas-tugas yang diberikan kepada penulis adalah sebagai berikut:

- Pembelajaran mengenai berbagai macam *framework* dan Bahasa seperti *framework* Laravel, Flutter serta Bahasa PHP, HTML, CSS dan Dart. Terdapat Pembelajaran juga mengenai *database* yang akan digunakan yaitu PostgreSQL.
- Mempelajari *Scope of Work, Business Flow* serta *design* mengenai proyek yang akan dilakukan.
- Mengerjakan modul yang diberikan yaitu modul CMS dalam CRM perusahaan.
- Mengerjakan *website* dinamis dimana data *website* tersebut dapat dirubah dari modul CMS yang sudah dibuat.

3.3 Uraian Pelaksanaan Magang

Uraian pada pelaksanaan magang adalah penulis memiliki fokus utama pada bagian pembelajaran secara umum mengenai Laravel, PostgreSQL, Dart dan mempelajari proyek ATA agar meningkatkan pengetahuan terhadap permasalahan yang akan diselesaikan. Selanjutnya adalah memulai pengerjaan mengenai proyek ATA akan tetapi, karena ada kendala dan situasi, fokus utama penulis dialihkan

untuk mengerjakan pengembangan CRM dengan modul CMS di dalamnya. Fokus selanjutnya adalah mengerjakan *website* klien yang dapat mengubah data *website* tersebut menggunakan CMS yang sudah dikembangkan penulis.

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (Minggu 1 - Minggu 10)

Minggu Ke -	Pekerjaan yang dilakukan
1-2	Mempelajari *Scope of Work* yang akan dikerjakan untuk proyek ATA, Melakukan persiapan dengan pembelajaran mengenai *framework* Laravel secara umum seperti *Routing, Middleware, Controllers*. Membuat *Application Programming Interface* (API) *Login* dan *Register* menggunakan Laravel, serta Pembelajaran *database* PostgreSQL secara umum.
3	Melanjutkan pembelajaran *database* PostgreSQL, membuat *database* untuk data *user* pada proyek ATA, memasukkan data *User* menggunakan Postman, dan pembelajaran *framework* Flutter serta pembelajaran bahasa Dart secara umum.
4	Mempelajari *Business Flow* ATA Health Web, melanjutkan pembelajaran *framework* Flutter mengenai *Flutter Notification* serta bahasa Dart, dan mulai mengerjakan koneksi *Front End* Flutter ke API *Back End* Laravel.
5-6	Melanjutkan pekerjaan koneksi *Front End*, memperbaiki *error* seperti *error* pada bagian *Front End* Flutter dan *error Cross-Origin Resource Sharing* (CORS), serta mempelajari *Business Process*ATA Health.
7	Mengubah API *Register* dan API *Login* dengan menambahkan *error handling* dan bentuk *response* berupa *JavaScript Object Notation* (JSON), *testing* API menggunakan Postman, dan Mengerjakan *Back End* mengenai Hubungi Kami serta pembuatan *form* yang *user* dapat kirim ke Email *Admin*.
8-9	Persiapan untuk perpindahan ke Pekerjaan Modul CMS dan *website*, mulai Mempelajari proyek Modul CMS dan *website*, dan mulai membuat *function index* serta *function table* pada Modul CMS fitur Berita.
10	Mengerjakan Fitur CRUD pada Modul CMS, Melakukan *bugfixing* dan *testing* untuk *table* pada Modul CMS Berita, dan mulai Mengerjakan *function index* dan *table* pada Modul CMS Fitur Keanggotaan IMI.

Tabel 3.2. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (Minggu 11 - Minggu 16)

Minggu Ke -	Pekerjaan yang dilakukan
11	Mengerjakan Fitur CRUD pada Modul CMS fitur Keanggotaan IMI, Mempelajari *design* yang akan digunakan untuk *Front End website*, dan Mengerjakan Bagian Beranda dan Tentang Kami.
12-13	Mengerjakan bagian awal *page Product*, Kemitraan, FAQ dan *blog*. Memperbaiki *error* dan mulai nyambungkan Modul CMS ke *website*.
14-16	Mengerjakan Tampilan *website* yang dapat menunjukkan data berdasarkan Modul CMS yang sudah dibuat (Menyambungkan Modul CMS ke Tampilan *website*)

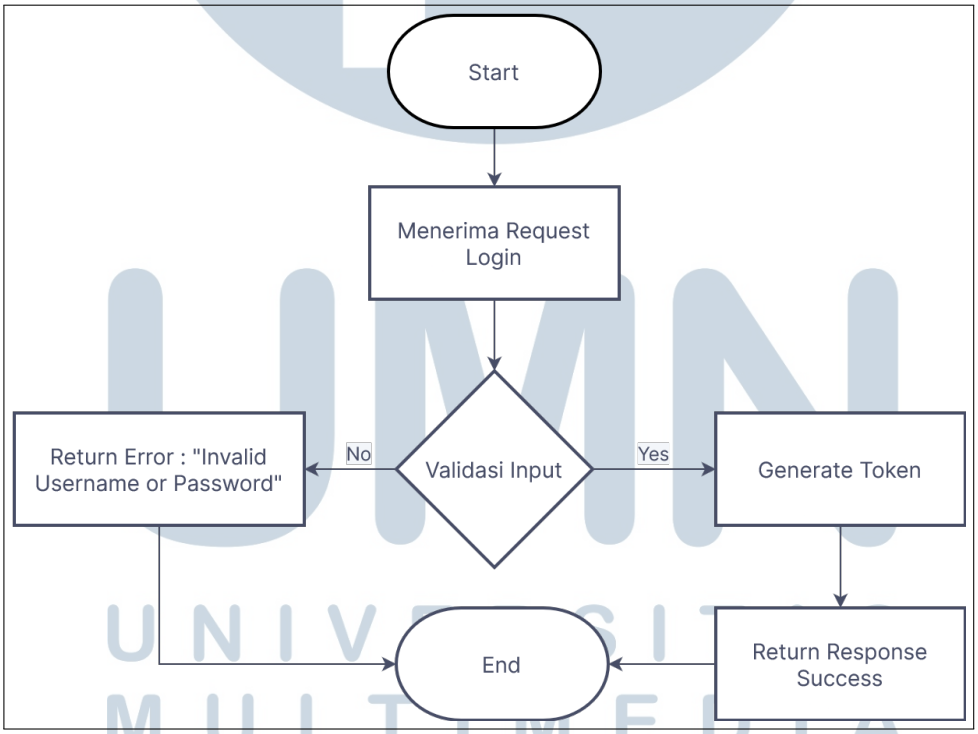
3.3.1 Minggu 1-2

Pada minggu pertama, fokus utama penulis terdapat pada pembelajaran *Scope of Work* terhadap proyek yang akan dilakukan yaitu proyek ATA. Selanjutnya, penulis melakukan persiapan untuk mengerjakan proyek ATA dengan pembelajaran mengenai *framework* Laravel seperti *Routing, Middleware dan Controllers*. Lalu, penulis melakukan latihan *Back End* dengan pembuatan API *Login* dan *Register* dan menggunakan pembelajaran *database* PostgreSQL karena penulis ingin meningkatkan pemahaman terhadap *database* yang ingin digunakan oleh perusahaan.

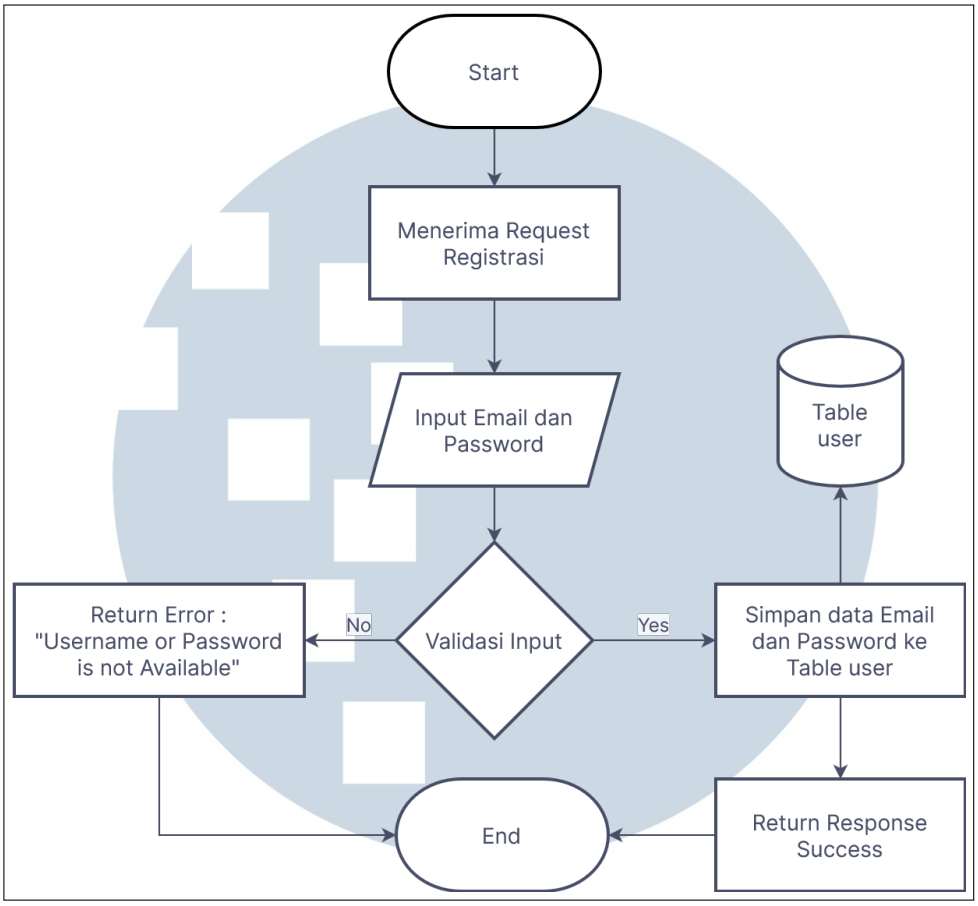
Scope of Work proyek ATA dapat dilihat dari gambar 3.1, kemudian *Flowchart* latihan API *Login* dan *Register* dapat dilihat dari gambar 3.2 dan gambar 3.3.

NO.	EPIC	FEATURES	SUB FEATURES	DESCRIPTION	PHASE
A	Akun	Profil	Super Admin		1
			Admin		1
B	Statistik/Grafik	1. Member 2. Penjualan 3. Sukses, Pending & Gagal			1
C	Member	Data Member	1. Perusahaan 2. Individu		1
D	Banner	1. Setting & Edit 2. Unggah 3. Delate			1
E	Promo	1. Setting & Edit 2. Unggah 3. Delate			1
F	Artikel	1. Setting & Edit 2. Unggah 3. Delate 4. Tag			1
G	Produk	1. Kategori 2. Setting & Edit 2. Unggah 3. Delate			1
H	Tentang Kami	1. Tentang 2. Syarat dan Ketentuan 3. Legalitas 4. Kontak			1

Gambar 3.1. Potongan dari *Scope of Work* proyek ATA



Gambar 3.2. *Flowchart* Latihan API *Login*

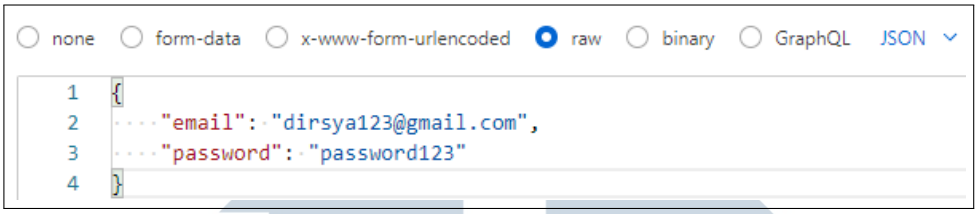


Gambar 3.3. *Flowchart* Latihan API *Register*

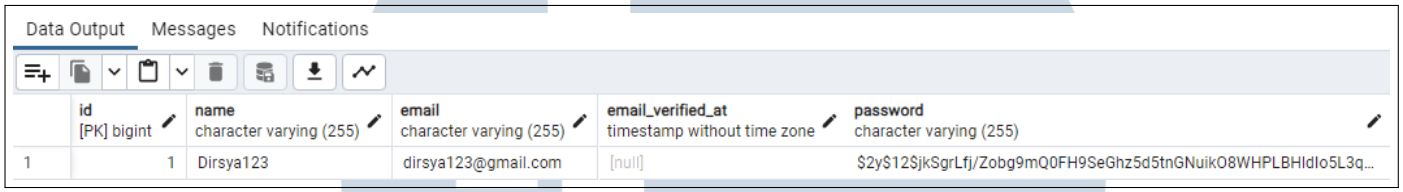
3.3.2 Minggu 3

Pada minggu ini, penulis melanjutkan pembelajaran *database* PostgreSQL yang dilakukan minggu lalu dan juga membuat *database* untuk data *user* yang nantinya akan dilakukan *testing input* data menggunakan Postman. Penulis juga melakukan pembelajaran *framework* Flutter serta pembelajaran bahasa Dart agar meningkatkan pemahaman untuk mengerjakan proyek-proyek yang berhubungan dengan *mobile*.

Input Field data *user* melalui Postman dapat dilihat dari gambar 3.4, kemudian hasil data *User* yang berhasil dimasukkan melalui Postman dapat dilihat dari gambar 3.5



Gambar 3.4. *Input Field* data *User*



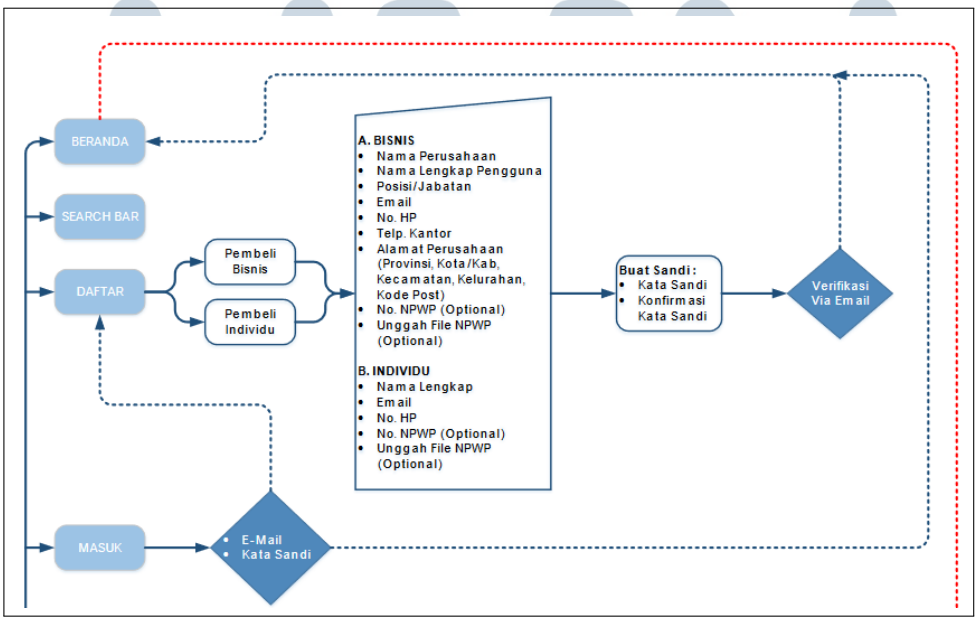
	id [PK] bigint	name character varying (255)	email character varying (255)	email_verified_at timestamp without time zone	password character varying (255)
1	1	Dirsya123	dirsya123@gmail.com	[null]	$2y$12$jkSgrLfj/Zobg9mQ0FH9SeGhz5d5tnGNuikO8WHPLBHIdIo5L3q...

Gambar 3.5. Data *user* yang berhasil dimasukkan ke *table*

3.3.3 Minggu 4

Pada minggu selanjutnya, penulis melakukan pembelajaran mengenai *Business Flow* ATA agar meningkatkan pemahaman terhadap proyek yang akan dikerjakan, Melanjutkan pembelajaran *framework* Flutter untuk persiapan jika mendapatkan tugas yang membutuhkan proyek di bidang *mobile* dan juga penulis mulai mengerjakan koneksi *Front End* Flutter ke API *Back End* Laravel.

Business Flow proyek ATA dapat dilihat dari gambar 3.6.

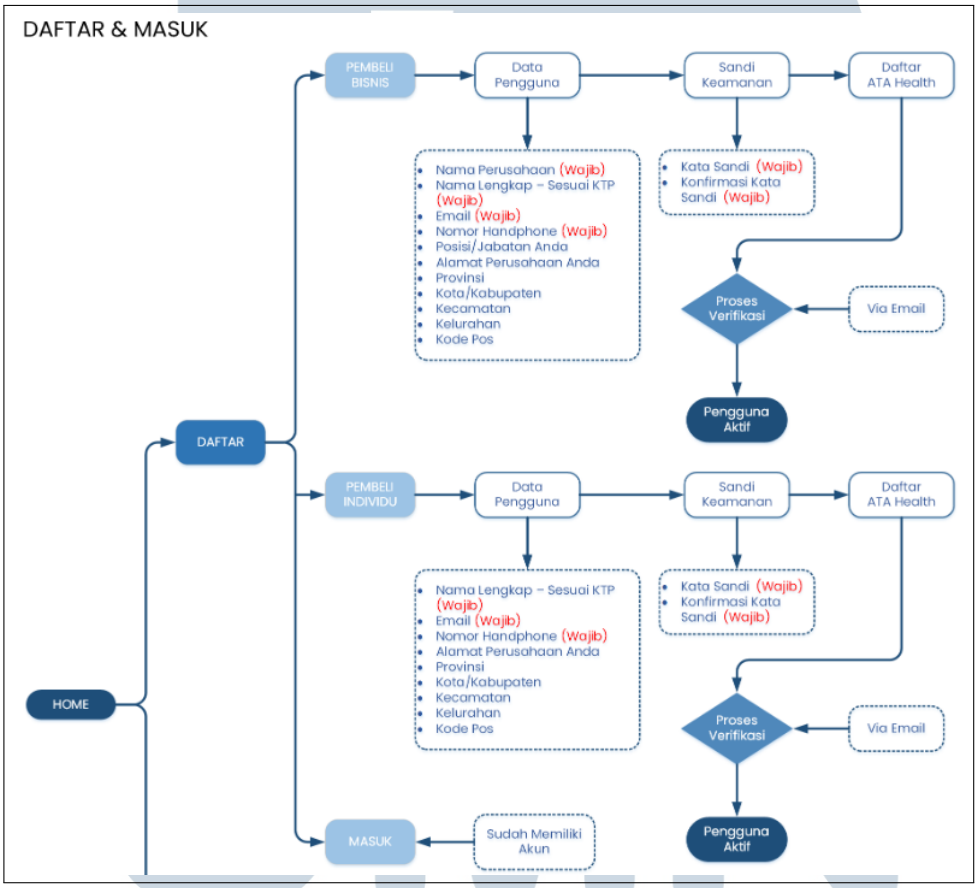


Gambar 3.6. Potongan *Business Flow* proyek ATA [1]

3.3.4 Minggu 5-6

Pada Minggu ke-5, penulis menlanjutkan pengerjaan koneksi *Front End*. Penulis juga melakukan perbaikan *error* dan mempelajari *Business Process* ATA Health untuk memahami proses berjalannya aplikasi ATA Health dengan lebih dalam.

Business Process proyek ATA dapat dilihat dari gambar 3.7.



Gambar 3.7. Potongan *Business Process* proyek ATA [1]

3.3.5 Minggu 7

Pada Minggu ke-7, penulis melakukan tugas yaitu Mengubah API *Register* dan API *Login* dengan menambahkan *error handling* dan bentuk *response* berupa JSON, *testing* API menggunakan Postman, dan Mengerjakan *Back End* mengenai Hubungi Kami serta Pembuatan *form* yang *user* dapat kirim ke Email *Admin*.

A. Perubahan API *Register* dan *Login*

Penulis melakukan perubahan API *Register* dan *Login* dengan menambahkan *error handling* dan bentuk *response* berupa JSON. Proses API *Register* adalah pertama akan dilakukan validasi *input* untuk registrasi, termasuk *field-field* yang wajib diisi dan *Field* tambahan seperti *password* minimal 8 karakter, ada huruf besar dan angka dan jika *user* adalah akun perusahaan. Selanjutnya, akan melakukan validasi *request* sesuai dengan validasi *input* di tahap sebelumnya. Jika gagal, akan *return* dengan pesan *error*. Jika berhasil, akan membuat *user* baru dengan data yang diterima dari *request* dan menyimpan *user* ke *database*. Terakhir, akan membuat *token* untuk validasi dan menampilkan hasil yang dikirim dalam *response* JSON.

Selanjutnya adalah proses API *Login*. Proses API *Login* yang telah dibuat penulis adalah pertama akan dilakukan validasi *input* yang diterima dari *request login*, yaitu jika *input* email dan *password* diisi dengan benar lalu akan mencari *user* di *database* berdasarkan email yang telah dibuat dan pengecekan jika *user* ditemukan serta *password* yang diberikan cocok dengan *hash password* di *database*. Jika email atau *password* salah maka akan mengembalikan *Response* dengan pesan *error* dan kode *status 401*. Jika berhasil, akan membuat *token* untuk autentikasi dan menyiapkan hasil yang akan dikirim dalam *response* JSON. Penulis juga membuat *error handling* dalam bentuk *exception* jika terjadi *error* selama proses *Login*. Setelah API selesah dibuat, penulis melakukan *testing* melalui Postman.

Potongan kode API *Register* dapat dilihat dari gambar 3.8, kemudian potongan kode API *Login* dapat dilihat dari gambar 3.9 serta hasil *response* JSON dari gambar 3.10

```php
public function userRegister(Request $request)
{
    // validasi untuk register (password minimal 8 karakter dan ada Uppercase+Angka)
    $array = [
        'is_perusahaan' => 'required|boolean', 'name' => 'required|string', 'email' => 'required|email',
        'password' => 'required|string', 'nomor_hp' => 'required|string', 'alamat' => 'required|string',
        'provinsi' => 'required|string', 'kota' => 'required|string', 'kecamatan' => 'required|string',
        'kode_pos' => 'required|string', 'nama_perusahaan' => 'nullable|required_if:is_perusahaan,true|string',
        'jabatan' => 'nullable|required_if:is_perusahaan,true|string',
    ];
    $validator = Validator::make($request->all(),$array);
    if ($validator->fails())
    {
        throw new InvalidArgumentException($validator->errors());
    }
    try
    {
        // Membuat user baru dengan data yang diterima
        $user = User::create([
            'name' => $request->name,'email' => $request->email,
            'password' => Hash::make($request->password),
            'nomor_hp' => $request->nomor_hp, 'alamat' => $request->alamat,
            'provinsi' => $request->provinsi,'kota' => $request->kota,
            'kecamatan' => $request->kecamatan, 'kode_pos' => $request->kode_pos,
        ]);

        if ($request->is_perusahaan)
        {
            $user['company_name'] = $request->nama_perusahaan;
            $user['position'] = $request->jabatan;
        }

        $user->save();
        $token = $user->createToken('authToken')->plainTextToken;

        // Menyiapkan hasil yang akan dikirimkan dalam respons JSON
        $result = [
            'success' => true,'code' => 201,
            'message' => 'Get List Paket Success',
            'errors' => null,'data' => $user,
            'token'=> $token
        ];

    } catch (\Exception $e) {
        $result = [
            'success' => false, 'code' => ($e->getCode() == 0 ? 500 : $e->getCode()),
            'message' => ($e->getCode() == 0 ? 'Error' : $e->getMessage()),
            'errors' => $e->getMessage(),
        ];
    }
    // Mengembalikan respons dalam format JSON
    return response()->json($result);
}
```


Gambar 3.8. Potongan Kode API *Register*

```php
public function userLogin(Request $request)
{
    try
    {
        // Validasi Input
        $request->validate([
            'email' => 'required|email',
            'password' => 'required',
        ]);

        $user = User::where('email', $request->email)->first();


        if (!$user || !Hash::check($request->password, $user->password))
        {

            return response()->json(['message' => 'Invalid email or password'], 401);
        }


        $token = $user->createToken('authToken')->plainTextToken;

        // Menyiapkan hasil yang akan dikirimkan dalam respons JSON
        $result =
        [
            'success' => true,
            'code' => 200,
            'message' => 'Login berhasil',
            'errors' => null,
            'data' => $user->name,
            'id' =>$user->id,
            'token'=> $token
        ];
    }

    catch (\Exception $e)
    {

        $result = [
            'success' => false,
            'code' => ($e->getCode() == 0 ? 500 : $e->getCode()),
            'message' => ($e->getCode() == 0 ? 'Error' : $e->getMessage()),
            'errors' => $e->getMessage(),
        ];
    }

    // Mengembalikan respons dalam format JSON
    return response()->json($result);
}
```


Gambar 3.9. Potongan Kode API *Login*

Gambar 3.10. Hasil *response* JSON setelah *input* menggunakan Postman

B. Pembuatan *Form* dan Hubungi Kami

Penulis mengerjakan pembuatan *form* dengan proses sebagai berikut, validasi *input* dari *request*, membuat kontak baru dengan data yang diterima dan menyimpannya ke *database*. Lalu mengirim email ke *Admin* dengan data kontak yang baru dibuat. Untuk Hubungi Kami, penulis menggunakan fitur *redirect* dari masing-masing *social media* seperti WhatsApp, Instagram, dan Tiktok.

3.3.6 Minggu 8-9

Pada minggu ini, Penulis melakukan peralihan dan persiapan untuk perpindahan ke pekerjaan modul CMS dan *website*. Setelah itu penulis mempelajari CMS untuk meningkatkan pemahaman terhadap modul CMS yang akan dikerjakan dan mulai membuat 2 *function* yaitu *Index* dan *Table* pada Modul CMS.

A. *Function Index*

Untuk *function Index*, pertama akan mengambil kategori berita dari *database* yang memiliki tipe ”Kategori” dan mengurutkannya berdasarkan kolom ’order’ secara *ascending* dan menyiapkan elemen data yang akan digunakan di halaman berita. Setelah itu akan mengembalikan *view* berita dengan data elemen serta kategori berita. Elemen menggunakan *Breadcrumb*, yaitu sebuah sebutan untuk desain sistem navigasi sekunder untuk menunjukkan lokasi halaman situs pengguna berada tanpa adanya pemeriksaan struktur *Uniform Resource Locator* (URL). *Breadcrumb* pada umumnya diletakkan pada posisi halaman atas situs dan umumnya dipakai pada *e-commerce* yang mempunyai banyak kategori. [4]

Potongan Kode *Function Index* dapat dilihat dari gambar 3.11

```
public function index()
{

  $qCategory = Category::query()
  ->where('type','=','Kategori')
  ->orderBy('order', 'ASC')
  ->get()
  ->toArray();

  $element =
  [
        'module' => "Berita",
        'open'   => "berita",
        'active'   => "berita",
        'breadcrumb' => true,
        'breadcrumbs' => [
            [
                'name' => 'Beranda',
                'active' => 0,
                'link'  => route('dashboard.index'),
            ],
            [
                'name' => 'Berita',
                'active' => 1,
                'link'  => route('berita.index'),
            ],
        ],
  ];

  return view('pages.berita.index', compact('element','qCategory'));
}
```

Gambar 3.11. Potongan Kode *Function Index*

B. *Function Table*

Selanjutnya adalah *function Table*, *function table* akan mendaftar kolom yang digunakan untuk *sorting* dan menentukan arah *sorting* tersebut. *Function Table* akan mendapatkan jumlah data yang ditampilkan dan *sorting* dari *request*. Lalu, *function* akan membuat *query* untuk mengambil data dari tabel dan menyiapkan *array* untuk data yang akan dikirimkan ke *view*. Setelah itu, *function* akan menghitung total data tanpa *filter* dan data dengan *filter* setelah menambahkan *limit*, *offset*, dan *order* pada *query*. Hasil akan dibuat menjadi data JSON untuk dikirimkan sebagai *response* dan mengembalikan *response*.

Potongan Kode *function Table* dapat dilihat dari gambar 3.12, kemudian hasil tampilan *Table* dapat dilihat dari gambar 3.13

```php
public function table(Request $request)
{
  $columns = array(0 =>'id', 1 =>'judul', 2 =>'deskripsi', 3 =>'kategori');

  $dir = $request->input('order.0.dir');
  $limit = $request->input('length');
  $start = $request->input('start');
  $order = $columns[$request->input('order.0.column')];

  $query = DB::table([redacted])
  ->join([redacted])
  ->select([redacted]);

  $searchValue = $request->input('search.value');
    if (!empty($searchValue)) {
        $query->where(function ($query) use ($searchValue) {
            $query
            ->where([redacted])
            ->orWhere([redacted]);
        });
    }

  $totalData = $query->count();
  $member = $query->offset($start)->limit($limit)->orderBy($order,$dir)->get();
  $totalFiltered = $totalData;
  $data = array();

  if(!empty($member)){
    foreach ($member as $s){
      $url_edit   = route('berita.edit',$s->id);
      $url_delete = route('berita.destroy',$s->id);
      $delete = '<a href="#" onclick="confirmDelete(event, \''.$url_delete.'\')" class="btnDelete"><i class="fas fa-eraser text-danger mr-2"></i></a>';
      $edit = '<a href="'.$url_edit.'" class="btnEdit"><i class="fas fa-edit"></i> </a>';

      $nestedData['id'] = $s->id; $nestedData['judul'] = $s->judul; $nestedData['deskripsi'] = $s->deskripsi;
      $nestedData['namakategori'] = $s->namakategori; $nestedData['kategori'] = $s->kategori;
      $nestedData['active'] = ($s->active == 't') ? 'Aktif' : 'Tidak Aktif';
      $nestedData['dibuat'] = Carbon::parse($s->created_at)->format('d M Y H:i');
      $nestedData['gambar'] = isset($s->gambar) ? '<img src="' . Storage::disk('s3')->temporaryUrl($s->gambar, now()->addDay()) . '" alt="Image" width="100" height="100">' : '';
      $nestedData['action'] = $edit." ".$delete;
      $data[] = $nestedData;
    }
  }
  $json_data = array(
    "draw"            => intval($request->input('draw')),
    "recordsTotal"    => intval($totalData),
    "recordsFiltered" => intval($totalFiltered),
    "data"            => $data
  );

  echo json_encode($json_data);
}
```

Gambar 3.12. Potongan Kode *Function Table*

ID	Gambar	Judul	Deskripsi	Kategori	Status	Dibuat	Action
1	Image	Balapan liar kolong	123	Mobil	Aktif	26 Apr 2024 14:13	
2		Balapan anak langit	abcdefadfaaaa	Motor	Tidak Aktif	26 Apr 2024 14:14	
3		Drifting buat Beginner	321	Mobil	Aktif	06 May 2024 23:32	

Gambar 3.13. Hasil Tampilan *Table*

3.3.7 Minggu 10

Untuk minggu ke-10, penulis mengerjakan fitur CRUD pada Modul CMS dengan *function-function* seperti *Create*,*Store*,*Edit*,*Update*, dan *Destroy*. Setelah itu, penulis mulai mengerjakan *function Index* dan *Table* untuk fitur Modul CMS selanjutnya yaitu fitur Keanggotaan IMI.

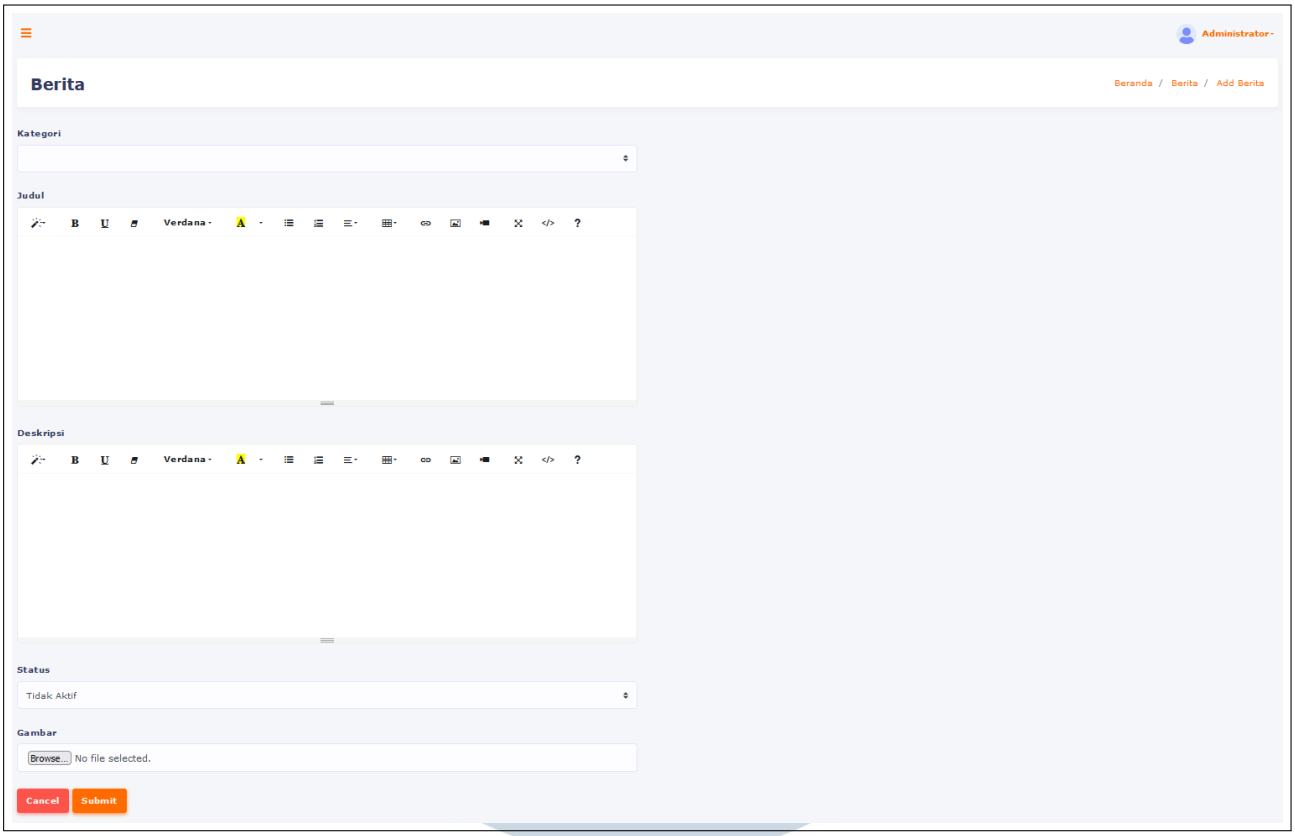
A. *Function Create dan Store*

Function Create akan mendapatkan data kategori yang memiliki tipe 'kategori' berdasarkan 'order' secara *ascending* dan menyusun elemen dengan informasi modul dan *breadcrumb*. Setelah itu, akan melakukan inisialisasi data dengan null agar tidak terjadi *error* dan mengembalikan tampilan halaman dengan bentuk *Form* dan juga variabel yang disediakan.

Potongan Kode *Function Create* dapat dilihat dari gambar 3.14, kemudian hasil tampilan *form* dapat dilihat dari gambar 3.15

```php
function create(Request $request)
{

  $qCategory = Category::query()
  ->where('type','=','Kategori')
  ->orderBy('order', 'ASC')
  ->get()
  ->toArray();

  $element =
    [
        'module' => "Berita",
        'open'   => "berita",
        'active'   => "berita",
        'breadcrumb' => true,
        'breadcrumbs' => [
            [
                'name' => 'Beranda',
                'active' => 0,
                'link'  => route('dashboard.index'),
            ],
            [
                'name' => 'Berita',
                'active' => 1,
                'link'  => route('berita.index'),
            ],
            [
              'name' => 'Add Berita',
              'active' => 1,
              'link'  => route('berita.create'),
            ],
        ],
    ];

  $data = null;
  return view('pages.berita.form', compact('element', 'data', 'qCategory'));
}
```

Gambar 3.14. Potongan Kode *Function Create*



Gambar 3.15. Tampilan *Form* untuk Mengisi Data

Ketika *user* mengisi *form* tersebut dan mengklik submit, akan menjalankan *function Store* untuk menyimpan data ke *Table* dengan tahap-tahap berikut, yaitu validasi *input* yang diterima dari *request*, jika gagal maka akan mengembalikan *response* dengan *error*. Setelah itu, akan mulai transaksi *database* dan membuat *auto-increment* ID dengan mendapatkan ID terbesar dari tabel dan menambahkannya dengan 1 untuk ID baru. Lalu, terdapat pengecekan jika ada *file* gambar atau tidak. *Function Store* akan melakukan pengisian *array input* dengan data yang divalidasi dan memasukkan data baru ke dalam tabel. Jika sudah, akan *commit* transaksi dan mengembalikan *response* sukses jika berhasil dan *rollback* jika gagal.

Potongan Kode *Function Store* dapat dilihat dari gambar 3.16,

```
public function store(Request $request)
{
    $validator = Validator::make($request->all(), [
        'judul' => 'required','deskripsi' => 'required',
        'kategori' => 'required','active' => 'required',
    ]);

    if ($validator->fails()) {
        return [
            'success' => false, 'code' => 422,
            'message' => 'Error Validasi','errors' => $validator->errors(),
        ];}
    try
    {
        DB::beginTransaction();

        $maxId = DB::table([redacted])->max('id');
        $newId = $maxId + 1;

        if(isset($request->clubPicture) && $request->hasFile('clubPicture')){
            $filepp = $request->file('clubPicture');
            $filenamepp = [redacted] . time() . '.' . $filepp->getClientOriginalExtension();
            Storage::disk('s3')->put([redacted].$filenamepp, file_get_contents($filepp));
            $input['gambar'] = [redacted].$filenamepp;
        }

        else{$input['gambar'] = null;}

        $input['id'] = $newId; $input['kategori'] = $request->kategori;
        $input['judul'] = $request->judul; $input['deskripsi'] = $request->deskripsi;
        $input['active'] = $request->active === "Aktif" ? true : false;
        $input['user_created'] = Session::get('id');
        $input['created_at'] = date('Y-m-d H:i:s');

        DB::table([redacted])->insert($input);

        $pesan = "Berita berhasil dibuat";
        DB::commit();

        $result = [
            'success' => true,'code' => 201,
            'message' => $pesan,'errors' => null,
        ];}
    catch (\Exception $e) {
        DB::rollback();
        $result = [
            'success' => false,'code' => $e->getCode(),
            'message' => $e->getMessage(),'errors' => $e->getMessage(),];
    }

    return $result;
}
```

Gambar 3.16. Potongan Kode *Function Store*

B. *Function Edit dan Update*

Function Edit akan melakukan proses mendapatkan semua kategori dan inisialisasi *array* element untuk mengambil modul, navigasi dan *breadcrumb* yang sama. Akan tetapi, data yang di ambil berdasarkan ID dari tabel dan akan mengembalikan *view form* seperti pada *function Create* untuk proses *edit* Data.

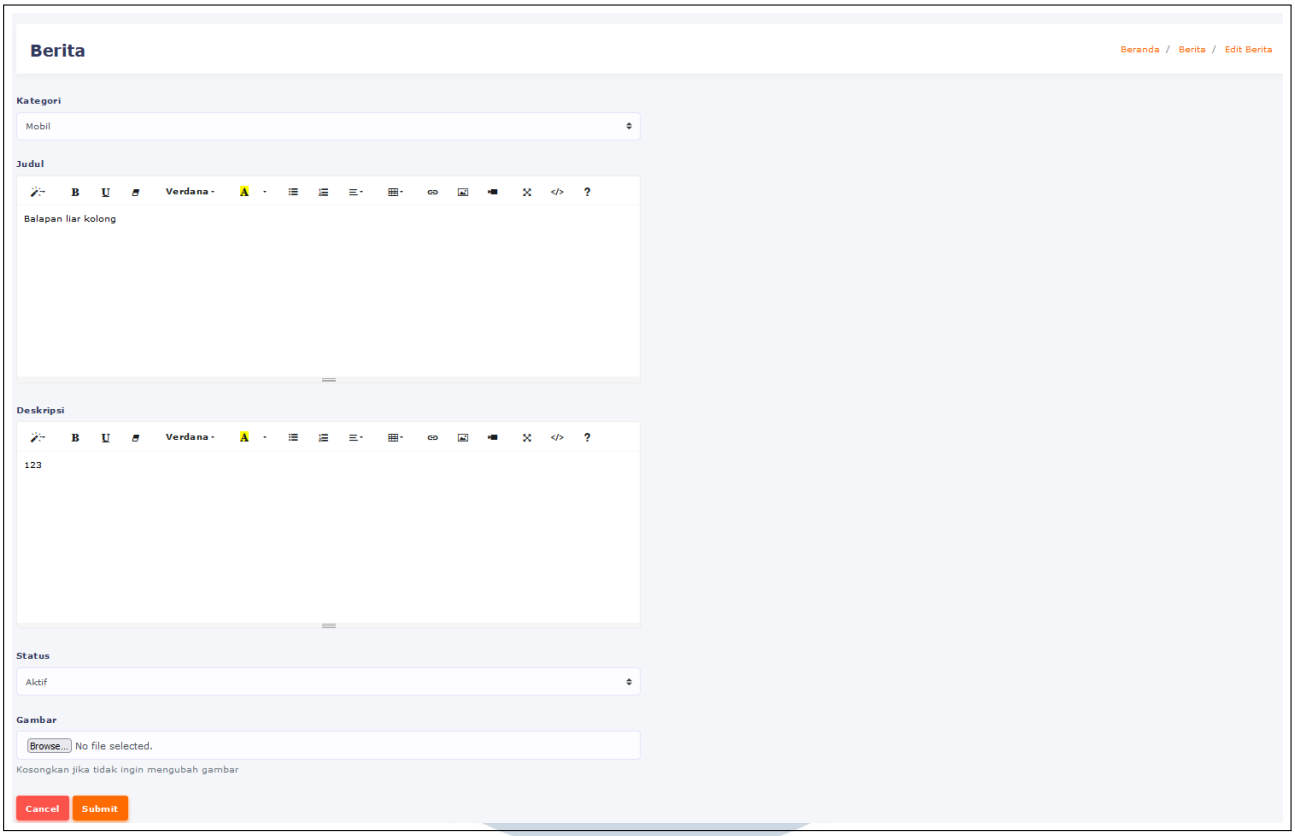
Potongan Kode *Function Create* dapat dilihat dari gambar 3.17,

```
public function edit($id)
  {

    $qCategory = Category::query()
    ->where('type','=','Kategori')
    ->orderBy('order', 'ASC')
    ->get()
    ->toArray();

      $element = [
        'module' => "Berita",
        'open'   => "berita",
        'active'   => "berita",
        'breadcrumb' => true,
        'breadcrumbs' => [
              [
                  'name' => 'Beranda',
                  'active' => 0,
                  'link'  => route('dashboard.index'),
              ],
              [
                  'name' => 'Berita',
                  'active' => 1,
                  'link'  => route('berita.index'),
              ],
              [
                  'name' => 'Edit Berita',
                  'active' => 1,
                  'link'  => route('berita.edit', $id),
              ],
          ],
      ];

      $data = DB::table([redacted])->where('id', $id)->first();
      return view('pages.berita.form', compact('element','data' ,'qCategory'));
  }
```

Gambar 3.17. Potongan Kode *Function Edit*

Ketika *function Edit* sudah dilakukan, akan muncul *form* dengan data yang sudah diambil berdasarkan ID dari tabel. Ketika *user* sudah selesai dengan proses *edit* dan klik *submit*, akan melakukan *function Update*.

Tampilan *Form* Edit dapat dilihat dari gambar 3.18



Gambar 3.18. Tampilan *Form* Edit

Function Update akan melakukan tahap-tahap yang mirip seperti Store, yaitu validasi *input* yang diterima dari *request*, terjadinya transaksi *database* akan tetapi tidak membuat *auto-increment* ID dikarenakan sudah ada ID sebelumnya. Proses pengecekan gambar dan pengisian *array input* dengan data yang sudah divalidasi dan dibuat lalu dimasukkan kedalam tabel. Terdapat juga *commit* transaksi jika *response* sukses dan *rollback* jika gagal

Potongan Kode *Function Update* dapat dilihat dari gambar 3.19

```
public function update(Request $request, $id)
{
    $validator = Validator::make($request->all(), [
        'judul' => 'required','deskripsi' => 'required',
        'kategori' => 'required','active' => 'required'
    ]);

    if ($validator->fails()) { return [
            'success' => false,'code' => 422,
            'message' => 'Error Validation',
            'errors' => $validator->errors(),
        ];
    }
    try
    {
        DB::beginTransaction();
        $data = DB::table([redacted])->where('id', $id)->first();
        if (!$data) {
            throw new \Exception('Berita not found', 404);
        }
        $input = [];

        if ($request->hasFile('clubPicture')) {
            $filepp = $request->file('clubPicture');
            $filenamepp = [redacted] . time() . '.' . $filepp->getClientOriginalExtension();
            Storage::disk('s3')->put([redacted] . $filenamepp, file_get_contents($filepp));
            $input['gambar'] = [redacted] . $filenamepp;
        }

        $input['judul'] = $request->judul; $input['deskripsi'] = $request->deskripsi;
        $input['kategori'] = $request->kategori;
        $input['active'] = $request->active === "Aktif" ? true : false; $input['updated_at'] = now();

        DB::table([redacted])->where('id', $id)->update($input);
        DB::commit();
        $result =
        [
            'success' => true,'code' => 201,
            'message' => 'Berita berhasil diedit',
            'errors' => null,
        ];
    }
    catch (\Exception $e){
        DB::rollback();
        $result =
        [
            'success' => false,'code' => $e->getCode(),
            'message' => $e->getMessage(),'errors' => $e->getMessage(),
        ];
    }
    return $result;
}
```

Gambar 3.19. Potongan Kode *Function Update*

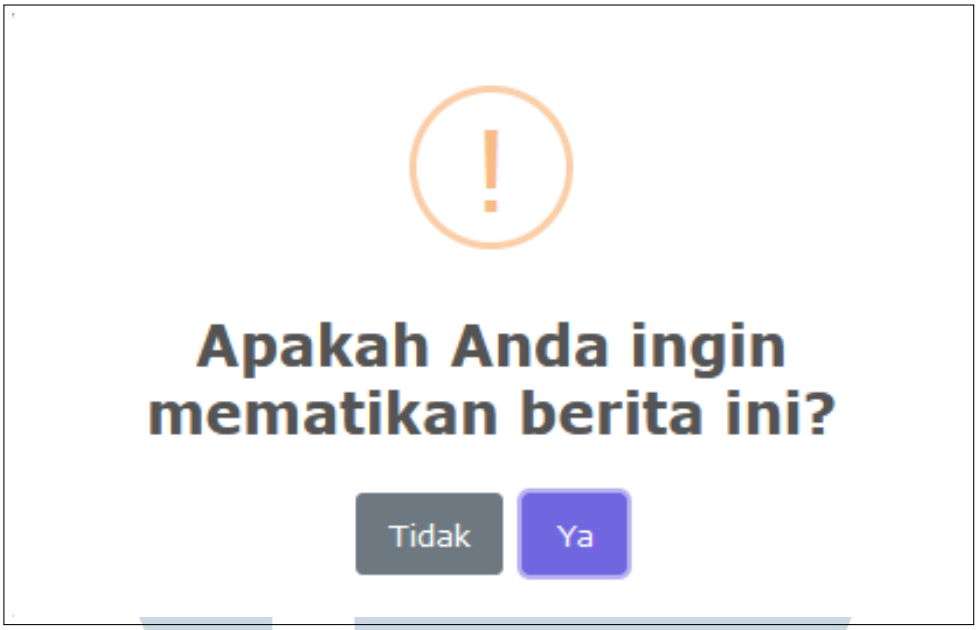
C. *Function Delete*

Untuk *function Delete*, akan terjadi proses yang berbeda dari *function Delete* pada umumnya. *Function Delete* yang akan dilakukan adalah pengecekan jika data yang ingin dihapus ada atau tidak lalu proses perubahan *field* 'active' dari 'true' menjadi 'false' karena klien tidak mau ada data yang terhapus di *database*. Untuk tampilannya terdapat *Sweet Alert* (SWAL) yang akan mengingatkan pengguna untuk konfirmasi melakukan *function Delete* atau tidak.

Potongan Kode *Function Delete* dapat dilihat dari gambar 3.20, kemudian Tampilan SWAL dapat dilihat dari gambar 3.21

```
DB::beginTransaction();

$data = DB::table([redacted])->where('id', $id)->first();
if (!$data)
{
    throw new \Exception('Berita not found', 404);
}

DB::table([redacted])->where('id', $id)->update(['active' => false]);

DB::commit();

$result =
[
    'success' => true,
    'code' => 200,
    'message' => 'Berita menjadi tidak aktif',
    'errors' => null,
];

return redirect()->route('berita.index')->with('success', 'Berita menjadi tidak aktif');
```

Gambar 3.20. Potongan Kode *Function Delete*

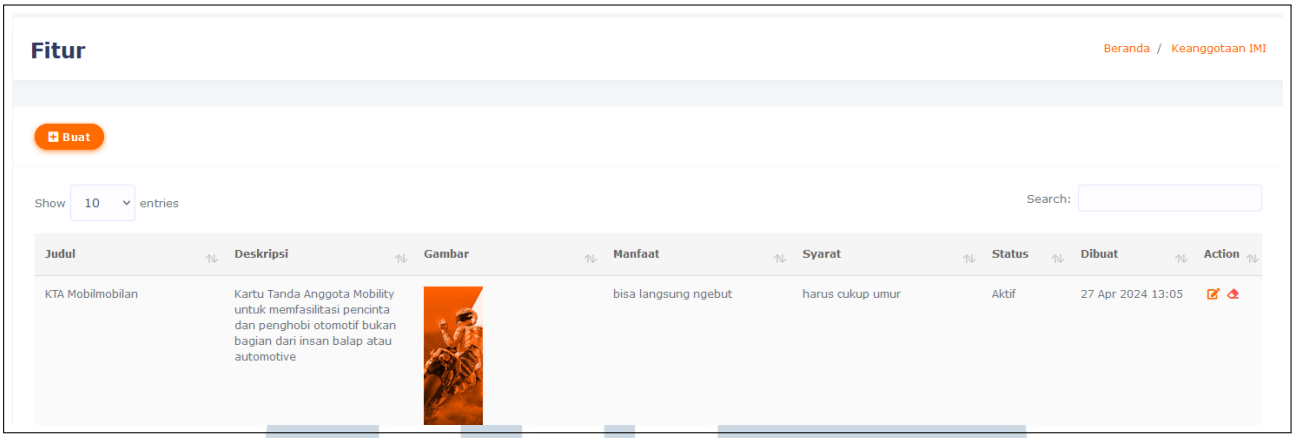


Gambar 3.21. Tampilan SWAL

3.3.8 Minggu 11

Pada minggu ini, penulis mengerjakan *function* CRUD dalam Modul CMS yang memiliki proses seperti CRUD fitur berita akan tetapi berbeda tempat yaitu terletak di fitur keanggotaan IMI. proses tersebut adalah membuat *function Create* dan *Store* untuk membuat dan menyimpan data di *table database*, *Edit* dan *Update* untuk melakukan perubahan data yang sudah dimiliki, dan *Delete* untuk melakukan *soft deletion* data di *table database* dengan cara merubah *field* 'active' dari 'true' menjadi 'false'. Penulis juga mulai mempelajari *design* yang akan digunakan untuk *Front End website* dimana data *Front End website* tersebut akan tersinkronisasi dengan Modul CMS yang sudah dibuat.

Tampilan Tabel Fitur dapat dilihat dari gambar 3.22



Gambar 3.22. Tampilan Tabel Fitur

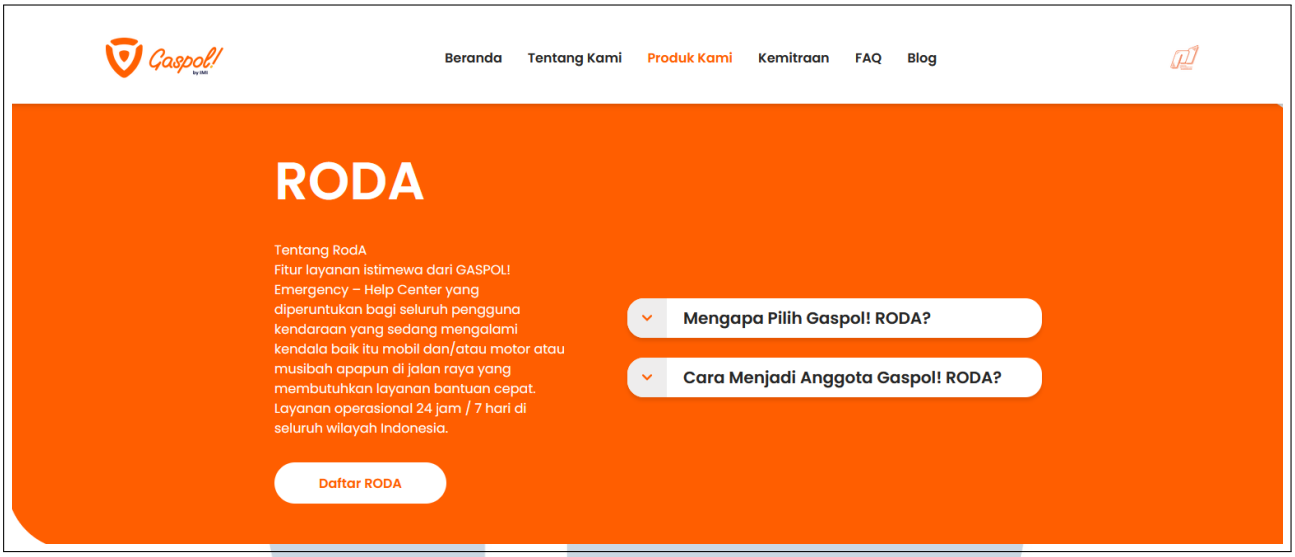
3.3.9 Minggu 12-13

Pada minggu ke-12, penulis mengerjakan bagian awal dari *page Product*, *blog*, dan FAQ. Penulis juga melakukan *error fixing* dan mulai proses sinkronisasi modul CMS kepada *website* yang sedang dibuat.

Potongan Kode Product dapat dilihat dari gambar 3.23, kemudian gambar tampilan Product dapat dilihat dari gambar 3.24

```
@if ($product['tipe'] == 0)
    <div class="rounded-se-[80px] rounded-es-[80px] bg-[#FAFAFA] w-full flex items-center justify-center">
        <div class="pt-[72px] pb-[56px] max-[992px]:pb-[72px]">
            <div class="container mx-auto px-24 max-[992px]:px-12 max-[576px]:px-8">
                <div class="flex items-center gap-[72px] max-[992px]:flex-col max-[992px]:gap-4">
                    <div class="max-w-[357px] max-[992px]:max-w-full">
                        <h2 class="text-[64px] font-semibold leading-[64px] mb-10">{{ $product['name'] }}</h2>
                        <div>{!! $product['descre'] !!}</div>
                        <a href="#"
                            class="mt-[30px] text-white bg-primary h-[50px] inline-flex justify-center items-center px-[54px] rounded-[100px] font-semibold leading-none">Daftar
                            {{ $product['name'] }}</a>
                    </div>
                    <div>
                        <ul class="mt-[54px] flex flex-col max-w-[504px] gap-4">
                            @foreach ($product['rodas'] as $roda)
                                <li x-data="accordion('{{ $roda['id'] }}one')">
                                    <div @click="handleClick()"
                                        class="bg-white rounded-[20px] flex flex-row justify-between items-center cursor-pointer shadow-[0px_4px_4px_0px_rgba(0,0,0,0.10)]">
                                        <div class="bg-primary p-[18px] rounded-ss-[20px] rounded-es-[20px]">
                                            <svg :class="handleRotate()"
                                                class="w-3 h-3 text-white transform transition-transform duration-500"
                                                aria-hidden="true" xmlns="http://www.w3.org/2000/svg" fill="none"
                                                viewBox="0 0 10 6">
                                                <path stroke="currentColor" stroke-linecap="round"
                                                    stroke-linejoin="round" stroke-width="2" d="M9 5 5 1 1 5" />
                                            </svg>
                                        </div>
                                        <span
                                            class="{{ strlen($roda['heading']) > 40 ? 'px-5 w-full font-semibold text-base leading-[1.3]' : 'px-5 w-full font-semibold text-xl' }}
                                        </div>
                                        <div x-ref="tab" :style="handleToggle()"
                                            class="bg-white mt-2 rounded-[20px] overflow-hidden max-h-0 duration-500 transition-all">
                                            <div class="text-gray-900 p-10">{!! $roda['body'] !!}</div>
                                        </div>
                                    </li>
                                @endforeach
                            </ul>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
@else
```

Gambar 3.23. Potongan Kode *Page Product*

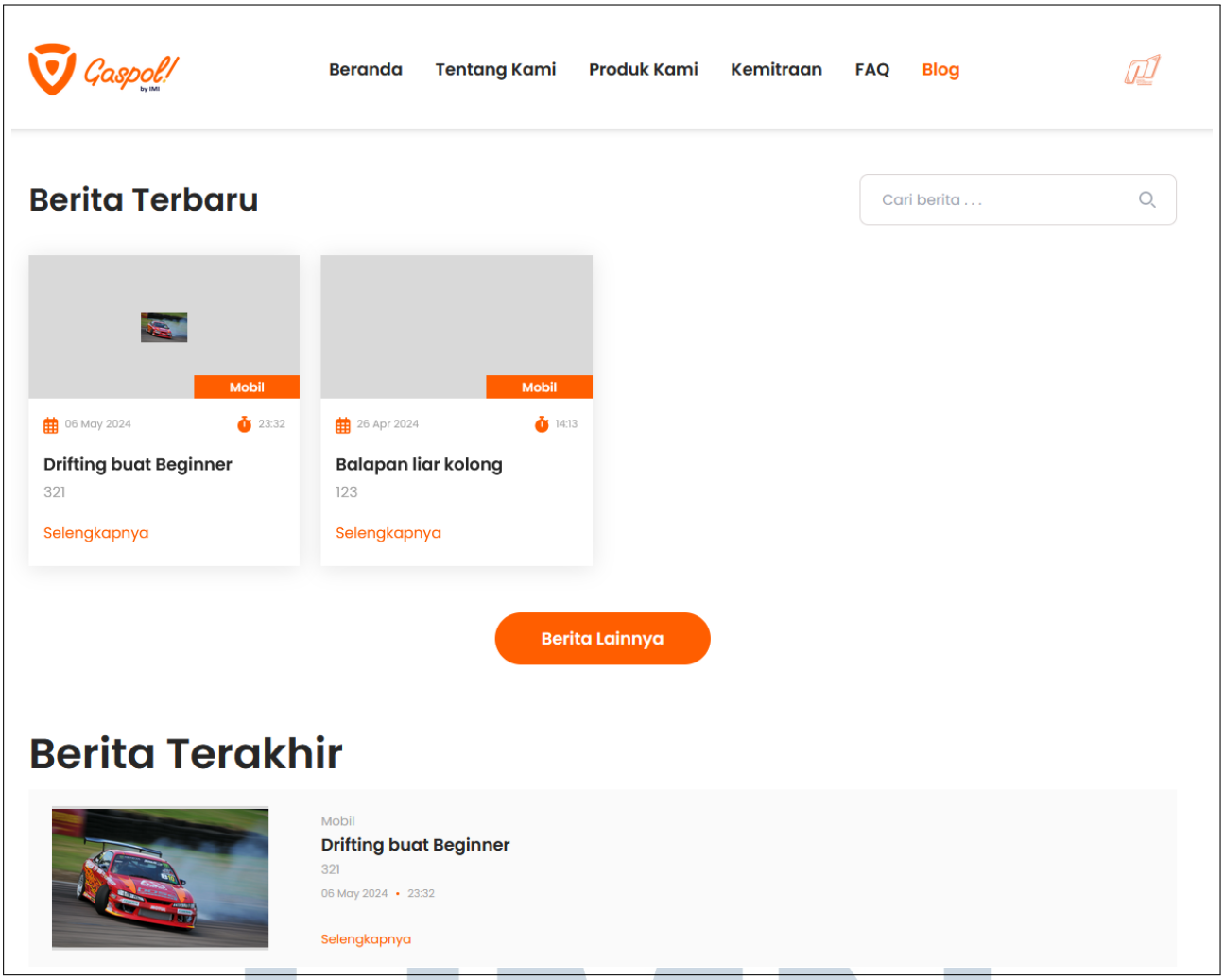


Gambar 3.24. Tampilan *Page Product*

Potongan Kode *Blog* dapat dilihat dari gambar 3.25, kemudian gambar tampilan *Blog* dapat dilihat dari gambar 3.26

```
<div class="container mx-auto px-24 max-[992px]:px-10">
    <div class="flex justify-between items-center">
        <h2 class="text-3xl text-center font-semibold leading-none">Berita Terbaru</h2>
        <form>
            <label for="default-search"
                class="mb-2 text-sm font-medium text-gray-900 sr-only dark:text-white">Search</label>
            <div class="relative">
                <input type="search" id="news-search" name="news-search"
                    class="block w-full py-[13px] px-5 text-sm text-gray-900 border border-[#DAD4D4] rounded-lg bg-white focus:ring-blue-500 focus:border-blue-500"
                    placeholder="Cari berita . . ." required>
                <div class="absolute inset-y-0 end-0 flex items-center pe-5 pointer-events-none">
                    <svg class="w-4 h-4 text-gray-500 dark:text-gray-400" aria-hidden="true"
                        xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 20 20">
                        <path stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
                            d="m19 19-4-4m0-7A7 7 0 1 1 1 8a7 7 0 0 1 14 0Z" />
                    </svg>
                </div>
            </div>
        </form>
    </div>
    <div class="grid grid-cols-4 gap-5 mt-[28px] max-[1024px]:grid-cols-2 max-sm:grid-cols-1">
```

Gambar 3.25. Potongan Kode *Page Blog*



Gambar 3.26. Tampilan *Page Blog*

Potongan Kode FAQ dapat dilihat dari gambar 3.27, kemudian gambar tampilan FAQ dapat dilihat dari gambar 3.28

```
<div class="w-full h-[532px] flex items-center justify-center bg-[#D9D9D9]">
    <img src="/images/empty-img.png" class="w-[156px]" alt="" />
</div>
<div class="pt-[68px] pb-[52px]">
    <div class="container mx-auto px-24 max-[992px]:px-8">
        <h2 class="text-[2.5rem] text-center font-semibold leading-none">Pertanyaan Seputar Gaspol!</h2>
        <hr class="max-w-[140px] bg-primary h-[5px] rounded-[50px] mx-auto mt-4 mb-11" />
        <div class="flex justify-center gap-11 max-[992px]:flex-col max-[992px]:gap-6">
            <ul class="mt-[54px] flex flex-col max-w-[504px] gap-3 max-[992px]:max-w-full">
                @foreach ($faqs as $faq)
                    <li x-data="accordion({{ $faq['id'] }})">
                        <div @click="handleClick()"
                            class="flex flex-row justify-between items-center cursor-pointer bg-white shadow-[0px_0px_4px_0px_rgba(0,0,0,0.25)]">
                            <div class="bg-primary p-[18px] rounded-ss-[5px] rounded-es-[5px]">
                                <svg :class="handleRotate()"
                                    class="w-3 h-3 text-white transform transition-transform duration-500"
                                    aria-hidden="true" xmlns="http://www.w3.org/2000/svg" fill="none"
                                    viewBox="0 0 10 6">
                                    <path stroke="currentColor" stroke-linecap="round" stroke-linejoin="round"
                                        stroke-width="2" d="M9 5 5 1 1 5" />
                                </svg>
                            </div>
                            <span class="px-5 w-full font-semibold text-base">{{ $faq['heading'] }}</span>
                        </div>
                        <div x-ref="tab" :style="handleToggle()"
                            class="mt-2 bg-white overflow-hidden max-h-0 duration-500 transition-all rounded-[5px] shadow-[0px_0px_4px_0px_rgba(0,0,0,0.25)]">
                            <p class="text-gray-900 px-4 py-5">{{ $faq['body'] }}</p>
                        </div>
                    </li>
                @endforeach
            </ul>
            <div class="max-[992px]:mx-auto">
                <img src="/images/faq/faq-image.jpg" class="max-w-[414px] max-[992px]:max-w-full" alt="">
            </div>
        </div>
    </div>
</div>
```

Gambar 3.27. Potongan Kode Page FAQ

Gambar 3.28. Tampilan Page FAQ

3.3.10 Minggu 14 - Minggu 16

Pada Minggu-Minggu ini, fokus utama penulis ada dalam pekerjaan tampilan *website* yang mengambil data-data dari *table* dan modul CMS yang sudah dibuat. Data-data yang diambil dari modul CMS adalah Berita dan Keanggotaan IMI.

A. Tampilan Berita

Terdiri dari *function Index* dan Detail, *function Index* melakukan proses membuat *query* untuk mengambil data dari tabel dan melakukan *join*. Setelah itu mengecek apakah ada parameter 'news-search' yang dikirim melalui *request* GET dan jika ada akan menambahkan kondisi pencarian berdasarkan judul berita. Selanjutnya adalah menambahkan pengurutan berdasarkan tanggal pembuatan secara *descending* dan mendeklarasi *array* kosong untuk menyimpan hasil data

yang sudah di proses. Lalu, memasukkan setiap *item* ke dalam *array* newss dan jika sudah akan mengembalikan *view* dengan data berita yang sudah diproses. *Function* Detail juga melakukan proses *query* yang sama dan menformat data ke *array* dengan struktur yang sama dan mengembalikan *view* dengan data tersebut. Data-data yang hanya ditunjuk pada *website* adalah data yang memiliki *field* 'active' true.

Potongan Kode *function Index* dan Detail dapat dilihat dari gambar 3.29 dan gambar 3.30, kemudian gambar *table* berita dan perbandingannya di *website* dapat dilihat dari gambar 3.31 dan gambar 3.32

```
public function index()
{

  $queryBerita = DB::table([redacted])
  ->join('category as x', [redacted], '=', 'x.id')
  ->select([redacted], '[redacted]')
  ->where([redacted], true);

  if(isset($_GET['news-search'])){
    $queryBerita = $queryBerita->where(DB::raw( [redacted] ), 'like', '%' . strtolower($_GET['news-search']) . '%');
  }

  $queryBerita = $queryBerita
  ->orderBy('created_at', 'DESC')
  ->get()
  ->toArray();

  $newss = [];
  $cloop = 1;
  foreach($queryBerita as $berita)
  {
    $newss[] = [
      'id'=> $berita->id,
      'datePublish' => Carbon::parse($berita->created_at)->format('d M Y'),
      'timePublish' => Carbon::parse($berita->created_at)->format('H:i'),
      'title' => $berita->judul,
      'description' => $berita->deskripsi,
      'category' => $berita->namakategori,
      'gambar' => isset($berita->gambar) ? Storage::disk('s3')->temporaryUrl($berita->gambar, now()->addDay()) : '/images/empty-img.png',

    ];
    $cloop++;
  }

    return view('news/index',
        [
            'newss'=>$newss
        ]);
```

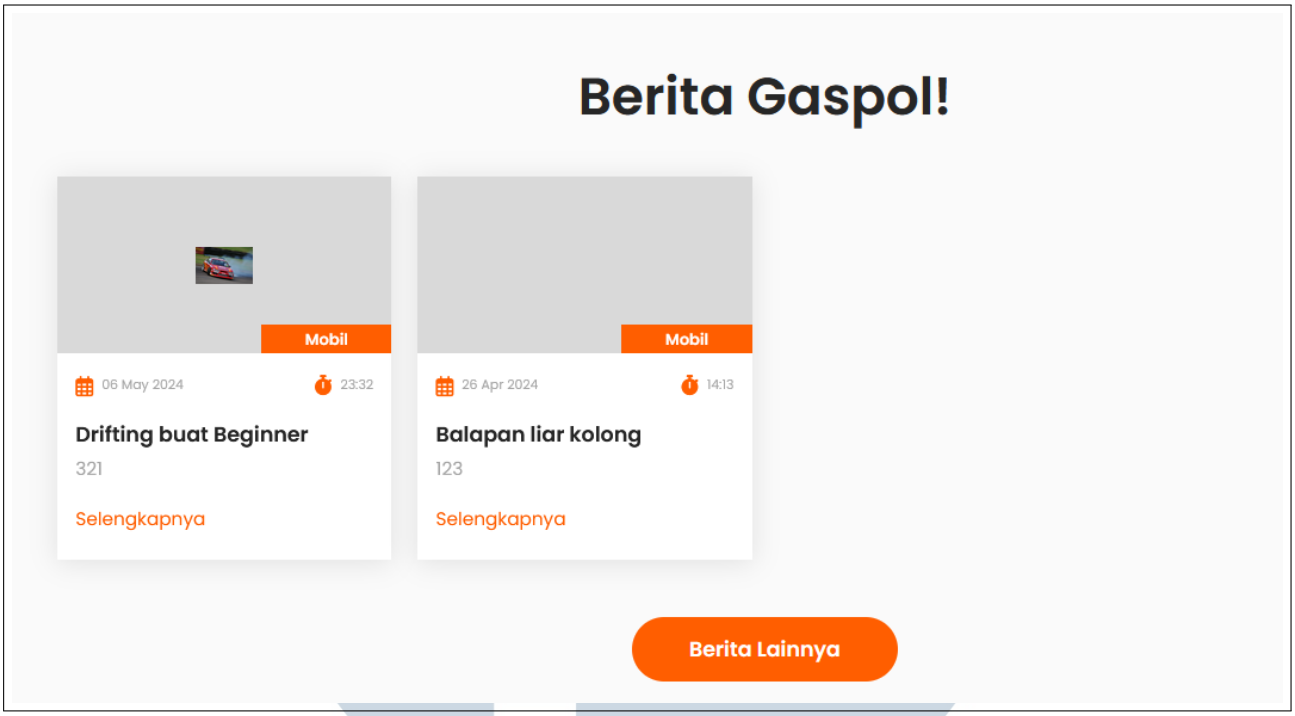
Gambar 3.29. Potongan Kode *Index* Berita

```php
public function detail($id)
{
    $berita = DB::table([redacted])
        ->join('category as x', [redacted], '=', 'x.id')
        ->select([redacted], [redacted])
        ->where([redacted], $id)
        ->where([redacted], true)
        ->first();

    if (!$berita)
    {
        abort(404);
    }

    $news = [
        'id' => $berita->id,
        'datePublish' => Carbon::parse($berita->created_at)->format('d M Y'),
        'timePublish' => Carbon::parse($berita->created_at)->format('H:i'),
        'title' => $berita->judul,
        'description' => $berita->deskripsi,
        'category' => $berita->namakategori,
        'gambar' => isset($berita->gambar) ? Storage::disk('s3')->temporaryUrl($berita->gambar, now()->addDay()) : '/images/empty-img.png',
    ];

    return view('news.detail', ['news' => (object) $news]);
}
```

Gambar 3.30. Potongan Kode Detail Berita

ID	Gambar	Judul	Deskripsi	Kategori	Status	Dibuat	Action
1	Image	Balapan liar kolong	123	Mobil	Aktif	26 Apr 2024 14:13	
2		Balapan anak langit	abcdefadfaaaa	Motor	Tidak Aktif	26 Apr 2024 14:14	
3		Drifting buat Beginner	321	Mobil	Aktif	06 May 2024 23:32	

Gambar 3.31. Tampilan di CMS



Gambar 3.32. Tampilan di *Page* Berita

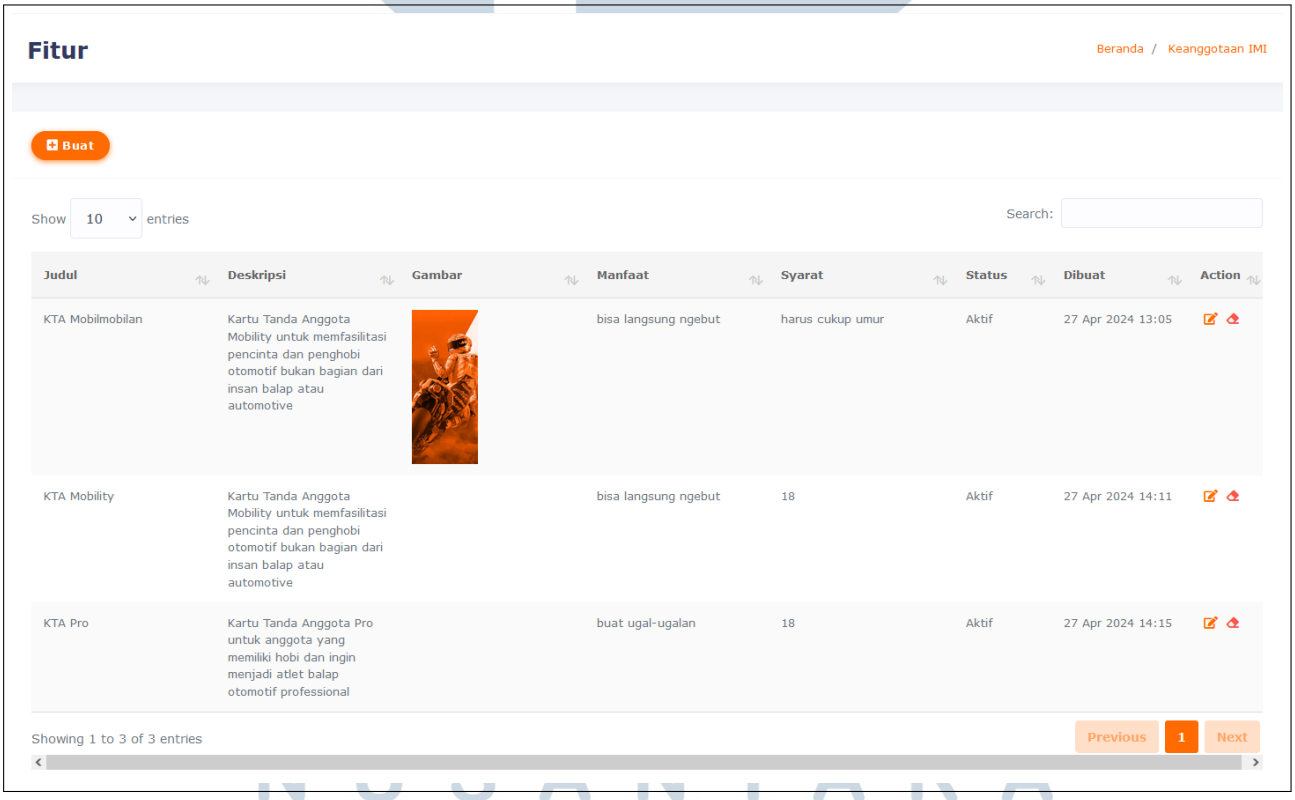
B. Tampilan Keanggotaan IMI

Terdapat *function Index* dimana *function* tersebut akan membuat *query* untuk mengambil data dari tabel yang aktif dan memiliki ID yang sesuai dengan *request*. Selanjutnya, akan mendeklarasi *array* kosong untuk menyimpan hasil data dan memasukkan setiap *item member* ke dalam *array*. Lalu, akan mengembalikan *view* dengan data yang sudah diproses.

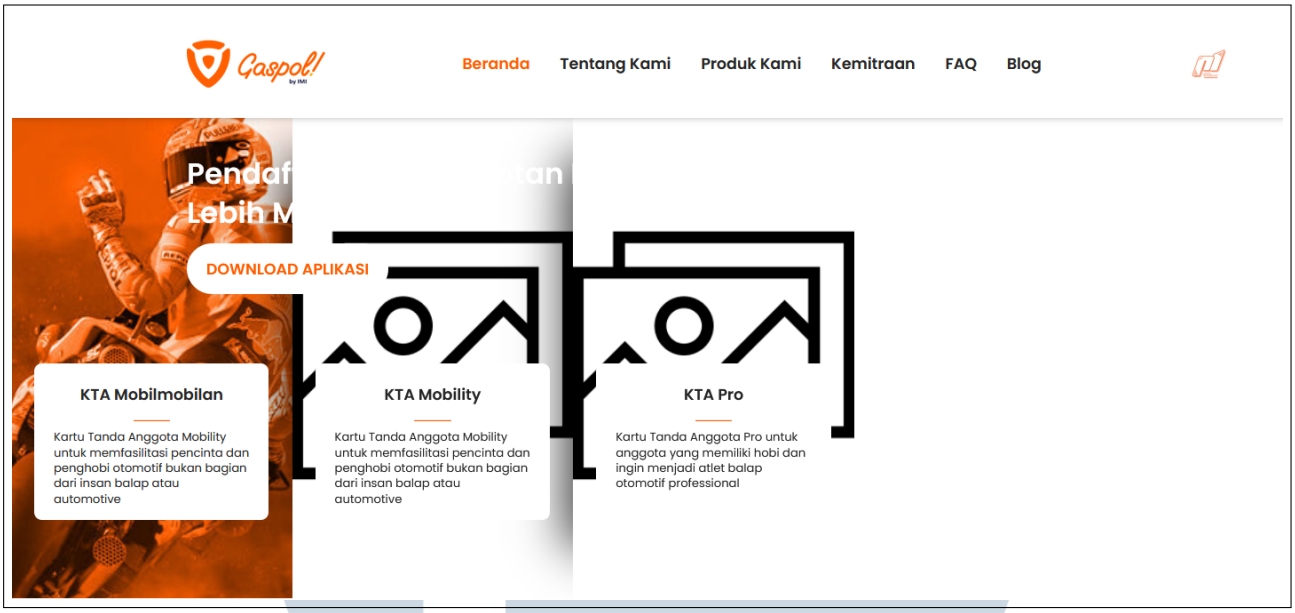
Potongan Kode *Function Index* dapat dilihat dari gambar 3.33 dan gambar, kemudian gambar *table* fitur dan perbandingannya di *website* dapat dilihat dari gambar 3.34 dan gambar 3.35

```php
class MemberController extends Controller
{
    public function index($request)
    {

        $queryMembers = DB::table([redacted])
        ->where('active','=','t')
        ->where('id','=', $request)
        ->orderBy('id', 'ASC')
        ->get()
        ->toArray();

        $members = [];
        foreach($queryMembers as $member){
          $members[] = [
            'id' => $member->id,
            'heading' => $member->name,
            'body' => $member->description,
            'imgName' => isset($member->file_banner) ? Storage::disk('s3')->temporaryUrl($member->file_banner, now()->addDay()) : '/images/empty-img.png',
            'syarat' => $member->syarat_ketentuan,
            'manfaat' => $member->manfaat,
          ];

        }

        return view('member/index',
            [
                'members'=>$members,
            ]);
    }
}
```

Gambar 3.33. Potongan Kode *Index Member*



Gambar 3.34. Tampilan di CMS Fitur



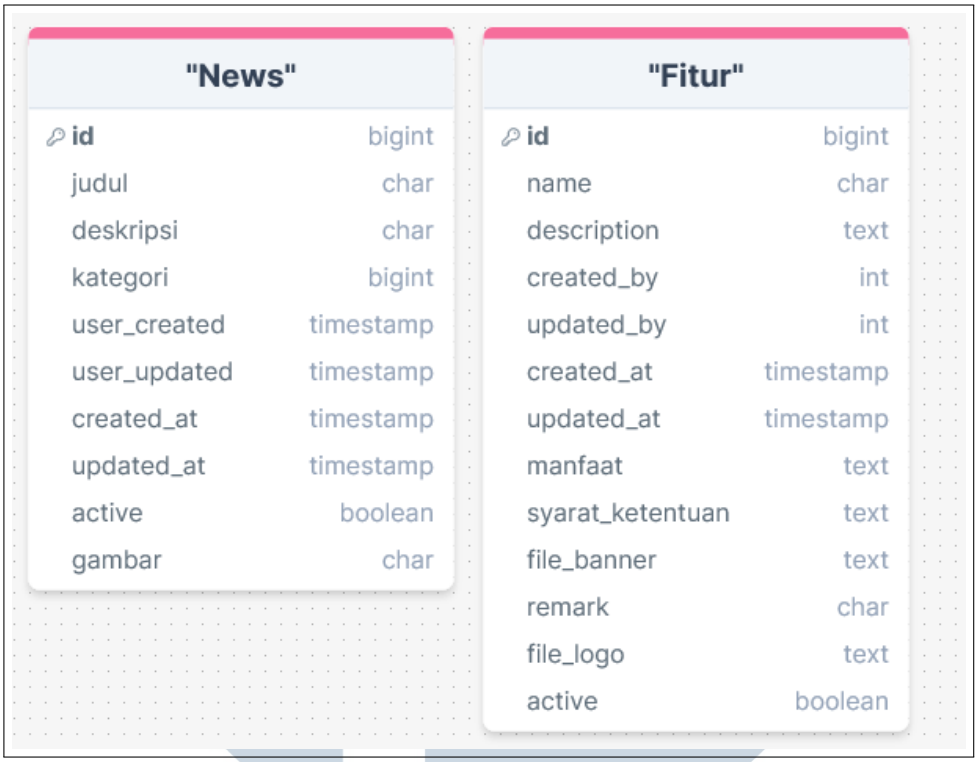
Gambar 3.35. Tampilan di *Page Member*

3.3.11 *Database*

Penulis menggunakan *database* PostgreSQL untuk proyek yang penulis kerjakan. Berikut adalah *Database Schema* yang penulis gunakan untuk proyek modul CMS dan *website*.

- **Fitur**
  Tabel ini menyimpan informasi tentang fitur-fitur keanggotaan yang tersedia
- **News**
  Tabel ini menyimpan informasi mengenai berita yang akan ditunjukkan di *blog website*

Database Schema dapat dilihat dari gambar 3.36



Gambar 3.36. Gambar *Database Schema*

3.4 Kendala dan Solusi yang Ditemukan

Selama proses pengerjaan Modul CMS, ada beberapa kendala dalam mengerjakan proyek ini. Berikut adalah kendala yang didapatkan yaitu:

- Kurangnya pengetahuan tentang CMS dan unsur-unsur didalamnya
- Keterbatasan pemahaman terhadap cara kerja CRM
- Kurangnya komunikasi dan koordinasi karena penulis mengerjakan proyek WFH
- Pengalaman mengatasi kendala teknis yang tidak terduga dalam pengembangan modul CMS
- Terjadi perpindahan *database* yang mengakibatkan pengerjaan proyek menjadi terhambat

Dengan kendala-kendala ini, penulis menemukan solusi untuk kendala tersebut yaitu:

- Membaca dokumentasi mengenai CMS untuk meningkatkan pemahaman terhadap CMS tersebut
- Mempelajari proyek atau studi kasus yang sudah terjadi dan membutuhkan CRM sebagai solusinya
- Terdapat alat komunikasi yang efektif dan dilakukan secara berkala seperti group WhatsApp, Zoom Meeting, dan Google Meet
- Meningkatkan kemampuan dalam pemecahan masalah dan komunikasi dengan rekan kerja agar dapat menyelesaikan kendala teknis yang dimiliki
- Mempersiapkan *pekerjaan* dengan matang agar tidak terhambat ketika terjadi perpindahan *database* selanjutnya