BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pada pelaksanaan kerja magang dibagian Pengembangan *e-Government*, tidak ada struktur pasti dari Diskominfo sendiri, namun, secara sistem kerja, terdapat beberapa personel yang membantu kerja magang Pengembangan *e-Government* yang digambarkan sebagai berikut :

Gambar 3.1 Struktur Sistem Kerja Magang Divisi Pengembangan e-Government

Berikut masing-masing peran personel dalam gambar 3.1:

1. ***Supervisor***

   Supervisor memiliki peranan tertinggi dan penting dalam pengelolaan proyek dan pengembangan keterampilan peserta magang. Tugas utama supervisor adalah memberikan proyek yang harus dikerjakan oleh intern, serta mengecek hasil kerja secara berkala untuk memastikan bahwa semua tugas dilakukan sesuai standar yang ditentukan. Selain itu, supervisor juga

berfungsi sebagai pendukung yang siap membantu intern jika mengalami kesulitan dalam menyelesaikan tugas. Dalam konteks konsultasi, supervisor sering kali menjawab pertanyaan dengan pendekatan yang sederhana namun efektif, seperti, "Kalau datanya seperti ini, baiknya bagaimana?" Hal ini membantu intern untuk berpikir kritis dan kreatif. Selain itu, supervisor bertanggung jawab untuk menginput evaluation grade 1 dan evaluation grade 2, serta memberikan approval untuk seluruh daily task yang telah diselesaikan. Supervisor tidak hanya berfungsi sebagai pengawas, tetapi juga sebagai mentor yang berkontribusi terhadap pengembangan kemampuan intern.

2. Asisten *Supervisor*

Asisten supervisor berperan sebagai jembatan antara intern dan supervisor, terkesan lebih fleksibilitas dalam konsultasi dan bantuan. Meskipun tidak memberikan penilaian seperti evaluation grade 1 dan evaluation grade 2, asisten supervisor sangat membantu intern dalam menyelesaikan tugas-tugas mereka. Ketika supervisor tidak dapat hadir, asisten supervisor mengambil alih peran tersebut, memberikan bimbingan dan dukungan yang diperlukan untuk memastikan intern dapat melanjutkan pekerjaannya tanpa hambatan. Pendekatan yang lebih santai dan terbuka dari asisten supervisor memungkinkan intern untuk merasa lebih nyaman dalam mengajukan pertanyaan dan meminta bantuan. Asisten supervisor berkontribusi pada lingkungan kerja yang kolaboratif dan mendukung, yang sangat penting bagi perkembangan intern.

3. *Intern* / Pemagang

Sebagai intern atau pemagang, individu bertanggung jawab untuk melaksanakan pekerjaan sesuai dengan instruksi yang diberikan oleh supervisor maupun asisten supervisor. Pemagang diharapkan mematuhi regulasi yang telah disepakati selama diskusi awal, yang mencakup etika kerja dan standar kualitas. Intern diharuskan untuk mengisi daily task yang

nantinya akan di-review dan di-approve oleh supervisor. Melalui proses ini, intern belajar untuk mengelola waktu dan tanggung jawab mereka dengan baik, serta memahami pentingnya komunikasi yang efektif dalam lingkungan kerja. Selain itu, pengalaman ini memberikan kesempatan bagi intern untuk mengembangkan keterampilan praktis yang relevan dengan bidang studi mereka, sekaligus membangun jaringan profesional yang berharga. Peran intern bisa dikatakan sangat krusial dalam mencapai tujuan proyek serta mendukung kelancaran operasional tim.

3.2 Tugas dan Uraian Kerja Magang

Pada bagian tugas dan uraian kerja magang akan dijelaskan tahapan serta penjelasan dari masing-masing tahapan pengerjaan *project* magang secara detail. *Project* yang diberikan berjudul "Identifikasi dan Evaluasi Respon Pengaduan Masyarakat Menggunakan Algoritma Random Forest dan K-Means Clustering" dengan tujuan penelitian yang menjadi acuan pengerjaan project selama kegiatan kerja magang. *Project* ini akan berkaitan langsung dengan pengembangan sistem di bidang *e-Government* pada Dinas Komunikasi dan Informatika Kota Tangerang yang mana membantu meningkatkan kualitas layanan masyarakat. Selama kegiatan pengerjaan *project*, pemagang selalu didampingi oleh *supervisor* beserta asisten *supervisor* yang sedia membantu, menjawab pertanyaan ataupun konsultasi. Berikut tabel yang menjelaskan aktivitas kerja magang maupun tahapan pengerjaan

Tabel 3.1 Realisasi Kegiatan Kerja Magang

No.	Aktivitas	Waktu Pelaksanaan	Koordinasi
Februari			
1.	Mempelajari bahasa CodeIgniter 3 dengan langsung mengimplementasikannya pada pembuatan website yang mengimplementasikan CRUD, validasi, registrasi, *login*, dan *logout* pada page-page dalam *website.*	Minggu-1 s.d Minggu-2	*Supervisor*
2.	Mempelajari *JavaScript* dengan memasang dan menyambungkan *template bootstrap* dan *template dashboard* yang sudah di modifikasi dengan *login* dan *logout.*	Minggu-3 s.d Minggu 4	*Supervisor*

No.	Aktivitas	Waktu Pelaksanaan	Koordinasi
3.	Memodifikasi *dashboard* dan menambahkan tombol yang akan membawa ke halaman *datatable*	Minggu-3	*Supervisor*
4.	Menambahkan CSRF token, dan fitur detail dengan modal	Minggu-4	*Supervisor*
Maret			
5.	Penerimaan project evaluasi respon pengaduan masyarakat yang akan dikerjakan berupa dataset seputar data pengaduan dan materi project seperti tujuan penelitian.	Minggu-1	*Supervisor*
6.	Melakukan data preparation (drop, info, pengubahan tipe data, statistic tanggal, describe, shape, columns) dan data preprocessing (hitung NaN, NaT, dan None)	Minggu-1	*Supervisor*
7.	Melakukan data *preprocessing* kedua (*Natural Language Processing*)	Minggu-1	*Supervisor*
8.	Melakukan *distribution visualization* menyesuaikan masing-masing tujuan penelitian	Minggu-2	*Supervisor*
9.	Melakukan *modeling* pertama dengan *K-means*	Minggu-2	*Supervisor*
10.	Melakukan *modeling* kedua dengan *Random Forest*	Minggu-2	*Supervisor*
11.	Menjadikan hasil-hasil *modeling* menjadi API	Minggu-3 s.d Minggu 4	*Supervisor*
April			
12.	Mengerjakan *website dashboard* sesuai ketentuan yang diberikan	Minggu 1 s.d Minggu 4	*Supervisor*
Mei			
13.	Melakukan pengujian aplikasi *website* oleh *supervisor*	Minggu 1 s.d Minggu 2	*Supervisor*
Juni			
14.	Menyusun laporan magang yang berisikan informasi kegiatan kerja magang dan pengerjaan proyek.	Minggu 3 Mei s.d Minggu 4 Juni	*Supervisor*
Februari - Juni			
15.	Menunjukkan kedisiplinan dengan hadir tepat waktu di kantor sesuai dengan jadwal yang di sepakati	Minggu 1 Februari s.d Minggu 4 Juni	*Supervisor*

3.2.1. Mempelajari bahasa CodeIgniter 3 dengan langsung mengimplementasikannya pada pembuatan website yang mengimplementasikan CRUD, validasi, registrasi, *login*, dan *logout* pada page-page dalam *website.*

Pada saat memulai magang pada satu bulan pertama, pemagang akan mempelajarari *framework CodeIgnitor*3. Hasil pembelajaran akan langsung di implementasikan dalam pembuatan website. Penugasan *website* ini merupakan

website sederhana yang mampu melakukan *Create, Update, Read, Delete* (CRUD). Berikut merupakan Langkah-langkah pembuatan website sederhana.

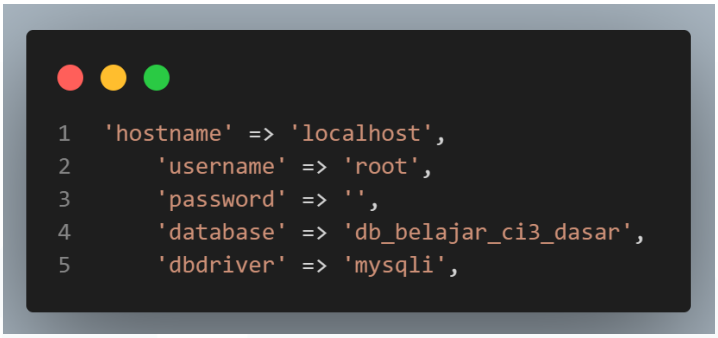
```
1  $autoload['libraries'] = array('database', 'session', 'form_validation');
```

Gambar 3.2 Autoload.php

Gambar 3.2 menunjukkan *autoload.php* yang berfungsi memuat *resource* otomatis saat aplikasi web dijalankan. *Code* diatas berfungsi memuat *library-library* yang ingin digunakan. Contoh *library* yang dimaksud seperti fitur database yang bisa langsung digunakan tanpa harus *load* ulang, fitur *session* untuk menyimpan data *login,* dan *form_valid* untuk memvalidasi input dari input dari form secara otomatis. Untuk mempermudah pengurusan *database,* dilakukan penginstalan Laragon.

Gambar 3.3 Logo Laragon

Laragon merupakan *local development environment* yang menyediakan *server database* seperti MySQL yang digunakan selama kegiatan kerja magang. Laragon dinilai cocok dengan pengembangan dengan *framework CodeIgnitor* 3. Sehingga bisa digunakan untuk membantu pembuatan aplikasi *web* yang berbasis *database* khususnya *website CRUD* yang sedang dikerjakan. Ketika Laragon berhasil tersambung dengan baik, kita bisa melanjutkan tahap pembuatan website sebagai berikut.



Gambar 3.4 database.php

Gambar 3.4 menunjukkan bagian *database.php* yang berfungsi untuk mengatur koneksi *database* yang akan digunakan. pada *line* 1 , digunakan alamat *server localhost* yang cocok untuk *server* local seperti Laragon. Selanjutnya pada *line* 2, *root* akan menjadi *username default* untuk Laragon. Pada *line* 3 yakni *password* kosong yang menjadi *password default* untuk Laragon. Dan *line 4* yakni db_belajar_ci3_dasar yang menjadi nama *database* yang akan digunakan.

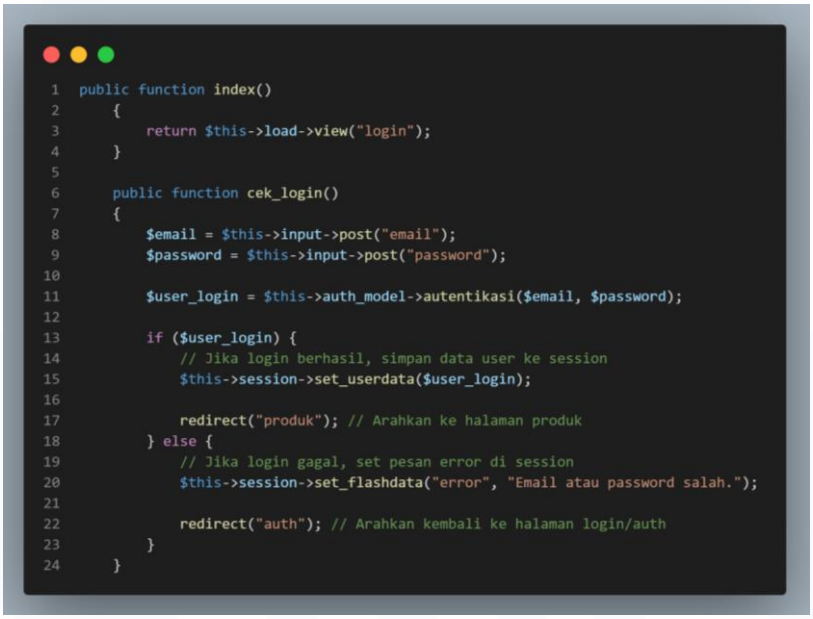
Gambar 3.5 Config.php

Gambar 3.4 menentukan URL utama aplikasi web yang dibuat. URL yang utama aplikasi menjadi URL yang digunakan ketika ingin mengakses *website.* Namun, perlu diperhatikan penamaan *file* apakah sudah sama dengan yang ada di URL utama. Hal ini dikarenakan URL harus disesuaikan dengan nama folder proyek yang dikerjakan di Laragon, khususnya www.

Gambar 3.6 routes.php

Routes biasanya digunakan untuk mendefinisikan rute atau *flow* aplikasi web. Rute yang didefinisikan akan menghubungkan URL yang diakses dengan fungsi atau metode dalam *controller*, dan memanggil rute yang didefinisikan ketika aplikasi diakses termasuk akses yang dibatasi. Seperti di *line* 1 yang mengarahkan *user* ke halaman auth atau *login* secara otomatis. Hal ini juga berlaku untuk semua rute atau *flow website.*

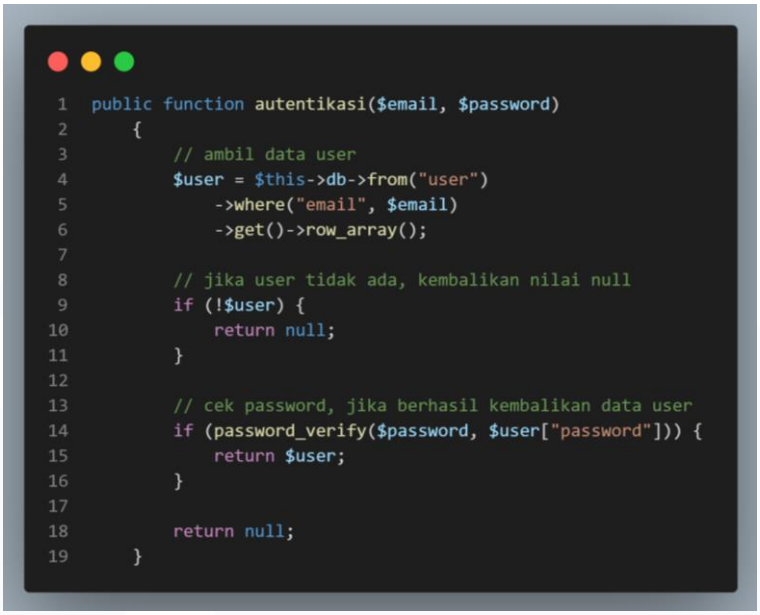
Diawal pembelajaran, *supervisor* menekankan bahwa *framework CodeIgnitor* 3 membangun aplikasi web dengan struktur yang rapih dan efisien. Definisi struktur yang rapih dan efisien ini didukung dengan 3 komponen utama *CodeIgnitor 3* yang dikenal dengan MVC atau *Model-View-Controller*. *Controller* akan menjadi penghubung antara *Model* dan *View*. ***Controller*** akan menerima *input,* proses *input* atau data dengan ***Model*** dan mengarahkan data ke ***View*** untuk ditampilkan.



```
1  public function index()
2      {
3          return $this->load->view("login");
4      }
5
6      public function cek_login()
7      {
8          $email = $this->input->post("email");
9          $password = $this->input->post("password");
10
11         $user_login = $this->auth_model->autentikasi($email, $password);
12
13         if ($user_login) {
14             // Jika login berhasil, simpan data user ke session
15             $this->session->set_userdata($user_login);
16
17             redirect("produk"); // Arahkan ke halaman produk
18         } else {
19             // Jika login gagal, set pesan error di session
20             $this->session->set_flashdata("error", "Email atau password salah.");
21
22             redirect("auth"); // Arahkan kembali ke halaman login/auth
23         }
24     }
```

Gambar 3.7 Fungsi index dan login di controller/Auth.php

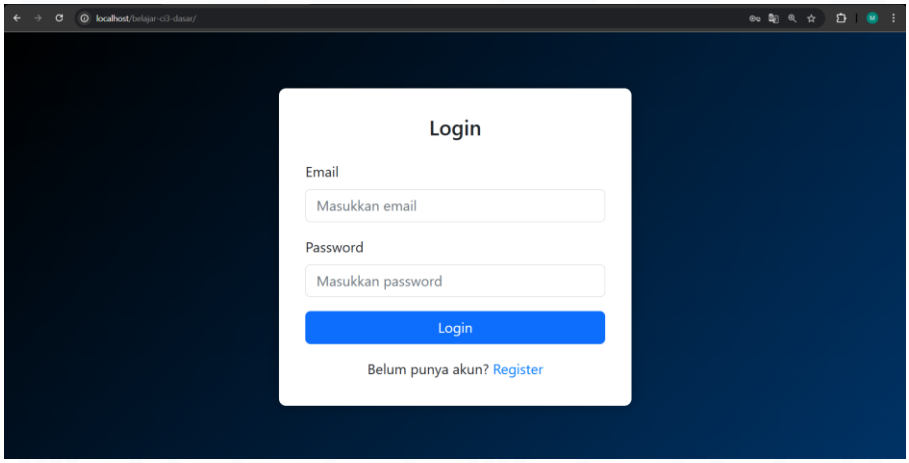
Fungsi index merupakan rute *default*. Fungsi index pada gambar 3.5 akan menampilkan halaman *login* ketika URL diakses. Setelah diarahkan ke halaman login, user akan menginput *email* dan *password* untuk *login.* Fungsi cek login diatas akan memverifikasi *email* dan *password* dengan metode autentikasi. Jika autentikasi berhasil, data *user* akan disimpan dalam *session*.



```php
public function autentikasi($email, $password)
    {
        // ambil data user
        $user = $this->db->from("user")
            ->where("email", $email)
            ->get()->row_array();

        // jika user tidak ada, kembalikan nilai null
        if (!$user) {
            return null;
        }

        // cek password, jika berhasil kembalikan data user
        if (password_verify($password, $user["password"])) {
            return $user;
        }

        return null;
    }
```

Gambar 3.8 Model/user_model.php

User_model menghubungkan aplikasi *website* dengan *database*. Proses autentikasi pada saat login akan terjadi di *user_model* yakni *query* akan mencari *user* berdasarkan *email* yang di *input*, jika *user* teridentifikasi, model akan memverifikasi *password* dengan *password* yang tersimpan di *database.* Sebaliknya jika *email* atau *password* tidak teridentifikasi, akan muncul *pop up* bahwa *email atau password* salah. Jika *password* cocok, model akan mengembalikan data *user.*



Gambar 3.9 Halaman login

Gambar 3.9 menunjukkan tampilan halaman login. Dimana *user* akan diminta menginput *email* dan *password* untuk *login*. Namun jika *user* masih belum memiliki akun, *user* bisa memilih menu *register.* Menu register memberikan kesempatan bagi *user* baru untuk membuat akun. Untuk menu dan halaman *register* pun memiliki langkah yang sama sebagai berikut.

```php
public function register()
{
    $this->load->view('register');
}

public function process_register()
{
    $nama     = $this->input->post('nama');
    $email    = $this->input->post('email');
    $password = $this->input->post('password');

    // Cek apakah email sudah terdaftar
    if ($this->auth_model->cek_email($email)) {
        $this->session->set_flashdata('error', 'Email sudah terdaftar, silakan gunakan email lain.');
        redirect('auth/register');
    } else {
        // Jika belum terdaftar, lakukan insert ke database
        $data = [
            'nama'     => $nama,
            'email'    => $email,
            'password' => password_hash($password, PASSWORD_DEFAULT) // Hash Password
        ];

        $this->auth_model->register($data);

        $this->session->set_flashdata('success', 'Pendaftaran berhasil! Silakan login.');
        redirect('auth');
    }
}
```

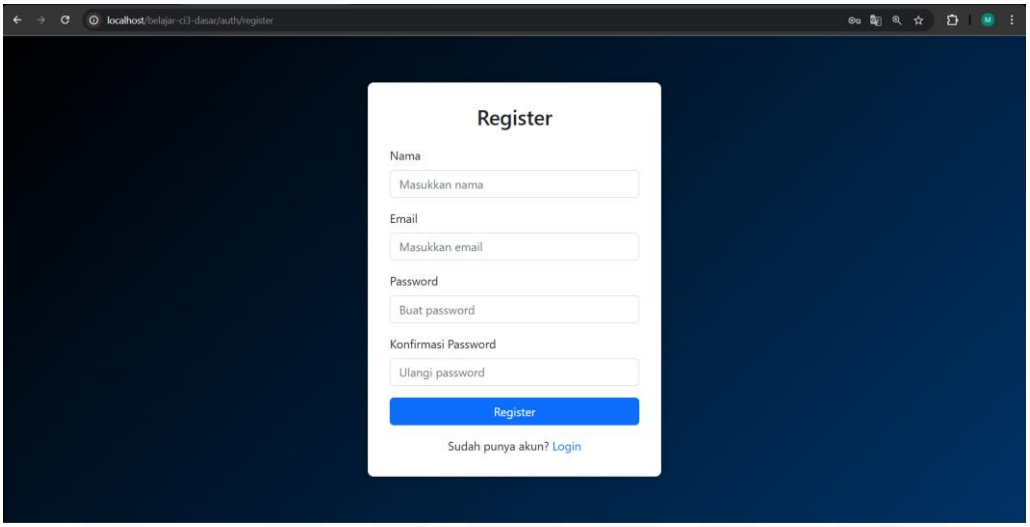
Gambar 3.10 controller/Auth.php

Fungsi *process_register* akan membantu *user* dalam proses registrasi. Fungsi ini akan meminta data seperti nama, *email,* dan *password* sebagai *input.* Lalu, fungsi *auth_model* akan memeriksa apakah *email* sudah terdaftar. Jika *email* belum terdaftar, akan dilakukan *hash-password* lalu menyimpan data *user*. Jika proses registrasi berhasil, *user* akan diarahkan kembali ke halaman *login.*

```php
public function register($data)
{
    $this->db->insert('user', $data);
}
```

Gambar 3.11 model/user_model.php

Code model pada gambar 3.11 akan membantu proses registrasi yang menerima parameter $data yang mewakili nama, *email,* dan *password*. Pada line 3 digunakan objek atau *library database* atau db. Hal tersebut dilakukan untuk memanggil fungsi *insert* yang akan membawa data ke tabel *user* dalam *database.* Jika auth dan model terintegrasi dengan baik, maka halaman register akan muncul sebagai berikut.



Gambar 3.12 Halaman register

Gambar 3.12 merupakan hasil *controller, model,* dan *views* untuk halaman registrasi. *User* bisa menginput nama, *email, password*, dan *password* kedua untuk konfirmasi. Jika *user* sudah berhasil melakukan proses registrasi, user akan dikembalikan ke halaman *login* untuk melakukan *login* terlebih dahulu sebelum memasuki *main page website.* Disediakan juga menu *login* jika *user* sudah memiliki akun.

Saat *user* berhasil registrasi dan *login, user* akan memasuki *main page* atau halaman utama. *Main page* merupakan halaman utama yang menyediakan daftar produk yang terdaftar di *database.* Terdapat beberapa kolom dalam tabel daftar produk. Beberapa kolom tersebut terdiri dari nama, harga produk, jumlah stok produk, serta kolom yang menyediakan menu aksi untuk mengedit produk dan menghapus produk. Namun untuk Langkah pertama, akan dibuat halaman utama terlebih dahulu sebagai berikut.

```
1  public function __construct()
2      {
3          parent::__construct();
4          $this->load->model("produk_model");
5      }
6
7      public function index()
8      {
9          $list_produk = $this->produk_model->getAll();
10         $this->load->view("produk", ["list_produk" => $list_produk]);
11     }
```

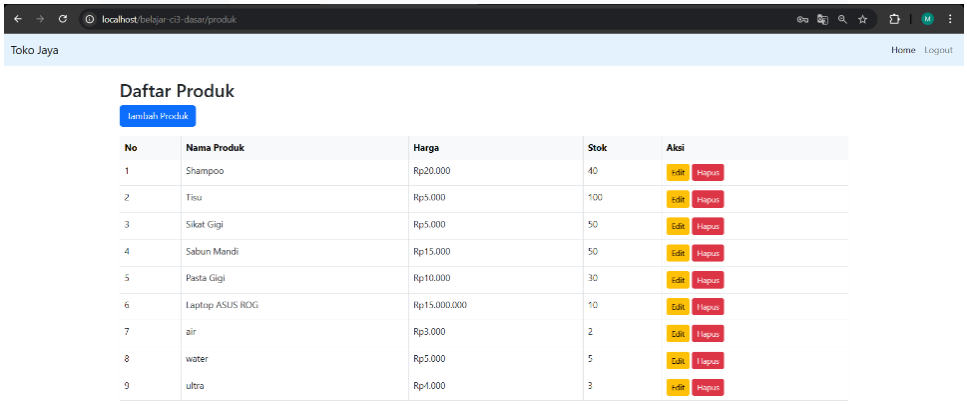
Gambar 3.13 Controller/Produk.php

Bagian *construct* pada *controller* gambar 3.13 akan menangani bagian pengambilan dan tampilan data produk pada halaman utama. Konstruktor pada *line* 1 akan memanggil konstruktor dari kelas induk dan memuat produk model dari kelas model sehingga bisa berinteraksi dengan data produk. Fungsi index yang dimulai dari *line* 7 akan menggunakan metode *getAll* di *line* 9 dari kelas model untuk menampilkan daftar semua produk. Kemudian, data produk yang diperoleh akan ditampilkan dengan *view* produk dan meneruskan variabel list_produk yang berisikan data produk ke tampilan halaman utama.

```
1  public function getAll()
2      {
3          $data = $this->db->from("produk")->get()->result_array();
4
5          return $data;
6      }
```

Gambar 3.14 model/Produk_model.php

Fungsi *form* dan *get,* metode *getAll* pada kelas model akan mengambil data produk dari *database*. Metode ini akan menggunakan objek *database* $this->db pada *line* 3 untuk melakukan query dengan memanggil fungsi *form* untuk menentukan tabel produk. Selanjutnya hasil *query* akan diambil dalam bentuk array dengan *result_array* di *line 3.* sebagai output, akan ditampilkan data produk.



Gambar 3.15 Tampilan Main page website

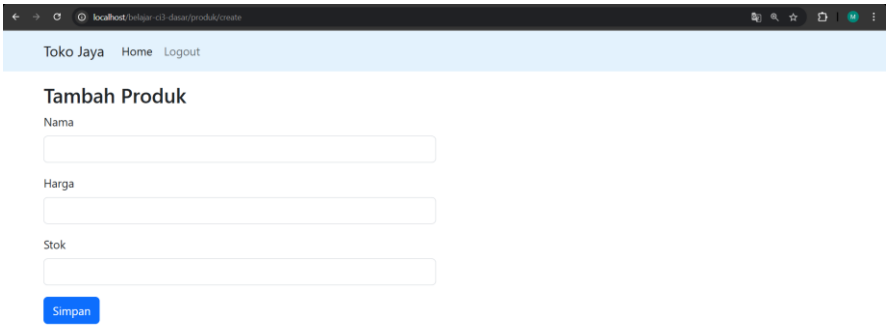
Gambar 3.15 merupakan hasil dari *controller, model,* dan *view* yang terkoordinasi dengan baik. *Main page* mencerminkan aspek kedua dari CRUD yakni *Read*. Gambar 3.15 menampilkan tabel daftar produk. Tabel tersebut terdiri dari atas beberapa kolom yang menampilkan nama toko, harga produk, jumlah stok dan kolom khusus menu aksi untuk edit dan hapus produk.

Gambar 3.16 Controller/Produk.php

Setelah memasuki *main page*, *user* bisa melakukan salah satu elemen CRUD yakni *create* atau tambah produk. *Code* pada gambar 3.16 berfungsi menampilkan halaman tambah produk. Metode *create* dengan *code* di *line 3* akan memuat tampilan bernama tambah_produk. Nantinya, menu tambah produk ini memiliki tombolnya sendiri. Yang ketika diketik akan mengarah ke halaman khusus tambah produk.

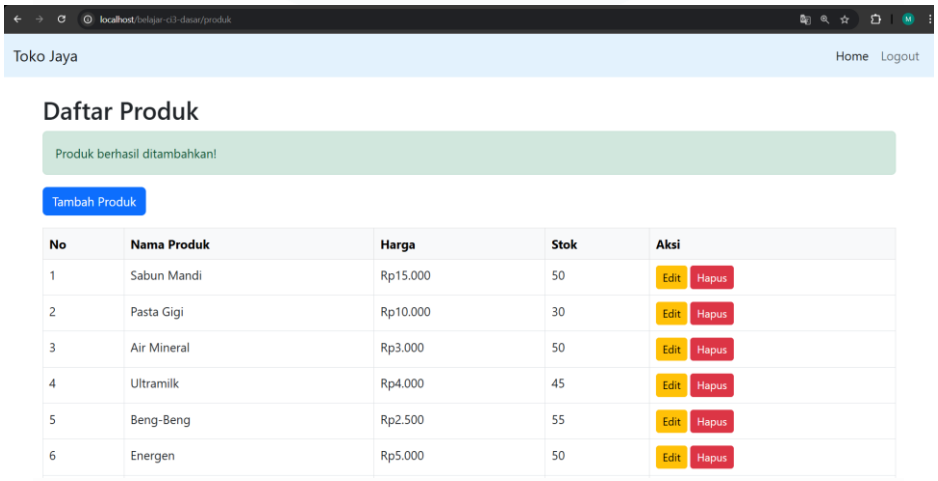
Gambar 3.17 model/Produk_model.php

Kelas model yang mencakup *code* pada gambar 3.17 berfungsi untuk menambahkan produk baru ke database. *Code* pada *line* 2 akan memanggil fungsi *insert* yang menambahkan atau menyisipkan data produk baru ke tabel produk. Metode tersebut akan mengembalikan nilai *Boolean* yang menunjukkan berhasil atau tidaknya operasi. Model berfungsi untuk membantu mengelola data produk berinteraksi dengan *database.*



Gambar 3.18 Tampilan tambah produk

Setelah mengisi nama produk, harga, dan stok, langkah selanjutnya merupakan memilih tombol simpan. Ketika salah satu *form* informasi tidak diisi, proses tambah produk tidak akan bisa dilanjutkan dan muncul *pop up* bahwa *form* harus diisi. Selanjutnya ketika tombol tersebut dipencet, maka akan ada *pop up* sebagai bentuk validasi apakah yakin ingin menambahkan produk. Jika memilih "ya" pada *pop up* tersebut, maka akan langsung kembali ke halaman utama dengan tabel daftar produk yang sudah mengandung informasi data produk baru.



Gambar 3.19 Tampilan Produk Berhasil di Tambahkan

Saat kembali ke halaman utama setelah proses tambah produk, akan muncul *pop up* bahwa produk berhasil ditambahkan. *Pop up* ini menjadi tanda bahwa proses tambah produk berhasil dilakukan. Hal tersebut di implementasikan agar ada keterangan aksi apa yang baru saja dilakukan. Selain menambahkan produk, *user* dapat melakukan *update* atau edit produk.

```php
public function edit($id)
    {
        $produk = $this->produk_model->get_produk_by_id($id);

        if (!$produk) {
            show_404();
        }

        $this->load->view("edit_produk", ["produk" => $produk]);
    }
```

Gambar 3. 20 controller/Produk.php

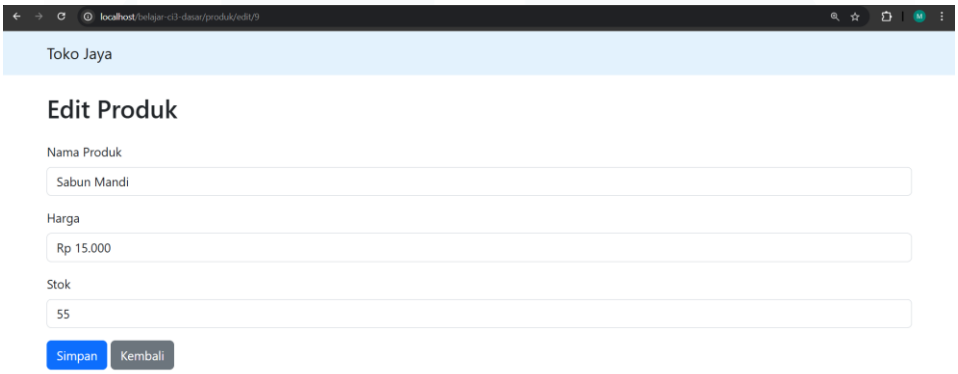
Ketika *user* ingin melakukan edit produk, pada *controller* akan dibuat fungsi yang akan membawa *user* ke halaman edit produk. Proses edit atau *update* produk menerima ID atau $id yang digunakan untuk mengambil data produk. Sehingga saat memasuki halaman edit produk, ditampilkan data-data produk yang tersimpan di *database* dan siap di edit. Data yang ditampilkan merupakan sebuah *output* yang diterima.

```php
public function update_produk($id, $data) {
        $this->db->where('id', $id);
        return $this->db->update('produk', $data);
    }

    public function edit_produk($id)
    {
    return $this->db->get_where("produk", ["id" => $id])->row_array();
    }
```

Gambar 3.21 model/Produk_model.php

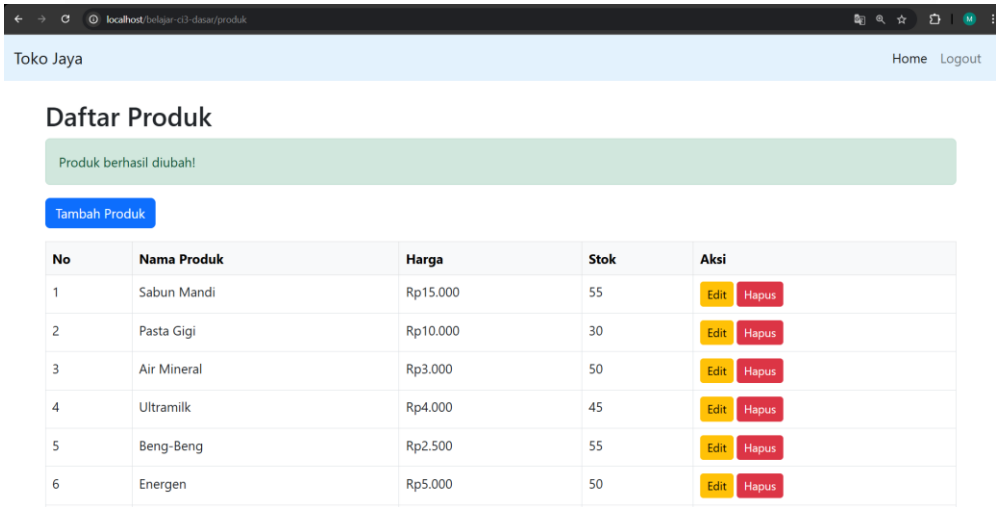
Pada gambar 3.21 terdapat 2 metode dalam membantu proses edit produk. *Function* pertama pada *line 1* akan memperbaharui informasi produk yang sudah tersimpan di *database*. *Code* pada *line* 2 akan menetapkan kondisi untuk memilih produk berdasarkan ID. Selanjutnya metode *update* pada *line* 3 akan digunakan untuk memperbarui data dalam tabel produk dengan data baru yang diterima

sebagai parameter $data. *Function* kedua pada *line 6* akan mengambil data produk berdasarkan ID. *Code* dari *line* 8 digunakan untuk mengambil informasi dan mengembalikannya sebagai array baris tunggal. Kedua metode tersebut bekerja sama untuk melancarkan proses edit atau *update* produk.



Gambar 3.22 Halaman Edit atau Update Produk

Berikut tampilan halaman edit produk. Pada gambar 3.22 terlihat *form* sudah terisi akan data-data yang tersimpan di *database.* Ketika data sudah diubah, menu simpan akan dipencet. Selanjutnya, akan muncul *pop up* untuk validasi apakah yakin ingin mengubah data produk.



Gambar 3.23 Tampilan Produk Berhasil di Edit

Jika "ya" dipilih pada *pop up* validasi yang muncul, *user* akan kembali diarahkan ke halaman utama. Namun, terdapat *pop up* atau notifikasi yang

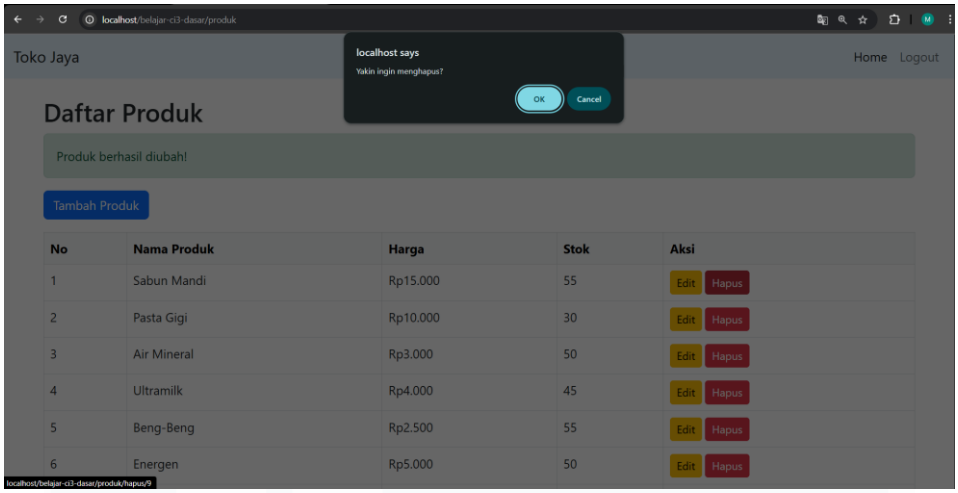
memberitahukan bahwa data produk berhasil diubah. Fungsi dari *pop up* tersebut merupakan tanda bahwa pembaharuan produk berhasil dilakukan. Fungsi lain dari *pop up* tersebut merupakan memberitahukan *user* kegiatan atau aksi apa yang baru saja terjadi.

Gambar 3.24 controller/Produk.php

Pada gambar 3.24 merupakan *code* pada *controller* untuk proses hapus produk. Metode hapus akan menerima parameter $id yang merupakan ID produk yang ingin dihapus. *Code* dari *line* 3 digunakan model untuk melakukan operasi penghapusan pada *database*. Setelah proses penghapusan berhasil, *code* pada *line* 4 akan memberikan *feedback* berupa pesan bahwa operasi penghapusan berhasil dilakukan. Dan *line* 5 akan mengarahkan *user* ke halaman utama.

Gambar 3.25 model/Produk_model.php

Code pada gambar 3.25 merupakan *code* model yang berfungsi mengapus. Metode hapus pada *line* 1 akan menerima parameter $id yang merupakan ID produk yang ingin dihapus. *Code line* 2 digunakan untuk menghapus entri dari tabel produk berdasarkan ID. Jika tombol aksi hapus dipencet, akan muncul *pop up* sebagai validasi seperti berikut.

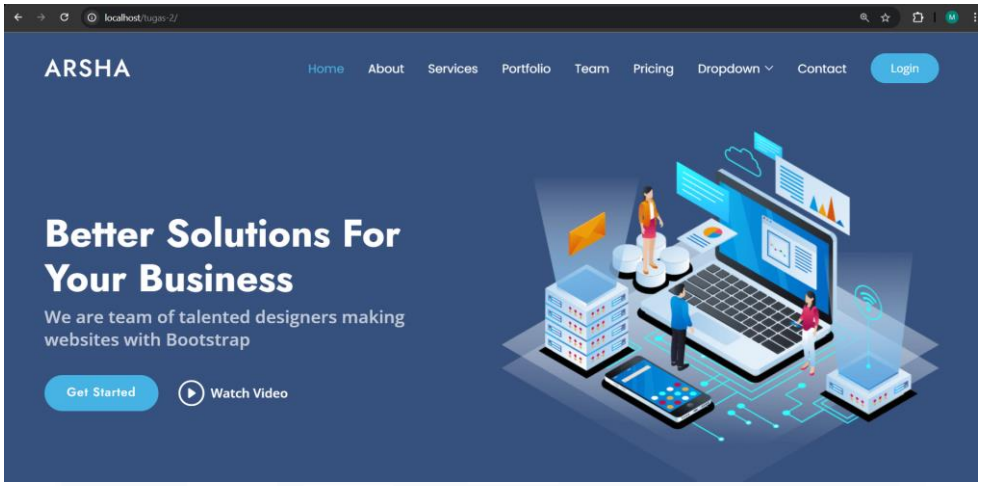


Gambar 3.26 Tampilan Saat Ingin Menghapus Produk

Jika "ya" dipilih, maka proses penghapusan akan terjadi. Setelah itu *user* akan kembali ke halaman utama dengan adanya *pop up* produk berhasil dihapus. *Pop up* ini menandakan proses penghapusan berhasil dilakukan. Selain menjadi tanda, *pop up* juga memberitahukan aksi apa yang baru saja terjadi.

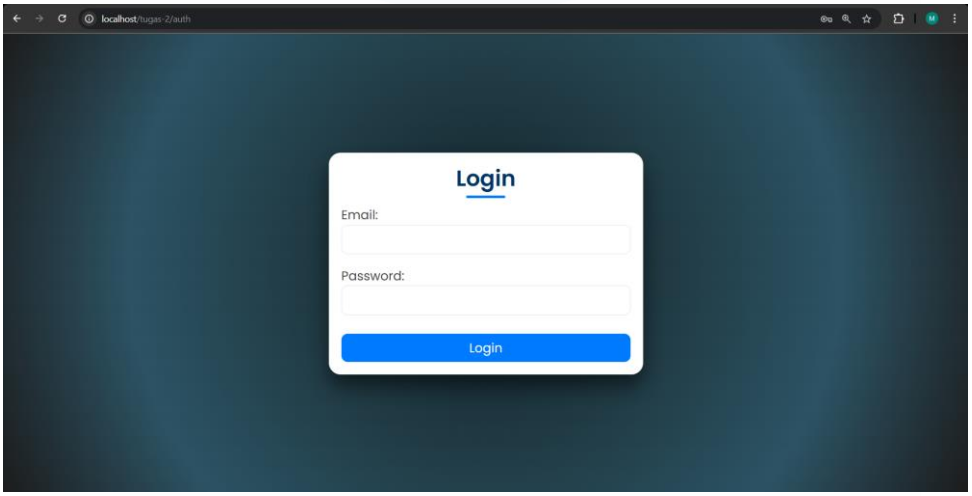
3.2.2 Mempelajari JavaScript dengan memasang dan menyambungkan *template* bootstrap dan *template* dashboard yang sudah di modifikasi dengan login dan logout.

Setelah *website* sederhana diselesaikan, diberikan tugas kedua oleh *supervisor* berupa 2 *template* berbeda yang harus di integrasikan. *Template* pertama adalah *template* profil perusahaan yang didapatkan dari sebuah *website.* Sedangkan *template* kedua merupakan *template dashboard* yang didapatkan dari sebuah *website* juga. Kedua *template* tersebut harus di integrasikan dengan adanya *login* dan *logout*.



Gambar 3.27 Tampilan template Arsha

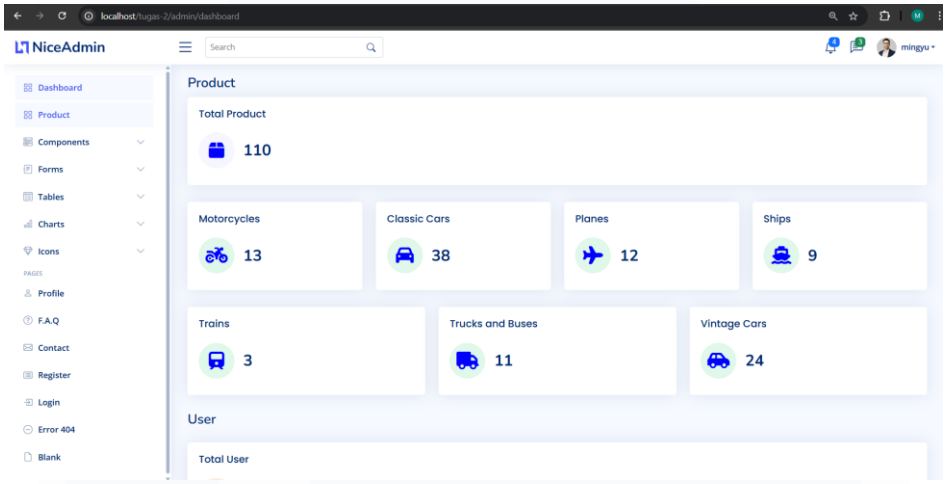
Gambar 3.27 menunjukkan tampilan *template website* Arsha yang menjadi tugas kedua. Menyesuaikan dengan perintah pengerjaan tugas ini yang hanya sebatas menyambungkan dengan adanya *login,* menu login dibuat menjadi tombol. Tampilan tombol yang berbeda dengan tombol lainnya agar menonjolkan *login*. Peletakan tombol di pojok kanan juga agar terpusat dan tidak tercampur.



Gambar 3.28 Tampilan menu login

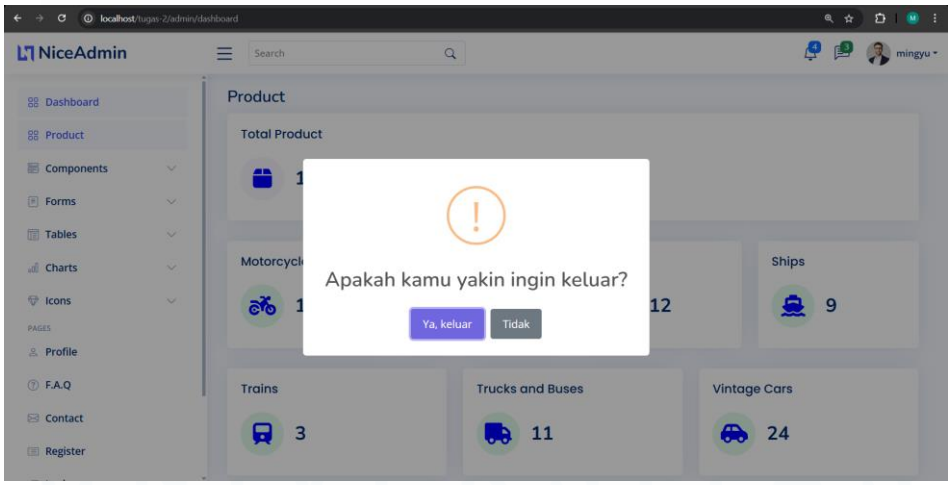
Gambar 3.28 merupakan tampilan ketika tombol *login* di pilih. Saat *login, email* dan *password* akan diminta sebagai inputan data untuk masuk ke *website.* Namun jika salah satu *form* tidak terisi, akan ada *pop up* bahwa *form* wajib diisi.

Jika salah satu *form* memiliki kesalahan, maka *user* tidak akan bisa memasuki *website*.



Gambar 3.29 Tampilan dashboard

Saat berhasil *login, user* akan memasuki *website*. *Dashboard* pada gambar 3.29 terdiri atas beberapa macam jenis informasi. Setiap informasinya terdiri atas beberapa *section card. Section card* tersebut memiliki judul informasi, jumlah data yang ingin ditampilkan serta logo untuk mendukung judul informasi.



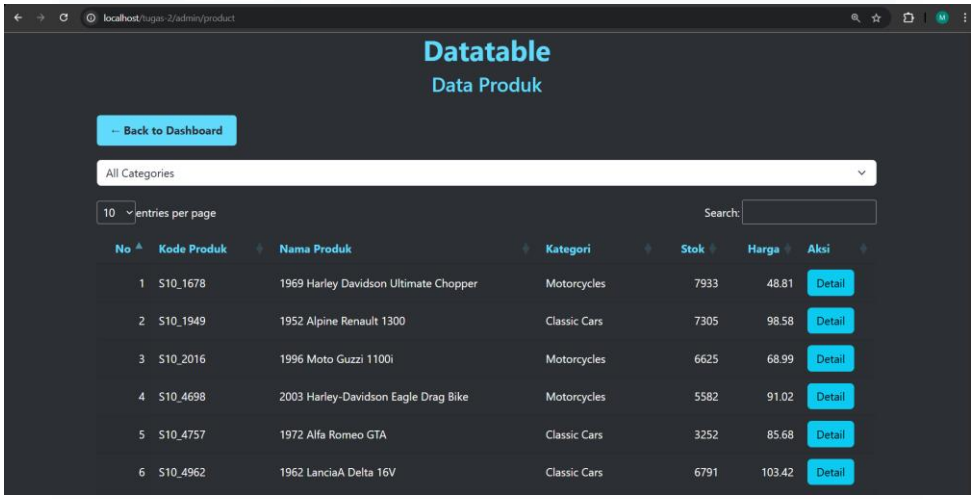
Gambar 3.30 Validasi untuk logout

Ketika *user* ingin *logout,* terdapat notif apakah *user* yakin ingin keluar. Jika *user* memilih "ya", maka *user* akan otomatis kembali ke halaman *login.* Namun jika *user* memilih "tidak", *user* akan kembali ke *dashboard*. Notif digunakan sebagai

bentuk konfirmasi atau validasi apakah aksi yang akan dilakukan *user* benar ingin dilakukan.

3.2.3 Memodifikasi *dashboard* dan menambahkan tombol yang akan membawa ke halaman *datatable*

Section card pertama pada aspek *product* yang berjudul *total product* dijadikan *tombol* yang jikalau di pilih akan membawa *user* pada halaman yang menampilkan *datatable*



Gambar 3.31 Tampilan datatable

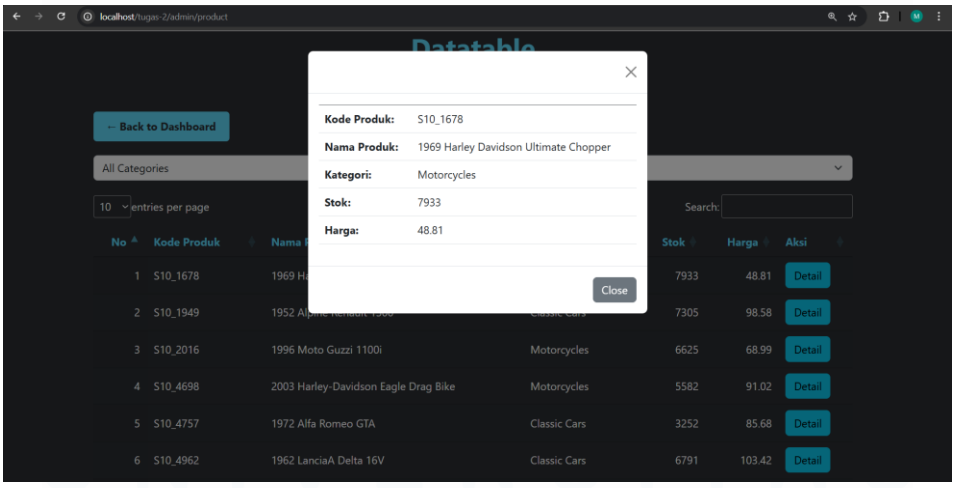
Gambar 3.31 menampilkan halaman *datatable* yang memiliki judul "*Datatable"* sub judul "Data Produk". Tombol "*back to dashboard"* yang akan membawa *user* kembali ke halaman *dashboard*. Selanjutnya terdapat opsi kategori untuk mempersempit kategori pencarian dan jumlah entri atau output yang diinginkan. *User* juga bisa langsung melakukan pencarian dengan mengetik nama kendaraan yang ingin dicari.

3.2.4 Menambahkan CSRF token, dan fitur detail dengan modal

```php
$config['csrf_protection'] = FALSE;
$config['csrf_token_name'] = 'csrf_test_name';
$config['csrf_cookie_name'] = 'csrf_cookie_name';
$config['csrf_expire'] = 7200;
$config['csrf_regenerate'] = FALSE;
$config['csrf_exclude_uris'] = array();
```

Gambar 3.32 Code config.php

Cross-Site Request Forgery atau CSRF merupakan serangan yang akan mengeksploitasi kepercayaan aplikasi *website* terhadap pengguna yang sudah terdaftar atau terautentikasi. Sedangkan CSRF *token* merupakan *unique value* yang di hasilkan *server* yang disertakan dalam setiap *request user*. Pada *line* 1, csrf *token* dibuat menjadi *false* agar perlindungan CSRF menjadi tidak aktif. Hal ini bisa diubah menjadi *true* agar perlindungan terhadap CSRF menjadi aktif.



Gambar 3.33 Hasil tombol detail berupa modal

Modal merupakan elemen *user interface* yang muncul diatas konten halaman tanpa mengalihkan *user* dari halaman yang dibuka. Jika tombol detail pada kolom

aksi dipilih, maka modal akan muncul. Modal disini berfungsi menampilkan informasi detail mengenai produk. Sehingga sangat berfungsi jika *user* ingin mengetahui informasi detail produk.

3.2.5 Penerimaan proyek evaluasi respon pengaduan masyarakat yang akan dikerjakan berupa *dataset* seputar data pengaduan dan materi proyek seperti tujuan penelitian

Setelah menyelesaikan satu bulan pertama masa belajar, di berikan proyek dari. Proyek ini diberikan langsung oleh *supervisor* dengan judul "Identifikasi dan Evaluasi Respon Pengaduan Masyarakat Menggunakan Algoritma *Random Forest* dan *K-means Clustering*". Untuk mendukung proyek penelitian yang diberikan, diberikan *dataset* yang merupakan data pengaduan yang diterima Diskominfo dalam dua tahun terakhir yakni 2024 dan 2025. Dataset diberikan dalam bentuk *link* xlsx.

Pemberian tujuan penelitian ini dilakukan untuk memastikan batasan dalam penelitian. Sehingga batasan serta apa yang ingin di teliti dalam proyek yang diberikan sudah jelas. *Deadline* yang ditetapkan oleh *supervisor* merupakan hasil diskusi yang sudah disepakati. Hasilnya, proyek penelitian yang diberikan mulai dikerjakan dan berjalan lancar selama pengerjaan.

3.2.6 Melakukan *data preparation* (*drop, info,* pengubahan tipe data, statistic tanggal, *describe*) dan *data preprocessing* (hitung NaN, NaT, dan None)

Setelah menerima *dataset* dan tujuan penelitian, dilakukan *data preparation* dan *data preprocessing*. Kedua kegiatan ini memiliki fungsinya masing-masing. *Data preparation* yang berfungsi menyaring informasi penting, sedangkan *data preprocessing* berfungsi dalam mengidentifikasi dan menangani data yang hilang. Baik *data preparation* maupun *data preprocessing* memiliki tahapan dan fungsinya masing-masing.

	id_pengaduan	jenis_pengaduan	nama_dinas	status_label	id_user	id_unor	kategori_hashtag	nama_pelapor	alamat_pelapor	kecamatan_pelapor
0	127961	2	DINAS KEPENDUDUKAN DAN PENCATATAN SIPIL	selesai	NaN	14	PENDUDUK	NaN	NaN	NaN
1	128218	1	Bidang Pelayanan Kesehatan\r	belum	1.399190e+05	110	PUSKEMAS	NaN	NaN	NaN
2	128217	1	DINAS PERHUBUNGAN	proses	8.632000e+03	12	LAMPU PJU	MUHAMMAD RIDWAN W	PURI MEGAH C2 NO.52	CIPONDOH
3	128216	1	KECAMATAN KARANG TENGAH	belum	1.697649e+09	39	PJU KEC. KARANG TENGAH	SUBUR	KP. PEDURENAN	KARANG TENGAH
			KECAMATAN				PJU KEC.		KP.	

Gambar 3.34 *Hasil* import dataset

Gambar 3.34 menunjukkan hasil *import* atau data yang diberikan oleh *supervisor*. Data memiliki terdapat beberapa kolom data kategorikal maupun numerik. Setiap barisnya mewakili data dari masing-masing ID pengaduan. Pada gambar 3.34 juga terlihat terdapat baris yang mengandung beberapa nilai NaN.

```
df = df.drop(columns=['id_pengaduan', 'jenis_pengaduan', 'id_user', 'id_unor',
                      'nama_pelapor',  'user_sosmed', 'username_sosmed', 'lat','lng'])
df
```

Gambar 3.35 *Code untuk menghapus kolom yang tidak diperlukan*

Setelah melihat kolom apa saja yang ada di *dataset*, terdapat beberapa kolom yang tidak relevan sehingga perlu dihapus. *Code* pada gambar 3.35 memperlihatkan terdapat sembilan kolom yang dihapus. Sembilan kolom yang dihapus diantaranya ID pengaduan, jenis pengaduan, ID yang mengajukan laporan pengaduan, ID unor, nama pelapor, social media yang digunakan serta nama pengguna sosmed yang mengajukan pengaduan, *latitude* dan *longtitude*. Penghapusan kolom ini dilakukan dengan tujuan penelitian hanya meneliti data yang relevan dan dibutuhkan.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17478 entries, 0 to 17477
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   nama_dinas         16695 non-null  object
 1   status_label       17478 non-null  object
 2   kategori_hashtag   17476 non-null  object
 3   alamat_pelapor     2836 non-null   object
 4   kecamatan_pelapor  2650 non-null   object
 5   kelurahan_pelapor  2533 non-null   object
 6   isi_pengaduan      17478 non-null  object
 7   lokasi             17412 non-null  object
 8   tgl_pengaduan      17478 non-null  datetime64[ns]
 9   tgl_proses         16621 non-null  object
 10  tgl_selesai        16498 non-null  object
dtypes: datetime64[ns](1), object(10)
memory usage: 1.5+ MB
```

Gambar 3.36 *Informasi* dataset

Setelah menghapus kolom-kolom yang tidak diperlukan, code info() dijalankan untuk mengetahui informasi seputar *dataset*. Dari gambar 3.36 terdapat beberapa hal yang perlu diperhatikan. Hal-hal yang perlu diperhatikan diantaranya terdapat 17.477 data dengan total 11 kolom. Namun, terdapat beberapa kolom yang datanya tidak terisi atau *null*.

```
1  # mengubah tipe data kolom lokasi menjadi string
2
3  df['lokasi'] = df['lokasi'].astype(str)
```

Gambar 3.37 *Code* untuk mengubah tipe data kolom lokasi

pada gambar 3.36 dapat terlihat bahwa tipe data kolom lokasi merupakan object. Hal ini akan menganggu jalannya analisis data sehingga kolom lokasi akan diubah menjadi format *string*. Hal ini akan memudahkan *data preprocessing*. Selain memudahkan *data preprocessing*, pengubahan data alamat juga membantu mempermudah analisis data alamat.

```
# mengubah kolom menjadi tipe data tanggal

df['tgl_pengaduan'] = pd.to_datetime(df['tgl_pengaduan'], errors='coerce')
df['tgl_proses'] = pd.to_datetime(df['tgl_proses'], errors='coerce')
df['tgl_selesai'] = pd.to_datetime(df['tgl_selesai'], errors='coerce')
```

Gambar 3.38 *Code* untuk mengubah tipe data kolom tanggalan

Selain kolom alamat, tiga kolom tanggalan juga terlihat sebagai *object* pada gambar 3.36. Sehingga dilakukan pengubahan pada tiga kolom tanggalan dan gambar 3.38 memperlihatkan *code* yang dijalankan. Tiga kolom yang diubah menjadi tipe data tanggal diantaranya tanggal pengaduan masuk, tanggal pengaduan di proses, dan tanggal pengaduan diselesaikan.

```
# statistik tanggal-tanggal

df['durasi_proses'] = (df['tgl_proses'] - df['tgl_pengaduan']).dt.days
df['durasi_selesai'] = (df['tgl_selesai'] - df['tgl_proses']).dt.days
df['total_durasi'] = (df['tgl_selesai'] - df['tgl_pengaduan']).dt.days
```

Gambar 3.39 Statistik tanggalan

Setelah tiga kolom tanggalan diubah menjadi tipe data tanggal, dilakukan penghitungan atau statistic data. Hal ini dilakukan untuk mempermudah analisis data nantinya. Terdapat tiga penghitungan yang dihasilkan statistic tanggalan pada gambar 3.39 diantaranya lama pengaduan di proses, lama pengaduan diselesaikan, dan total durasi pengaduan diproses. Tiga penghitungan statistik ini akan mempermudah dalam menganalisis waktu.

	tgl_pengaduan	tgl_proses	tgl_selesai	durasi_proses	durasi_selesai	total_durasi
count	17478	16620	16497	16620.000000	16495.000000	16497.000000
mean	2024-08-02 09:17:36.932372224	2024-08-02 14:52:20.926293760	2024-08-03 05:42:34.959022848	1.910590	0.523128	2.506941
min	2024-01-01 00:10:33	2009-01-01 09:23:00	2009-01-01 09:23:00	-5572.000000	-46.000000	-5572.000000
25%	2024-04-18 13:45:00.750000128	2024-04-20 09:10:04	2024-04-22 09:38:11	0.000000	0.000000	0.000000
50%	2024-08-01 09:29:31	2024-08-01 11:45:35	2024-08-02 09:24:37	0.000000	0.000000	0.000000
75%	2024-11-22 10:11:57.500000	2024-11-21 14:45:41	2024-11-22 08:43:01	1.000000	0.000000	2.000000
max	2025-03-06 15:12:31	2025-03-28 07:05:10	2025-03-28 07:05:10	289.000000	34.000000	289.000000
std	NaN	NaN	NaN	44.535728	1.807133	44.711583

Gambar 3.40 Hasil statistik ringkasan data numerik

Setelah mengubah tipe data, *code describe*() dijalankan untuk mengetahui kesimpulan data numerik pada *dataset*. Pada gambar 3.40 terlihat tiga kolom hasil statistic tanggalan sudah termasuk. Pada hasil statistik juga terlihat seluruh data merupakan data tanggalan. Hal ini menjadi perhatian besar dimana tanggalan akan menjadi data yang penting dan banyak di analisis.

```
# Menghitung jumlah NaT dalam kolom tgl_proses
jumlah_nat = df['tgl_proses'].isna().sum()
print("Jumlah pengaduan yang berisikan NaN di tgl_proses (tidak direspon) :", jumlah_nat)
```

```
Jumlah pengaduan yang berisikan NaN di tgl_proses (tidak direspon) : 858
```

```
# Menghitung jumlah NaT dalam kolom tgl_proses
jumlah_nat = df['tgl_selesai'].isna().sum()
print("Jumlah pengaduan yang berisikan NaT di tgl_proses (tidak diselesaikan) :", jumlah_nat)
```

```
Jumlah pengaduan yang berisikan NaT di tgl_proses (tidak diselesaikan) : 981
```

```
# Menghitung jumlah NaN
jumlah_nan = df.isna().sum().sum()
print(f'Jumlah NaN : {jumlah_nan}')
```

```
Jumlah NaN : 49861
```

Gambar 3.41 *Data Preprocessing*

Pada gambar 3.41 terdapat beberapa detail yang perlu diperhatikan. Pada tahap *data proprocessing* dilakukan tiga pengidentifikasian nilai. Dari tiga tahap tersebut dapat disimpulkan beberapa hal. Tiga informasi yang dihasilkan *data proprocessing* diantaranya terdapat 858 laporan yang tidak di proses, 981 laporan yang tidak terselesaikan, dan terdapat 49.861 nilai NaN.

3.2.7 Melakukan data preprocessing kedua (Natural Language Processing)

Setelah melakukan *data preprocessing* pertama untuk memproses data numerik, dilakukan *data preprocessing* kedua untuk data kategorikal atau non numerik. *Data Preprocessing* kedua dilakukan dengan *Natural Language Processing* (NLP). NLP memungkinkan komputer memahami, menafsirkan, dan menghasilkan bahasa yang digunakan manusia. NLP sangat penting untuk dilakukan pada penelitian ini dikarenakan data pengaduan yang banyaknya dalam bentuk kalimat dan paragraf. Hal ini membutuhkan usaha ekstra dalam memproses datanya. Sehingga dilakukan NLP untuk membantu mempermudah proses pengerjaan penelitian ini.

```
# menghapus tanda baca yang terdapat pada data

pattern = r'["\'!?\*,;<>/()]+'
```

```
# menghapus pattern yang ditemukan (string)

df['isi_pengaduan'] = df['isi_pengaduan'].str.replace(pattern,'')
df['lokasi'] = df['lokasi'].str.replace(pattern,'')
```

```
# mengubah semua huruf menjadi lowercase

df['isi_pengaduan'] = df['isi_pengaduan'].str.lower()
df['lokasi'] = df['lokasi'].str.lower()
df['nama_dinas'] = df['nama_dinas'].str.lower()
```

Gambar 3.42 *Text Standardization dan Cleaning Regex*

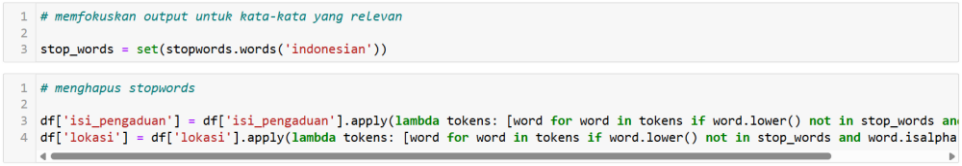
Pada gambar 3.42 dilakukan beberapa hal dalam melakukan NLP. *Code* pada *cell* pertama *line 3* dijalankan untuk menghapus karakter-karakter yang ingin dihapus dari data teks. Selanjutnya *code* pada *cell* kedua dijalankan untuk menghapus pola yang di identifikasi. Dan *code* pada *cell* ketiga akan merubah semua huruf pada kolom isi pengaduan, lokasi, dan nama dinas menjadi *lowercase*.

```
# tokenisasi teks (pengubahan baris teks menjadi daftar kata)
from nltk.tokenize import word_tokenize

df['isi_pengaduan'] = df['isi_pengaduan'].apply(lambda x: word_tokenize(x))
```

```
# tokenisasi teks (pengubahan baris teks menjadi daftar kata)

df['lokasi'] = df['lokasi'].apply(lambda x: word_tokenize(x))
```

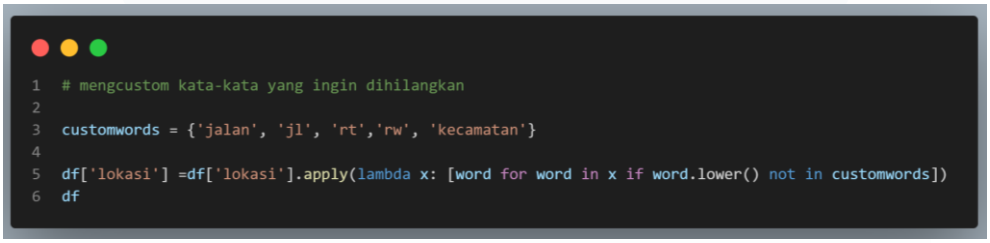
Gambar 3.43 *Word Tokenization*

Code pada gambar 3.43 merupakan *word tokenization* yang dilakukan untuk proses NLP. *Code* pada *line* 2 *cell* pertama akan mengimpor fungsi *word_tokenize.* Lalu *code* pada *line* 4 *cell* pertama dan *code* pada *line* 3 *cell* 2 akan menggunakan metode *apply*() untuk menerapkan fungsi lambda pada kolom isi pengaduan dan lokasi. Fungsi lambda berguna dalam memecah setiap teks dalam kolom menjadi daftar kata.



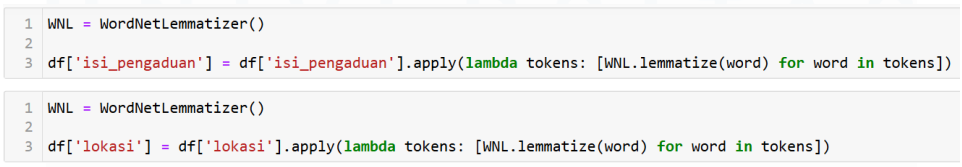
Gambar 3.44 *Stopword Removal*

Selanjutnya dilakukan *Stopword Removal* dalam proses NLP. Terdapat 2 jenis *code* yang dijalankan untuk membantu NLP. *Code* pada *line* 3 cell pertama dijalankan untuk mengambil daftar kata-kata umum dalam bahasa Indonesia yang di inisiasikan dengan *stopwords*. Digunakan *set*() untuk menyimpan daftar kata-kata atau *stopwords* dalam struktur yang efisien. Selanjutnya *code* pada *cell* kedua dijalankan untuk mempertahankan kata-kata yang tidak termasuk dalam *stopwords* pada kolom isi pengaduan dan lokasi.



Gambar 3.45 *Costumized Stopword*

Code pada gambar 3.45 merupakan contoh *stopword* yang di kustomisasi. Proses kustomisasi ini dilakukan sesuai kebutuhan jika dirasa daftar kata dalam *stopword* masih kurang efektif. Hal ini biasanya dilakukan pada kolom yang datanya dalam bentuk kalimat atau paragraf. Kustomisasi *stopword* dilakukan untuk kolom lokasi dan isi pengaduan.

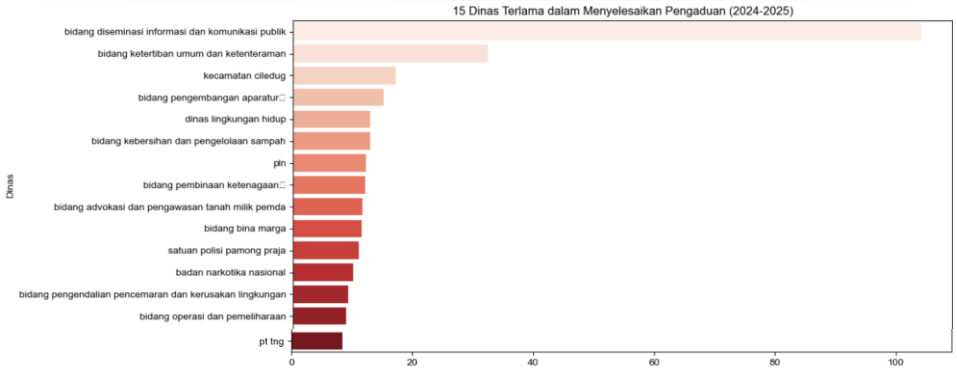


Gambar 3.46 *Word Lemmatization*

Pada tahap terakhir NLP, dilakukan *word lemmatization*. *Word Lemmatization* dilakukan untuk mengubah kata menjadi kata dasar. Hal ini membantu proses analisis data dalam cara yang lebih konsisten. Hal ini di implementasikan pada kolom isi pengaduan dan lokasi yang rentan akan kata yang dirubah.

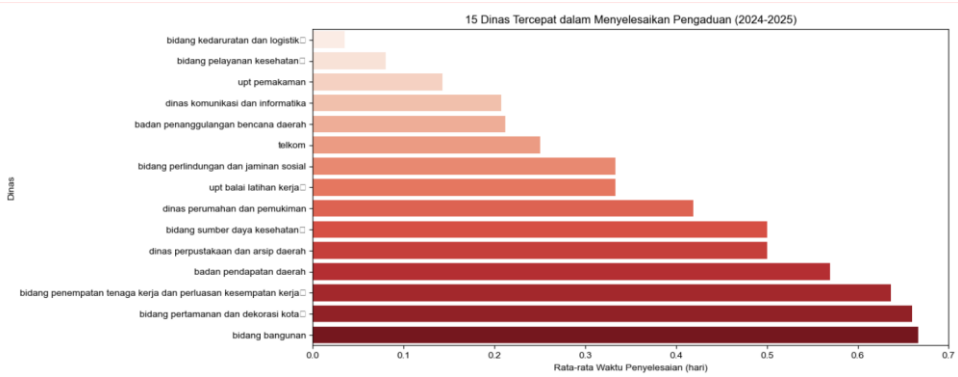
3.2.8 Melakukan *distribution visualization* menyesuaikan masing-masing tujuan penelitian

Setelah membersihkan dan mempersiapkan data, akan dilakukan visualisasi distribusi. Visualisasi distribusi merupakan visualisasi yang digunakan untuk menunjukkan nilai-nilai dalam *dataset* yang terdistribusi. Jumlah visualisasi distribusi bisa melebihi tujuan penelitian untuk menyediakan informasi yang lebih baik. Visualisasi distribusi akan menjawab masing-masing tujuan penelitian.



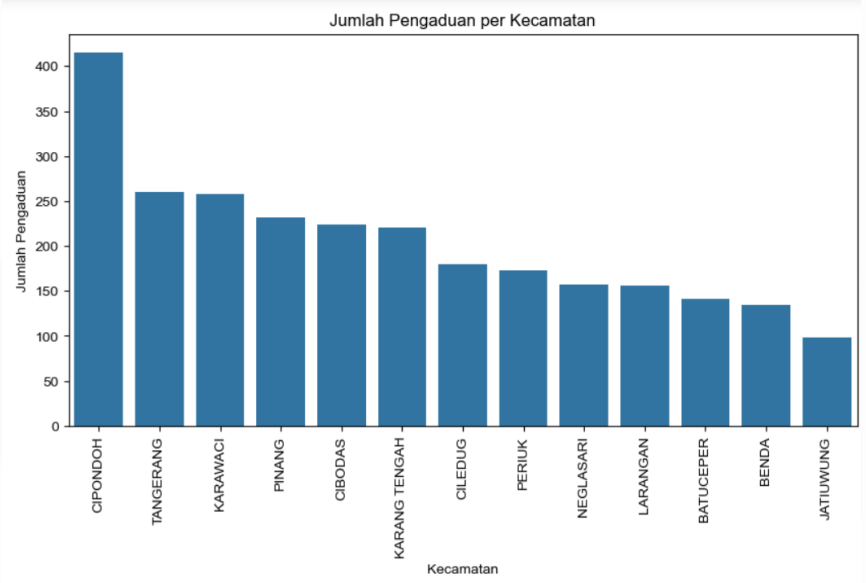
Gambar 3.47 *Visualisasi 1* tujuan penelitian 1

Pada tujuan penelitian pertama, diminta untuk mengidentifikasi dinas yang menyelesaikan pengaduan paling lambat maupun cepat. Visualisasi distribusi 1 pada gambar 3.47, terlihat bahwa Bidang Diseminasi Informasi dan Komunikasi Publik menjadi dinas terlama dalam menyelesaikan pengaduan. Hal ini dikarenakan dinas tersebut membutuhkan waktu selama 104 hari untuk menyelesaikan pengaduan yang diterima. Hal ini menjadi evaluasi untuk dinas terkait agar mempercepat penyelesaian laporan pengaduan yang diterima.



Gambar 3.48 visualisasi 2 tujuan penelitian 1

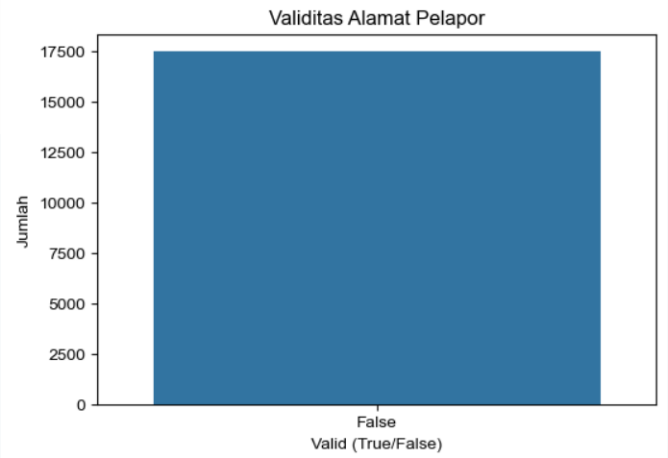
Visualisasi kedua pada gambar 3.48 masih dilakukan untuk menjawab tujuan penelitian pertama. Terlihat semua dinas yang termasuk dalam 15 dinas tercepat dalam menyelesaikan pengaduan, menyelesaikan pengaduannya dalam waktu kurang dari 1 hari. Namun dinas tercepat merupakan dinas kedaruratan dan logistik. Dinas tersebut menyelesaikan laporan pengaduan dalam waktu 0.3 hari.



Gambar 3.49 Visualisasi 1 tujuan penelitian 2

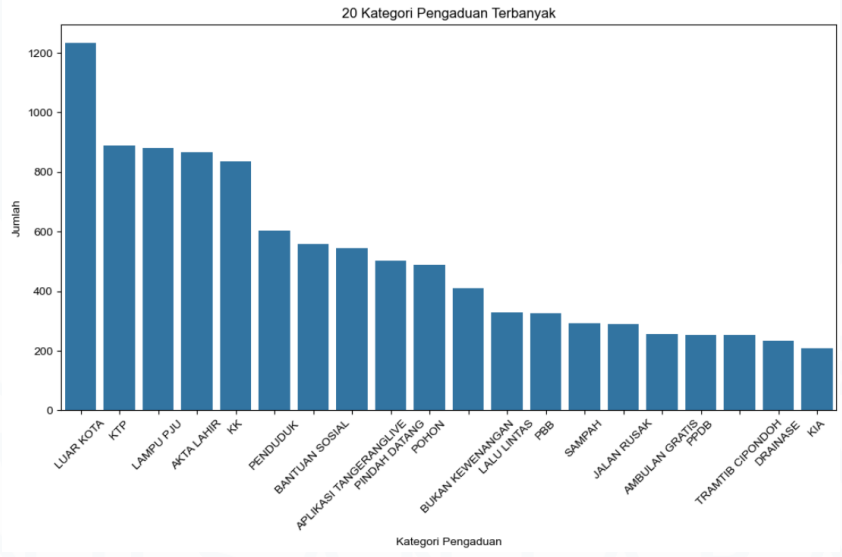
Tujuan penelitian kedua merupakan mengidentifikasi wilayah dengan jumlah pengaduan tertinggi. Pada gambar 3.49, tujuan penelitian terjawab dengan kecamatan untuk hitungan wilayahnya. Kecamatan Cipondoh menjadi kecamatan dengan jumlah pengaduan tertinggi dengan lebih dari 400 laporan. Sedangkan

kecamatan dengan paling sedikit jumlah laporannya jatuh kepada Kecamatan Jatiuwung.



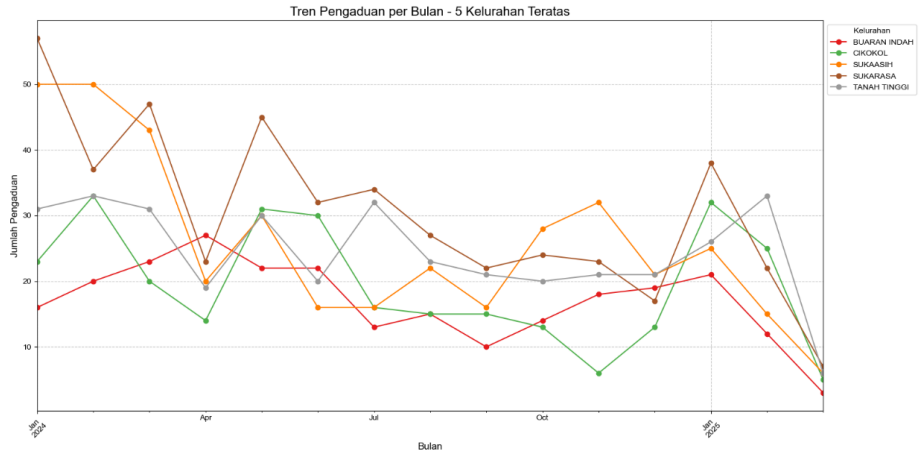
Gambar 3.50 Visualisasi 1 tujuan penelitian 3

Tujuan penelitian ketiga merupakan analisis kesesuaian alamat pelapor dengan alamat yang dilaporkan. Hal ini perlu dilakukan untuk mengetahui validitas laporan yang masuk. Namun pada gambar 3.50 yang menunjukkan visualisasi untuk menjawab tujuan penelitian 3, ditemukan bahwa seluruh alamat yang dilaporkan tidak sesuai dengan alamat pelapor. Hal ini menjadi evaluasi untuk Diskominfo dalam merespon pengaduan yang diterima.



Gambar 3.51 Visualisasi 1 untuk tujuan penelitian 4

Selanjutnya untuk visualisasi pada gambar 3.51 memperlihatkan jawaban untuk menjawab tujuan penelitian 4 yakni kategori pengaduan yang paling banyak dikeluhkan. Terlihat bahwa kategori pengaduan Luar Kota menjadi kategori paling banyak masuk dalam laporan pengaduan. Setelah dicari tahu, laporan kategori ini masuk dari wilayah diluar Kota Tangerang. Hal ini juga menjadi bahan evaluasi bagi Diskominfo maupun dinas lainnya dalam mensosialisasikan *platform* pengaduan mereka kepada masyarakat disekitarnya.



Gambar 3. 52 Visualisasi 1 tujuan penelitian 5

Visualisasi pada gambar 3.52 memperlihatkan tren pengaduan berdasarkan kelurahan. Kelurahan Sukarasa menjadi kelurahan dengan kategori pengaduan paling banyak dikeluhkan. Tiga hal yang paling dikeluhkan di Kelurahan Sukarasa diantaranya Lampu PJU, Tramtib Cipondoh, dan Pohon. hal ini membuktikan bahwa Kelurahan Sukarasa membutuhkan pengembangan dan perbaikan pada fasilitas umumnya.

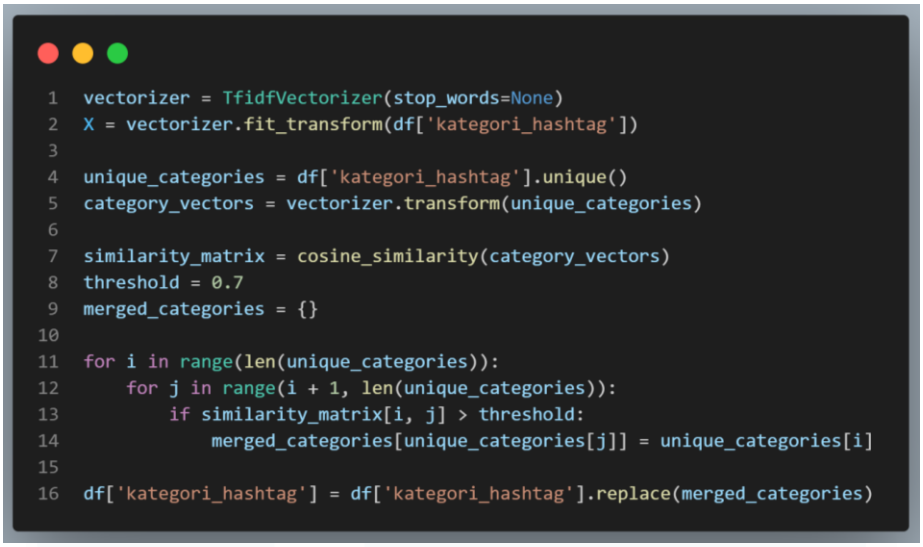
	frekuensi
tangerang	8109
kota	5127
kecamatan	2565
kelurahan	2460
ktp	2114
anak	1996
lampu	1904
warga	1556
rumah	1546
nama	1478

Gambar 3.53 tujuan penelitian 6

Tujuan penelitian keenam merupakan kata-kata atau frasa yang paling sering muncul dalam laporan. Pada gambar 3.53, Tangerang menjadi frasa paling banyak muncul dalam laporan dengan frekuensi 8.109 kali. Hal ini menunjukkan bahwa masalah yang terjadi akan menyangkut nama Kota Tangerang. Hal ini menjadi bahan evaluasi bagi Kota Tangerang dalam menyediakan pelayanan publik yang lebih baik.

3.2.9 Melakukan *modeling* pertama dengan *k-means*

Setelah data siap, tahap selanjutnya merupakan pemodelan atau *modeling*. Pada tahap pemodelan pertama, digunakan algoritma *k-means. K-means* digunakan untuk mengelompokkan pengaduan berdasarkan lokasi dan kategori. Analisis dilakukan untuk mengidentifikasi wilayah dengan jumlah pengaduan tertinggi. Dilakukan beberapa tahapan dalam pemodelan dengan *k-means*.



```
vectorizer = TfidfVectorizer(stop_words=None)
X = vectorizer.fit_transform(df['kategori_hashtag'])

unique_categories = df['kategori_hashtag'].unique()
category_vectors = vectorizer.transform(unique_categories)

similarity_matrix = cosine_similarity(category_vectors)
threshold = 0.7
merged_categories = {}

for i in range(len(unique_categories)):
    for j in range(i + 1, len(unique_categories)):
        if similarity_matrix[i, j] > threshold:
            merged_categories[unique_categories[j]] = unique_categories[i]

df['kategori_hashtag'] = df['kategori_hashtag'].replace(merged_categories)
```

Gambar 3.54 Pengelompokkan Kategori

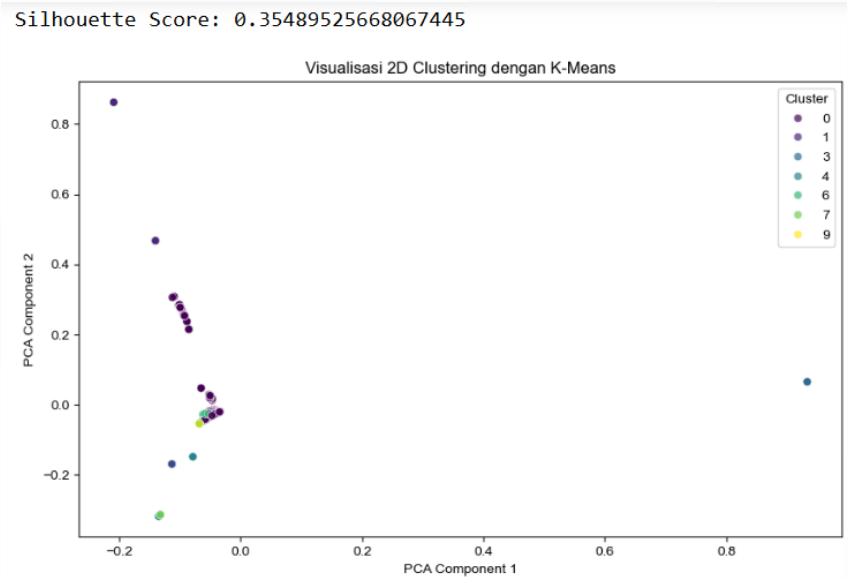
Setelah mengimpor *library-library* yang dibutuhkan, *code* pada gambar 3.54 dijalankan. *Code* tersebut merupakan *code* untuk membantu menghitung kesamaan untuk kategori yang bisa dikatakan unik. Sehingga kategori yang unik atau sedikit frekuensinya akan digabungkan dengan kategori yang mirip. Hal ini dilakukan untuk mengurangi jumlah kategori dan memfokuskan penelitian pada kategori yang potensial.

```
wcss = []
k_values = range(2, 11)
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(k_values, wcss, marker='o')
plt.xlabel('Jumlah Cluster (k)')
plt.ylabel('WCSS')
plt.title('Metode Elbow untuk Menentukan k Optimal')
plt.show()
```

Gambar 3.55 Menghitung jumlah *cluster*

Nilai k dalam konteks *clustering* merupakan jumlah *cluster* yang ingin dibentuk. Menentukan nilai k harus dilakukan dengan tepat karna jika tidak tepat

jumlahnya, data tidak dikelompokkan dengan baik. Pada gambar 3.55 digunakan metode *elbow* untuk menentukan nilai k dengan memplot WCSS terhadap k. Hasil dari nilai k merupakan 10.



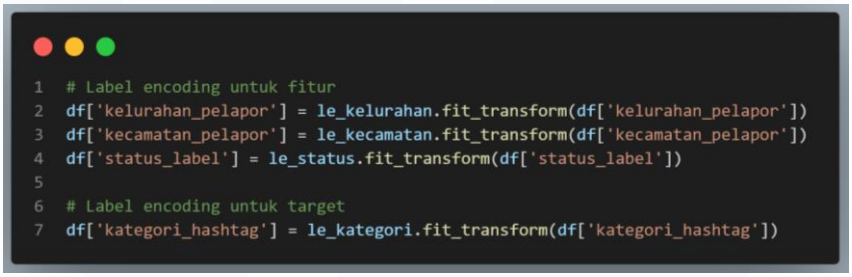
Gambar 3.56 Visualisasi hasil clustering

Pada gambar 3.56 terlihat bahwa *clustering* terlihat kurang baik. Hal ini dipengaruhi oleh nilai *silhouette* yang rendah yakni 35%. Nilai k = 10 menghasilkan pengelompokkan yang masih tumpang tindih. Telah dilakukan penambahan atau pengurangan nilai k untuk melihat visualisasi *clustering* yang dihasilkan. Namun, pemodelan pada gambar 3.56 menjadi tidak efektif sehingga hasil terbaik yang bisa didapatkan dari analisis yang dilakukan.

Dari 10 *cluster* yang terbentuk, masing-masing *cluster* memiliki pengelompokannya masing-masing. *Cluster 0 terkait fasilitas* dan masalah lalu lintas, *cluster* 1 tentang lampu PJU, *cluster* 2 tentang Kartu Keluarga, *cluster*3 tentang luar kota, *cluster* 4 tentang akta kelahiran, *cluster* 5 tentang aplikasi TangerangLive, *cluster* 6 tentang fasilitas, *cluster* 7 tentang KTP, *cluster 8* tentang penduduk, dan *cluster* 9 tentang kelistrikan. Dapat disimpulkan pengelompokkan sudah mendekati sempurna. Namun tetap perlu dilakukan analisis lebih lanjut untuk *clustering* yang lebih baik.

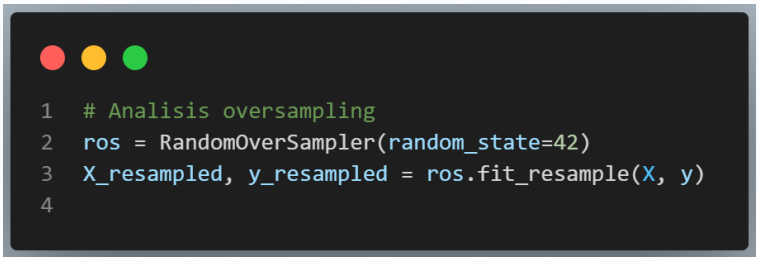
3.2.10 Melakukan *modeling* kedua dengan *random forest*

Selanjutnya dilakukan prediksi hashtag kategori dengan menggunakan *Random Forest*. Pemodelan kedua ini dilakukan dengan tujuan memprediksi kategori yang akan dilaporkan berdasarkan kelurahan, kecamatan, dan status pengaduan. Hal ini penting untuk dilakukan karena dengan memprediksi kategori pengaduan apa yang akan dilaporkan pada satu kecamatan atau kelurahan, maka kecamatan atau kelurahan tersebut dapat melakukan tindakan preventif terlebih dahulu. Dilakukan beberapa tahapan dalam melakukan pemodelan ini.



Gambar 3.57 Label encoding untuk fitur dan target

Setelah mengimpor *library-library* yang dibutuhkan, dilakukan *encode* data untuk mengubah data kategorikal menjadi numerikal. Hal ini dikarenakan *Random forest* tidak bisa menangani data kategorikal secara langsung. Namun hal ini dapat membantu mempermudah pemodelan. Kolom-kolom yang mengalami *encoding* merupakan kolom-kolom yang menjadi fitur dan target dalam proses prediksi ini.



Gambar 3.58 *Oversampling Analysis*

Setelah *encoding*, dilakukan penentuan fitur dan target. Kolom kelurahan pelapor, kecamatan pelapor, dan status label menjadi fitur. Sedangkan kolom kategori hashtag menjadi target prediksi. Setelahnya dilakukan analisis *oversampling* pada gambar 3.58. *RandomOverSampler* mengatasi adanya

ketidakseimbangan yang teridentifikasi pada *dataset*. Hal ini akan membantu meningkatkan sampel dari kelas minoritas.

```
# Membagi data menjadi training dan testing
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.2, random_state=42)
```

Gambar 3.59 Pembagian data untuk pelatihan dan pengujian data

Selanjutnya dilakukan pembagian data untuk tahap *training testing.* Pada penelitian ini, 80% data akan digunakan untuk *training.* Sedangkan 20% data akan digunakan untuk *testing*. Hal ini selalu dilakukan untuk menguji keberhasilan model.

Setelah membagi data, dilakukan penyusunan parameter untuk *GridSearch*. *GridSearch* berguna dalam pencarian grid. Pencarian grid ini sangat baik untuk menemukan kombinasi terbaik untuk model *random forest.* Setelah melakukan *GridSearch,* akan dilakukan inisiasi model *random forest classifier* untuk klasifikasi.

Kemudian, tahapan yang dilakukan selanjutnya merupakan melakukan *GridSearchCV*. *GridSearchCV* dilakukan untuk mencari parameter terbaik. Yang membedakannya, *GridSearchCV* menemukan parameter terbaik berdasarkan akurasi. Dan menggunakan 3-fold *cross-validation* untuk evaluasi.

```
# Mengambil model terbaik
best_model = grid_search.best_estimator_

# Melakukan prediksi dengan model terbaik
y_pred = best_model.predict(X_test)
```

Gambar 3.60 Pengambilan model terbaik dan prediksi

Code pada *line 2* akan membantu mengambil model terbaik. Setelahnya, akan dilakukan prediksi dengan model terbaik yang tadi di identifikasi. Prediksi yang dilakukan dengan model terbaik akan menguji 20% data *training*. Selanjutnya dilakukan evaluasi model.

```
Akurasi Model: 0.3627
```

Gambar 3.61 Hasil evaluasi model

Diakhir penelitian, evaluasi akurasi model dilakukan. Pada gambar 3.60 terlihat bahwa akurasi model sebesar 36%. Dapat dikatakan bahwa hasil pemodelan terbilang rendah dan masih perlu perbaikan. Namun dari pihak Diskominfo menerima hasil penelitian sementara dan tidak memerintahkan meneliti ulang. Pihak Diskominfo menenekankan untuk melanjutkan ke tahap proyek selanjutnya.

3.2.11 Menjadikan hasil-hasil *modeling* menjadi API

Setelah menyelesaikan analisis dan pemodelan data, tahap selanjutnya yang dilakukan merupakan menjadikan hasil pemodelan menjadi API. API atau *Application Programming Interface* merupakan protocol yang memungkinkan berbagai *software* untuk berinteraksi. Fungsi dari API sendiri diantaranya mengintegrasikan sistem dan aplikasi, otomatisasi penugasan, dan fungsionalitas bagi eksternal. Namun, pada proyek ini, API yang dibuat cukup berbeda pada API pada umunya.

API yang dibuat disimpan dalam folder yang sama dengan proyek *website.* API disimpan dalam *file* berjudul *app.py* yang tersimpan dalam folder *my_flask_app. Folder* tersebut tersimpan dalam *folder* laksa yang menjadi *folder* proyek *website*.

```
{
  "nama_dinas": "Bidang Perlindungan dan Jaminan Sosial",
  "rata_rata_hari": 1.0
},
```

Gambar 3.62 Durasi penyelesaian laporan dalam hitungan hari

API pertama menghasilkan data rata-rata durasi yang dibutuhkan masing-masing dinas untuk menyelesaikan laporan pengaduan yang masuk. Gambar 3.62 merupakan salah satu contoh *output* dari hasil API yang dibuat. *Endpoint* atau *url* dari API pertama ini adalah http://127.0.0.1:5000/ratarata_dinas. API ini nantinya akan di visualisasikan dan ditampilkan dalam *website.*


```
{
  "jumlah_pengaduan": 185,
  "nama_dinas": "DINAS PERHUBUNGAN"
},
```


Gambar 3.63 Jumlah laporan yang diselesaikan dalam 1 hari

API kedua menghasilkan data jumlah laporan yang diselesaikan dinas dalam waktu 1 hari. Gambar 3.63 merupakan salah satu contoh informasi yang dihasilkan oleh API kedua ini. *Endpoint* atau *url* API kedua adalah http://127.0.0.1:5000/laporan-1-hari. Secara keseluruhan, API kedua ini menghasilkan jumlah pengaduan yang diselesaikan serta dinas dari jumlah laporan yang diselesaikan paling banyak.


```
{
  "jumlah_hari": 0.20758585340850846,
  "nama_dinas": "DINAS KOMUNIKASI DAN INFORMATIKA"
},
```


Gambar 3.64 Dinas Tercepat menyelesaikan pengaduan

API ketiga menghasilkan data 15 dinas tercepat dalam menyelesaikan laporan. Gambar 3.64 merupakan salah satu contoh informasi yang dihasilkan oleh API ketiga ini. *Endpoint* atau *url* API kedua adalah http://127.0.0.1:5000/top-15-dinas-tercepat. Secara keseluruhan, API ketiga ini menghasilkan data 15 dinas yang menyelesaikan pengaduan paling cepat.


```
{
  "jumlah_hari": 104.0,
  "nama_dinas": "Bidang Diseminasi Informasi dan Komunikasi Publik"
},
```


Gambar 3.65 Dinas Terlama menyelesaikan pengaduan

API keempat menghasilkan data 15 dinas terlama dalam menyelesaikan laporan. Gambar 3.65 merupakan salah satu contoh informasi yang dihasilkan oleh API keempat ini. *Endpoint* atau *url* API kedua adalah http://127.0.0.1:5000/top-15-dinas-terlama. Secara keseluruhan, API keempat ini menghasilkan data 15 dinas yang menyelesaikan pengaduan paling lama.


```
{
  "2024-M01": {
    "BABAKAN": 1,
    "BUARAN INDAH": 3,
    "CIKOKOL": 7,
    "GERENDENG": 0,
    "KELAPA INDAH": 3,
    "KOANG JAYA": 0,
    "NAMBO JAYA": 0,
    "PABUARAN TUMPENG": 1,
    "PASAR BARU": 0,
    "PORIS PLAWAD INDAH": 0,
    "PORIS PLAWAD UTARA": 2,
    "SUKAASIH": 5,
    "SUKAJADI": 0,
    "SUKARASA": 25,
    "SUKASARI": 5,
    "TANAH TINGGI": 13
  },
```


Gambar 3.66 Jumlah pengaduan kecamatan secara mingguan

API kelima menghasilkan jumlah pengaduan yang diterima oleh kecamatan dalam hitungan mingguan. Gambar 3.66 merupakan salah satu contoh informasi yang dihasilkan oleh API kelima ini. *Endpoint* atau *url* API kelima adalah http://127.0.0.1:5000/pengaduan-mingguan. Secara keseluruhan, API keempat ini menghasilkan data dari minggu pertama tahun 2024 hingga minggu sepuluh tahun 2025.


```
{
  "2024-M01": {
    "BABAKAN": 2,
    "BUARAN INDAH": 16,
    "CIKOKOL": 23,
    "GERENDENG": 1,
    "KELAPA INDAH": 14,
    "KOANG JAYA": 2,
    "NAMBO JAYA": 0,
    "PABUARAN TUMPENG": 1,
    "PASAR BARU": 4,
    "PORIS PLAWAD INDAH": 13,
    "PORIS PLAWAD UTARA": 15,
    "SUKAASIH": 50,
    "SUKAJADI": 1,
    "SUKARASA": 57,
    "SUKASARI": 14,
    "TANAH TINGGI": 31
  },
```


Gambar 3.67 Jumlah pengaduan kecamatan secara bulanan

API keenam menghasilkan jumlah pengaduan yang diterima oleh kecamatan dalam hitungan bulanan. Gambar 3.67 merupakan salah satu contoh informasi yang dihasilkan oleh API kelima ini. *Endpoint* atau *url* API keenam adalah http://127.0.0.1:5000/pengaduan-bulanan. Secara keseluruhan, API keempat ini menghasilkan data dari bulan pertama tahun 2024 hingga bulan tiga tahun 2025.


```
{
  "jumlah_pengaduan": 455,
  "kelurahan_pelapor": "SUKARASA"
},
```


Gambar 3.68 Jumlah laporan pengaduan berdasarkan kelurahan

API 7 menghasilkan data 10 kelurahan dengan jumlah pengaduan paling banyak. Gambar 3.68 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-7 ini. *Endpoint* atau *url* API ke-7 adalah http://127.0.0.1:5000/10-pengaduan-kelurahan. Secara keseluruhan, API keempat ini menghasilkan 10 data jumlah pengaduan per-kelurahan yang diurutkan dari yang terbesar.


```
{
  "jumlah_invaliditas": 415,
  "kecamatan_pelapor": "CIPONDOH"
},
```

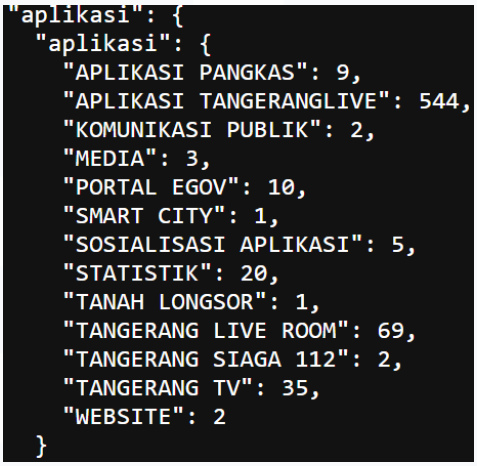

Gambar 3.69 Invaliditas laporan

API 8 menghasilkan data jumlah laporan yang tidak valid berdasarkan kecamatan pelapor. Gambar 3.69 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-8 ini. *Endpoint* atau *url* API ke-8 adalah http://127.0.0.1:5000/jumlah-invaliditas. Secara keseluruhan, API ke-8 ini menghasilkan data jumlah pengaduan tidak valid yang diurutkan dari yang terbesar.


```
  [
    "Jl. Tanah Seratus No.112, Sudimara Jaya, Kecamatan Ciledug,
Kota Tangerang, Banten 15151, Indonesia. \r\nPatokan: SDN sudimara
Timur 2 dan SDN Sudimara 5",
    1
  ],
```

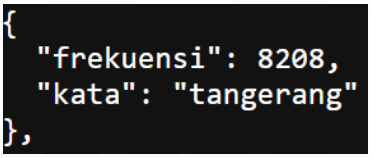

Gambar 3.70 Validitas laporan

API 9 menghasilkan data jumlah laporan yang valid. Gambar 3.70 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-9 ini. *Endpoint* atau *url* API ke-9 adalah http://127.0.0.1:5000/validitas-laporan. Secara keseluruhan, API ke-9 ini menghasilkan data jumlah pengaduan valid.



Gambar 3.71 Frasa dominan dalam kategori pengaduan

API 10 menghasilkan data jumlah frasa yang banyak muncul pada kategori pengaduan. Gambar 3.71 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-10 ini. Endpoint atau url API ke-10 adalah http://127.0.0.1:5000/frasa-dominan. Secara keseluruhan, API ke-10 ini menghasilkan data jumlah frasa.



Gambar 3.72 Frasa dominan dalam laporan

API 11 menghasilkan data jumlah frasa yang banyak muncul pada laporan. Gambar 3.72 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-11 ini. Endpoint atau url API ke-11 adalah http://127.0.0.1:5000/analisis-teks. Secara keseluruhan, API ke-11 ini menghasilkan data jumlah frasa yang banyak muncul dalam laporan.

```
{
  "jumlah_pengaduan": 1233,
  "kategori_hashtag": "LUAR KOTA"
},
```


Gambar 3.73 Jumlah pengaduan berdasarkan hashtag kategori

API 12 menghasilkan data jumlah hashtag kategori yang paling banyak diterima. Gambar 3.73 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-12 ini. Endpoint atau url API ke-12 adalah http://127.0.0.1:5000/api/keluhan. Secara keseluruhan, API ke-12 ini menghasilkan data jumlah hashtag kategori yang diterima.


```
{
  "BABAKAN": [
    {
      "jumlah": 7,
      "kategori_hashtag": "POHON"
    },
    {
      "jumlah": 7,
      "kategori_hashtag": "LALU LINTAS"
    },
    {
      "jumlah": 5,
      "kategori_hashtag": "JALAN RUSAK"
    }
  ],
```


Gambar 3.74 Tiga kategori hashtag paling banyak dilaporkan berdasarkan kelurahan

API 13 menghasilkan data jumlah tiga hashtag kategori yang paling banyak diterima setiap kelurahan. Gambar 3.74 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-13 ini. Endpoint atau url API ke-13 adalah http://127.0.0.1:5000/top3_kategori_kelurahan. Secara keseluruhan, API ke-13 ini menghasilkan data tiga hashtag kategori yang paling banyak diterima kelurahan.


```
{
  "jumlah_pengaduan": 260,
  "kecamatan_pelapor": "TANGERANG"
},
```


Gambar 3.75 Jumlah pengaduan berdasarkan kecamatan

API 14 menghasilkan data jumlah pengaduan berdasarkan kecamatan. Gambar 3.75 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-14 ini. Endpoint atau url API ke-14 adalah http://127.0.0.1:5000/jumlah-pengaduan-wilayah. Secara keseluruhan, API ke-14 ini menghasilkan data jumlah pengaduan berdasarkan kecamatan pelapor.


```
{
  "jumlah": 2861,
  "nama_dinas": "DINAS KEPENDUDUKAN DAN PENCATATAN SIPIL"
},
```


Gambar 3.76 Jumlah pengaduan yang tidak diselesaikan

API 15 menghasilkan data jumlah pengaduan yang tidak diselesaikan berdasarkan dinas. Gambar 3.76 merupakan salah satu contoh informasi yang dihasilkan oleh API ke-15 ini. Endpoint atau url API ke-15 adalah http://127.0.0.1:5000/dinas-tidak-merespon. Secara keseluruhan, API ke-15 ini menghasilkan data jumlah pengaduan yang tidak diselesaikan oleh dinas.

3.2. 12 Mengerjakan *website dashboard* sesuai ketentuan yang diberikan

Setelah semua API siap, pengerjaan *website dashboard* dilakukan. *Website dashboard* yang dikerjakan akan menampilkan beberapa visualisasi. Visualisasi-visualisasi ditampilkan dalam kategori masing-masing. Visualisasi dalam *dashboard* juga terdiri atas beberapa jenis.

```
1  $autoload['helper'] = array('url');
```

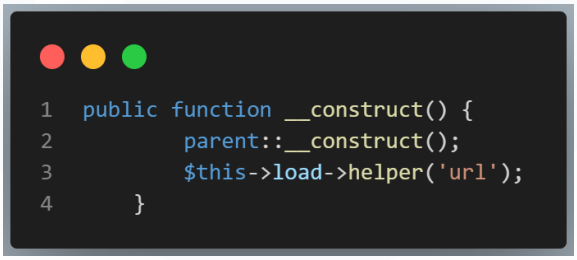
Gambar 3.77 *autoload*.php

Autoload.php dalam *framework CodeIgniter* digunakan secara otomatis untuk memuat komponen tertentu. Hal ini mengurangi kebutuhan untuk membuat komponen secara manual di *controller, model,* dan *view*. *Code* pada gambar 3.77 menunjukkan *code autoload* untuk komponen *helper*. Fungsi *helper* merupakan menyediakan utilitas tertentu seperti memanipulasi *url*.

```
$route['default_controller'] = 'dashboard';
$route['404_override'] = '';
$route['translate_uri_dashes'] = FALSE;
$route['dashboard'] = 'dashboard';
```

Gambar 3.78 *routes.php*

Code pada gambar 3.78 merupakan *code* yang terdapat pada *routes.php*. *routes.php* akan membantu menginisiasikan alur pada *website.* Pada *line* 1 terlihat jika *url* diakses, *controller dashboard* akan otomatis terpanggil. Sehingga secara *default* akan menuju *dashboard* saat *website* diakses.



```
public function __construct() {
        parent::__construct();
        $this->load->helper('url');
    }
```

Gambar 3.79 Controller

Pada gambar 3.79 dapat terlihat *code* untuk *controller.* Adanya *controller* berfungsi dalam mendefinisikan konstruktor untuk sebuah *controller*. Konstruktor dari kelas induk akan dipanggil untuk memastikan semua inisialisasi dasar dilakukan. *Helper* untuk *url* juga dilakukan untuk memberikan akses ke fungsi-fungsi terkait *url*.

```
public function index() {

        $this->load->view('backend/header');
        $this->load->view('backend/navbar');
        $this->load->view('backend/dashboard', $data);
        $this->load->view('backend/footer');

        $rata2_dinas = $this->get_api('http://127.0.0.1:5000/ratarata_dinas');
        $data['rata2_dinas'] = $rata2_dinas;
```

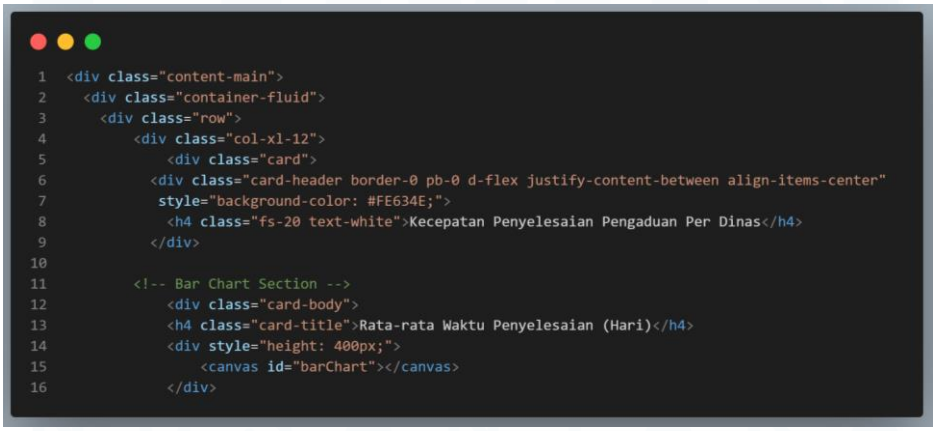
Gambar 3.80 Fungsi index

Fungsi index pada gambar 3.80 merupakan metode *default* yang dipanggil ketika tidak ada metode lain yang akan ditentukan dalam URL. Fungsi index pada *controller* juga banyak digunakan untuk menampilkan halaman utama. *Code* pada *line* 3 sampai 6 akan memuat tampilan terkait membuka halaman *web*. *Code* pada *line* 8 berguna dalam menyimpan hasil pemanggilan metode *get_api* dan memanggil metode tersebut untuk melakukan permintaan HTTP ke API. Semua API yang akan digunakan akan melalui hal yang sama.

```
1  private function get_api($url) {
2          $client = curl_init($url);
3          curl_setopt($client, CURLOPT_RETURNTRANSFER, true);
4          $response = curl_exec($client);
5          curl_close($client);
6          return json_decode($response, true);
7      }
```

Gambar 3.81 Metode *get_api*

Gambar 3.81 merupakan cara umum untuk berinteraksi dengan API dalam *aplikasi website* yang berbasis PHP. *Code* pada *line* 1 mendefinisikan metode *get_api* yang hanya bisa diakses dalam kelas yang sama. Metode tersebut juga menerima satu parameter yaitu $url yang merupakan alamat API yang nantinya diakses. Metode *get_api* menginisialisasi sesi cURL, mengatur opsi untuk mengembalikan hasil sebagai *string*, menjalankan permintaan, menutup sesi, hingga mengembalikan hasil dalam bentuk array PHP.



```
1  <div class="content-main">
2    <div class="container-fluid">
3      <div class="row">
4          <div class="col-xl-12">
5              <div class="card">
6            <div class="card-header border-0 pb-0 d-flex justify-content-between align-items-center"
7             style="background-color: #FE634E;">
8              <h4 class="fs-20 text-white">Kecepatan Penyelesaian Pengaduan Per Dinas</h4>
9            </div>
10
11         <!-- Bar Chart Section -->
12             <div class="card-body">
13             <h4 class="card-title">Rata-rata Waktu Penyelesaian (Hari)</h4>
14             <div style="height: 400px;">
15                 <canvas id="barChart"></canvas>
16             </div>
```

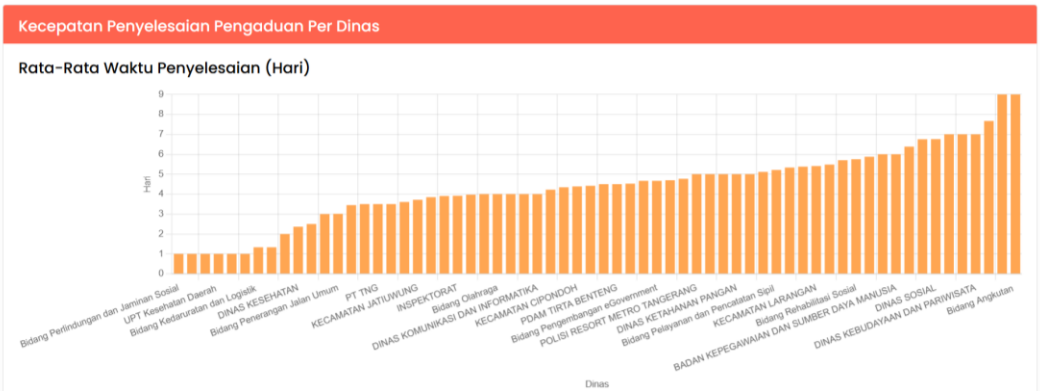
Gambar 3.82 Views untuk section 1 dan visualisasi 1

Code pada gambar 3.82 merupakan bagian dari struktur halaman *website* yang menggunakan *layout* responsive. *Code* mencakup *header* dan bagian konten untuk *bar chart* untuk kecepatan penyelesaian pengaduan per dinas. Disediakan elemen canvas untuk menggunakan skrip *JavaScript*. Struktur tersebut memastikan konten terpisah dengan baik namun tetap tampil teratur dan responsif.

```
1  async function apiBarChart() {
2      const dataFromPHP = <?= json_encode($rata2_dinas) ?>;
3
4      const labels = dataFromPHP.map(item => item.nama_dinas);
5      const values = dataFromPHP.map(item => item.rata_rata_hari);
6
7      const ctx = document.getElementById("barChart").getContext("2d");
```

Gambar 3.83 JavaScript untuk visualisasi 1

Code pada gambar 3.83 mendefinisikan fungsi asinkron apiBarChart yang bertugas untuk mengambil data dari PHP dan mempersiapkannya untuk grafik batang. *Code* pada *line 2* sampai 5 akan membuat data dari PHP di-encode menjadi JSON dan dipetakan untuk mendapatkan label dan nilai yang akan digunakan dalam grafik. Selain itu, *code* pada *line 7* akan dijalankan untuk mengambil konteks dari elemen canvas sehingga memungkinkan penggambaran grafik menggunakan JavaScript. Fungsi ini merupakan bagian penting dari integrasi frontend dan backend dalam aplikasi web.



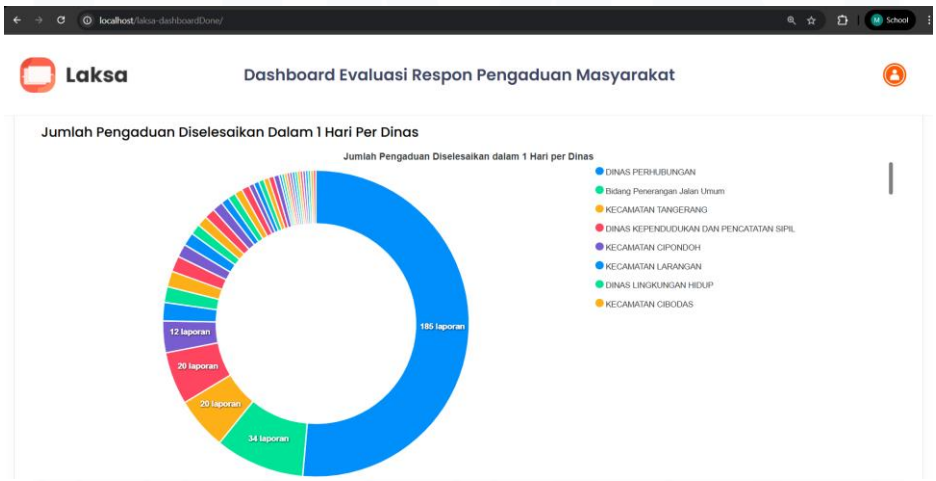
Gambar 3.84 Tampilan visualisasi 1 dalam section 1

Kecepatan Penyelesaian Pengaduan Per Dinas menjadi bagian pertama. Bagian pertama ini berisikan beberapa visualisasi salah satunya *bar chart* yang memperlihatkan rata-rata waktu yang diperlukan dinas untuk menyelesaikan laporan. Dapat dilihat bahwa Dinas Bidang Perlindungan dan Jaminan Sosial menjadi dinas tercepat dalam menyelesaikan laporan yakni dalam waktu 1 hari. Sedangkan Dinas Bidang Angkutan menjadi dinas yang memakan waktu paling banyak dalam menyelesaikan pengaduan yakni 9 hari.

```
<script src="https://cdn.jsdelivr.net/npm/apexcharts@3.41.0"></script>
<script>
const dataFromPHP = <?php echo json_encode($laporan1hari ?? []); ?>;

// Ambil series dan labels untuk donut chart
const series = dataFromPHP.map(item => item.jumlah_pengaduan);
const labels = dataFromPHP.map(item => item.nama_dinas.trim());
```

Gambar 3.85 JavaScript untuk visualisasi 2

Code pada gambar 3.86 mempersiapkan data yang diperlukan untuk menggambar grafik donut menggunakan ApexCharts. Library ApexCharts dimuat dari CDN, dan data dari PHP diambil dan diolah untuk mendapatkan nilai dan label yang akan digunakan dalam grafik. Langkah ini penting untuk menampilkan data secara visual dalam aplikasi web.



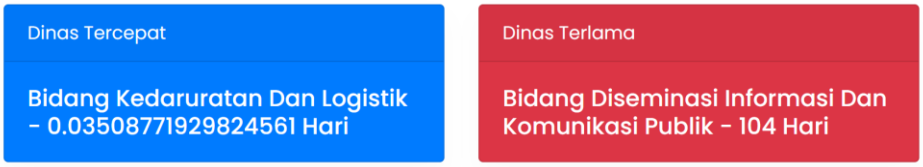
Gambar 3.86 Tampilan visualisasi 2 dalam section 1

Visualisasi kedua pada gambar 3.87 menunjukkan *donut chart* yang memperlihatkan jumlah pengaduan yang diselesaikan oleh dinas dalam satu hari. Terdapat label jumlah laporan pada masing-masing bagian. Setiap warna mewakili setiap dinas dan terdapat keterangan warna di samping *donut chart*. Berdasarkan *donut chart*, Dinas Perhubungan menjadi dinas yang menyelesaikan laporan terbanyak dalam 1 hari.

```
const fastestData = <?= json_encode($dinastercepat) ?>;
    const slowestData = <?= json_encode($dinasterlama) ?>;

    if (fastestData.length > 0) {
        document.getElementById('fastestDinas').innerText =
            fastestData[0].nama_dinas + " - " + fastestData[0].jumlah_hari + " Hari";
    }

    if (slowestData.length > 0) {
        document.getElementById('slowestDinas').innerText =
            slowestData[0].nama_dinas + " - " + slowestData[0].jumlah_hari + " Hari";
    }
```

Gambar 3.87 *JavaScript* visualisasi 3 & 4 section

Code pada gambar 3.87 mengambil data tercepat dan terlambat dari server dalam format JSON. Bagian json_encode akan mengonversi data PHP menjadi format *JavaScript*. Kemudian, *code akan* memeriksa apakah masing-masing data memiliki elemen; jika ada, elemen HTML dengan ID fastestDinas dan slowestDinas akan diupdate untuk menampilkan nama dinas dan jumlah hari dari data tercepat dan terlambat. Informasi terkait dinas ditampilkan secara dinamis di halaman web jika data tersedia.



Gambar 3.88 Tampilan visualisasi 3 & 4 section 1

Pada gambar 3.88 terlihat *highlight box* yang memperlihatkan dinas tercepat dan terlama dalam menyelesaikan laporan. Disertakan juga keterangan waktu untuk

mendukung judul yang diberikan. *Font* yang dibuat dalam warna putih agar jelas dan mudah dibaca. Warna masing-masing *highlight* box dibuat berbeda namun tetap menyesuaikan tulisan.

```
function apiRankingTable() {
    const data = <?= json_encode($sepuluhkelurahan) ?>;
    const tableBody = document.getElementById("rankingTableBody");
    tableBody.innerHTML = ""; // Clear previous content

    data.forEach((item, index) => {
        const row = `<tr>
            <td>${index + 1}</td>
            <td>${item.kelurahan_pelapor}</td>
            <td>${item.jumlah_pengaduan}</td>
        </tr>`;
        tableBody.innerHTML += row;
    });
```

Gambar 3.89 *JavaScript* visualisasi 1 section 2

Code pada gambar 3.89 merupakan sebuah fungsi *JavaScript* untuk memperbarui tabel dengan data dari API. Data dari variabel PHP (`sepuluhkelurahan`) diubah menjadi format yang dapat digunakan dengan `json_encode`. Kemudian, fungsi mengambil elemen dengan ID `rankingTableBody` untuk menampung baris-baris tabel. Sebelum menambahkan konten baru, *code* akan mengatur `innerHTML` menjadi kosong. Selanjutnya, menggunakan `forEach` untuk iterasi membuat baris baru (`<tr>`). Di setiap baris, dua sel (`<td>`) akan menampilkan nama kelurahan dan jumlah pengaduan. Setelah itu, baris baru akan ditambahkan ke dalam tubuh tabel dengan mengupdate `innerHTML`.

Pengaduan yang di Terima

10 Wilayah Dengan Jumlah Pengaduan Tertinggi

No	Kelurahan	Jumlah Pengaduan
1	SUKARASA	455
2	SUKAASIH	390
3	TANAH TINGGI	367
4	CIKOKOL	291
5	BUARAN INDAH	255
6	SUKASARI	202
7	KELAPA INDAH	198
8	PORIS PLAWAD UTARA	135
9	PORIS PLAWAD INDAH	87
10	BABAKAN	58

Gambar 3.90 Tampilan visualisasi 1 section 2

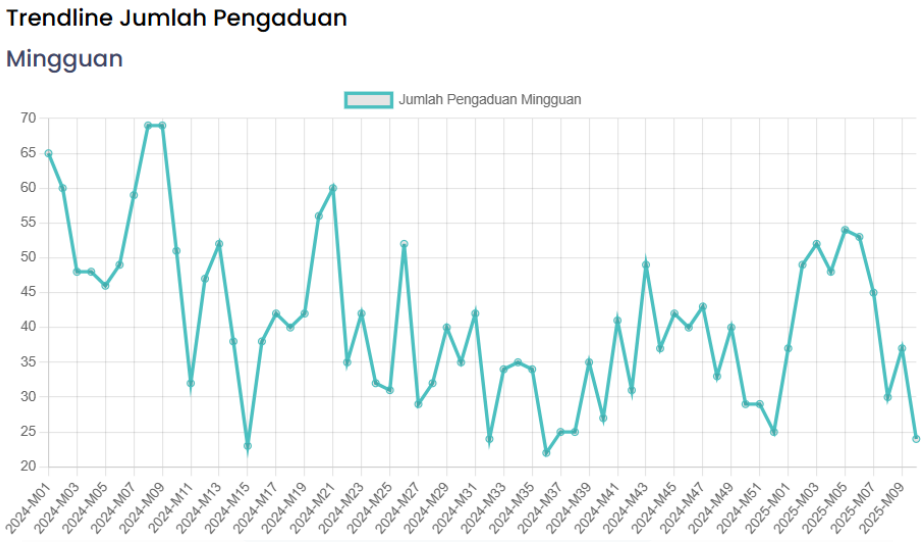
Gambar 3.90 menunjukkan tampilan *section* kedua dengan visualisasi pertama. Judul *section* yang merupakan Pengaduan yang di Terima menjadikan *section* ini smembahas seputar pengaduan. Pada visualisasi pertama, terlihat tabel yang memperlihatkan 10 kelurahan tertinggi yang menerima pengaduan. Hal ini dapat menjadi evaluasi bagi masing-masing kelurahan untuk melakukan perbaikan.

```
function renderTrendlineCharts() {
    const weeklyData = <?= json_encode($mingguan) ?>;
    const monthlyData = <?= json_encode($bulanan) ?>;

    const weeklyLabels = Object.keys(weeklyData);
    const monthlyLabels = Object.keys(monthlyData);

    const weeklyCounts = Object.values(weeklyData).map(obj =>
        Object.values(obj).reduce((a, b) => a + b, 0)
    );

    const monthlyCounts = Object.values(monthlyData).map(obj =>
        Object.values(obj).reduce((a, b) => a + b, 0)
    );

    const weeklyCtx = document.getElementById("weeklyTrendChart").getContext("2d");
    const monthlyCtx = document.getElementById("monthlyTrendChart").getContext("2d");
```

Gambar 3.91 JavaScript untuk visualisasi 3&4 section 2

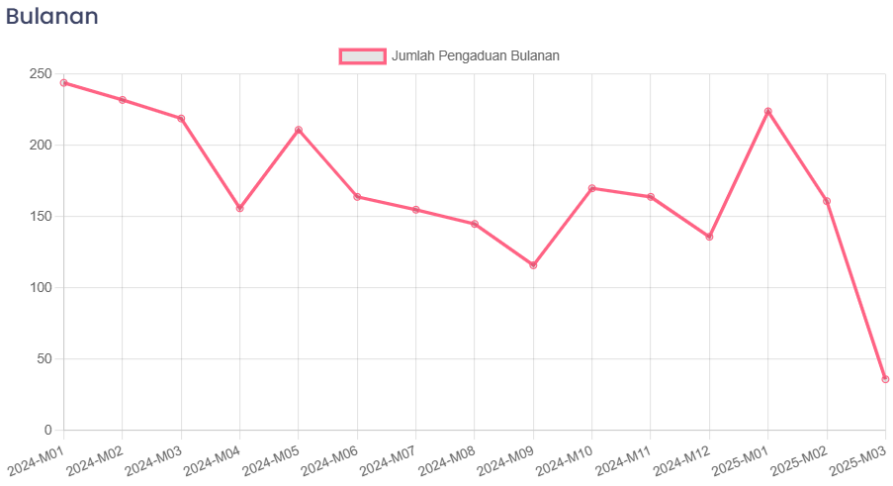
Code pada gambar 39 adalah fungsi *JavaScript* untuk menampilkan grafik tren mingguan dan bulanan. Data mingguan dan bulanan diubah menjadi format yang dapat digunakan melalui `*json_encode*`. Kunci kedua data disimpan dalam

`*weeklyLabels*` dan `*monthlyLabels*`. Nilai akan dihitung dengan metode `*reduce*` untuk menjumlahkan total pengaduan per minggu, dan disimpan dalam `*weeklyCounts*` dan *monthlyCounts*. Terakhir, konteks rendering untuk dua kanvas diambil untuk mempersiapkan menggambar grafik berdasarkan data yang telah diproses.



Gambar 3.92 Visualisasi 2 section 2

Pada *section* kedua, terdapat visualisasi untuk memperlihatkan *Trendline* jumlah pengaduan dalam hitungan mingguan dan bulanan. Gambar 3.92 merupakan visualisasi yang memperlihatkan *trendline* dalam hitungan mingguan. Terlihat bahwa pengaduan paling tinggi terjadi pada minggu ke-8 dan minggu ke-9. Baik minggu ke-8 dan minggu ke-9 memiliki 69 laporan.



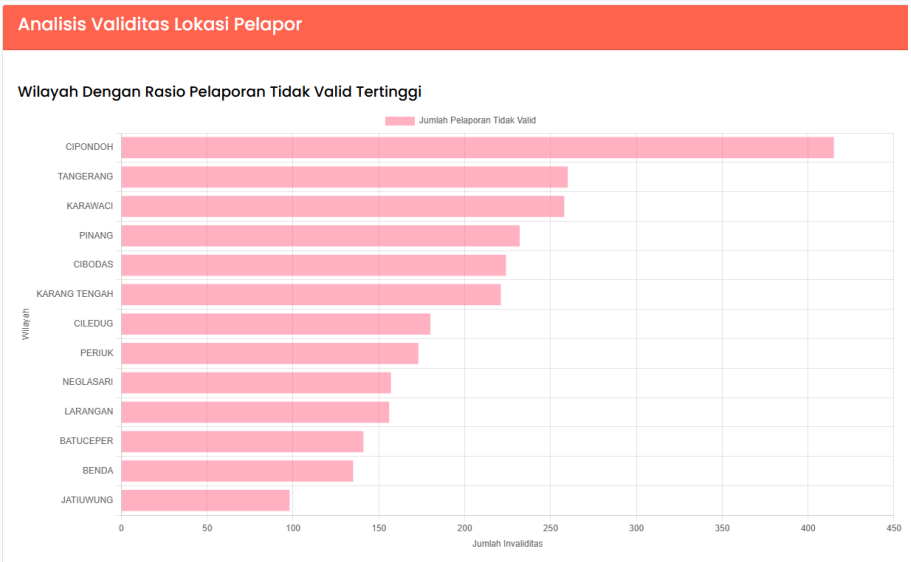
Gambar 3.93 Visualisasi 3 section 2

Gambar 3.93 memperlihatkan visualisasi *trendline* jumlah pengaduan dalam hitungan bulanan. Terlihat bahwa pengaduan paling tinggi terjadi pada bulan pertama tahun 2025 dan bulan pertama tahun 2025. Jumlah masing-masing laporan pada kedua waktu adalah 224 laporan. Sedangkan pengaduan terendah terjadi pada bulan ke-9 tahun 2024 dengan 116 laporan.

```
function renderInvalidReportingChart() {
    const data = <?= json_encode($jumlahinvalid) ?>;
    const labels = data.map(item => item.kecamatan_pelapor);
    const values = data.map(item => item.jumlah_invaliditas);

    const ctx = document.getElementById("invalidReportingChart").getContext("2d");
```

Gambar 3.94 JavaScript visualisasi 1 section 3

Gambar 3.94 merupakan *code* yang dirancang untuk menampilkan grafik terkait laporan yang tidak valid. Data akan diambil dari variabel (`jumlahlinvalid`) dan diubah menjadi format yang dapat digunakan dengan `*json_encode*`. Dilakukan pemetaan data untuk mendapatkan dua array untuk nama kecamatan dan jumlah invaliditas. Terakhir, konteks rendering akan digunakan untuk menggambar grafik berdasarkan data yang telah diproses.



Gambar 3.95 Visualisasi 1 section 3

Gambar 3.95 merupakan visualisasi pertama dalam *section* ketiga yang berjudul Analisis Validitas Lokasi Pelapor. Judul *section* ketiga memperlihatkan bahwa *section* akan berfokus pada validitas lokasi. *Chart* yang berjudul Wilayah Dengan Rasio Pelaporan Tidak Valid Tertinggi menjadi visualisasi pertama pada *section* ini. Terlihat bahwa wilayah memiliki laporan tidak valid tertinggi adalah Cipondoh dengan total 415 laporan.

```
1  function renderMismatchTable() {
2      const data = <?= json_encode($valid['invalid']) ?>;
3      const tableBody = document.getElementById("mismatchTableBody");
4      tableBody.innerHTML = ""; // Kosongkan isi lama
5
6      const maxItems = 20;
7      const limitedData = data.slice(0, maxItems);
```

Gambar 3.96 *JavaScript* visualisasi 2 *section* 3

Code pada gambar 3.96 merupakan sebuah fungsi untuk menampilkan tabel data yang menunjukkan laporan yang tidak sesuai. Data akan diambil dari variabel `valid` dan diubah menjadi format yang dapat digunakan melalui `json_encode`. Elemen `*mismatchTableBody*` diambil untuk menampung data tabel, dan isi sebelumnya dibersihkan dengan `*innerHTML*` menjadi kosong. Fungsi akan

membatasi jumlah item yang akan ditampilkan dengan *code* pada *line* 7, dana mengambil 20 item pertama dari data.

Contoh Laporan Dengan Mismatch Lokasi

No	Alamat Pelapor	Lokasi Kejadian
1	QP97+9W9, Sudimara Timur, Kecamatan Ciledug, Kota Tangerang, Banten 15151, Indonesia. Patokan: lapak barang bekas samping pemancingan	11
2	RMF2+9Q4, RT.005/RW.003, Buaran Indah, Kec. Tangerang, Kota Tangerang, Banten 15119, Indonesia. PATOKAN : GG Al Falah	4
3	QP97+9W9, Sudimara Timur, Kecamatan Ciledug, Kota Tangerang, Banten 15151, Indonesia. Patokan: lapak barang bekas samping pemancingan Jl. Masjid IX	3
4	Jl. Manggis I No.2-9, Tegal, Kecamatan Kemang, Kabupaten Bogor, Jawa Barat 16310, Indonesia. Patokan: perumahan bkr bogor	3
5	Jl. Tugu No.85, RT.003/RW.010, Cipondoh, Kec. Cipondoh, Kota Tangerang, Banten 15122, Indonesia. PATOKAN : Cipondoh	3

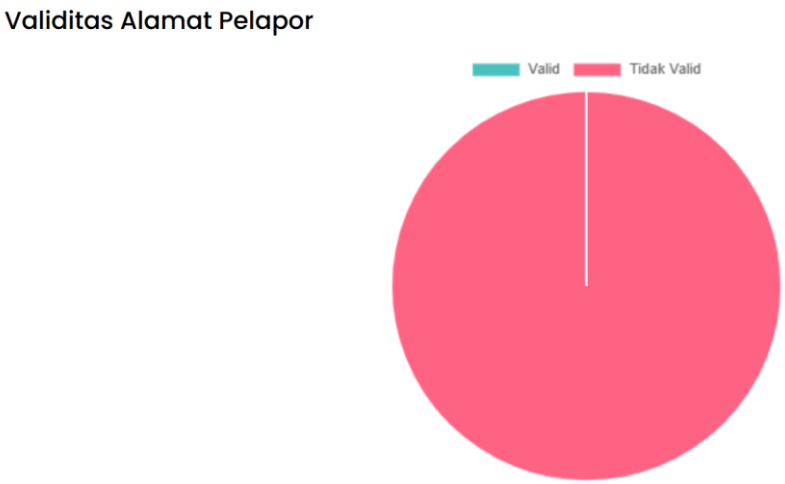
Gambar 3.97 Tampilan visualisasi 2 *section* 3

Gambar 3.97 menampilkan tabel dengan kolom no, alamat pelapor, dan lokasi kejadian. Hal tersebut memperlihatkan bahwa lokasi yang dilaporkan belum tentu cocok dengan alamat pelapor. 11 lokasi kejadian yang tidak cocok dengan alamat pelapor menjadi jumlah terbesar pada tabel ini. Sedangkan 2 lokasi yang tidak cocok dengan alamat pelapor menjadi jumlah tersedikit.

```
1  function renderValidityAddressChart() {
2      const data = <?= json_encode($alamatValid) ?>;
3
4      const ctx = document.getElementById("validityAddressChart").getContext("2d");
```

Gambar 3.98 JavaScript visualisasi 3 section 3

Gambar 3.98 memperlihatkan fungsi untuk menampilkan grafik terkait validitas alamat. Data akan diambil dan diubah menjadi format yang dapat digunakan dengan `*json_encode*`. Selanjutnya, konteks rendering diambil untuk menggambar grafik berdasarkan data yang telah diproses. Fungsi ini mempersiapkan visualisasi yang menunjukkan informasi mengenai validitas alamat yang diterima.



Gambar 3.99 visualisasi 3 section 3

Pie chart menjadi visualisasi ketiga dalam *section* ini. Visualisasi yang berjudul Validitas alamat pelapor akan memperlihatkan validitas alamat pelapor. Terdapat keterangan warna yang berada diatas *pie* chart. Berdasarkan gambar 3.99, terlihat bahwa 100% alamat pelapor tidak valid dengan alamat yang dilaporkan.

```
const treemapData = {
        name: "Kategori",
        children: Object.entries(disKategori).map(([kategori, isi]) => {
            const subKategori = isi[kategori];
            return {
                name: kategori,
                children: Object.entries(subKategori).map(([nama, jumlah]) => ({
                    name: nama,
                    value: jumlah
                }))
            };
        })
    };
```

Gambar 3.100 *JavaScript* visualisasi 1 section 4

Gambar 3.100 merupakan objek untuk mengatur struktur data untuk visualisasi treemap. Setiap kategori diubah menjadi objek dengan nama kategori dan anak-anaknya. Sedangkan setiap subkategori akan mengambil pasangan nama dan jumlahnya dan membentuk objek baru. Struktur data disiapkan untuk

menggambarkan hierarki kategori dan subkategori yang akan digunakan dalam visualisasi treemap.

Gambar 3.101 Tampilan visualisasi 1 *section* 4

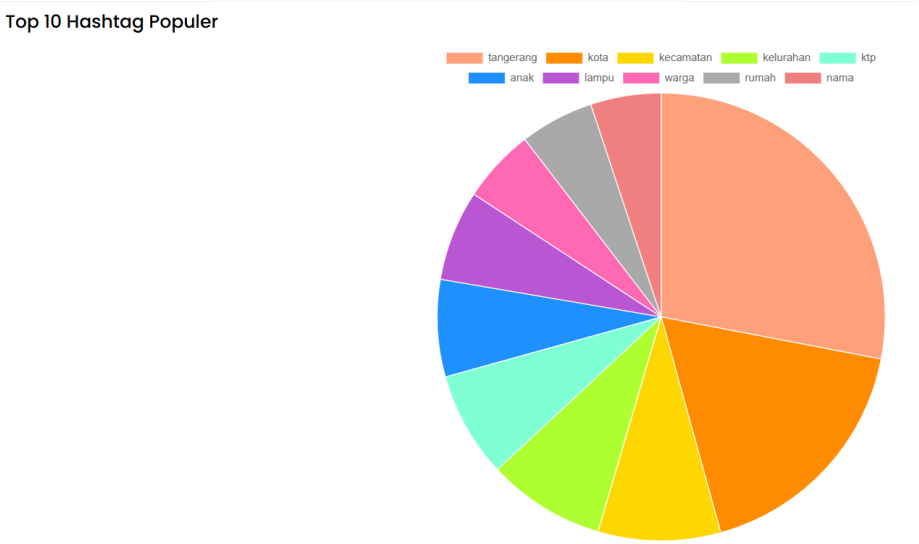
Gambar 3.101 memperlihatkan *section* keempat dalam *website* ini. Judul *section* yang merupakan Kategori Pengaduan berdasarkan Hashtag menjadikan *section* ini berfokus pada kategori pengaduan yang masuk. Visualisasi pertama yang berjudul Distribusi Hashtag Per Kategori merupakan *treemap* yang menampilkan masing-masing kategori dengan jumlah laporan yang dimiliki. Dalam setiap kategori terdapat *hashtag* kategori beserta jumlahnya juga. Pada gambar terlihat kategori *administrative* menjadi kategori terbanyak jumlah laporannya dengan total 8.111 laporan.

```
window.onload = function () {
    const hashtagData = <?php echo json_encode(array_slice($hashtagBanyak, 0, 10)); ?>;
    const labels = hashtagData.map(item => item.kata);
    const data = hashtagData.map(item => item.frekuensi);

    const ctx = document.getElementById('pieHashtag').getContext('2d');
```

Gambar 3.102 *JavaScript* visualisasi 2 *section* 4

Code pada gambar 3.102 memperlihatkan fungsi untuk mempersiapkan visualisasi pie chart untuk menampilkan distribusi frekuensi. Data akan diubah menjadi format JavaScript dengan menggunakan `json_encode`. Dan jumlah item yang diambil dibatasi menjadi 10. Selanjutnya, *code* memetakan data untuk mendapatkan dua array yang berisi nama hashtag dan frekuensi. Terakhir, konteks

rendering diambil dari elemen kanvas dan akan digunakan untuk menggambar grafik berdasarkan data yang telah diproses.



Gambar 3.103 Tampilan visualisasi 2 *section* 4

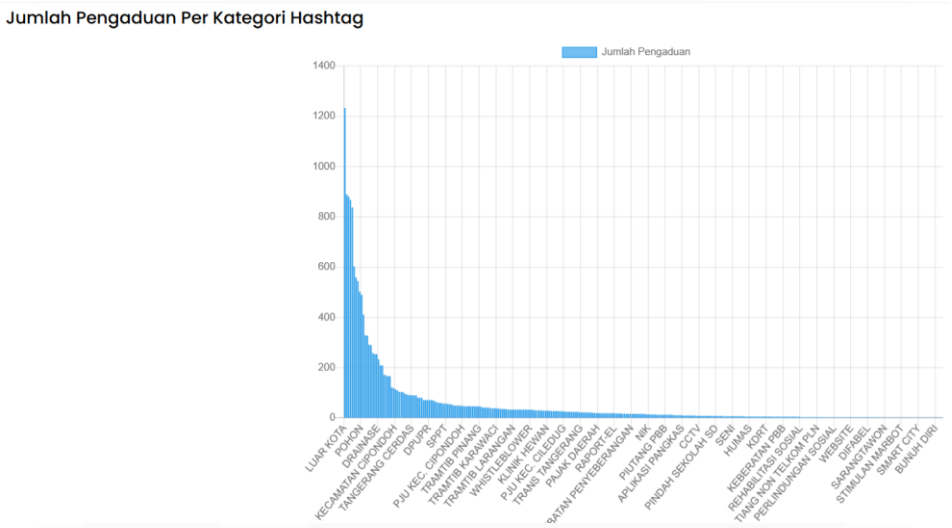
Gambar 3.103 memperlihatkan *pie chart* dengan judul *Top* 10 *Hashtag* Populer. Terdapat keterangan warna diatas *pie chart* untuk memperlihatkan masing-masing warna yang mewakili kategori. Kategori terbanyak dalam *pie chart* ini merupakan Tangerang dengan total 8.208 laporan. Sedangkan kategori dengan jumlah paling sedikit dalam visualisasi ini adalah nama dengan total 1.490 laporan.

```
document.addEventListener("DOMContentLoaded", function () {
    const data = <?= json_encode($hashtagChart) ?>;
    const labels = data.map(item => item.kategori_hashtag);
    const values = data.map(item => item.jumlah_pengaduan);

    const ctx = document.getElementById("barKeluhan").getContext("2d");
```

Gambar 3.104 *JavaScript* visualisasi 3 *section* 4

Code pada gambar 3.104 digunakan untuk mempersiapkan visualisasi grafik batang yang menunjukkan jumlah pengaduan per kategori hashtag. Data diambil akan diubah menjadi format *JavaScript* dengan `*json_encode*`. Akan ada dua *array* yang berisi kategori hashtag dan jumlah pengaduan. Kemudian, konteks rendering

diambil dari elemen kanvas untuk menggambar grafik berdasarkan data yang telah diproses. Fungsi ini bertujuan.



Gambar 3.105 Tampilan visualisasi 3 *section* 4

Pada gambar 3.105 terlihat visualisasi dengan judul Jumlah Pengaduan Per Kategori Hashtag. Terdapat keterangan warna diatas untuk memperjelas hal apa yang diwakili warna tersebut. Kategori hashtag terbanyak pada tabel merupakan Luar Kota dengan total 1.233 laporan. Sedangkan kategori dengan jumlah laporan paling sedikit merupakan bunuh diri dengan total 2 laporan.

```
const masalahKelurahan = <?= json_encode($masalahKelurahan); ?>;
const bubbleData = [];
Object.entries(masalahKelurahan).forEach(([kelurahan, masalahList]) => {
    masalahList.forEach((masalah, index) => {
        bubbleData.push({
            x: kelurahan,
            y: index + 1,
            r: Math.max(5, masalah.jumlah * 2),
            label: masalah.kategori_hashtag,
            backgroundColor: getRandomColor()
        });
    });
});
```

3.106 *JavaScript* visualisasi 1 *section* 5

Code pada gambar 3.106 akan mendefinisikan variabel `masalahKeluhan` yang berisi data dari PHP dan mengonversinya ke format JavaScript dengan

`*json_encode*`. Sebuah array kosong dibuat untuk menyimpan informasi visualisasi. Kode iterasi melalui akan menambahkan objek baru ke properti `x`, `y`, dan `label`, di mana `y` dihitung dengan mengalikan jumlah masalah dengan 2, dan `backgroundColor` diatur dengan fungsi `getRandomColor()`. Data ini disiapkan untuk visualisasi bubble chart yang menggambarkan masalah keluhan berdasarkan kategori.

Gambar 3.107 Tampilan visualisasi 1 section 5

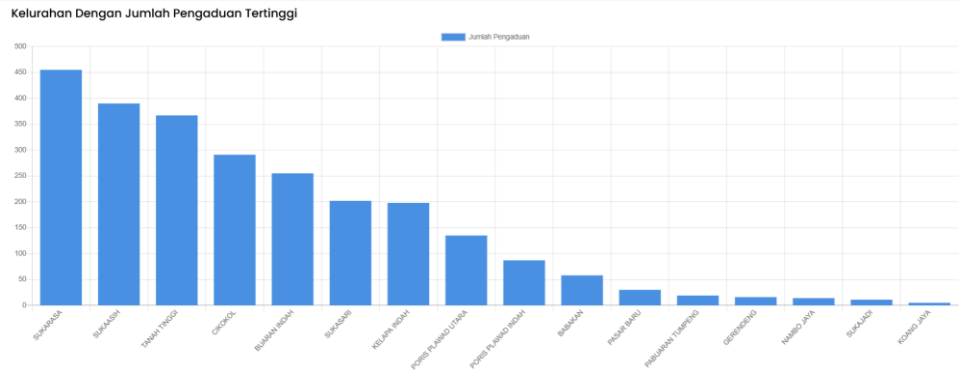
Section yang berjudul Analisis Laporan Lokasi Pelapor merupakan *section* yang berfokus pada menganalisis laporan pada kelurahan. Pada visualisasi pertama *section* ini, terdapat *bubble chart* yang memperlihatkan banyak laporan dengan kelurahan. Kelurahan dan laporan terbannyak merupakan lampu PJU di Kelurahan Sukarasa. Sedangkan jumlah laporan paling sedikit adalah 3 laporan yang terjadi di beberapa kelurahan.

```
document.addEventListener('DOMContentLoaded', function () {
    const dataFromPHP = <?= json_encode($pengaduanKelurahan, JSON_UNESCAPED_UNICODE); ?>;
    const labels = dataFromPHP.map(item => item.kelurahan_pelapor);
    const values = dataFromPHP.map(item => item.jumlah_pengaduan);

    const ctx = document.getElementById("pengaduanKelurahanChart").getContext("2d");
```

Gambar 3.108 *JavaScript* visualisasi 2 *section* 5

Code pada gambar 3.107 merupakan fungsi untuk mempersiapkan visualisasi grafik yang menunjukkan jumlah pengaduan berdasarkan pelapor. Akan dilakukan

beberapa tahapan untuk membuat visualisasi terakhir pada *section* terakhir *website* ini. Data akan diambil dan diubah menjadi format *JavaScript* menggunakan `*json_encode*`. Konteks rendering diambil dari elemen kanvas dengan ID `pengaduanKeluhanChart`, dan akan digunakan untuk menggambar grafik berdasarkan data yang telah diproses.



Gambar 3.109 Tampilan visualisasi 2 *section* 5

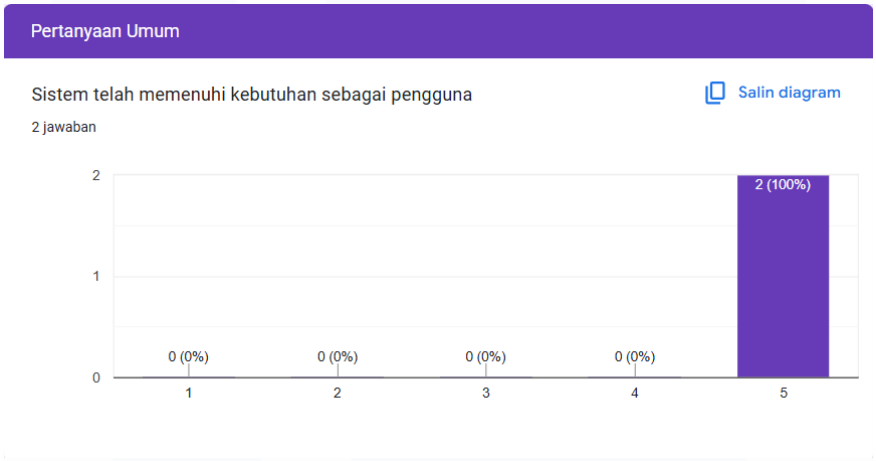
Gambar 3.108 merupakan *chart* yang menampilkan kelurahan yang menerima pengaduan tertinggi. Terdapat keterangan warna diatas *chart* sebagai petunjuk apa yang diwakili oleh warna. Kelurahan dengan jumlah pengaduan tertinggi merupakan Sukarasa dengan 455 laporan. Sedangkan kelurahan yang menerima laporan paling sedikit adalah Koang Jaya dengan total 5 laporan.

3.2.13 Melakukan pengujian aplikasi *website* oleh *supervisor*

Setelah menyelesaikan pengerjaan proyek aplikasi *website* yang diberikan, pengujian aplikasi dilakukan untuk memastikan fungsi berjalan dengan baik. Selain fungsi, pengujian ini juga dilakukan untuk memastikan bahwa tampilan aplikasi *website* sudah baik. Pengujian aplikasi *website* ini dilakukan dengan melibatkan *supervisor* dan asisten *supervisor* sebagai penguji.

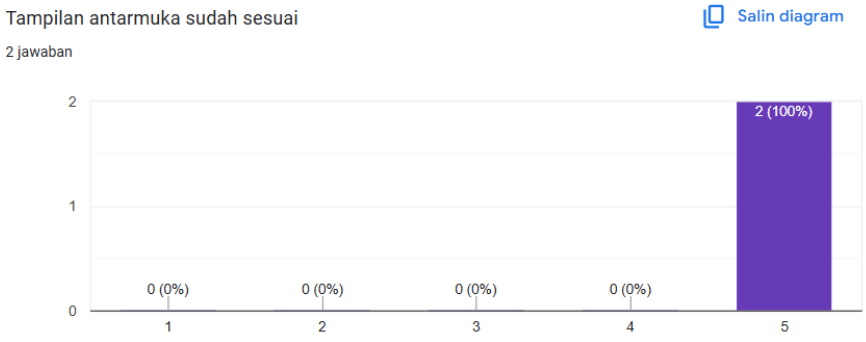
Google form digunakan untuk pengujian ini dan disediakan berbagai pertanyaan yang menyangkut fungsi dan tampilan aplikasi *website.* Sebagai jawaban, opsi nilai 1 sampai dengan 5 disediakan. Selanjutnya, *supervisor* dan asisten *supervisor* akan mengisi *google form* yang telah disediakan. Pengujian dan pengisian *google form* ini berlangsung sampai 2 hari.

Pada *google form* disediakan 2 bagian untuk membedakan fokus pertanyaan. Bagian pertanyaan merupakan bagian umum yang pertanyaannya secara umum mengenai kegunaan, tampilan, waktu respon, kendala, dan mudah dipahami. Sedangkan bagian kedua berfokus pada tampilan aplikasi *website* mengenai tampilan dan kejelasan. Masing-masing bagian memiliki 5 pertanyaan.



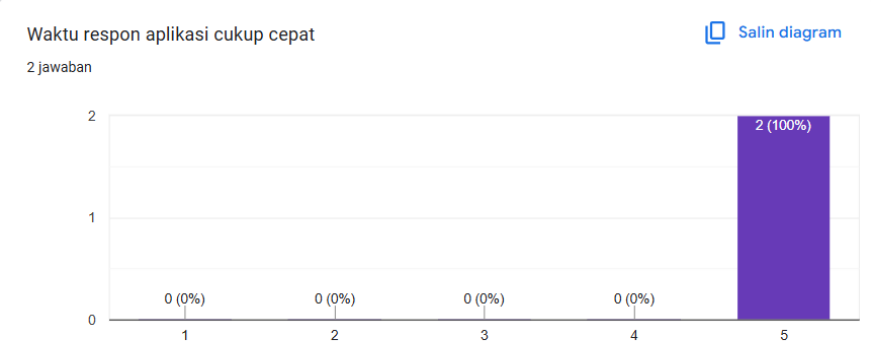
Gambar 3.110 Pertanyaan 1 bagian 1

Pertanyaan pertama pada bagian pertama menanyakan jika sistem sudah memenuhi kebutuhan penguji sebagai pengguna. Sebagai respon, kedua penguji memberikan nilai 5. Berdasarkan respon yang diberikan, dapat disimpulkan aplikasi *website* membantu mengevaluasi pengaduan yang masuk. Sehingga dari segi kegunaan, aplikasi *website* yang dibuat sudah berguna dengan baik.



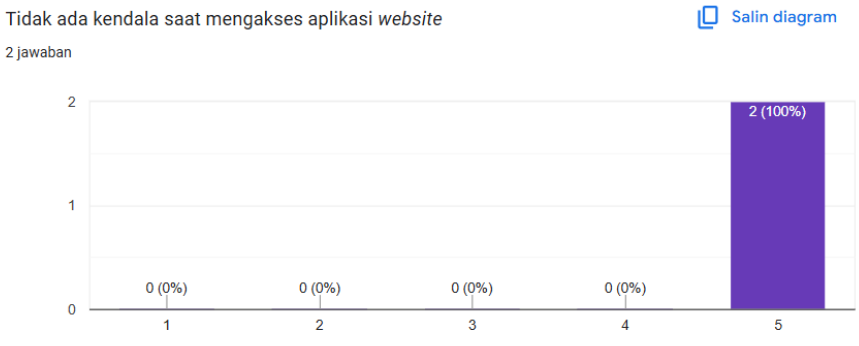
Gambar 3.111 Pertanyaan 2 bagian 1

Pertanyaan kedua pada bagian pertama menanyakan jika tampilan antarmuka sistem sudah sesuai dari perspektif penguji sebagai pengguna. Sebagai respon, kedua penguji memberikan nilai 5. Berdasarkan respon yang diberikan, dapat disimpulkan bahwa tampilan aplikasi *website* sudah sesuai. Sehingga dari segi tampilan secara umum, tampilan aplikasi *website* sudah baik.



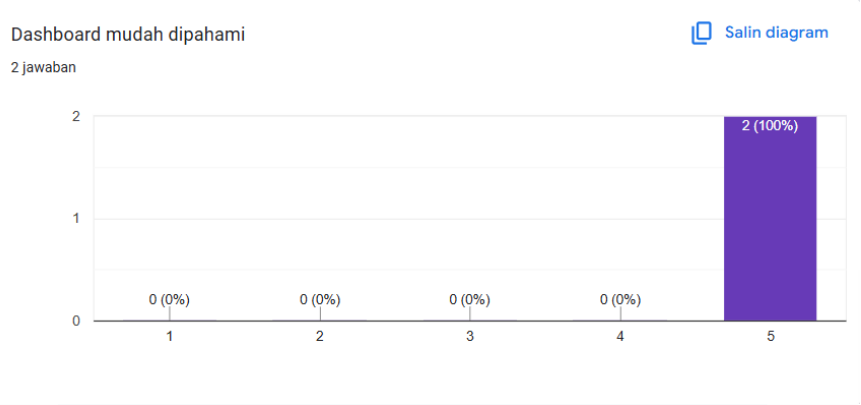
Gambar 3.112 Pertanyaan 3 bagian 1

Pertanyaan ketiga pada bagian pertama menanyakan jika waktu respon aplikasi *website* sudah bisa dikatakan cepat dari perspektif penguji sebagai pengguna. Sebagai respon, kedua penguji memberikan nilai 5. Berdasarkan respon yang diberikan, dapat disimpulkan bahwa waktu respon aplikasi website sudah bisa dikatakan cepat. Sehingga dari segi waktu, waktu respon aplikasi website sudah cepat.



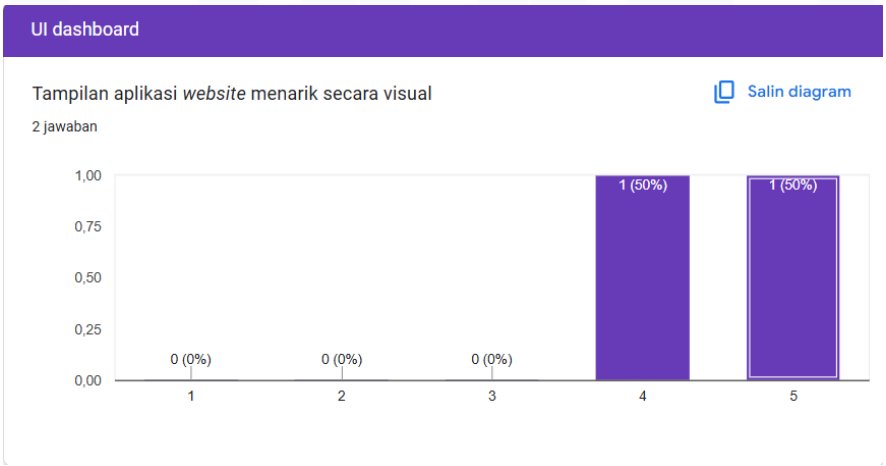
Gambar 3.113 Pertanyaan 4 bagian 1

Pertanyaan keempat pada bagian pertama menanyakan jika tidak ada kendala saat pengaksesan aplikasi *website*. Sebagai respon, kedua penguji memberikan nilai 5. Berdasarkan respon yang diberikan, dapat disimpulkan bahwa tidak ada kendala yang ditemukan saat aplikasi *website* diakses. Sehingga dari pengaksesan, aplikasi *website* sudah baik.



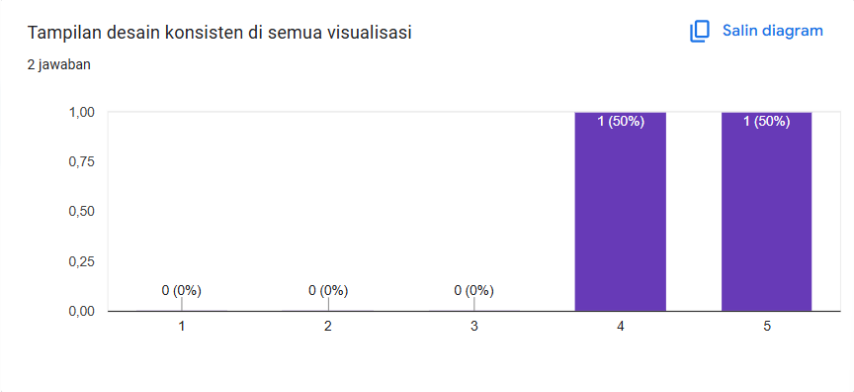
Gambar 3.114 Pertanyaan 5 bagian 1

Pertanyaan terakhir pada bagian pertama menanyakan jika tampilan antarmuka sistem mudah dipahami dari perspektif penguji sebagai pengguna. Sebagai respon, kedua penguji memberikan nilai 5. Berdasarkan respon yang diberikan, dapat disimpulkan bahwa tampilan aplikasi website sudah baik. Sehingga dari segi tampilan secara umum, tampilan aplikasi website sudah bisa dipahami dengan baik.



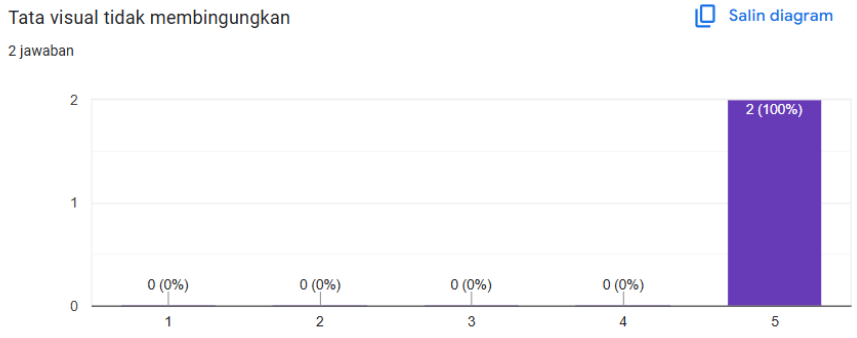
Gambar 3.115 Pertanyaan 1 bagian 2

Bagian kedua merupakan bagian yang berfokus pada tampilan aplikasi. Pertanyaan pertama dari bagian kedua menanyakan jika tampilan aplikasi *website* sudah menarik dari perspektif penguji sebagai pengguna. Terdapat 2 jenis respon yang didapatkan pada pertanyaan ini. Ada 1 buah nilai 4 dan 1 buah nilai 5 yang artinya tampilan aplikasi *website* belum bisa dikatakan sepenuhnya menarik. Hal ini menjadi evaluasi agar tampilan bisa dibuat lebih menarik lagi.



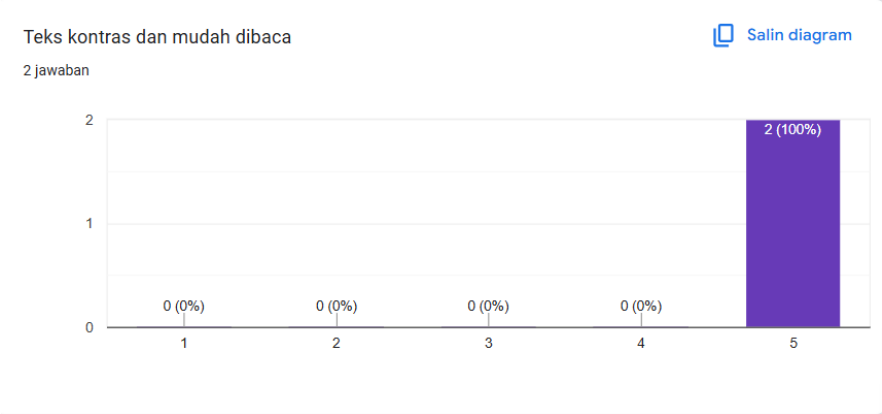
Gambar 3.116 Pertanyaan 2 bagian 2

Pertanyaan kedua dari bagian kedua menanyakan jika tampilan aplikasi *website* sudah konsisten secara keseluruhan dari perspektif penguji sebagai pengguna. Terdapat 2 jenis respon yang didapatkan pada pertanyaan ini. Ada 1 buah nilai 4 dan 1 buah nilai 5 yang artinya tampilan aplikasi *website* belum bisa dikatakan sepenuhnya konsisten. Hal ini menjadi evaluasi agar tampilan bisa dibuat lebih konsisten dan perhatikan detail-detail lain



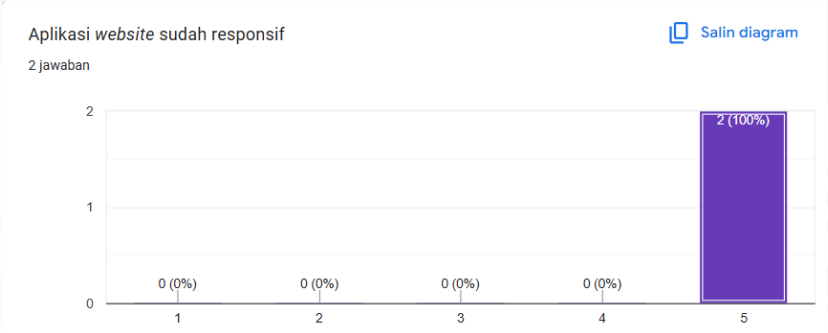
Gambar 3.117 Pertanyaan 3 bagian 2

Pertanyaan ketiga dari bagian kedua menanyakan jika susunan visualisasi dalam aplikasi *website* sudah baik dari perspektif penguji sebagai pengguna. Sebagai respon, kedua penguji memberikan nilai 5. Berdasarkan respon yang diberikan, dapat disimpulkan bahwa susunan visualisasi aplikasi *website* sudah baik. Sehingga dari segi tampilan, tampilan aplikasi website sudah.



Gambar 3.118 Pertanyaan 4 bagian 2

Pertanyaan keempat dari bagian kedua menanyakan jika tampilan aplikasi website sudah kontras sehingga mudah dibaca dari perspektif penguji sebagai pengguna. Sebagai respon, kedua penguji memberikan nilai 5. Berdasarkan respon yang diberikan, dapat disimpulkan bahwa tampilan aplikasi website sudah kontras dalam artian pemilihan dan penggunaan warna sudah membuat informasi dalam aplikasi website mudah terbaca. Sehingga dari segi tampilan secara umum, tampilan aplikasi website sudah bisa dibaca dengan baik.



Gambar 3.119 Pertanyaan 5 bagian 2

Pertanyaan terakhir dari bagian kedua menanyakan jika tampilan aplikasi website sudah responsive dari perspektif penguji sebagai pengguna. Sebagai respon, kedua penguji memberikan nilai 5. Berdasarkan respon yang diberikan, dapat disimpulkan bahwa aplikasi website sudah responsif. Sehingga dari segi respon secara umum, tampilan aplikasi website sudah merespon dengan baik.

3.2.14 Menyusun laporan magang yang berisikan informasi kegiatan kerja magang dan pengerjaan proyek

Setelah menyelesaikan pengerjaan proyek yang diberikan, dilakukan penyusunan dan penulisan laporan MBKM *Internship*. Laporan yang dibuat terdiri atas 4 bab penting. Bab pertama akan menjadi pendahuluan untuk laporan kerja magang. Bab ini akan menjelaskan latar belakang permasalahan proyek, pendekatan yang digunakan untuk melakukan pengerjaan proyek, tahapan kerja magang, dan tabel linimasa untuk mengetahui hal apa yang akan dilakukan selama kerja magang.

Selanjutnya bab 2 akan menjelaskan mengenai lokasi dimana kerja magang dilakukan yakni Diskominfo. Pada bab ini, dijelaskan sejarah, visi misi, dan struktur organisasi Diskominfo. Disertakan gambar logo dan struktur organisasi Diskominfo untuk membantu pemahaman tentang Diskominfo. Selanjutnya, dijelaskan juga struktur organisasi, inti Diskominfo, serta bagian atau divisi tempat kerja magang dilakukan.

Bab 3 membahas hasil kerja yang dilakukan selama kerja magang. Disertakan tabel realisasi yang menyesuaikan dengan tabel linimasa pada bab 1. Hal ini dilakukan untuk memastikan bahwa kegiatan kerja magang yang dilakukan di tabel realisasi, sesuai dengan perencanaan kegiatan kerja magang di tabel linimasa. Dijelaskan juga detail apa saja pekerjaan yang dilakukan dengan dokumentasi hasil kerja maupun *code*.

Bab 4 akan menjadi penutup laporan ini. Pada bab ini terdapat kesimpulan pengerjaan proyek yang dikerjakan. Kesimpulan akan didasarkan pada keberhasilan

yang didapatkan dari pengerjaan proyek. Selanjutnya pada bab 4 terdapat bagian untuk menjelaskan kendala-kendala yang dihadapi selama kerja magang dengan solusi yang menyelesaikan kendala-kendala tersebut. Terdapat juga saran untuk perusahaan, universitas, dan mahasiswa yang akan melakukan MBKM *internship*.

Laporan ini dilengkapi dengan daftar pustaka untuk referensi dan lampiran. Lampiran yang disertakan pada laporan ini terdiri atas beberapa hal. Hal-hal yang dilampirkan dalam laporan diantaranya surat pengantar yang diberikan ke Diskominfo sebelum magang, kartu MBKM, *daily task* MBKM, lembar verifikasi laporan MBKM, surat penerimaan MBKM (*LoA*), lampiran pengecekan hasil turnitin dan semua hasil karya tugas selama MBKM. Penulisan laporan dilakukan dan dilaporkan secara berkala kepada dosen pembimbing.

3.2.15 Menunjukkan kedisiplinan dengan hadir tepat waktu di kantor sesuai dengan jadwal yang di sepakati

Pada rapat pertama pendiskusian kerja magang, terdapat beberapa hal yang di diskusikan. Hal-hal tersebut diantaranya jadwal pembagian hari WFO dan WFH, jam masuk atau memulai kerja magang dan jam selesai kerja atau jam pulang, serta pakaian yang dikenakan selama WFO. Dari hasil diskusi yang dilakukan, disepakati untuk melakukan tiga hari WFO dan 2 hari WFH asal tetap melaporkan hasil kerja 2 hari WFH. Selanjutnya disepakati bahwa pukul 08.00 WIB di wajibkan untuk memulai jam kerja. Pukul 12.00 WIB hingga 13.00 WIB untuk waktu istirahat makan siang dan pukul 17.00 diperbolehkan pulang. Untuk pakaian selama kerja magang WFO, diwajibkan untuk mengenakan pakaian yang formal dan sopan. Hal-hal tersebut disepakati oleh pihak Diskominfo dan termasuk kedalam *Letter of Acceptance* sehingga wajib diperhatikan dan dijalankan dengan penuh komitmen.

3.3 Kendala yang Ditemukan

Selama masa praktek kerja magang dilaksanakan, terdapat beberapa kendala yang ditemukan. Berikut kendala dan penjelasannya dalam bentuk point :

1. **Kurangnya pengalaman dalam menggunakan *Framework CodeIgniter***

   Salah satu kendala utama yang dihadapi selama magang di Dinas Komunikasi dan Informatika adalah kurangnya pengalaman dalam menggunakan framework CodeIgniter. Framework ini merupakan alat yang digunakan oleh tim untuk pengembangan aplikasi, namun bagi pemagang, CodeIgniter masih terbilang asing. Keterbatasan pengetahuan dan pengalaman dalam penggunaan framework ini menjadi tantangan tersendiri, yang berdampak pada efektivitas waktu dalam menyelesaikan proyek. Akibatnya, proses pengerjaan proyek menjadi lambat dan mempengaruhi keseluruhan timeline yang telah ditetapkan.

2. **Sebagian besar data yang disediakan Diskominfo merupakan data *unstructured***

   Setelah satu bulan pertama belajar berakhir, supervisor memberikan data beserta sejumlah poin penting untuk pengerjaan proyek. Namun, data yang disediakan banyaknya tidak terstruktur, dengan banyak nilai null dan tidak mencerminkan kondisi real-time. Selain itu, karena proyek ini berkaitan dengan evaluasi respon pengaduan, terdapat banyak pengaduan yang harus diproses terlebih dahulu sebelum melanjutkan analisis lebih lanjut. Hal ini menuntut perhatian ekstra untuk memastikan bahwa data yang digunakan dalam evaluasi adalah akurat dan relevan, sehingga hasil proyek dapat memberikan wawasan yang berharga.

3. **Terbatasnya akses *website* referensi**

   Salah satu kendala yang dihadapi selama kerja magang adalah terbatasnya akses ke website resmi milik Diskominfo. *Website* tersebut hanya bisa diakses oleh orang-orang tertentu yang memiliki wewenang, seperti pegawai internal Diskominfo. Dikarenakan status sebagai mahasiswa magang, hal tersebut menyebabkan terhalangnya pengaksesan ke *website* tersebut. Hal ini menjadi tantangan karena dashboard yang sedang dikembangkan seharusnya bisa menyesuaikan atau terintegrasi dengan tampilan dan alur dari website aslinya. Akibatnya, beberapa bagian

dari dashboard dibuat berdasarkan asumsi atau arahan lisan dari pembimbing.

3.4 Solusi yang dilakukan

Walaupun menemukan beberapa kendala selama kerja magang, ditemukan juga solusi untuk masing-masing kendala yang dihadapi sebagai berikut :

1. **Memperbanyak konsultasi dengan *superisor* dan asisten *supervisor***

   Untuk mengatasi kendala tersebut, pemagang mengambil langkah proaktif dengan memperbanyak komunikasi dan konsultasi dengan supervisor dan asisten supervisor. Ketika menghadapi kebingungan atau kendala, seperti error dalam kode, pemagang aktif mencari waktu untuk berdiskusi dan meminta bantuan. Baik supervisor maupun asisten supervisor sangat mendukung dan memberikan respons yang positif, sehingga pemagang merasa didukung dalam proses pembelajaran. Pada awal periode magang, mereka juga diberikan rekomendasi channel YouTube yang relevan dengan materi yang dipelajari, serta akses ke tutorial yang disusun oleh asisten supervisor di platform Notion. Pendekatan ini tidak hanya membantu pemagang untuk memahami framework CodeIgniter dengan lebih baik, tetapi juga memperkuat hubungan kerja yang positif antara pemagang dan tim, menciptakan lingkungan belajar yang kolaboratif dan mendukung.

2. **Memperdalam proses *data preprocessing* dan *data preparation***

   Untuk mengatasi kendala yang ditemukan pada data, telah dilakukan beberapa langkah strategis, termasuk data preprocessing dan data preparation. Proses data preprocessing bertujuan untuk membersihkan data dari nilai-nilai null dan inkonsistensi, sehingga kualitas data yang digunakan dapat ditingkatkan. Selanjutnya, data preparation dilakukan untuk mengorganisir dan menyusun data agar sesuai dengan kebutuhan analisis. Langkah-langkah ini diharapkan menghasilkan hasil analisis yang lebih akurat dan relevan, memberikan wawasan yang lebih baik dalam evaluasi respon pengaduan. Upaya ini tidak hanya meningkatkan keandalan data,

tetapi juga membantu dalam memperdalam analisis yang diperlukan untuk pengambilan keputusan yang lebih tepat.

3. **Memperbanyak konsultasi dan permintaan konfirmasi tentang ketepatan progress pekerjaan**

   Untuk mengatasi kendala terbatasnya akses ke *website* referensi, langkah solusi yang dilakukan adalah membuat tampilan dashboard semirip mungkin dengan versi aslinya berdasarkan informasi yang didapat, seperti dari contoh desain sebelumnya atau hasil diskusi dengan *supervisor*. Komunikasi dengan *supervisor* dilakukan secara rutin untuk meminta masukan dan memastikan bahwa hasil dashboard sudah sesuai dengan kebutuhan instansi, meskipun tidak dapat melihat langsung sistem produksinya.