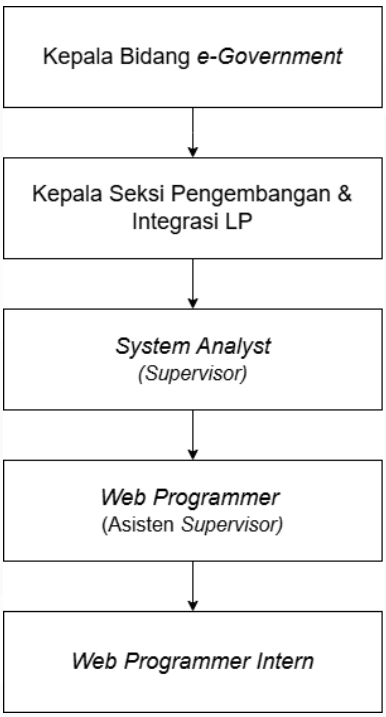
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pelaksanaan kerja magang dilakukan di Dinas Komunikasi dan Informatika Kota Tangerang, khususnya di bidang *e-Government* di bagian pengembangan dan integrasi layanan publik. Secara umum, Dinas Komunikasi dan Informatika Kota Tangerang bertugas untuk memastikan bahwa seluruh sistem digital dan teknologi yang digunakan di instansi pemerintahan bisa berjalan dengan baik, efisien, dan bisa mendukung pelayanan publik yang transparan serta mudah diakses oleh masyarakat. Bidang *e-Government* sendiri dalam Dinas Komunikasi dan Informatika Kota Tangerang, berfokus untuk mengembangkan dan mengintegrasikan berbagai aplikasi untuk membantu dalam mengelola pemerintahan dan meningkatkan layanan publik secara digital. Tujuannya adalah untuk menciptakan sistem pemerintahan yang lebih modern. Melalui penggunaan aplikasi, informasi dan layanan publik bisa diakses masyarakat secara online kapan saja dan di mana saja. Dalam menjalani program magang, posisi yang dijalani adalah sebagai *Web Programmer*, di mana fokus utama berkaitan dengan pengembangan aplikasi berbasis *website*. Salah satu proyek utama yang dikerjakan selama masa magang adalah proyek analisis data *stunting*. Dalam proyek ini, dilakukan proses pengumpulan, pengolahan, dan analisis data yang berkaitan dengan kondisi *stunting* di wilayah Kota Tangerang. Setelah proses analisis data selesai, hasilnya kemudian disajikan dalam bentuk *dashboard* interaktif. *Dashboard* ini merupakan tampilan visual yang menyajikan data dan informasi secara ringkas, menarik, dan mudah dipahami. *Dashboard* ini bertujuan agar informasi tentang *stunting* mudah dipahami, baik oleh pihak internal pemerintah maupun masyarakat umum Kota Tangerang. Dengan begitu, pemerintah dapat menyampaikan data secara lebih transparan, dan masyarakat juga bisa lebih sadar akan pentingnya pencegahan *stunting*.

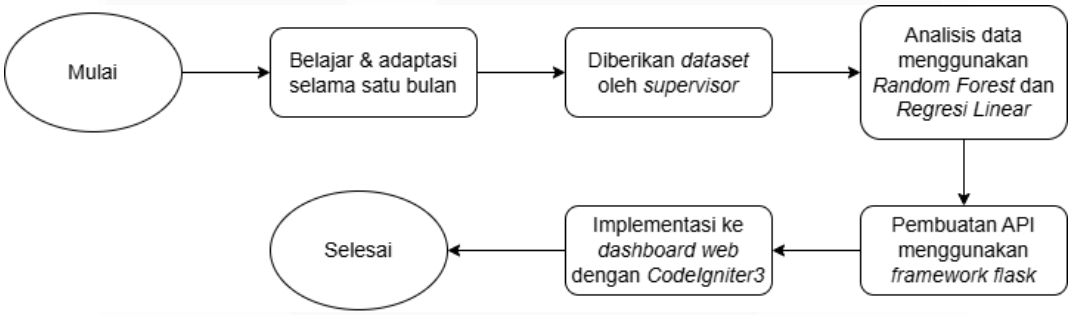


Gambar 3.1 Struktur Koordinasi selama Proses Magang

(Sumber: Divisi Layanan Publik Dinas Komunikasi dan Informatika Kota Tangerang)

Gambar 3.1 menunjukkan struktur koordinasi selama proses pelaksanaan kerja magang. Kegiatan kerja dilakukan berdasarkan struktur koordinasi yang jelas dan sistematis. Struktur ini sangat penting agar setiap pekerjaan bisa berjalan sesuai arahan, terorganisir, dan bisa memberikan hasil yang maksimal. Struktur koordinasi dimulai dari Kepala Bidang *e-Government*, yang memiliki tanggung jawab dalam mengawasi dan memastikan semua program dan proyek dalam bidang *e-Government* berjalan sesuai dengan visi dan misi dinas. Selanjutnya adalah Kepala Seksi Pengembangan dan Integrasi Layanan Publik yang bertugas untuk mengatur jalannya proyek-proyek dan memastikan bahwa pengembangan aplikasi serta sistem digital sudah sesuai dengan kebutuhan layanan publik. Kepala Seksi tidak hanya memantau secara umum, tetapi juga terkadang secara langsung mengecek hasil pekerjaan magang, memberikan *feedback,* serta menanyakan *progress* proyek yang sedang dikerjakan. Selanjutnya adalah *supervisor* dengan peran sebagai *System Analyst. Supervisor* bertugas untuk memberikan arahan mengenai tugas yang sedang dikerjakan dan memastikan bahwa tugas yang dilakukan sudah sesuai

dengan kebutuhan instansi. Koordinasi dengan *supervisor* dilakukan secara harian, baik melalui diskusi langsung maupun diskusi melalui *WhatsApp* saat *work from home*. Selain itu, terdapat juga *Web Programmer* sebagai asisten *supervisor*, yang mendampingi dan membantu dalam menjelaskan detail tugas. Jika *supervisor* sedang sibuk atau mengurus hal lain, asisten *supervisor* ini menjadi tempat pertama untuk berdiskusi mengenai kendala teknis atau proses kerja. Terakhir, posisi *Web Programmer Intern* yang bertugas untuk mengerjakan proyek-proyek yang sudah diberikan dengan bimbingan dan arahan dari *supervisor* maupun asisten *supervisor*. Dengan adanya koordinasi yang jelas, proses kerja magang menjadi lebih terarah.



Gambar 3.2 Alur Kerja Magang

(Sumber: Divisi Layanan Publik Dinas Komunikasi dan Informatika Kota Tangerang)

Gambar 3.2 menunjukkan alur kerja magang di Dinas Komunikasi dan Informatika Kota Tangerang sebagai *Web Programmer* di bidang *e-Government*. Pada bulan pertama, fokusnya adalah belajar dan beradaptasi dengan sistem yang digunakan. Setelah proses adaptasi, *supervisor* memberikan dataset yang berisi data *stunting*. Data tersebut kemudian dianalisis menggunakan bahasa pemrograman Python, dengan bantuan model *Random Forest* dan Regresi Linear. Setelah proses analisis data selesai, hasilnya diubah menjadi API *(Application Programming Interface)* dengan menggunakan *framework flask*. Kemudian, diintegrasikan ke dalam *dashboard web* dengan menggunakan *framework CodeIgniter* 3, agar data yang sudah diproses bisa ditampilkan dalam bentuk visual yang lebih mudah dipahami. Setelah semua proses selesai dan mendapat persetujuan dari supervisor, maka tugas dinyatakan selesai.

3.2 Tugas dan Uraian Kerja Magang

Pelaksanaan kerja magang melibatkan berbagai tugas yang berhubungan dengan pengembangan sistem di bidang *e-Government* pada Dinas Komunikasi dan Informatika Kota Tangerang. Setiap tugas yang diberikan bertujuan untuk membantu meningkatkan layanan digital di pemerintahan. Pekerjaan yang dilakukan mengikuti tahapan yang sudah ditentukan. Realisasi dari kerja magang di Dinas Komunikasi dan Informatika Kota Tangerang pada bidang *e-Government* dapat dilihat pada Tabel 3.1.

Tabel 3.1 Timeline Realisasi Kerja Magang pada Dinas Komunikasi dan Informatika Kota Tangerang

No.	Aktivitas	Mulai	Selesai	Koordinasi	Divisi
1	**Melakukan pengembangan aplikasi berbasis *web* menggunakan bahasa pemrograman PHP dengan *framework CodeIgniter* dan database MySQL**				
1.1	Belajar membuat *website* CRUD dengan *CodeIgniter* 3.	03/02/25	07/02/25	*Supervisor*	LP (Layanan Publik)
1.2	Melakukan pemasangan template b*ootstrap* dengan fitur *login, logout,* membuat halaman baru dengan menampilkan data tabel, menambahkan CSRF token, dan menambahkan fitur detail dengan modal.	10/02/25	14/02/25	*Supervisor*	LP (Layanan Publik)
2	**Mengidentifikasi serta menganalisis permasalahan dalam proyek yang sedang berjalan**				
2.1	Menganalisis data *stunting* dengan melakukan *business understanding,* data *understanding* dan data *preparation*.	03/03/25	22/04/25	*Supervisor*	LP (Layanan Publik)
3	**Menyusun solusi yang efektif untuk mengoptimalkan kinerja dan fungsionalitas aplikasi**				
3.1	Melakukan modeling sekaligus pelatihan dan evaluasi untuk algoritma *Random Forest.*	10/03/25	28/04/25	*Supervisor*	LP (Layanan Publik)
3.2	Melakukan modeling sekaligus pelatihan dan evaluasi untuk algoritma Regresi Linear.	11/03/25	28/04/25	*Supervisor*	LP (Layanan Publik)

No.	Aktivitas	Mulai	Selesai	Koordinasi	Divisi
3.3	Pembuatan API *(Application Programming*	29/04/25	16/05/25	*Supervisor*	LP (Layanan Publik)
3.4	Pemasangan *template Bootstrap dashboard* pada *CodeIgniter3.*	19/04/25	19/05/25	*Supervisor*	LP (Layanan Publik)
3.5	Mengintegrasikan data dalam bentuk ringkasan *(deployment)*	22/05/25	23/05/25	*Supervisor*	LP (Layanan Publik)
3.6	Membuat visualisasi yang menampilkan analisis per wilayah.	26/05/25	28/05/25	*Supervisor*	LP (Layanan Publik)
3.7	Melanjutkan dengan menampilkan faktor risiko.	02/06/25	04/06/25	*Supervisor*	LP (Layanan Publik)
3.8	Melanjutkan dengan menampilkan profil anak.	05/06/25	11/06/25	*Supervisor*	LP (Layanan Publik)
3.9	Melanjutkan dengan menampilkan analisis kinerja fasilitas kesehatan	12/06/25	13/06/25	*Supervisor*	LP (Layanan Publik)
4	**Melaksanakan pengujian sistem, termasuk User Acceptance Testing (UAT) dan System Integration Testing (SIT) untuk memastikan kualitas dan keandalan aplikasi**				
4.1	Melakukan pengujian fitur aplikasi oleh pengguna.	16/06/25	17/06/25	*Supervisor*	LP (Layanan Publik)
5	**Mematuhi jadwal kerja yang telah disepakati dan hadir di kantor sesuai dengan ketentuan yang berlaku**				
5.1	Menunjukkan disiplin dengan mematuhi jadwal kerja dan kehadiran di kantor.	03/02/25	30/06/25	*Supervisor*	LP (Layanan Publik)

(Sumber: Divisi Layanan Publik Dinas Komunikasi dan Informatika Kota Tangerang)

Proses pelaksanaan kerja magang menggunakan beberapa *tools*. *Tools* tersebut untuk mendukung tugas yang berhubungan dengan analisis data dan pengembangan *website.* Penggunaan *tools* ini membantu agar proses kerja menjadi lebih cepat dan rapi. Berikut merupakan *tools* yang digunakan selama program magang berlangsung:

1) Jupyter Notebook.

Jupyter Notebook merupakan sebuah aplikasi berbasis *website* yang digunakan untuk menulis dan menjalankan kode program [16]. Alat ini sangat

berguna karena bisa menampilkan hasilnya pada satu tempat yang sama. Dalam pelaksanaan tugas magang, *Jupyter Notebook* digunakan untuk proses pengolahan data menggunakan bahasa pemrograman *Python* [17].

2) Visual Studio Code.

Visual Studio Code merupakan *code* editor atau alat yang digunakan untuk menulis dan mengedit kode program [18]. Salah satu kelebihan *Visual Studio Code* adalah karena bisa dijalankan di berbagai sistem operasi seperti Windows, Mac, maupun Linux. Dengan menggunakan *Visual Studio Code*, proses magang akan menjadi lebih terstruktur.

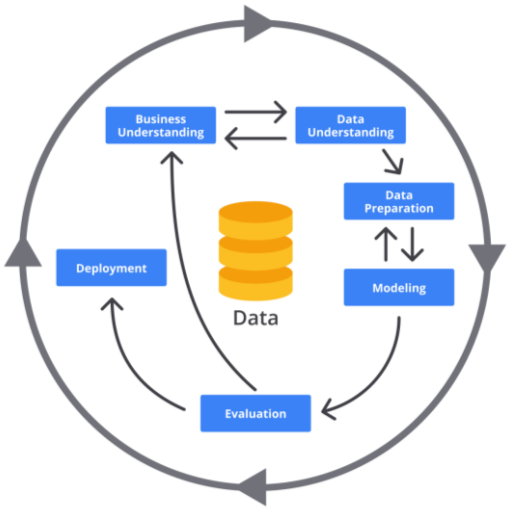
3) Postman.

Postman adalah sebuah aplikasi yang digunakan untuk menguji API *(Application Programming Interface).* API merupakan jembatan yang menghubungkan antara satu aplikasi dengan aplikasi lainnya. Dengan menggunakan Postman, bisa mengirim permintaan ke server lalu melihat respon yang diberikan oleh server tersebut [19].

4) Laragon.

Laragon adalah aplikasi yang bisa membantu membuat *website* ataupun aplikasi. Aplikasi ini bekerja seperti XAMPP [20]. Aplikasi ini sangat membantu karena semua kebutuhan utama seperti *web server*, *database*, dan PHP sudah tersedia di dalamnya.

Proses pelaksanaan kerja magang menggunakan *framework* CRISP-DM *(Cross Industry Standard Process for Data Mining*). CRISP – DM memberikan struktur kerja menjadi tahapan yang jelas. Dengan mengikuti tahapan CRISP-DM, proses analisis data selama magang menjadi lebih efisien dalam pengambilan keputusan.



Gambar 3.3 Tahapan framework CRISP – DM [12]

Gambar 3.3 menunjukkan tahapan dari CRISP-DM. CRISP – DM terdiri dari enam tahapan yang saling berhubungan. Setiap tahapan memiliki peran yang penting untuk memastikan bahwa proses analisis data berjalan dengan lancar. Tahap pertama adalah *Business Understanding*. Tahap ini bertujuan untuk memahami tujuan utama dari proyek atau masalah yang ingin diselesaikan [21]. Setelah mengetahui tujuan proyek, tahap selanjutnya adalah *Data Understanding*. Tahap ini adalah tahap untuk memahami data seperti mengenali dan memahami isi data [21]. Selanjutnya adalah tahap *Data Preparation*. Tahap ini adalah tahap untuk menyiapkan data [21]. Tahap ini melibatkan pembersihan data dan dipilih bagian pentingnya supaya siap diproses lebih lanjut. Setelah itu, tahap *Modeling*. Pada tahap ini dibangun dua model *machine learning* yaitu *Random Forest* dan *Regresi Linear* untuk mengklasifikasikan risiko *stunting*. Kedua model ini nantinya akan dibandingkan untuk melihat mana yang paling akurat. Setelah model dibuat, tahapan selanjutnya adalah *Evaluation*. Tahapan ini adalah tahap yang digunakan untuk mengevaluasai performa dari model yang sudah dibuat [21]. Tahapan terakhir adalah *Deployment*, model yang paling baik akan dimasukkan ke dalam sistem *dashboard* berbasis *website*. Hasil dari model akan ditampilkan secara visual agar mudah dipahami. Berikut ini merupakan penjelasan rinci mengenai tugas dan uraian kerja magang:

3.2.1 Melakukan Pengembangan Aplikasi Berbasis Web Menggunakan Bahasa Pemrograman PHP dengan Framework CodeIgniter dan Database MySQL

Tahapan awal kegiatan magang difokuskan pada proses belajar pengembangan aplikasi berbasis *website* menggunakan *framework CodeIgniter* 3. Kegiatan ini berlangsung selama satu bulan, dimulai dari tanggal 3 Februari 2025 hingga 28 Februari 2025. Fokus utamanya adalah untuk memahami struktur dasar aplikasi *website.*

3.2.1.1 Belajar Membuat Website CRUD dengan CodeIgniter 3

Langkah pertama dalam pengembangan aplikasi menggunakan *CodeIgniter 3* dimulai dengan pengaturan awal pada beberapa file konfigurasi. Pengaturan ini penting. Tujuannya agar sistem bisa berjalan dengan lancar.

```
1  $config['base_url'] = 'http://localhost/toko-sederhana/';
```

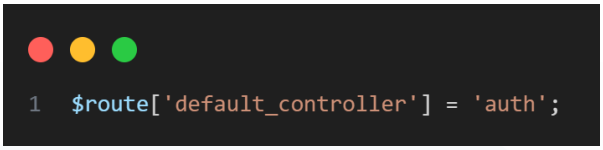
Gambar 3.4 Kode *config*.php

Gambar 3.4 merupakan kode *config.php. Base_url* menunjukkan lokasi aplikasi pada server lokal (Laragon). Alamat ini tentunya harus disesuaikan dengan nama folder tempat project *CodeIgniter* disimpan.

```
1  'hostname' => 'localhost',
2  'username' => 'root',
3  'password' => '',
4  'database' => 'tugass',
```

Gambar 3.5 Kode *database*.php

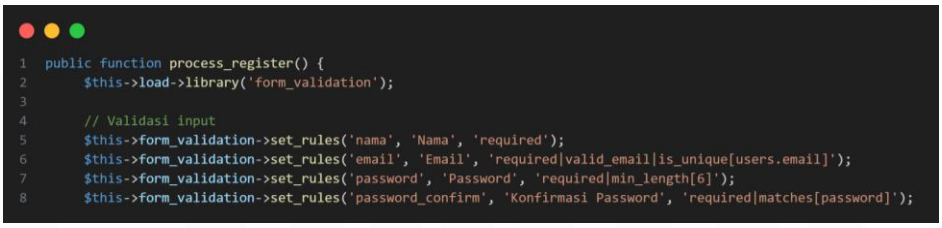
Gambar 3.5 merupakan kode *database.php.* File *database.php* digunakan untuk mengatur koneksi antara aplikasi dan *database.* Pengaturan ini penting karena jika tidak ada koneksi ke *database,* aplikasi tidak akan bisa menyimpan atau mengambil data.



Gambar 3.6 Kode *routes*.php

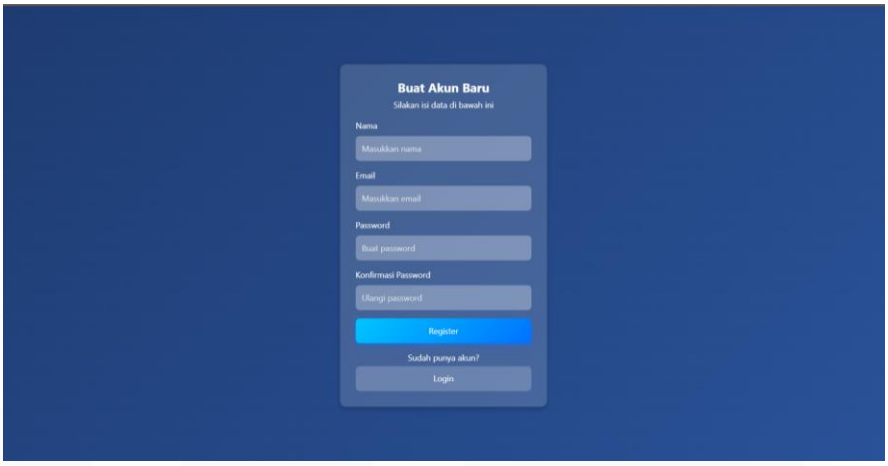
Gambar 3.6 merupakan kode *routes.php*. File *routes.php* digunakan untuk mengatur arah atau sebagai jalur masuk setiap kali pengguna membuka halaman di aplikasi. File ini menentukan *controller* mana yang akan dijalankan pertama kali saat aplikasi dibuka.

Tahapan selanjutnya adalah masuk ke *test case* pertama, yaitu membuat sebuah website sederhana dengan fitur CRUD *(Create, Read, Update, Delete)* menggunakan *CodeIgniter* 3. Pembuatan website CRUD ini merupakan langkah dasar yang penting, karena berfungsi untuk mengelola data dalam sistem. Tahap ini merupakan aktivitas dasar yang memungkinkan pengguna untuk menambahkan, melihat, memperbarui, dan menghapus data yang tidak diperlukan. Langkah awal yang dilakukan adalah *register,* yaitu proses pendaftaran akun agar pengguna dapat mengakses sistem dengan identitas masing-masing.



Gambar 3.7 Potongan Kode *Register* pada *controller*/*Auth*.php

Gambar 3.7 merupakan potongan kode *Register* pada *controller/Auth.php. Controller register* berfungsi untuk menangani proses pendaftaran akun baru oleh pengguna. Saat pengguna mengisi dan mengirimkan form, fungsi *process_register* akan melakukan validasi input seperti nama, *email, password,* dan konfirmasi *password.* Validasi ini digunakan untuk memastikan bahwa semua data telah diisi dengan benar. Setelah itu, pengguna akan diarahkan ke halaman *login* dengan pesan berhasil.



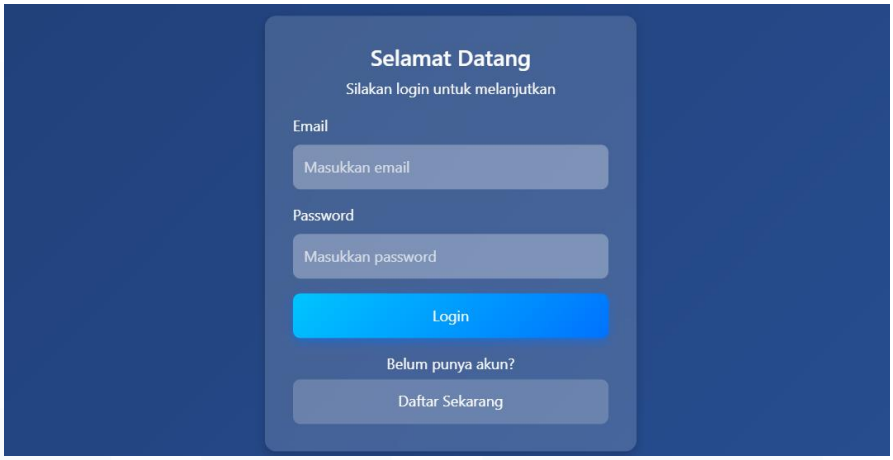
Gambar 3.8 Tampilan *Register*

Gambar 3.8 merupakan tampilan *register.* Tampilan ini merupakan halaman yang akan digunakan pengguna untuk membuat akun baru. Di dalamnya terdapat form yang harus diisi yaitu, nama, *email, password,* dan konfirmasi *password.* Setelah semua data diisi, pengguna tinggal menekan tombol "*Register"* untuk menyimpan data ke sistem. Jika sudah melakukan pendaftaran, pengguna bisa menekan tombol "*Login"*.

```
1  public function login() {
2      $this->load->view('login');
3  }
4
5  // Proses Login
6  public function cek_login() {
7      $email = $this->input->post("email");
8      $password = $this->input->post("password");
9
```

Gambar 3.9 Potongan Kode *Login* pada *controller*/*Auth*.php

Gambar 3.9 merupakan potongan kode *login* pada *controller/Auth.php.* Pada bagian ini, terdapat dua fungsi utama yang berperan penting dalam proses autentikasi pengguna. Pertama adalah fungsi *login*, yang digunakan untuk menampilkan halaman *login*. Halaman ini berisi *form* yang harus diisi oleh pengguna, seperti email dan *password*. Selanjutnya, terdapat fungsi cek_*login* yang bertugas untuk memproses data yang dikirimkan dari *form login*. Fungsi ini akan mengambil inputan dari pengguna, yaitu email dan *password*, lalu mencocokkannya dengan data yang ada di *database*.



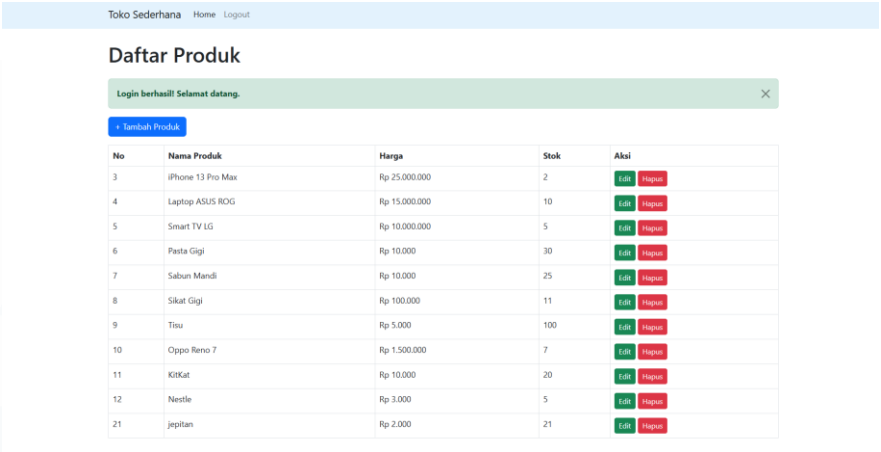
Gambar 3.10 Tampilan *Login*

Gambar 3.10 merupakan tampilan *login* pada *test case* pertama. Pengguna diminta untuk mengisi *email* dan *password,* lalu menekan tombol "*Login"* untuk masuk ke dalam sistem. Jika *login* berhasil, pengguna akan diarahkan ke tampilan halaman produk.

```
1  // Menampilkan daftar produk
2   public function index() {
3       $data['produk'] = $this->Produk_model->get_all_produk();
4       $this->load->view('produk', $data);
5   }
```

Gambar 3.11 Potongan Kode *Product* pada *controller*/Produk

Gambar 3.11 merupakan potongan kode *product* pada *controller/Produk.* Fungsi *index* memiliki tugas utama untuk mengambil seluruh data produk yang tersedia di dalam *database*. Proses pengambilan data ini dilakukan dengan bantuan model yang bernama Produk_model, melalui method atau fungsi khusus yaitu get_all_produk(). Setelah data produk berhasil diambil dari *database*, selanjutnya data tersebut dikirimkan ke bagian *view* untuk ditampilkan kepada pengguna dalam bentuk halaman yang lebih mudah dibaca. File *view* yang digunakan untuk menampilkan data ini bernama produk.php, yang akan menampilkan daftar produk. Dengan cara ini, pengguna dapat melihat semua produk yang telah dimasukkan ke dalam sistem secara lengkap dan rapi.



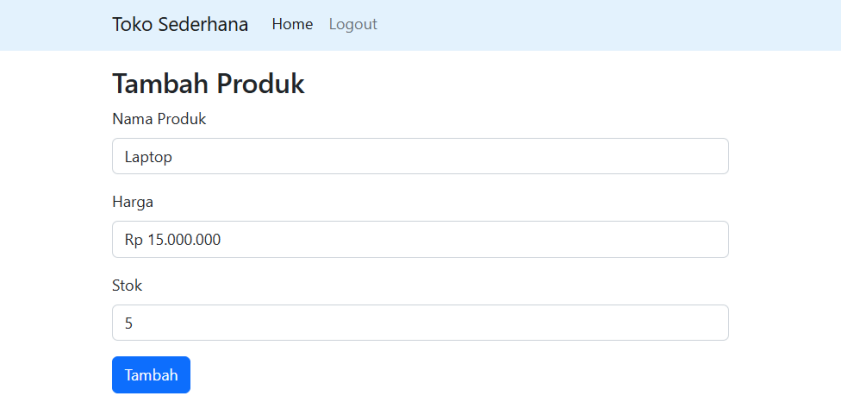
Gambar 3.12 Tampilan Daftar Produk

Gambar 3.12 merupakan tampilan daftar produk. Setelah proses *login* berhasil, pengguna akan diarahkan ke tampilan daftar produk. Tampilan ini berisi daftar produk dalam bentuk tabel. Daftar produk ini berisi no, nama produk, harga, stok, dan aksi yang mempunyai dua tombol yaitu tombol edit dan tombol hapus.

```
// Menyimpan produk baru dengan validasi
 public function store() {
     // Atur aturan validasi
     $this->form_validation->set_rules('nama', 'Nama Produk', 'required|min_length[3]');
     $this->form_validation->set_rules('harga', 'Harga', 'required|numeric');
     $this->form_validation->set_rules('stok', 'Stok', 'required|integer');

```

Gambar 3.13 Potongan Kode Tambah Produk pada *controllers*/Produk

Gambar 3.13 merupakan potongan kode tambah produk pada *controllers*/Produk. Fungsi *store* bertanggung jawab untuk menerima dan memproses data yang telah diinputkan oleh pengguna, seperti nama produk, harga, dan jumlah stok. Sebelum data disimpan ke dalam *database*, sistem terlebih dahulu melakukan validasi untuk memastikan bahwa data yang dikirim telah sesuai dengan ketentuan. Misalnya, sistem akan memeriksa apakah nama produk tidak boleh kosong, harga harus berupa angka, dan stok harus berupa bilangan bulat. Validasi ini bertujuan untuk menjaga konsistensi dan keakuratan data yang masuk ke dalam sistem. Dengan adanya fungsi ini, proses penambahan produk menjadi lebih terstruktur.



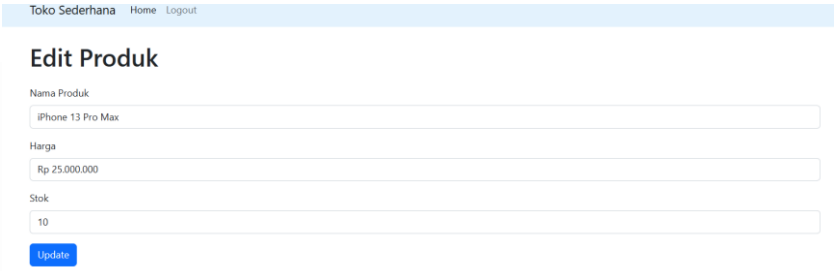
Gambar 3.14 Tampilan Tambah Produk

Gambar 3.14 merupakan tampilan tambah produk. Tampilan ini digunakan agar pengguna bisa menambahkan produk sesuai dengan yang diinginkan. Jika sudah selesai mengisi semua kolom, pengguna bisa menekan tombol “Tambah”.

```php
// Menampilkan form edit produk berdasarkan ID
public function edit($id) {
    $data['produk'] = $this->Produk_model->get_produk_by_id($id);
    if (!$data['produk']) {
        $this->session->set_flashdata('error', 'Produk tidak ditemukan');
        redirect('produk');
    }
    $this->load->view('edit_produk', $data);
}

// Memperbarui produk setelah diedit
public function update($id) {
    // Atur validasi
    $this->form_validation->set_rules('nama', 'Nama Produk', 'required|min_length[3]');
    $this->form_validation->set_rules('harga', 'Harga', 'required|numeric');
    $this->form_validation->set_rules('stok', 'Stok', 'required|integer');

```

Gambar 3.15 Potongan Kode Edit Produk pada *controllers*/Produk

Gambar 3.15 merupakan potongan kode edit produk pada *controllers/Produk.* Bagian ini berfungsi untuk menangani proses edit atau *update* data produk. Fungsi edit digunakan untuk menampilkan form edit berdasarkan id produk yang dipilih. Fungsi *update* digunakan untuk memproses data yang sudah diedit oleh pengguna. Di dalamnya terdapat validasi agar data yang dimasukkan sesuai.



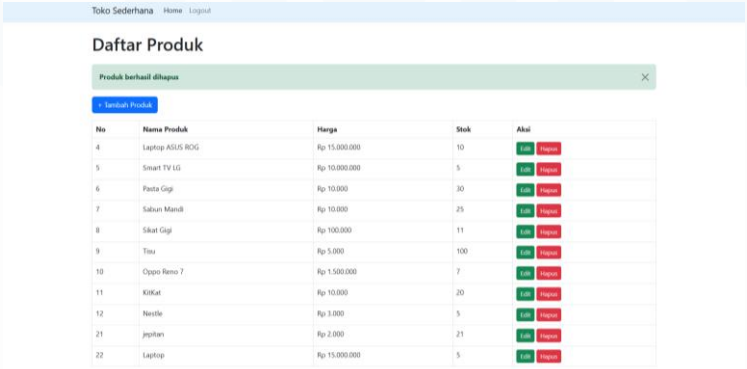
Gambar 3.16 Tampilan Edit Produk

Gambar 3.16 merupakan tampilan edit produk. Tampilan ini berisi nama produk, harga dan stok. Tampilan ini digunakan agar pengguna bisa melakukan pembaruan pada data yang sudah ada. Dengan adanya edit produk ini, misalnya jika stok bertambah, maka tidak perlu repot untuk membuat data baru, bisa menggunakan edit produk ini.

```php
public function hapus($id) {
    if ($this->Produk_model->hapus_produk($id)) {
        $this->session->set_flashdata('success', 'Produk berhasil dihapus');
    } else {
        $this->session->set_flashdata('error', 'Gagal menghapus produk');
    }
    redirect('produk');
}
```

Gambar 3.17 Potongan Kode Hapus Produk pada *controllers*/Produk

Gambar 3.17 merupakan potongan kode hapus produk pada *controllers/Produk.* Bagian ini berfungsi untuk menghapus data produk berdasarkan id nya. Setelah proses selesai, pengguna akan diarahkan kembali ke halaman daftar produk.



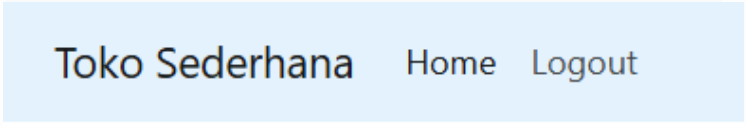
Gambar 3.18 Tampilan Ketika Produk Berhasil Dihapus

Gambar 3.18 merupakan tampilan ketika produk berhasil dihapus. Dikatakan berhasil jika muncul kotak dialog yang berisikan kalimat produk berhasil dihapus dan di daftar produknya, produk tersebut sudah tidak ada.

```php
public function logout() {
    $this->session->unset_userdata('id'); // Hapus sesi pengguna
    $this->session->set_flashdata('logout_message', 'Anda telah keluar dari sistem.');
    redirect('auth/login');
}
```

Gambar 3.19 Kode *Logout* pada *controllers*/*Auth*

Gambar 3.19 merupakan kode *logout* pada *controllers/Auth.* Fungsi ini digunakan untuk menghapus data sesi sehingga pengguna tidak lagi dianggap sedang *login*. Setelah itu, ditampilkan pesan bahwa pengguna telah keluar dan diarahkan kembali ke halaman *login* agar dapat masuk kembali jika diinginkan.



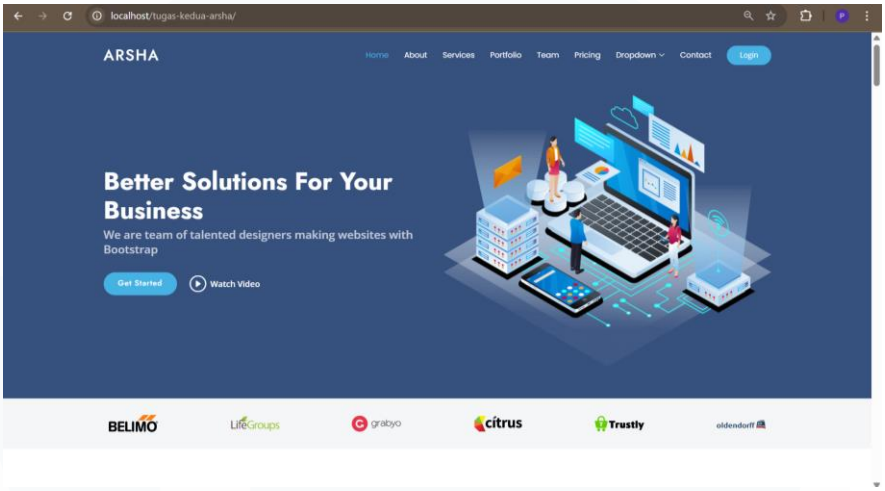
Gambar 3.20 Tampilan *Logout*

Gambar 3.20 merupakan tampilan *Logout.* Jika pengguna sudah selesai menggunakan sistem dan ingin keluar dari sistem, pengguna bisa menekan menu *Logout.* Menu ini adalah menu keluar dari sistem dan akan diarahkan kembali ke halaman awal yaitu *login.*

3.2.1.2 Melakukan Pemasangan Template Bootstrap dengan Fitur Login, Logout, Membuat Halaman Baru dengan Menampilkan Data Tabel, Menambahkan CSRF Token, dan Menambahkan Fitur Detail dengan Modal

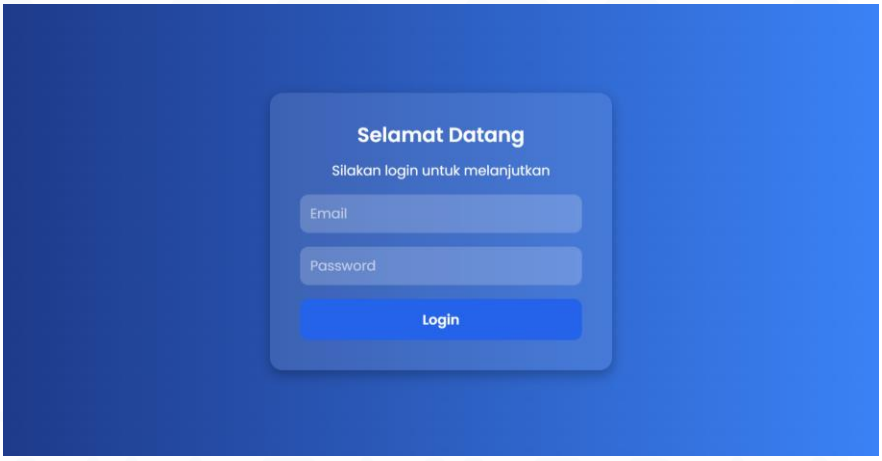
Langkah berikutnya adalah melakukan pemasangan template *bootstrap* pada *CodeIgniter 3* setelah berhasil membuat *website* CRUD di *test case* pertama. Pada tahap ini, fokus utamanya adalah pemasangan template *Bootsrap yang* digunakan untuk membuat tampilan *website* agar lebih menarik

dan rapi. *Template* ini sudah diberikan oleh *supervisor,* sehingga tidak perlu mendesain semuanya dari nol.



Gambar 3.21 Pemasangan *Template Bootsrap* pada *CodeIgniter3*

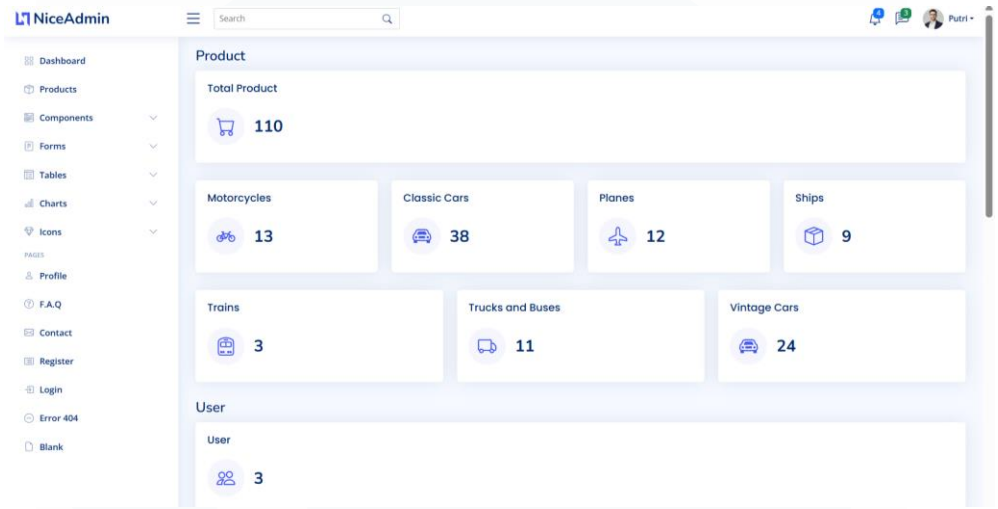
Gambar 3.21 merupakan tampilan ketika *template Bootsrap* berhasil dipasang pada *CodeIgniter 3*. Namun, pada tahap ini, semua tombol yang ada di halaman tersebut belum memiliki fungsi. Tombol-tombol tersebut hanya sekadar tampilan visual saja, dan belum terhubung dengan proses atau aksi tertentu di sistem. Satu-satunya tombol yang sudah bisa digunakan adalah tombol *"Login"*.



Gambar 3.22 Tampilan *Login* pada *Test Case* Kedua

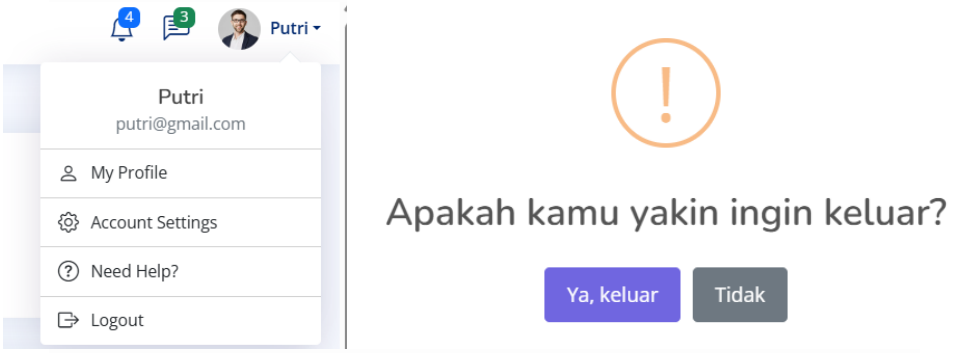
Gambar 3.22 merupakan tampilan *login* pada *test case* kedua. Tampilan ini menggunakan desain yang bersih dan modern. Pada form *login,* tedapat

inputan seperti email dan *password* yang harus diisi oleh pengguna, sebelum masuk ke *dashboard* admin.



Gambar 3.23 Tampilan *Dashboard* Admin

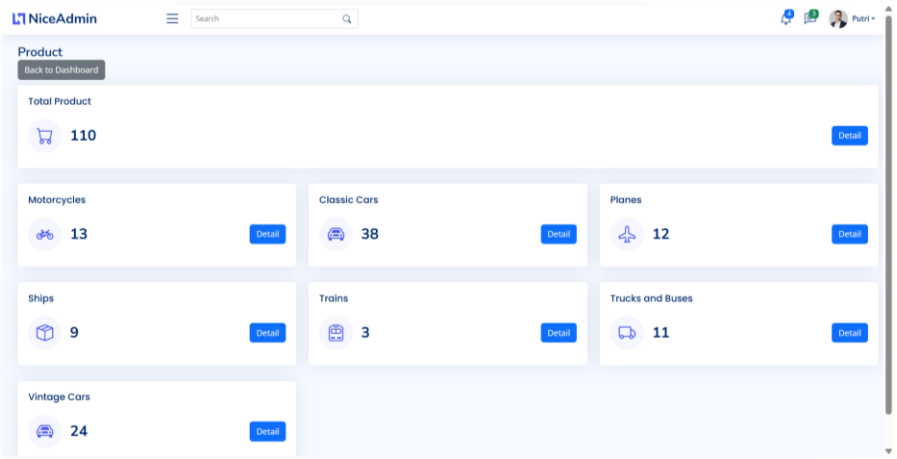
Gambar 3.23 merupakan tampilan *dashboard* admin. Tampilan ini ditampilkan saat pengguna berhasil *login.* Tampilan ini memberikan informasi tentang data-data yang ada di *database.* Akan tetapi, di bagian *sidebar,* hanya *dashboard* dan *product* saja yang bisa diklik. Tampilan ini memungkinkan pengguna untuk melihat informasi – informasi dengan mudah.



Gambar 3.24 Tampilan *Logout*

Gambar 3.24 merupakan tampilan *logout.* Tampilan ini berfungsi jika pengguna ingin mengakhiri sesi *loginnya.* Ketika pengguna menekan tombol *logout,* pengguna akan mendapatkan notifikasi, apakah pengguna tersebut benar-benar ingin melakukan *logout* atau tidak. Jika ya, maka pengguna akan

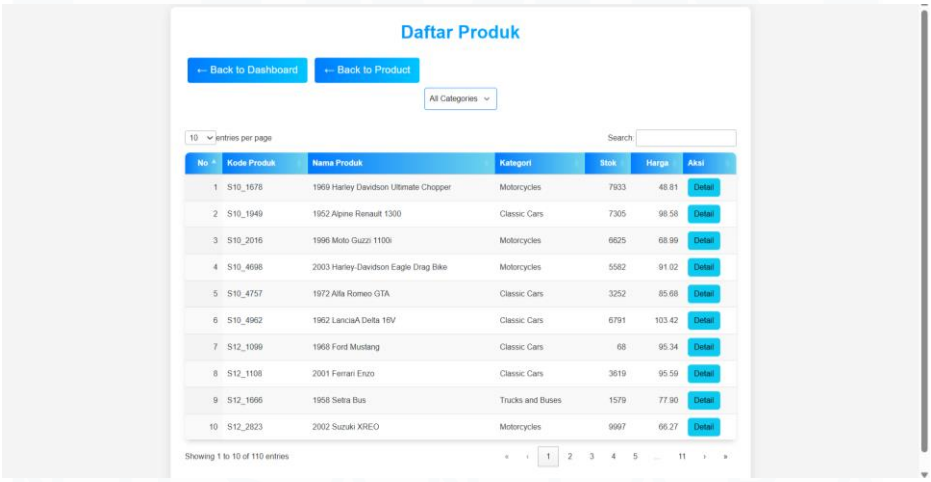
diarahkan ke halaman *login.* Jika tidak, maka pengguna akan tetap berada di halaman *dashboard* admin.



Gambar 3.25 Tampilan Ketika Menu Produk Ditekan

Gambar 3.25 merupakan tampilan ketika menu produk ditekan. Tampilan ini memberikan informasi mengenai informasi produk. Selain itu, juga diberikan tombol detail untuk melihat informasi lebih lengkapnya mengenai apa yang diklik.

Langkah selanjutnya adalah pembuatan halaman baru. Pembuatan halaman baru ini untuk menampilkan data dalam bentuk tabel. Pembuatan halaman ini bertujuan agar *website* menjadi lebih terstruktur dan mudah untuk dibaca.



Gambar 3.26 Tampilan Daftar Produk

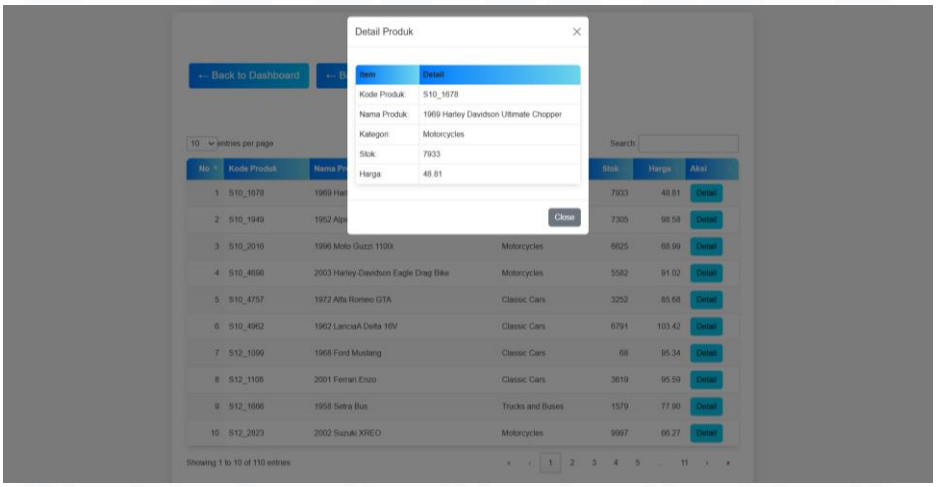
Gambar 3.26 merupakan tampilan daftar produk. Tampilan ini berisi informasi dari semua produk yang ada di *database.* Tampilan ini berisi kolom no, nama produk, harga produk, kategori, stok, harga, dan aksi dengan tombol *detail.*

Langkah selanjutnya adalah pembuatan halaman baru dengan menambahkan CSRF token dan menambahkan fitur detail dengan modal. CSRF token digunakan untuk meningkatkan keamanan saat pengguna melakukan interaksi dengan sistem. Sedangkan fitur detail menggunakan modal digunakan agar informasi bisa ditampilkan tanpa perlu membuka halaman baru.

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

Gambar 3.27 Kode CSRF *Token*

Gambar 3.27 merupakan kode CSRF *token.* CSRF *token* adalah kode unik yang digunakan untuk melindungi aplikasi dari serangan *CSRF (Cross-Site Request Forgery),* supaya data tidak bisa dikirim sembarangan oleh pihak lain. CSRF token ditandai dengan *protection* dan *regenerate* bernilai *true*.



Gambar 3.28 Tampilan Detail Produk Menggunakan Modal

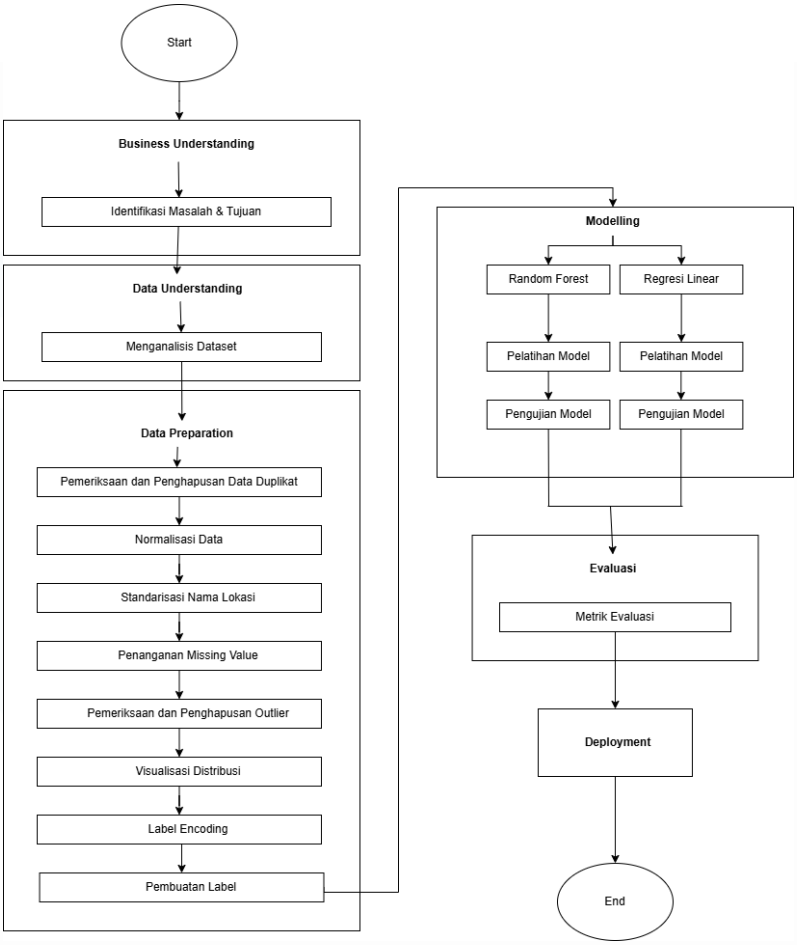
Gambar 3.28 merupakan tampilan detail produk menggunakan modal. Tampilan ini berisi kolom item dan detail, yang di mana item berisi kode produk, nama produk, kategori, stok, dan harga. Tampilan detail berisi informasi produk yang ada di *database.*

3.2.2 Mengidentifikasi serta Menganalisis Permasalahan dalam Proyek yang sedang Berjalan

Setelah menyelesaikan proses pembelajaran selama satu bulan, kegiatan magang dilanjutkan ke tahapan berikutnya. Pada tahap ini, fokus utamanya adalah mengidentifikasi dan menganalisis masalah yang terjadi dalam proyek Tujuannya adalah untuk memahami lebih dalam apa saja kendala yang dihadapi di lapangan. Tahapan ini berfokus terkait permasalahan dalam pengelolaan data *stunting.* Tahap ini penting sebagai langkah awal sebelum masuk ke proses pengembangan lebih lanjut.

3.2.2.1 Menganalisis Data Stunting dengan Melakukan Business Understanding, Data Understanding dan Data Preparation

Framework CRISP-DM *(Cross Industry Standard Process for Data Mining)* merupakan salah satu *framework* yang paling sering digunakan dalam dunia data *science* dan data *mining*. Pendekatan ini telah menjadi metode standar karena menyediakan alur kerja yang sistematis dalam proses analisis data. *Framework* ini juga memiliki tahapan yang jelas dan fokus pada tujuan analisis [22]. Setiap tahapan saling berhubungan tergantung pada kebutuhan dan hasil yang diperoleh. Kelebihan dari *framework* ini adalah kemampuannya untuk fokus pada tujuan analisis, sehingga setiap langkah yang diambil dapat diarahkan untuk menjawab permasalahan yang ingin diselesaikan melalui data. Adapun alur CRISP-DM secara keseluruhan dapat dilihat pada Gambar 3.29, yang menunjukkan bagaimana setiap tahap saling terhubung dan membentuk siklus kerja yang berkesinambungan.



Gambar 3.29 Alur CRISP-DM

Tahap pertama yang dilakukan adalah *Business Understanding.* Pada tahap ini, dilakukan pemahaman terhadap latar belakang dan tujuan dari proyek pengelolaan data *stunting* yang sedang dijalankan oleh instansi. Sebelumnya, pengelolaan data masih dilakukan secara manual menggunakan *Excel,* yang di mana hal ini menimbulkan permasalahan, yang menyebabkan proses analisis menjadi lambat dan kurang efisien. Ketika jumlah data yang dikelola semakin besar, *Excel* membutuhkan waktu lebih lama untuk membuka, menyimpan, ataupun memproses data, yang di mana hal ini bisa memakan waktu lebih lama dan lebih rentan terhadap *error.* Proses input data yang dilakukan secara manual juga bisa meningkatkan risiko kesalahan, seperti salah memasukkan angka, duplikasi data, atau salah penempatan kolom. Kesalahan seperti ini bisa mengacaukan hasil analisis.

Melihat permasalahan tersebut, dibutuhkan solusi yang efisien. Proyek ini bertujuan untuk melakukan analisis terhadap *dataset* stunting yang sudah diberikan oleh *supervisor.* Analisis data menggunakan algoritma *machine learning* yaitu *Random Forest* dan Regresi Linear, untuk melihat pola dan tren yang muncul dari data tersebut. Setelah dilakukan analisis, hasilnya dibuat menjadi API dengan *framework flask,* yang kemudian divisualisasikan dalam bentuk *dashboard website* menggunakan *CodeIgniter3.* Visualisasi ini bertujuan untuk mempermudah pihak instansi dalam memahami kondisi stunting di wilayah Kota Tangerang.

1) Dataset stunting_v1.

Setelah memahami latar belakang permasalahan serta kebutuhan instansi, kegiatan magang dilanjutkan ke tahap *Data Understanding.* Tahap ini dilakukan pada *dataset* stunting_v1. Dataset ini berisi kolom-kolom seperti berat badan anak (dalam kg), tinggi badan (dalam cm), ukuran lingkar kepala (dalam cm), usia saat pengukuran, jenis kelamin (1 untuk laki-laki, 2 untuk perempuan), dan label klasifikasi apakah anak tersebut mengalami *stunting* atau tidak (1 = *stunting*, 2 = tidak *stunting*). Untuk memahami isi dan struktur data, dilakukan analisis berikut:

a) Melihat ukuran dan informasi dataset.

Dataset ini memiliki 10.750 baris dan 7 kolom yang berisi data terkait *stunting*. Setiap kolom menyimpan jenis informasi yang berbeda, seperti berat badan, tinggi badan, usia ukur, dan lainnya, dengan tipe data yang sesuai. Selain itu, ditampilkan juga statistik dasar seperti nilai rata-rata (*mean*), standar deviasi (*std*), nilai minimum (*min*), dan maksimum (*max*) pada kolom-kolom numerik, yang dapat dilihat pada Gambar 3.30.

```
[19]: df.shape #dataset ini terdiri dari 10750 rows dan 7 columns

[19]: (10750, 7)

[21]: df.info() #melihat informasi dataset

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10750 entries, 0 to 10749
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   berat           10750 non-null  float64
 1   tinggi          10750 non-null  float64
 2   lingkar_kepala  10750 non-null  float64
 3   usia_ukur       10750 non-null  int64
 4   jenis_kelamin   10750 non-null  int64
 5   label           10750 non-null  int64
 6   tb_u            10750 non-null  object
dtypes: float64(3), int64(3), object(1)
memory usage: 588.0+ KB

[23]: df.describe() #melihat ringkasan statistik
```

[23]:

	berat	tinggi	lingkar_kepala	usia_ukur	jenis_kelamin	label
count	10750.000000	10750.000000	10750.000000	10750.000000	10750.000000	10750.000000
mean	11.977531	88.413466	56.824335	31.753488	1.447442	1.500093
std	5.132061	16.595087	637.207596	15.370818	0.497253	0.500023
min	2.750000	1.000000	4.000000	1.000000	1.000000	1.000000
25%	9.200000	77.200000	45.500000	19.000000	1.000000	1.000000
50%	12.000000	86.000000	47.500000	31.000000	1.000000	2.000000
75%	13.400000	95.000000	49.000000	45.000000	2.000000	2.000000
max	112.600000	129.600000	46552.000000	62.000000	2.000000	2.000000

Gambar 3.30 Melihat Ukuran dan Informasi *Dataset*

Selanjutnya adalah tahap *Data Preparation*. Tahap ini dilakukan pada *dataset* stunting_v1. Di tahap ini, data mulai dibersihkan, dicek satu per satu, dan dipersiapkan agar bisa langsung digunakan untuk proses modeling. Berikut adalah tahap untuk *data preparation* pada *dataset* stunting_v1:

a) Menghapus data kosong *(missing value).*

Gambar 3.31 adalah kode untuk memeriksa *missing value* dan dapat dilihat hasilnya bahwa tidak ada nilai yang hilang. *Missing value* adalah istilah yang digunakan ketika ada bagian data yang kosong atau belum terisi. Walaupun kelihatannya sepele, data kosong ini bisa menimbulkan masalah besar saat proses analisis. Data yang bersih sangat penting karena nilai yang hilang atau tidak valid dapat mempengaruhi hasil model dan membuatnya tidak akurat.

```
[41]: # Memeriksa nilai yang hilang di setiap kolom
missing_values = df.isnull().sum()
print("Nilai yang hilang sebelum penghapusan:")
print(missing_values)

Nilai yang hilang sebelum penghapusan:
berat            0
tinggi           0
lingkar_kepala   0
usia_ukur        0
jenis_kelamin    0
label            0
tb_u             0
dtype: int64

[43]: # Menghapus baris yang memiliki nilai null dan menyimpan hasilnya ke DataFrame baru
df_cleaned = df.dropna()

[45]: # Memeriksa nilai yang hilang di setiap kolom setelah penghapusan
missing_values_after = df_cleaned.isnull().sum()
print("Nilai yang hilang setelah penghapusan:")
print(missing_values_after)

Nilai yang hilang setelah penghapusan:
berat            0
tinggi           0
lingkar_kepala   0
usia_ukur        0
jenis_kelamin    0
label            0
tb_u             0
dtype: int64
```

Gambar 3.31 Memeriksa dan Menangani *Missing Value*

Proses analisis awal dilakukan menggunakan *dataset* stunting_v1. Namun, setelah analisis selesai, *supervisor* memberikan *dataset* baru yang lebih lengkap. *Dataset* stunting_v1 hanya berisi faktor-faktor fisik saja sedangkan *dataset* baru mencakup informasi tambahan seperti kecamatan, posyandu, dan puskesmas. Karena adanya perubahan struktur dan cakupan data, proses analisis harus diulang dari awal, mulai dari *data understanding*, *data preparation*, hingga *modeling*, dan *evaluasi*, agar hasil analisis sesuai dengan data yang terbaru.

2) Dataset stunting_v2.

Tahapan yang dilakukan adalah *Data Understanding* pada dataset stunting_v2. Pada tahap ini, *supervisor* memberikan *dataset* baru dengan cakupan informasi yang lebih lengkap. Dataset yang diberikan berformat .csv yang terdiri dari 42.296 *rows* dan 15 *columns*. Untuk memahami isi dan struktur data, dilakukan beberapa langkah analisis berikut:

a) Menganalisis *dataset.*

Dataset memiliki berbagai kolom yang memuat informasi penting yang berkaitan dengan anak dan status kesehatannya. Beberapa kolom yang

tersedia antara lain id, nik, nama lengkap, jenis kelamin, tanggal lahir, berat badan lahir, tinggi badan lahir, hingga data pengukuran seperti bb pengukuran, tb pengukuran, dll yang dapat dilihat pada Gambar 3.32. Data ini memberikan gambaran yang lebih lengkap dan mendetail dibandingkan *dataset* sebelumnya.

	id	no_kk	nik	nama_lengkap	jenis_kelamin	tanggal_lahir	bb_lahir	tb_lahir	nama_ortu	no_prop	...	tb_pengukuran	lila_pengukuran	bb_status
0	4731	NaN	3,67106E+13	AZURA FILZA	Perempuan	07/01/2021	2.8	49.0	a ginanjar	36.0	...	76.0	NaN	Berat Badan Normal
1	4819	NaN	3,67107E+15	FATISYAH BAHIRA	Perempuan	26/01/2021	2.7	0.0	ADE K	36.0	...	72.2	NaN	Berat Badan Normal
2	4988	NaN	3,67111E+15	SAHILA	Perempuan	24/03/2021	3.0	48.0	Rafid	36.0	...	69.0	NaN	Risiko Lebih
3	5203	NaN	3,67106E+15	OMAR NADIM SYAFIA	Laki-laki	13/01/2021	3.2	52.0	SYOPWANI	36.0	...	77.0	0.0	Berat Badan Normal
4	5326	3,67106E+15	3,67106E+15	DEAN ALTAF IRAWAN	Laki-laki	20/04/2021	3.2	50.0	muhamad irawan	36.0	...	85.0	0.0	Risiko Lebih
5	5334	3,67106E+15	3,67107E+15	MAHREEN KRASIVA AURORA	Perempuan	13/01/2021	3.5	50.0	M. RIZAL	36.0	...	72.0	0.0	Berat Badan Normal
6	5347	3,67106E+15	3,67107E+15	KHAWLA RANIA FELISHA	Perempuan	17/04/2021	3.4	51.0	M RIVAI	36.0	...	70.0	0.0	Berat Badan Normal
7	5348	3,67106E+15	3,67107E+15	FATIMAH AZZAHRA	Perempuan	10/01/2021	3.0	50.0	IRMAN SUDRAJAT	36.0	...	70.0	NaN	Berat Badan Normal
8	5349	NaN	3,67107E+15	SULISTYA SEKAR ARUM	Perempuan	22/02/2021	2.5	48.0	SAMSUDIN	36.0	...	71.0	0.0	Berat Badan Normal
9	5351	NaN	3,60118E+15	VISKA	Perempuan	27/03/2021	3.0	50.0	DEDI	36.0	...	70.0	NaN	Berat Badan

Gambar 3.32 Isi dari *Dataset* stunting_v2

Tahapan selanjutnya adalah dilakukan penghapusan kolom-kolom. Penghapusan dilakukan terhadap kolom-kolom yang dianggap tidak relevan atau tidak dibutukan untuk proses analisis lebih lanjut. Tujuannya adalah agar data menjadi lebih fokus, ringkas, dan memudahkan proses analisis. Proses penghapusan kolom dapat dilihat pada Gambar 3.33.

```
# Menghapus kolom yang tidak diinginkan
df = df.drop(columns=['id', 'no_kk', 'nik', 'nama_lengkap', 'tanggal_lahir', 'nama_ortu', 'no_prop', 'no_kab', 'no_kec', 'no_kel', 'rt', 'rw', 'alamat',
```

Gambar 3.33 Menghapus Kolom yang Tidak Diinginkan

Gambar 3.34 menunjukkan informasi umum dan ringkasan statistik dari *dataset* yang digunakan. Dari hasil informasi *dataset*, diketahui bahwa *dataset* ini memiliki 42.296 baris dan 15 kolom, dengan beberapa kolom masih memiliki data yang kosong. Analisis ini penting untuk memahami kualitas dan kondisi data sebelum dilakukan pembersihan lebih lanjut.

```
df.info() #melihat informasi dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42296 entries, 0 to 42295
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   jenis_kelamin   42295 non-null  object
 1   bb_lahir        42253 non-null  float64
 2   tb_lahir        42253 non-null  float64
 3   provinsi        42295 non-null  object
 4   kabupaten       42295 non-null  object
 5   kecamatan       42295 non-null  object
 6   kelurahan       42288 non-null  object
 7   puskesmas       42295 non-null  object
 8   posyandu        42194 non-null  object
 9   usia_saat_ukur  42291 non-null  object
 10  bb_pengukuran   42289 non-null  float64
 11  tb_pengukuran   41918 non-null  float64
 12  bb_status       42291 non-null  object
 13  tb_status       41167 non-null  object
 14  bb_tb_status    41658 non-null  object
dtypes: float64(4), object(11)
memory usage: 4.8+ MB
```

```
df.describe() #melihat ringkasan statistik
```

	bb_lahir	tb_lahir	bb_pengukuran	tb_pengukuran
count	42253.000000	42253.000000	42289.000000	41918.000000
mean	70.247848	51.250341	9.039132	70.637438
std	2804.117333	361.870887	74.259049	234.838132
min	0.000000	0.000000	0.000000	2.000000
25%	3.000000	48.000000	7.000000	65.000000
50%	3.000000	50.000000	8.000000	70.000000
75%	3.200000	50.000000	9.100000	75.000000
max	330050.000000	50896.000000	9145.000000	47018.000000

Gambar 3.34 Informasi dan Ringkasan Statistik *Dataset*

Setelah memahami isi dari dataset, tahap selanjutnya adalah tahap *Data Preparation* pada *dataset* stunting_v2. *Data Preparation* adalah salah satu tahap penting. Di tahap ini, data mulai dibersihkan, dicek satu per satu, dan dipersiapkan agar bisa langsung digunakan untuk proses modeling. Berikut adalah tahap untuk *data preparation* pada *dataset* stunting_v2:

a) Pemeriksaan dan penghapusan data duplikat.

Gambar 3.35 menunjukkan proses pemeriksaan data duplikat dalam *dataset*. Dari total 42.296 baris, ditemukan sebanyak 564 baris yang merupakan duplikat data. Keberadaan data duplikat ini bisa mempengaruhi hasil analisis, sehingga perlu dihapus agar hasil yang diperoleh lebih akurat.

```
# Cek jumlah baris sebelum hapus duplikat
print("Jumlah baris sebelum:", df.shape[0])

# Cek jumlah baris duplikat
print("Jumlah duplikat:", df.duplicated().sum())
```

```
Jumlah baris sebelum: 42296
Jumlah duplikat: 564
```

Gambar 3.35 Cek Duplikat

Gambar 3.36 menunjukkan hasil setelah proses penghapusan data duplikat dilakukan. Jumlah baris menjadi berkurang, yang awalnya 42.296 baris menjadi 41.732 baris, yang berarti 564 baris duplikat berhasil untuk

dihapus. Dengan penghapusan duplikat, *dataset* menjadi lebih bersih dan siap untuk dianalisis lebih lanjut.

```
# Menghapus duplikat
df = df.drop_duplicates()
```

```
# Cek jumlah baris setelah hapus duplikat
print("Jumlah baris setelah:", df.shape[0])

# Cek ulang apakah masih ada duplikat
print("Duplikat tersisa:", df.duplicated().sum())
```

```
Jumlah baris setelah: 41732
Duplikat tersisa: 0
```

Gambar 3.36 Penghapusan Duplikat

b) Normalisasi data.

Gambar 3.37 menunjukkan proses normalisasi data. Proses ini dilakukan dengan cara mengubah seluruh nilai teks pada beberapa kolom menjadi huruf kecil (*lowercase)*. Langkah ini penting agar semua penulisan menjadi konsisten, sehingga mempermudah dalam proses analisis data.

```
df['jenis_kelamin'] = df['jenis_kelamin'].str.lower()
df['provinsi'] = df['provinsi'].str.lower()
df['kabupaten'] = df['kabupaten'].str.lower()
df['kecamatan'] = df['kecamatan'].str.lower()
df['kelurahan'] = df['kelurahan'].str.lower()
df['puskesmas'] = df['puskesmas'].str.lower()
df['posyandu'] = df['posyandu'].str.lower()
df['usia_saat_ukur'] = df['usia_saat_ukur'].str.lower()
df['bb_status'] = df['bb_status'].str.lower()
df['tb_status'] = df['tb_status'].str.lower()
df['bb_tb_status'] = df['bb_tb_status'].str.lower()
```

Gambar 3.37 Mengubah Semua Teks Menjadi Huruf Kecil

Tabel 3.2 Contoh Hasil Normalisasi Data pada Kolom Jenis Kelamin

Sebelum	Sesudah
Perempuan	perempuan
Perempuan	perempuan
Perempuan	perempuan
Laki-laki	laki-laki
Laki-laki	laki-laki

Tabel 3.2 merupakan contoh hasil normalisasi data pada kolom jenis kelamin. Normalisasi dilakukan untuk menyamakan format penulisan agar

konsisten, seperti dari huruf kapital menjadi huruf kecil semua. Proses ini penting agar tidak terjadi perbedaan nilai akibat variasi penulisan yang seharusnya memiliki makna yang sama.

c) Standarisasi nama lokasi.

Gambar 3.38 menunjukkan proses standarisasi nama lokasi, yaitu dengan memperbaiki penulisan nama kecamatan agar lebih konsisten. Beberapa nama wilayah salah satunya nama kecamatan ditulis tanpa spasi, seperti "batuceper" yang seharusnya ditulis "batu ceper", dll. Langkah ini penting agar data lokasi tidak terbagi ke dalam kategori berbeda hanya karena perbedaan penulisan, sehingga analisis menjadi lebih akurat.

```
]: df['kecamatan'] = df['kecamatan'].replace('batuceper', 'batu ceper')
   print(df['kecamatan'].unique())
```

Gambar 3.38 Standarisasi Kecamatan & Kelurahan

Tabel 3.3 Contoh Hasil Standarisasi pada Kolom Kecamatan dan Kelurahan

Sebelum	Sesudah
batuceper	batu ceper
pasarbaru	pasar baru
koangjaya	koang jaya
nambojaya	nambo jaya
porisgaga	poris gaga

Tabel 3.3 menunjukkan contoh hasil standarisasi pada kolom kecamatan dan kelurahan. Standarisasi dilakukan untuk memisahkan kata yang sebelumnya tergabung. Hal ini bertujuan unutk meningkatkan konsistensi data.

d) Penanganan *missing value.*

Gambar 3.39 menunjukkan hasil pengecekkan terhadap data yang memiliki nilai kosong atau hilang *(missing value)* di setiap kolom. Dari hasil pengecekkan, terlihat bahwa beberapa kolom memiliki jumlah data kosong

yang cukup banyak. Mengetahui *missing value* ini sangat penting karena dapat mempengaruhi hasil analisis.

```
9]: # Memeriksa nilai yang hilang di setiap kolom
missing_values = df.isnull().sum()
print("Nilai yang hilang sebelum penghapusan:")
print(missing_values)

Nilai yang hilang sebelum penghapusan:
jenis_kelamin        1
bb_lahir            43
tb_lahir            43
provinsi             1
kabupaten            1
kecamatan            1
kelurahan            8
puskesmas            1
posyandu           100
usia_saat_ukur       2
bb_pengukuran        4
tb_pengukuran      374
bb_status            2
tb_status         1104
bb_tb_status       632
dtype: int64
```

Gambar 3.39 Pengecekkan *Missing Value*

Gambar 3.40 menunjukkan proses penghapusan baris-baris data yang memiliki nilai kosong *(missing value).* Semua baris yang mengandung nilai kosong dihapus agar *dataset* menjadi bersih dan siap dianalisis lebih lanjut. Langkah ini penting agar hasil analisis tidak keliru akibat data yang tidak lengkap.

```
]: # Menghapus baris yang memiliki nilai null dan menyimpan hasilnya ke DataFrame baru
df_cleaned = df.dropna()
```

Gambar 3.40 Menghapus *Missing Value*

Gambar 3.41 menunjukkan jumlah baris dan kolom setelah proses penghapusan *missing value.* Dataset yang awalnya berisi 41.732 setelah dihapus duplikat, kini menyisakan 40.398 baris dan 15 kolom data. Dengan melakukan pembersihan data, maka proses analisis bisa dilakukan dengan lebih akurat dan tidak terganggu oleh data yang tidak lengkap.

```
]: # Jumlah baris dan kolom setelah penghapusan missing values
print("Jumlah baris dan kolom setelah penghapusan missing values:")
print(df_cleaned.shape)

Jumlah baris dan kolom setelah penghapusan missing values:
(40398, 15)
```

Gambar 3.41 Setelah Penghapusan *Missing Value*

e) Pemeriksaan dan penghapusan *outlier*.

Gambar 3.42 menunjukkan proses pengecekkan *outlier* pada beberapa fitur numerik menggunakan metode *Z-score*. Data yang memiliki nilai *Z-Score* lebih dari 3 dianggap sebagai *outlier* karena nilainya jauh berbeda dari rata-rata. Langkah ini penting untuk memastikan kualitas data sebelum dilakukan analisis lebih lanjut.

```
[ ]: from scipy import stats

# Daftar fitur numerik yang ingin diperiksa
features = ['bb_lahir', 'tb_lahir', 'bb_pengukuran', 'tb_pengukuran']

# Menginisialisasi dictionary untuk menyimpan outlier
outliers_dict = {}

for feature in features:
    z_scores = stats.zscore(df_cleaned[feature])
    outliers = df_cleaned[abs(z_scores) > 3]  # Outlier dengan z-score lebih dari 3
    outliers_dict[feature] = outliers
    print(f"Jumlah outlier berdasarkan Z-score untuk {feature}: {len(outliers)}\n")
```

```
Jumlah outlier berdasarkan Z-score untuk bb_lahir: 3

Jumlah outlier berdasarkan Z-score untuk tb_lahir: 9

Jumlah outlier berdasarkan Z-score untuk bb_pengukuran: 10

Jumlah outlier berdasarkan Z-score untuk tb_pengukuran: 44
```

Gambar 3.42 Cek Outlier

Gambar 3.43 menunjukkan proses penghapusan *outlier* dari *dataset* yang sudah dibersihkan sebelumnya. Penghapusan dilakukan pada empat fitur numerik. Setelah dilakukan penghapusan, jumlah data berkurang yang awalnya 40.398 menajdi 40.333 baris, yang berarti 65 data *outlier* berhasil dihapus.

```
[ ]: from scipy import stats

# Menghapus outlier berdasarkan Z-score dari df_cleaned
for feature in ['bb_lahir', 'tb_lahir', 'bb_pengukuran', 'tb_pengukuran']:
    z_scores = stats.zscore(df_cleaned[feature])
    outliers = df_cleaned[abs(z_scores) > 3]  # Mengidentifikasi outlier dari df_cleaned
    df_cleaned = df_cleaned[~df_cleaned.index.isin(outliers.index)]  # Menghapus outlier dari df_cleaned

print("Jumlah data setelah menghapus outlier:", len(df_cleaned))
```

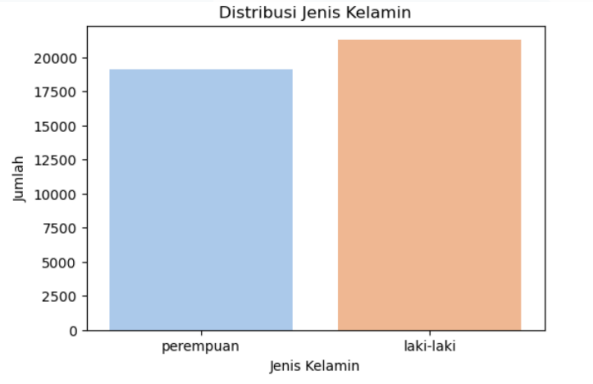
```
Jumlah data setelah menghapus outlier: 40333
```

Gambar 3.43 Setelah Menghapus Outlier

f) Visualisasi distribusi.

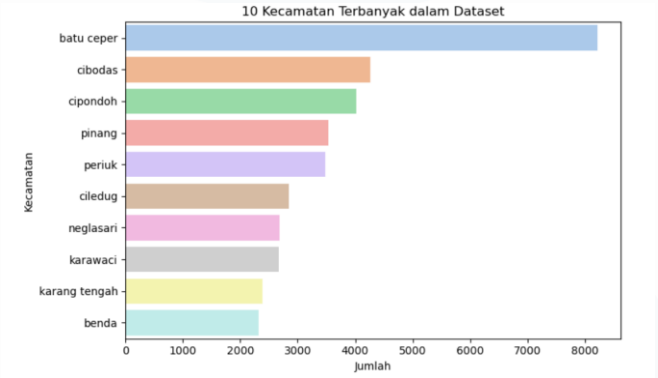
Langkah berikutnya adalah melakukan visualisasi distribusi untuk memudahkan dalam memahami data dengan lebih jelas. Gambar 3.44

menunjukkan distribusi jenis kelamin. Dari grafik yang ditampilkan, terlihat bahwa jumlah anak laki-laki sedikit lebih banyak dibandingkan dengan anak perempuan.



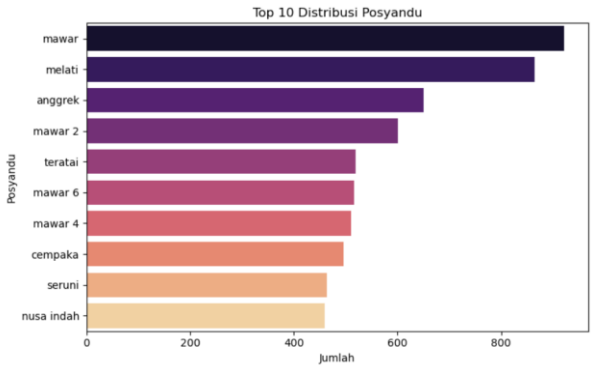
Gambar 3.44 Distribusi Jenis Kelamin

Gambar 3.45 menunjukkan 10 kecamatan terbanyak dalam *dataset*. Dari grafik, terlihat bahwa kecamatan Batu Ceper memiliki jumlah data terbanyak dibandingkan dengan kecamatan lainnya. Setelah Batu ceper, kecamatan yang juga sering muncul adalah Cibodas dan Cipondoh.



Gambar 3.45 Distribusi Kecamatan Terbanyak

Gambar 3.46 menunjukkan visualisasi distribusi posyandu terbanyak. Visualisasi menunjukkan bahwa Posyandu Mawar memiliki jumlah data paling banyak dalam *dataset*. Setelah Mawar, Posyandu Melati dan Anggrek menjadi yang terbanyak berikutnya.



Gambar 3.46 Distribusi Posyandu Terbanyak

g) Label encoding.

Tahapan selanjutnya adalah label encoding. Tahapan ini adalah tahap di mana proses mengubah data kategorikal (teks) menjadi angka agar bisa diproses oleh model *machine learning.* Kolom – kolom yang dikenai label encoding ada kolom jenis kelamin, provinsi, kabupaten, kecamatan, kelurahan, puskesmas, posyandu, usia saat ukur, bb status, tb status, dan bb tb status. Salah satunya adalah kolom jenis kelamin diubah dari "laki-laki" menjadi 1 dan "Perempuan" menjadi 2. Label encoding membantu menyederhanakan data tanpa menghilangkan makna kategorinya yang dapat dilihat pada Gambar 3.47.

```
# Ubah "Laki-Laki" menjadi 1 dan "perempuan" menjadi 2
df_cleaned['jenis_kelamin'] = df_cleaned['jenis_kelamin'].map({'laki-laki': 1, 'perempuan': 2})

# Cek hasilnya
print(df_cleaned['jenis_kelamin'].head())
```

```
0    2
1    2
2    2
3    1
4    1
Name: jenis_kelamin, dtype: int64
```

Gambar 3.47 Label Encoding Kolom Jenis Kelamin

h) Pembuatan label.

Tahapan selanjutnya adalah pembuatan label. Label ini adalah proses membuat kolom baru yang berisi kategori target yang ingin diprediksi, dalam hal ini apakah seorang anak mengalami *stunting* atau tidak berdasarkan kolom bb_tb_status. Gambar 3.48 menunjukkan pembuatan label. Nilai "gizi buruk" dan "gizi kurang" dianggap sebagai *stunting* dan diberi label 1, sedangkan

sisanya seperti “gizi baik” atau “obesitas” diberi label 2 sebagai tidak *stunting.* Proses ini penting karena akan digunakan sebagai target yang akan diprediksi dalam model *machine learning* nantinya.

```
def label_stunting(status):
    if status in ['gizi buruk', 'gizi kurang']:
        return 1  # Stunting
    else:
        return 2 # Tidak Stunting

df_cleaned['label'] = df_cleaned['bb_tb_status'].apply(label_stunting)
```

Gambar 3.48 Pembuatan Label

3.2.3 Menyusun Solusi yang Efektif untuk Mengoptimalkan Kinerja dan Fungsionalitas Aplikasi

Permasalahan yang sudah disebutkan sebelumnya, akan disusun solusi yang efektif untuk meningkatkan kinerja dan fungsionalitas aplikasi. Solusi yang dilakukan meliputi pembuatan model untuk membantu analisis data. Selain itu, solusi yang lainnya adalah evaluasi terhadap hasil model untuk memastikan akurasi, serta mengembangkan *dashboard* agar informasi dapat disajikan secara visual dan mudah dipahami.

3.2.3.1 Melakukan Modeling sekaligus Pelatihan dan Evaluasi untuk Algoritma Random Forest

Tahapan ini dilakukan pemodelan menggunakan dua algoritma yang berbeda yaitu *Random Forest* dan *Regresi Linear*. Kedua model tersebut dipilih karena kemampuan masing-masing dalam menangani masalah prediksi dengan pendekatan yang berbeda, di mana *Random Forest* lebih cocok untuk masalah klasifikasi dengan data yang kompleks, sementara *Regresi Linear* digunakan untuk memahami hubungan linier antara variabel. Pemilihan kedua model ini bertujuan untuk mengevaluasi kinerja masing-masing dalam memprediksi data *stunting* dan dibandingkan mana yang terbaik.

Pembagian data merupakan langkah penting sebelum membangun model *Machine Learning*. Tujuannya adalah untuk menguji seberapa baik

model bisa bekerja saat diberikan data baru yang belum pernah dilihat sebelumnya. Dalam proses ini, data dibagi menjadi dua bagian yaitu data *training* dan data *testing*. Data *training* (80%) digunakan untuk mengajarkan model mengenali pola dari data, sementara data *testing* (20%) digunakan untuk mengecek apakah model bisa memberikan prediksi yang akurat terhadap data baru. Pembagian data dengan rasio 80:20 banyak digunakan karena dianggap seimbang. Rasio ini memberikan cukup banyak contoh untuk proses belajar, namun tetap menyisakan data untuk mengukur keakuratan hasil. Pembagian ini juga membantu mencegah *overfitting* [23].

1) Dataset stunting_v1.

Gambar 3.49 menunjukkan pembagian data pada *dataset* stunting_v1. Pembagian data meliputi kolom berat, tinggi, lingkar_kepala, usia_ukur, dan jenis_kelamin digunakan sebagai fitur (X). Kolom label adalah target yang ingin diprediksi. Dengan cara ini, model bisa belajar hubungan antara data input (fitur) dan hasil akhirnya (label), lalu diuji apakah pemahamannya sudah cukup baik saat diberikan data *testing*.

```
[57]: # Memisahkan fitur dan target
      X = df_cleaned[['berat', 'tinggi', 'lingkar_kepala', 'usia_ukur', 'jenis_kelamin']]  # Fitur
      y = df_cleaned['label']  # Target

[58]: # Membagi data menjadi training set dan testing set (80% untuk training, 20% untuk testing)
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 3.49 Pembagian Data pada *Dataset* stunting_v1

2) Dataset stunting_v2.

Setelah proses pembersihan dan pelabelan data pada *dataset* stunting_v2 selesai dilakukan, langkah selanjutnya adalah membagi data menjadi dua bagian yaitu data latih *(training set)* dan data uji *(testing set)*. Pembagian ini bertujuan untuk melatih model *machine learning* menggunakan sebagian data (80%) dan menguji performa model menggunakan sisa data (20%). Gambar 3.50 menunjukkan pembagian data pada dataset stunting_v2. Dapat dilihat bahwa data dibagi menjadi fitur (X) yang berisi kolom-kolom

seperti jenis kelamin, bb lahir, tb lahir, usia saat ukur, bb pengukuran, dan tb pengukuran. Sedangkan target (Y) adalah kolom label.

```
# Memisahkan fitur dan target
X = df_cleaned[['jenis_kelamin', 'bb_lahir', 'tb_lahir','usia_saat_ukur', 'bb_pengukuran','tb_pengukuran']]  # Fitur
y = df_cleaned['label']  # Target

# Membagi data menjadi training set dan testing set (80% untuk training, 20% untuk testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 3.50 Pembagian Data pada *Dataset* stunting_v2

Pelatihan model *Random Forest* dilakukan dengan memberikan data pelatihan agar model bisa belajar dalam mengenali pola yang ada. Tujuannya adalah agar model bisa digunakan untuk memprediksi data baru dengan lebih akurat. Pada tahap ini, model *Random Forest* dilatih dengan menggunakan data latih yang telah dibagi.

1) Dataset stunting_v1.

Pelatihan model *Random Forest* pada *dataset* stunting_v1 dapat dilihat pada Gambar 3.51. Proses pelatihan ini dilakukan setelah tahap pembagian data selesai, agar model dapat mempelajari pola dari fitur-fitur yang tersedia. Hasil pelatihan tersebut akan digunakan untuk mengevaluasi performa model dalam melakukan prediksi pada data baru.

Pemilihan Model

```
# Melatih model menggunakan Random Forest
model = RandomForestClassifier(random_state=42)
```

Pelatihan Model

```
# Latih model menggunakan data pelatihan
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Gambar 3.51 Pelatihan Model *Random Forest Dataset* stunting_v1

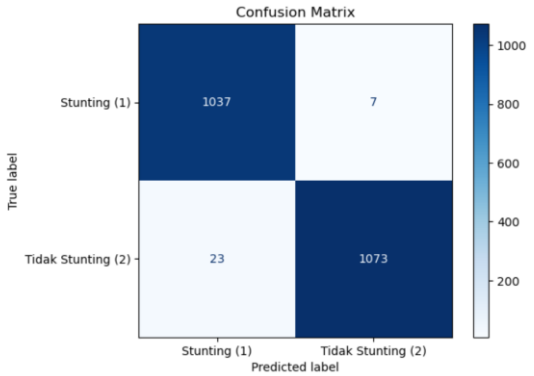
Hasil evaluasi model menunjukkan bahwa akurasi mencapai sekitar 99%, yang berarti model berhasil memprediksi *stunting* dan tidak *stunting* dengan sangat baik. Laporan klasifikasi memberikan informasi lebih detail tentang kinerja model. Untuk label "1" (*stunting*), *precision* adalah 98% dan *recall* 99%, menunjukkan bahwa model sangat baik dalam mengidentifikasi anak-anak yang mengalami *stunting* dan jarang melakukan kesalahan. Begitu

juga untuk label "2" (tidak *stunting*), *precision* adalah 99% dan *recall* 98%, menunjukkan bahwa model dapat dengan akurat mengklasifikasikan anak-anak yang tidak *stunting*. Hal ini menunjukkan bahwa model mampu memberikan prediksi yang dapat diandalkan dalam mengidentifikasi kondisi *stunting* pada anak berdasarkan data yang tersedia. Evaluasi model dapat dilihat pada Gambar 3.52.

```
Akurasi: 0.985981308411215
Laporan Klasifikasi:
              precision    recall  f1-score   support

           1       0.98      0.99      0.99      1044
           2       0.99      0.98      0.99      1096

    accuracy                           0.99      2140
   macro avg       0.99      0.99      0.99      2140
weighted avg       0.99      0.99      0.99      2140
```

Gambar 3.52 Pelatihan Model *Random Forest Dataset* stunting_v1

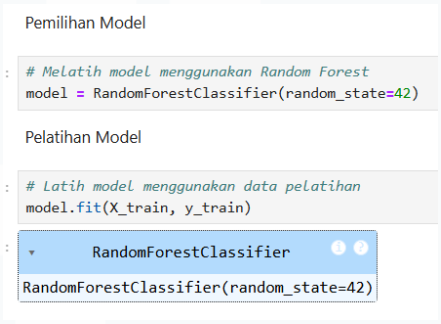
Model *Random Forest* menunjukkan hasil yang sangat baik dalam memprediksi *stunting* yang dapat dilihat pada Gambar 3.53. Model berhasil mengidentifikasi 1037 anak yang benar-benar *stunting* dan 1073 anak yang tidak *stunting* dengan tepat. Hanya ada 7 anak *stunting* yang salah diprediksi sebagai tidak *stunting*, dan 23 anak yang tidak *stunting* yang salah diprediksi sebagai *stunting*. Meskipun ada beberapa kesalahan, jumlahnya relatif kecil. Secara keseluruhan, model ini sangat efektif dalam membedakan antara anak yang mengalami *stunting* dan yang tidak. Hasil ini menunjukkan bahwa model dapat diandalkan untuk membantu dalam identifikasi *stunting*.



Gambar 3.53 *Confusion Matrix* Model *Random Forest Dataset* stunting_v1

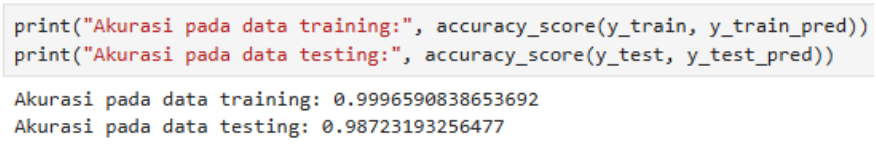
2) Dataset stunting_v2.

Tahap ini dilakukan pelatihan model menggunakan algoritma *Random Forest* pada *dataset* stunting_v2. Model dilatih menggunakan data latih yang sebelumnya telah dibagi yang dapat dilihat pada Gambar 3.54. Proses ini memungkinkan model untuk mempelajari pola dari data, dengan status *stunting.* Setelah pelatihan selesai, model siap untuk dievaluasi menggunakan data uji.



Gambar 3.54 Pelatihan Model *Random Forest* pada *Dataset* stunting v_2

Langkah selanjutnya adalah evaluasi performa model untuk mengetahui seberapa baik model dalam memprediksi kondisi *stunting* pada anak. Hasil evaluasi menunjukkan bahwa model memiliki akurasi sebesar 99.97% pada data *training* dan 98.72% pada data *testing* yang dapat dilihat pada Gambar 3.55. Nilai akurasi yang tinggi dan tidak terpaut jauh antara data *training* dan data *testing* menunjukkan bahwa model bekerja secara konsisten dna tidak mengalami *overfitting.*



Gambar 3.55 Akurasi *Training* dan *Testing*

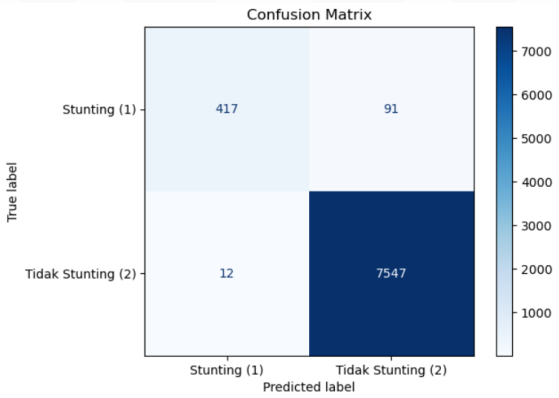
Gambar 3.56 menunjukkan hasil evaluasi model *Random Forest.* Evaluasi ini dilakukan pada dataset stunting_v2 menggunakan data *testing.* Model memiliki akurasi sebesar 98.72% dengan *precision* dan *recall* yang sangat tinggi.

Laporan Klasifikasi (Testing):

	precision	recall	f1-score	support
1	0.97	0.82	0.89	508
2	0.99	1.00	0.99	7559
accuracy			0.99	8067
macro avg	0.98	0.91	0.94	8067
weighted avg	0.99	0.99	0.99	8067

Gambar 3.56 Evaluasi Model *Random Forest Dataset* stunting_v2

Hasil evaluasi bisa direpresentasikan menggunakan *confusion matrix. Confusion matrix* membantu dalam melihat bagaimana model dapat mengklasifikasikan data berdasarkan kondisi sebenarnya dan prediksi model. Dapat dilihat pada Gambar 3.57 menunjukkan bahwa 417 kasus *True Positive* yaitu anak yang benar-benar mengalami *stunting* dan berhasil dikenali oleh model. Terdapat 91 kasus *False Negative,* yang di mana anak yang sebenarnya mengalami *stunting* tidak terdeteksi oleh model, sehingga diprediksi sebagai tidak *stunting.* Untuk kategori *False Positive,* terdapat 12 anak yang sebenarnya tidak mengalami *stunting,* namun salah diklasifikasikan oleh model sebagai *stunting.* Terakhir, sebanyak 7.547 anak termasuk dalam kategori *True Negative,* artinya anak yang tidak *stunting* berhasil diprediksi dengan benar oleh model.



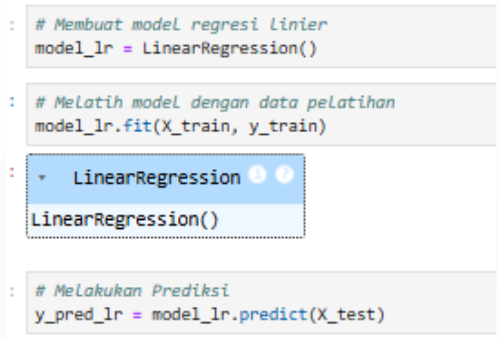
Gambar 3.57 *Confusion Matrix* Model *Random Forest Dataset* stunting_v2

3.2.3.2 Melakukan Modeling sekaligus Pelatihan dan Evaluasi untuk Algoritma Regresi Linear

Langkah selanjutnya adalah membangun model menggunakan algoritma *Regresi Linear.* Algoritma ini termasuk salah satu metode yang sering digunakan dalam analisis data karena mampu memprediksi nilai suatu variabel berdasarkan hubungan antara data yang ada. Setelah model selesai dilatih, langkah selanjutnya adalah mengujinya menggunakan data uji (*testing* data). Data uji ini tidak pernah dilihat oleh model sebelumnya, sehingga bisa digunakan untuk melihat seberapa baik model bisa memprediksi berdasarkan pengetahuan yang sudah dipelajarinya.

1) Dataset stunting_v1.

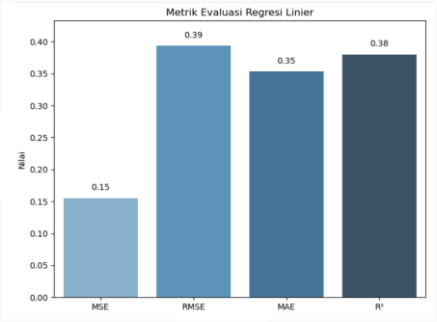
Metrik yang digunakan untuk mengevaluasi hasil prediksi antara lain *Mean Squared Error (MSE), R² Score, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE)*, dan Akurasi. Penggunaan metrik tersebut bertujuan untuk mengetahui seberapa baik model dalam memprediksi. Pelatihan model *Regresi Linear* pada *dataset* stunting_v1 dapat dilihat pada Gambar 3.58.



Gambar 3.58 Pelatihan Model Regresi Linear *Dataset* stunting_v1

Hasil evaluasi model *regresi linear* menunjukkan bahwa *Mean Squared Error* (MSE) sebesar 0.15, yang menandakan perbedaan antara nilai yang diprediksi dan nilai sebenarnya cukup kecil. R² Score yang diperoleh sebesar 0.38 menunjukkan bahwa model ini mampu menjelaskan sekitar 38% variasi data. Meskipun demikian, nilai akurasi yang mencapai 79% menunjukkan bahwa model dapat memprediksi data dengan cukup baik, meskipun hasilnya tidak sebaik model *Random Forest* yang dapat dilihat pada Gambar 3.59.

```
Akurasi : 0.794392523364486
```



Gambar 3.59 Evaluasi Model Regresi Linear *Dataset* stunting_v1

2) Dataset stunting_v2.

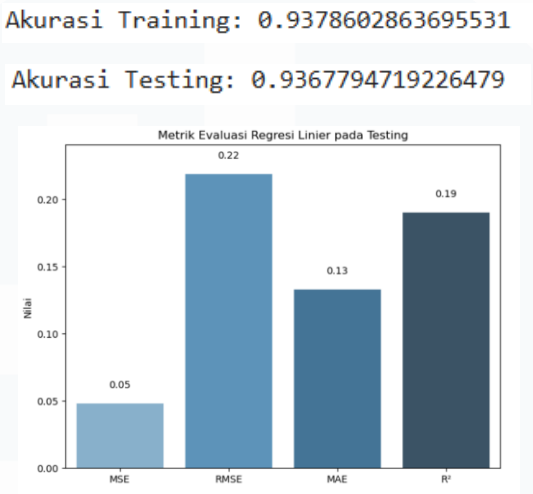
Tahapan selanjutnya adalah proses pelatihan model *regresi linear* pada *dataset* stunting_v2 seperti pada Gambar 3.60. Model ini digunakan untuk mempelajari hubungan antara fitur-fitur dengan label *stunting.* Model dilatih dengan data pelatihan agar bisa belajar dari pola yang ada.

```
188]: # Melatih model dengan data pelatihan
      model_lr.fit(X_train, y_train)

188]: LinearRegression
      LinearRegression()

190]: # Melakukan Prediksi
      y_pred_lr = model_lr.predict(X_test)
```

Gambar 3.60 Pelatihan Model Regresi Linear *Dataset* stunting_v2

Hasil evaluasi menunjukkan bahwa meskipun akurasi model mencapai 93.67%, model ini tidak dapat memprediksi kondisi *stunting* dengan baik. Angka R^2 Score yang sangat kecil hanya 19% menandakan bahwa model hanya dapat menjelaskan sebagian kecil dari hubungan antara fitur-fitur dengan hasil prediksi. Hal ini menunjukkan bahwa model regresi linear tidak efektif dalam menangkap pola dari data yang ada. Selain itu, MSE *(Mean Squared Error),* RMSE *(Root Mean Squared Error),* dan MAE (*Mean Absolute Error)* juga menunjukkan bahwa model ini masih sering memberikan prediksi yang jauh dari nilai seharusnya. Hasil evaluasi dapat dilihat pada Gambar 3.61. Akurasi pada data *testing* juga menunjukkan nilai yang hampir sama yaitu 93.78% yang

menandakan bahwa performa model konsisten mesipun tetap kurang optimal dalam melakukan prediksi stunting.



Gambar 3.61 Evaluasi Model Regresi Linear *Dataset* stunting_v2

Perbandingan antara *Random Forest* dan *Regresi Linear* menunjukkan bahwa model *Random Forest* memiliki kinerja yang lebih baik, dengan akurasi yang mencapai 98%. Di sisi lain, model *regresi linear* menunjukkan hasil yang lebih rendah dengan R^2 Score 0.19 dan akurasi sebesar 93%. Hal ini disebabkan oleh perbedaan karakteristik dari kedua algoritma dalam memproses dan memahami pola pada data. *Random Forest* adalah model klasifikasi yang memang dirancang untuk menangani target kategorikal. Sementara itu, *Regresi Linier* merupakan algoritma regresi yang lebih cocok untuk target berupa angka kontinu, bukan kelas. *Regresi linear* bekerja dengan mengasumsikan hubungan linier (garis lurus) antara variabel input (fitur) dan output (target). Dalam konteks *stunting,* hubungan antar variabel seperti usia, berat badan, tinggi badan, dan lain-lain tidak bisa djelaskan hanya dengan satu garis lurus. Sehingga, *Regresi Linear* tidak mampu untuk memahami pola-pola yang rumit ini, yang membuat prediksinya menjadi kurang akurat. Sedangkan model *Random Forest* adalah algoritma yang menggabungkan banyak pohon keputusan. Setiap pohon belajar dari bagian data yang berbeda, dan mampu untuk menangkap hubungan non-linier. Sehingga, akurasi *Random Forest* jauh lebih tinggi dibandingkan dengan *Regresi Linear.*

3.2.3.3 Pembuatan API (Application Programming Interface) dengan Flask

Proses analisis data dan pengujian beberapa model *machine learning* sudah selesai dilakukan. Dari beberapa model yang dicoba, hasil akhir yang digunakan adalah algoritma *Random Forest* pada *dataset* stunting_v2. Setelah model terbaik dipilih, langkah selanjutnya adalah pembuatan API *(Application Programming Interface).* API ini dibuat menggunakan *framework* flask. Fungsi dari API ini adalah sebagai jembatan yang menghubungkan model *machine learning* yang sudah dibuat dengan sistem lain seperti *website* atau *dashboard* yang akan dibuat [24].

Model yang telah berhasil dibuat kemudian disimpan ke dalam sebuah file bernama model_rf.pkl. Penyimpanan ini bertujuan agar model tidak perlu dilatih lagi dari awal setiap kali ingin digunakan. Setelah model disimpan dalam file model_rf.pkl, langkah selanjutnya adalah membuat file app.py untuk membangun API menggunakan *framework flask.* Dalam hal ini, app.py berfungsi untuk mengatur alur kerja API yang melibatkan interaksi antar pengguna dengan model *machine learning* yang sudah dilatih. Setiap kali program app.py dijalankan, *flask* secara otomatis akan memberikan alamat lokal untuk mengakses aplikasi, yaitu http://127.0.0.1:5000. Alamat tersebut digunakan untuk menguji dan menjalankan API secara lokal di komputer sebelum dipublikasikan ke server. Melalui URL ini, seluruh *endpoint* yang telah dibuat dapat diakses dan diuji menggunakan *browser* atau aplikasi seperti Postman. Beberapa *endpoint* sudah selesai dilakukan. Untuk *endpoint* yang pertama adalah *endpoint*/total-anak yang dapat dilihat pada Gambar 3.62. *Endpoint* ini menggunakan metode GET dan berfungsi untuk menampilkan jumlah total data anak yang terdapat dalam *dataset.* Hasilnya akan ditampilkan dalam format JSON.

```
{
  "total_anak": 42296
}
```


Gambar 3.62 *Endpoint* /total-anak

Gambar 3.63 menunjukkan *endpoint* /risiko-stunting. Endpoint /risiko-stunting menggunakan metode GET dan digunakan untuk menampilkan jumlah anak yang memiliki risiko *stunting* berdasarkan status gizi. Hasil akhir berupa jumlah total anak yang masuk kategori tersebut yaitu sebesar 2624, dan ditampilkan dalam format JSON.


```
{
  "jumlah_risiko_stunting": 2624
}
```


Gambar 3.63 *Endpoint* /risiko-stunting

Gambar 3.64 menunjukkan *endpoint* /posyandu-puskesmas-aktif. *Endpoint* ini menggunakan metode GET dan fungsinya adalah untuk menampilkan jumlah posyandu dan puskesmas yang aktif. Hasil akhirnya adalah jumlah posyandu aktif sebesar 599 dan jumlah puskesmas aktif sebesar 39. Hasil ini ditampilkan dalam format JSON.


```
{
  "jumlah_posyandu_aktif": 599,
  "jumlah_puskesmas_aktif": 39
}
```

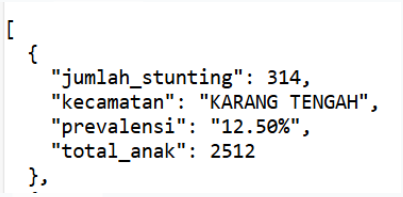

Gambar 3.64 *Endpoint*/posyandu-puskesmas-aktif

Gambar 3.65 menunjukkan *endpoint*/tren-risiko-bulanan. *Endpoint* ini menggunakan metode GET dan fungsinya adalah untuk menampilkan tren jumlah anak yang mengalami s*tunting* setiap bulan. Selain jumlah, *endpoint* ini juga menampilkan perubahan dibanding bulan sebelumnya dan memberikan status apakah kasus *stunting* sedang “naik”, “turun”, atau “tetap”.


```
[
  {
    "bulan_tahun": "2022-01",
    "jumlah_stunting": 296,
    "perubahan_dari_bulan_sebelumnya": 0,
    "status": "tetap"
  },
```


Gambar 3.65 Contoh *Endpoint* /tren-risiko-bulanan

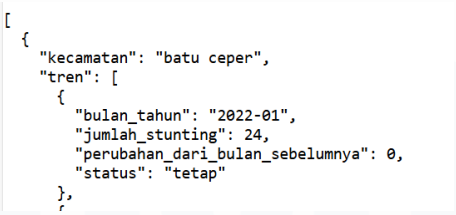
Gambar 3.66 menunjukkan *endpoint* /top10-wilayah-prevalensi. *Endpoint* ini menggunakan metode GET dan fungsinya adalah untuk menampilkan 10 kecamatan dengan persentase (prevalensi) *stunting* tertinggi. Adapun kecamatan dengan persentase tertinggi adalah Karang Tengah, Periuk, Pinang, Ciledug, Batu Ceper, Karawaci, Benda, Cibodas, Cipondoh, dan Larangan.



```
[
  {
    "jumlah_stunting": 314,
    "kecamatan": "KARANG TENGAH",
    "prevalensi": "12.50%",
    "total_anak": 2512
  },
```


Gambar 3.66 Contoh *Endpoint*/top10-wilayah-prevalensi

Gambar 3.67 menunjukkan *endpoint* /tren-stunting-per-wilayah. Endpoint ini menggunakan metode GET dan digunakan untuk menampilkan tren jumlah kasus stunting di tiap kecamatan dari bulan ke bulan. Setiap data akan menunjukkan apakah jumlah kasus naik, turun, atau tetap dibanding bulan sebelumnya, sehingga kita bisa melihat pola perkembangan *stunting* di suatu wilayah.



```
[
  {
    "kecamatan": "batu ceper",
    "tren": [
      {
        "bulan_tahun": "2022-01",
        "jumlah_stunting": 24,
        "perubahan_dari_bulan_sebelumnya": 0,
        "status": "tetap"
      },
```


Gambar 3.67 Contoh *Endpoint*/tren-stunting-per-wilayah

Gambar 3.68 menunjukkan *endpoint* /status-gizi. *Endpoint* ini menggunakan metode GET dan fungsinya adalah untuk menampilkan jumlah anak yang termasuk dalam kategori *stunting* dan tidak *stunting.* Hasilnya akan dikembalikan dalam format JSON. Dapat dilihat pada gambar bahwa status gizi *stunting* sebesar 2.624 anak dan status gizi tidak *stunting* sebesar 39.672 anak.

```
{
  "Stunting": 2624,
  "Tidak Stunting": 39672
}
```


Gambar 3.68 *Endpoint* /status-gizi

Gambar 3.69 menunjukkan *endpoint* /bb-tb-lahir-rendah. *Endpoint* ini menggunakan metode GET dan fungsinya adalah untuk mengelompokkan anak berdasarkan kondisi berat badan dan tinggi badan saat lahir. Jika berat badan kurang dari 2.500 gr atau tinggi badan kurang dari 47 cm, maka anak dikategorikan sebagai lahir dengan BB atau TB rendah. *Endpoint* ini membagi data ke dalam empat kategori yaitu Normal sebanyak 36.519, BB Rendah sebanyak 1.986, TB Rendah sebanyak 2.802, dan BB & TB Rendah sebanyak 989.


```
{
  "BB & TB Rendah": 989,
  "BB Rendah": 1986,
  "Normal": 36519,
  "TB Rendah": 2802
}
```


Gambar 3.69 *Endpoint*/bb-tb-lahir-rendah

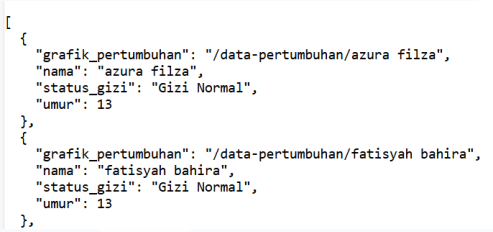
Gambar 3.70 menunjukkan *endpoint* /faktor-risiko. *Endpoint* ini menggunakan metode GET dan digunakan untuk menampilkan data rata-rata beberapa indikator kesehatan anak berdasarkan wilayah kecamatan. Data yang ditampilkan berupa rata-rata berat dan tinggi badan saat lahir dan saat pengukuran, serta usia rata-rata anak dalam bulan.


```
[
  {
    "bb_lahir": 2.89,
    "bb_pengukuran": 8.17,
    "kecamatan": "BATUCEPER",
    "tb_lahir": 47.56,
    "tb_pengukuran": 71.55,
    "usia_bulan": 10
  },
```


Gambar 3.70 *Endpoint*/faktor-risiko

Gambar 3.71 menunjukkan *endpoint* /profil-anak. *Endpoint* ini menggunakan metode GET dan digunakan untuk menampilkan informasi dasar anak-anak yang tercatat dalam data pengukuran gizi. Data yang ditampilkan

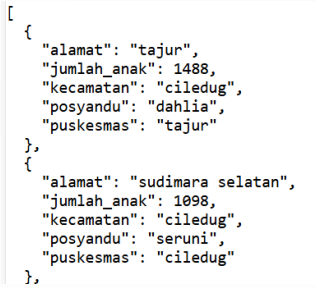
mencakup nama anak, umur saat pengukuran, status gizinya dan link menuju grafik pertumbuhan anak tersebut.



```
[
  {
    "grafik_pertumbuhan": "/data-pertumbuhan/azura filza",
    "nama": "azura filza",
    "status_gizi": "Gizi Normal",
    "umur": 13
  },
  {
    "grafik_pertumbuhan": "/data-pertumbuhan/fatisyah bahira",
    "nama": "fatisyah bahira",
    "status_gizi": "Gizi Normal",
    "umur": 13
  },
```


Gambar 3.71 Contoh *Endpoint* /profil-anak

Gambar 3.72 menunjukkan *endpoint* /fasilitas-kesehatan. *Endpoint* ini menggunakan metode GET dan digunakan untuk digunakan untuk menampilkan data fasilitas kesehatan berdasarkan wilayah. Respons dari endpoint ini berisi informasi tentang lokasi posyandu, puskesmas, dan jumlah anak yang tercatat di wilayah tersebut.

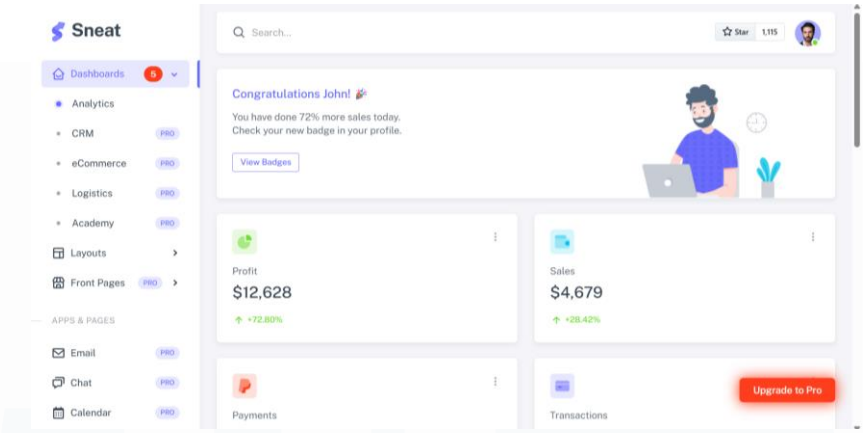


```
[
  {
    "alamat": "tajur",
    "jumlah_anak": 1488,
    "kecamatan": "ciledug",
    "posyandu": "dahlia",
    "puskesmas": "tajur"
  },
  {
    "alamat": "sudimara selatan",
    "jumlah_anak": 1098,
    "kecamatan": "ciledug",
    "posyandu": "seruni",
    "puskesmas": "ciledug"
  },
```


Gambar 3.72 Contoh *Endpoint* /fasilitas-kesehatan

3.2.3.4 Pemasangan Template Bootstrap Dashboard pada CodeIgniter3

Supervisor memberikan *template* untuk implementasi *dashboard* yang nantinya akan digunakan sebagai tampilan utama sistem. *Template* tersebut dipasang langsung ke dalam proyek CI3 agar dapat disesuaikan dengan alur dan struktur *framework* tersebut. Proses pemasangan dilakukan dengan menyalin file index.html, kemudian memisahkan bagian-bagian seperti *header, navbar,* dan *footer* untuk disesuaikan ke dalam struktur CI3. Gambar 3.73 merupakan *template dashboard* yang sudah berhasil dipasang pada CI3.



Gambar 3.73 Pemasangan *template dashboard* pada CI3

3.2.3.5 Mengintegrasikan Data dalam Bentuk Ringkasan (Deployment)

Langkah pertama pada tahap *deployment* dalam pengembangan aplikasi menggunakan *CodeIgniter 3* dimulai dengan pengaturan awal pada beberapa file konfigurasi. Pengaturan ini penting Tujuannya adalah agar sistem bisa berjalan dengan lancar.

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

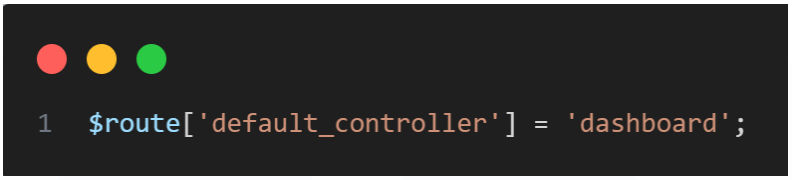
Gambar 3.74 Kode *autoload*.php

Gambar 3.74 merupakan kode *autoload.php.* Kode tersebut digunakan untuk memuat *library* tanpa harus memanggil satu per satu. *Library session* digunakan untuk menyimpan data pengguna sementara dan curl berguna untuk mengambil atau mengirim data dari dan ke API.

```
1  $config['base_url'] = 'http://localhost/project-stunting/';
```

Gambar 3.75 Kode *config*.php

Gambar 3.75 merupakan kode *config.php.* Kode ini digunakan untuk menentukan alamat dasar dari *website* yang sedang dibuat. Tahap ini penting agar sistem tahu di mana letak folder project saat dijalankan lewat *browser.*



```php
$route['default_controller'] = 'dashboard';
```

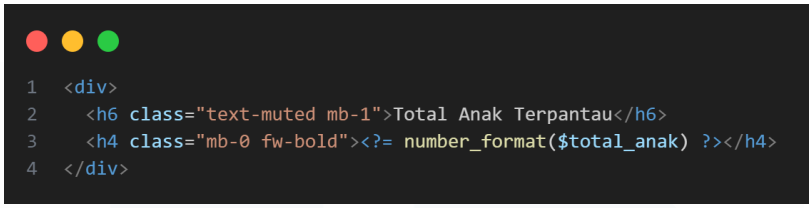
Gambar 3.76 Kode *route*.php

Gambar 3.76 merupakan kode *route.php.* Kode ini digunakan untuk menentukan *controller* mana yang pertama kali dijalankan saat *website* dibuka. Dalam hal ini, *controller dashboard* yang akan langsung ditampilkan sebagai halaman awal.

```php
public function index()
    {
        $data['total_anak'] = json_decode($this->curl->simple_get('http://127.0.0.1:5000/total-anak'))->total_anak ?? 0;
        $data['risiko_stunting'] = json_decode($this->curl->simple_get('http://127.0.0.1:5000/risiko-stunting'))->jumlah_risiko_stunting ?? 0;
        $data['posyandu_puskesmas'] = json_decode($this->curl->simple_get('http://127.0.0.1:5000/posyandu-puskesmas-aktif')) ?? (object)[];

        $response = json_decode($this->curl->simple_get('http://127.0.0.1:5000/tren-risiko-bulanan'));
        $tren_risiko = [];

        if ($response) {
            foreach ($response as $item) {
                $tren_risiko[] = [
                'bulan' => $this->format_bulan_indonesia($item->bulan_tahun),
                'jumlah' => $item->jumlah_stunting
                ];
            }
        }
```

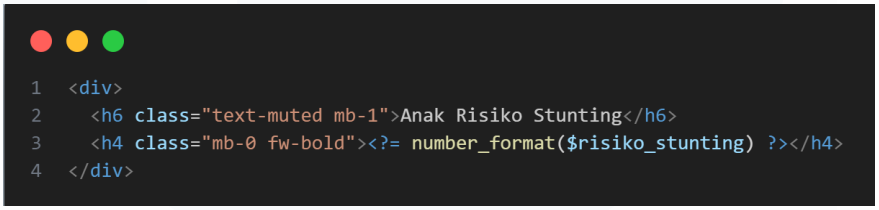
Gambar 3.77 Kode *controller/dashboard*.php

Gambar 3.77 menunjukkan kode *controller/dashboard.php.* Fungsi *index* digunakan untuk menampilkan halaman utama *dashboard* pada *website.* Sistem mengambil data dari API lokal menggunakan *library* curl, seperti total anak yang terpantau (/total-anak), jumlah anak yang berisiko *stunting* (/risiko-*stunting),* dan informasi posyandu serta puskemas yang aktif (/posyandu-puskesmas-aktif). Data yang diambil disimpan ke dalam array $data untuk ditampilkan ke *views.* Selanjutnya, untuk endpoint /tren-risiko-bulanan, diolah menggunakan foreach, di mana setiap item bulan diubah formatnya ke dalam bahasa Indonesia menggunakan fungsi format_bulan_indonesia(), lalu hasilnya disimpan ke dalam array $tren_risiko. Semua data dikumpulkan ke dalam variabel $data[‘tren_risiko’]. Setelah data siap, halaman *dashboard* akan ditampilkan dengan memuat empat bagian tampilan yaitu *header, navbar, dashboard,* dan *footer* dari folder *backend.*



Gambar 3.78 Potongan Kode *views/backend/dashboard*.php Bagian Total Anak

Gambar 3.78 menunjukkan potongan kode *views/backend/dashboard.php* bagian total anak. Kode ini digunakan untuk menampilkan informasi mengenai total anak yang terpantau di *website.* Nilai yang muncul diambil dari data yang ada di *controller* yaitu variabel $total_anak dan ditampilkan menggunakan fungsi *number_format()* agar tampil dengan format angka yang memiliki pemisah ribuan.



Gambar 3.79 Potongan Kode *views/backend/dashboard*.php Bagian Risiko *Stunting*

Gambar 3.79 menunjukkan potongan kode *views/backend/dashboard.php* bagian risiko *stunting*. Kode ini digunakan untuk menampilkan informasi mengenai jumlah anak yang teridentifikasi memiliki risiko *stunting.* Nilai yang muncul diambil dari data yang ada di *controller* yaitu variabel $risiko_stunting dan ditampilkan menggunakan fungsi *number_format()* agar tampil dengan format angka yang memiliki pemisah ribuan.

Gambar 3.80 Potongan Kode *views/backend/dashboard*.php Bagian Posyandu & Puskesmas Aktif

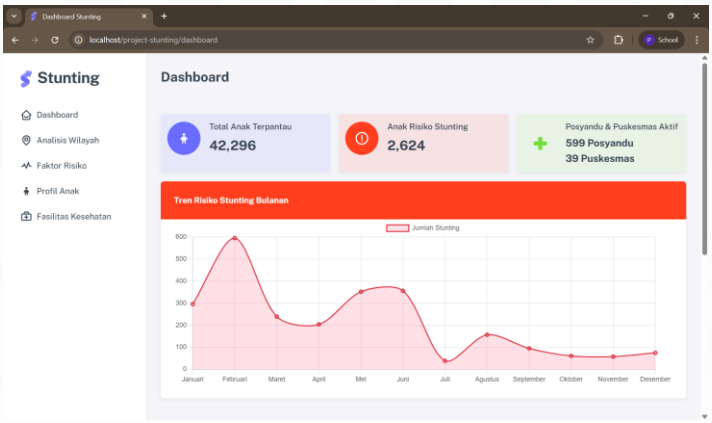
Gambar 3.80 menunjukkan potongan kode *views/backend/dashboard.php* bagian posyandu & puskesmas aktif. Kode ini

digunakan untuk menampilkan jumlah posyandu dan puskemas yang masih aktif beroperasi. Nilai data diambil dari *controller* dengan variabel $posyandu_puskesmas, kemudian ditampilkan menggunakan fungsi *number_format()* agar hasilnya mudah dibaca dengan pemisah ribuan.

```
<canvas id="trenRisikoChart" style="width: 100%; height: 100%;"></canvas>

<script>
  const ctx = document.getElementById('trenRisikoChart').getContext('2d');

  new Chart(ctx, {
    type: 'line',
    data: {
      labels: <?= json_encode(array_column($tren_risiko, 'bulan')) ?>,
      datasets: [{
        label: 'Jumlah Stunting',
        data: <?= json_encode(array_column($tren_risiko, 'jumlah')) ?>
      }]
    }
    }
  );
</script>
```

Gambar 3.81 Potongan Kode *views/backend/dashboard*.php Bagian Tren Risiko Bulanan

Gambar 3.81 menunjukkan potongan kode *views/backend/dashboard.php* bagian tren risiko bulanan. Kode ini digunakan untuk menampilkan grafik garis (*line chart)* yang merepresentasikan jumlah anak dengan risiko *stunting* setiap bulan. Elemen <canvas> berfungsi untuk menampilkan grafik. Kemudian, *library* Chart.js digunakan untuk membuat grafik berdasarkan data bulan dan jumlah kasus *stunting* yang diambil dari variabel $tren_risiko yang ada di *controller.*



Gambar 3.82 Tampilan *Dashboard Stunting*

Gambar 3.82 menunjukkan tampilan dari *dashboard stunting* yang disaijkan dalam bentuk ringkasan utama *(overview cards).* Tampilan ini menampilkan beberapa informasi penting yaitu total anak yang terpantau sebanyak 42,296, jumlah anak dengan risiko *stunting* sebanyak 2,624 anak, serta jumlah posyandu dan puskesmas aktif sebanyak 599 posyandu dan 39 puskesmas. Selain itu, ditampilkan juga tren risiko *stunting* bulanan dalam bentuk *line chart*. Dari grafik tersebut, dapat dilihat bahwa jumlah anak dengan risiko *stunting* tertinggi terjadi pada bulan Februari.

3.2.3.6 Membuat Visualisasi yang Menampilkan Analisis per Wilayah

Analisis per wilayah dibuat untuk mengetahui kondisi *stunting* secara lebih spesifik berdasarkan lokasi. Terdapat dua jenis visualisasi yang ditampilan yaitu diagram batang untuk top 10 wilayah dengan prevalensi tertinggi dan diagram garis yang menampilkan perkembangan jumlah kasus *stunting* tiap bulan per wilayah. Visualisasi ini penting untuk memahami wilayah mana yang perlu perhatian lebih.

```php
$top10_response = json_decode($this->curl->simple_get('http://127.0.0.1:5000/top10-wilayah-prevalensi'));
$top10_wilayah = $top10_response ?? [];

$labels = [];
$dataPrevalensi = [];
foreach($top10_wilayah as $wilayah) {
    $labels[] = $wilayah->kecamatan;
    // Menghapus tanda % dan mengubah ke float
    $prevalensi_bersih = floatval(str_replace('%', '', $wilayah->prevalensi));
    $dataPrevalensi[] = $prevalensi_bersih;
}
```

Gambar 3.83 Potongan Kode *controller*/analisis_wilayah.php Bagian Top 10 Wilayah dengan Prevalensi Tertinggi

Gambar 3.83 menunjukkan potongan kode *controller/analisis_wilayah.php* bagian visualisasi pertama yaitu top 10 wilayah dengan prevalensi tertinggi. Potongan kode ini digunakan untuk mengambil data dari API yang berisi 10 wilayah dengan prevalensi stuting tertinggi. Data yang diambil disimpan ke dalam variabel $top10_wilayah. Untuk nama kecamatan, dimasukkan ke dalam array $labels dan nilai prevalensinya dibersihkan dari tanda % diubah menjadi angka (float), lalu dimasukkan ke array $dataPrevalensi.

```
const labels = <?= json_encode($labels); ?>;
const dataPrevalensi = <?= json_encode($dataPrevalensi); ?>;

const ctxBar = document.getElementById('top10Chart').getContext('2d');

const top10Chart = new Chart(ctxBar, {
  type: 'bar',
  data: {
    labels: labels,
    datasets: [{
      label: 'Prevalensi Stunting (%)',
      data: dataPrevalensi,
    }]
  },
});
```

Gambar 3.84 Potongan Kode *views/backend*/analisis_wilayah.php Bagian Top 10 Wilayah Prevalensi *Stunting* Tertinggi

Gambar 3.84 menunjukkan potongan kode *views/backend/analisis_wilayah.php* yang menampilkan grafik batang untuk 10 wilayah dengan prevalensi *stunting* tertinggi. Nama wilayah diambil dari variabel *labels,* dan angka prevalensinya diambil dari dataPrevalensi. Grafik dibuat menggunakan library Chart.js dan ditampilkan di elemen dengan id top10Chart.

```
$tren_response = json_decode($this->curl->simple_get('http://127.0.0.1:5000/tren-stunting-per-wilayah'));
$tren_response = $tren_response ?? [];

$bulan_labels = [];
foreach($tren_response as $wilayah) {
    foreach($wilayah->tren as $entry) {
        if(!in_array($entry->bulan_tahun, $bulan_labels)) {
            $bulan_labels[] = $entry->bulan_tahun;
        }
    }
}
sort($bulan_labels); // Mengurutkan bulan

$tren_data = [];
foreach($tren_response as $wilayah) {
    $data_jumlah_stunting = [];
    foreach($bulan_labels as $bulan) {
        $found = false;
        foreach($wilayah->tren as $entry) {
            if($entry->bulan_tahun == $bulan) {
                $data_jumlah_stunting[] = $entry->jumlah_stunting;
                $found = true;
                break;
            }
        }
        if(!$found) {
            $data_jumlah_stunting[] = 0;
        }
    }
```

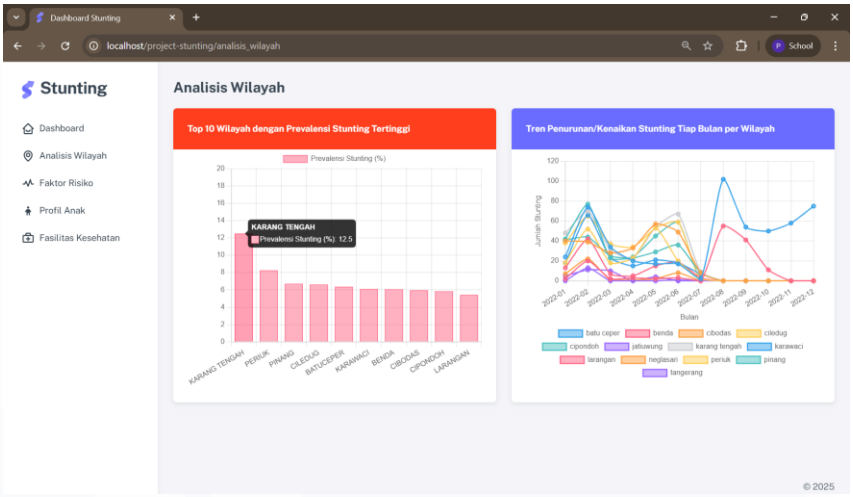
Gambar 3.85 Potongan Kode *controller*/analisis_wilayah.php Bagian Tren Penurunan/Kenaikan *Stunting* Tiap Bulan per Wilayah

Gambar 3.85 merupakan potongan kode *controller/analisis_wilayah.php* bagian tren penurunan/kenaikan stunting tiap bulan per wilayah. Kode ini mengambil data dari API tren-s*tunting*-per-wilayah menggunakan CURL, lalu hasilnya diubah ke dalam bentuk array. Data yang diambil berisi informasi jumlah *stunting* per kecamatan untuk tiap bulan. Setelah berhasil diambil, sistem akan mengumpulkan semua bulan yang muncul dari setiap kecamatan agar bisa dibuat label waktu. Label ini dikumpulkan di dalam array $bulan_labels. Untuk setiap kecamatan, dibuat daftar jumlah *stunting* berdasarkan urutan bulan. Jika suatu bulan tidak ada data *stunting*maka akan diisi dengan angka nol. Semua data yang sudah disusun disimpan dalam variabel $tren_data.

```
const bulanLabels = <?= json_encode($bulan_labels); ?>;
const trenDatasets = <?= json_encode($tren_data); ?>;

const ctxLine = document.getElementById('trenChart').getContext('2d');

const trenChart = new Chart(ctxLine, {
  type: 'line',
  data: {
    labels: bulanLabels,
    datasets: trenDatasets
  }
});
```

Gambar 3.86 Potongan Kode *views/backend/*analisis_wilayah.php Bagian Tren Penurunan/Kenaikan *Stunting* Tiap Bulan per Wilayah

Gambar 3.86 menunjukkan potongan kode *views/backend/analisis_wilayah.php* bagian tren penurunan/kenaikan *stunting* tiap bulan per wilayah. Kode ini digunakan untuk menampilkan grafik garis yang menunjukkan perubahan jumlah kasus *stunting* setiap bulan di berbagai wilayah. Variabel bulanLabels berisi daftar bulan yang digunakan sebagai label sumbu X, sedangkan trenDatasets berisi data jumlah *stunting* dari masing-masing wilayah untuk setiap bulan. Grafik ini dibuat menggunakan *library* Chart.js dan ditampilkan pada elemen <canvas> dengan ID trenChart.



Gambar 3.87 Tampilan Analisis Wilayah

Gambar 3.87 merupakan tampilan analisis wilayah. Tampilan ini terdiri dari dua visualisasi, yaitu visualisasi pertama menampilkan top 10 wilayah dengan prevalensi *stunting* tertinggi dalam bentuk diagram batang, di mana Karang Tengah menunjukkan prevalensi stunting tertinggi. Visualisasi kedua menampilkan tren penurunan/kenaikan *stunting* tiap bulan per wilayah yang ditampilkan dalam grafik garis *(line chart),* yang memungkinkan untuk melihat perkembangan jumlah kasus stunting dari waktu ke waktu di masing-masing wilayah.

3.2.3.7 Melanjutkan dengan Menampilkan Faktor Risiko

Faktor risiko digunakan untuk mengetahui penyebab utama *stunting* pada anak. Terdapat dua jenis visualisasi, yaitu diagram pie dan radar chart. Visualisasi ini membantu dalam mengidentifikasi faktor apa yang berkontribusi terhadap *stunting* dan wilayah mana yang paling banyak terdampak oleh faktor-faktor tersebut, sehingga intervensi bisa lebih tepat sasaran.

```php
$bb_tb_response = json_decode($this->curl->simple_get('http://127.0.0.1:5000/bb-tb-lahir-rendah'));
$bb_tb_data = (array)($bb_tb_response ?? []);
$data['bb_tb_data'] = $bb_tb_data;
```

Gambar 3.88 Potongan Kode *controller*/faktor_risiko.php Bagian Proporsi BB/TB Lahir Rendah

Gambar 3.88 menunjukkan potongan kode *controller/faktor_risiko.php* bagian proporsi bb/tb lahir rendah. Kode ini mengambil data dari API bb-tb-lahir-rendah menggunakan metode GET. Hasil dari API diubah menjadi array dan disimpan ke dalam variabel $bb_tb_data. Data ini kemudian dikirim ke *views* melalui $data[‘bb_tb_data’] untuk ditampilkan pada diagram pie.

```
const pieData = <?= json_encode($bb_tb_data); ?>;
const pieLabels = Object.keys(pieData);
const pieValues = Object.values(pieData);

new Chart(document.getElementById('pieChart'), {
  type: 'pie',
  data: {
    labels: pieLabels,
    datasets: [{
      data: pieValues,
    }]
  }
});
```

Gambar 3.89 Potongan Kode *views/backend/*faktor_risiko.php Bagian Proporsi BB/TB Lahir Rendah

Gambar 3.89 menunjukkan potongan kode *views/backend//faktor_risiko.php* bagian proporsi bb/tb lahir rendah. Kode ini mengambil data dari $bb_tb_data dan mengubahnya menjadi format JavaScript menggunakan json_encode. Label dan nilai dari data dipisah menggunakan Object.keys() dan Object.values(). Kemudian, data tersebut digunakan untuk membuat diagram pie menggunakan Chart.js pada elemen pieChart.

```
$faktor_risiko_response = json_decode($this->curl->simple_get('http://127.0.0.1:5000/faktor-risiko'));
$faktor_risiko_response = $faktor_risiko_response ?? [];

$kecamatan_labels = [];
$bb_lahir = [];
$tb_lahir = [];
$bb_pengukuran = [];
$tb_pengukuran = [];
$usia_bulan = [];

foreach ($faktor_risiko_response as $item) {
    $kecamatan_labels[] = $item->kecamatan;
    $bb_lahir[] = $item->bb_lahir;
    $tb_lahir[] = $item->tb_lahir;
    $bb_pengukuran[] = $item->bb_pengukuran;
    $tb_pengukuran[] = $item->tb_pengukuran;
    $usia_bulan[] = $item->usia_bulan;
}

$data['kecamatan_labels'] = $kecamatan_labels;
$data['radar_data'] = [
    ['label' => 'BB Lahir', 'data' => $bb_lahir],
    ['label' => 'TB Lahir', 'data' => $tb_lahir],
    ['label' => 'BB Pengukuran', 'data' => $bb_pengukuran],
    ['label' => 'TB Pengukuran', 'data' => $tb_pengukuran],
    ['label' => 'Usia (bulan)', 'data' => $usia_bulan]
];
```

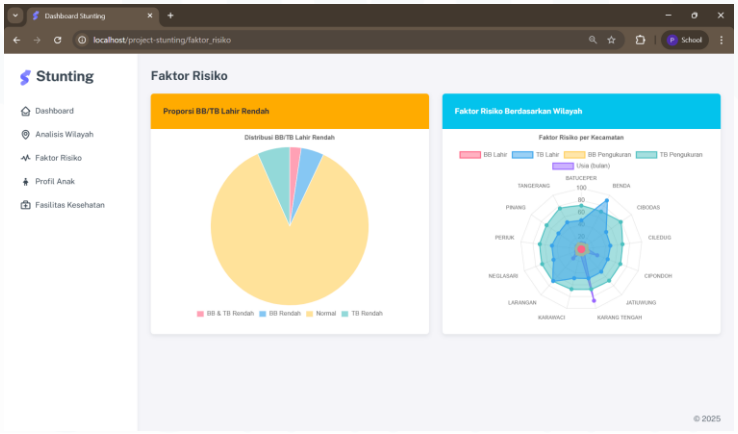
Gambar 3.90 Potongan Kode *controller/*faktor_risiko.php Bagian Faktor Risiko Berdasarkan Wilayah

Gambar 3.90 menunjukkan potongan kode *controller/faktor_risiko.php* bagian faktor risiko berdasarkan wilayah. Sitem mengambil data dari API /faktor-risiko lalu memprosesnya menjadi beberapa array. Masing-masing nilai dimasukkan ke dalam array sesuai kecamatan. Data ini kemudian disusun dalam bentuk array radar_data yang akan digunakan pada *radar chart* di tampilan (*views).*

```
const radarLabels = <?= json_encode($kecamatan_labels); ?>;
const radarRawData = <?= json_encode($radar_data); ?>;

const radarDatasets = radarRawData.map((item, index) => ({
  label: item.label,
  data: item.data
}));

const ctxRadar = document.getElementById('radarChart').getContext('2d');
new Chart(ctxRadar, {
  type: 'radar',
  data: {
    labels: radarLabels,
    datasets: radarDatasets
  }
});
```

Gambar 3.91 Potongan Kode *views/backend/*faktor_risiko.php Bagian Faktor Risiko Berdasarkan Wilayah

Gambar 3.91 menunjukkan potongan kode *views/backend/faktor_risiko.php* bagian faktor risiko berdasarkan wilayah. Kode ini digunakan untuk membuat *radar chart* dengan *library* Chart.js. Nilai label kecamatan dan data risiko diambil dari variabel $kecamatan_labels dan $radar_data, lalu dikonversi ke format JavaScript menggunakan json_encode, kemudian ditampilkan dalam bentuk *radar chart.*



Gambar 3.92 Tampilan Faktor Risiko

Gambar 3.92 menunjukkan tampilan faktor risiko, yang terdiri dari dua visualisasi. Visualisasi pertama adalah diagram pie yang menampilkan proporsi bb/tb lahir rendah, di mana mayoritas termasuk kategori normal, diikuti oleh tb rendah sebagai salah satu faktor risiko utama. Visualisasi kedua adalah *radar chart* yang menunjukkan nilai faktor risiko berdasarkan wilayah. Titik yang jauh dari pusat menunjukkan nilai risiko yang tinggi, sedangkan titik yang dekat menunjukkan risiko yang rendah. Dari grafik terlihat bahwa nilai tb lahir di Kecamatan Benda cukup tinggi karena posisinya jauh dari pusat.

3.2.3.8 Melanjutkan dengan Menampilkan Profil Anak

Profil anak disajikan dalam bentuk data tabel untuk menampilkan informasi mengenai masing-masing anak. Data ini berisi nama, umur dalam bulan, dan status gizi. Penyajian dalam bentuk tabel memudahkan proses identifikasi dan analisis kondisi setiap anak.

```php
$response = $this->curl->simple_get('http://127.0.0.1:5000/profil-anak');
$data_anak = json_decode($response);

$data['daftar_anak'] = $data_anak ?? [];
```

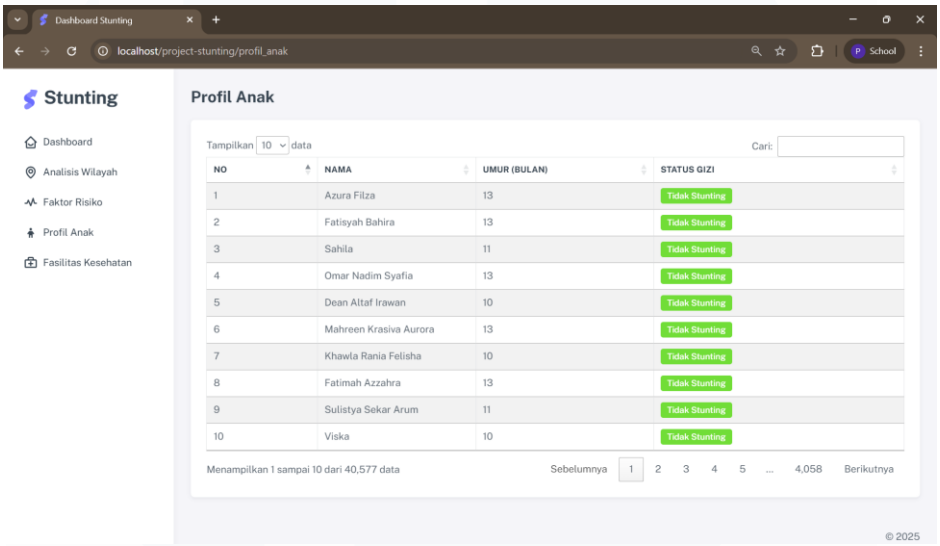
Gambar 3.93 Potongan Kode *controller*/profil_anak.php

Gambar 3.93 menunjukkan potongan kode *controller/profil_anak.php.* Kode ini digunakan untuk mengambil data profil anak dari API lokal menggunakan metode GET. Data yang diterima kemudian diubah dari format JSON menjadi array dengan json_decode. Hasilnya disimpan dalam variabel $data[‘daftar_anak’] untuk ditampilkan di *views.*

```php
<td>
  <?php
    $status = strtolower($anak->status_gizi);
    if ($status === 'gizi buruk' || $status === 'gizi kurang') {
      echo '<span class="badge bg-danger">Stunting</span>';
    } else {
      echo '<span class="badge bg-success">Tidak Stunting</span>';
    }
  ?>
</td>
```

Gambar 3.94 Potongan Kode *views/backend/*profil_anak.php

Gambar 3.94 merupakan potongan kode *views/backend/profil_anak.php.* Kode ini menampilkan status gizi anak. Jika statusnya gizi buruk atau kurang, akan ditandai sebagai *stunting* dengan warna merah, sisanya tidak *stunting* dengan warna hijau.

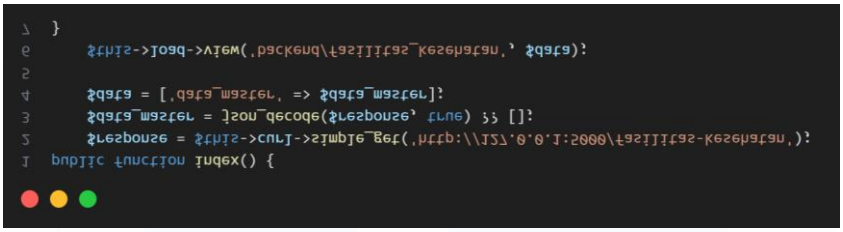


Gambar 3.95 Tampilan Profil Anak

Gambar 3.95 merupakan tampilan profil anak. Tabel tersebut menampilkan nama, umur dalam bulan, dan status gizi masing-masing anak. Tabel juga dilengkapi fitur pencarian untuk memudahkan pengguna dalam melihat data.

3.2.3.9 Melanjutkan dengan Menampilkan Analisis Kinerja Fasilitas Kesehatan

Fasilitas kesehatan disajikan dalam bentuk data tabel untuk menampilkan informasi layanan kesehatan di tiap wilayah. Data ini berisi nama puskesmas, kecamatan, alamat, posyandu, dan jumlah anak yang terpantau. Penyajian dalam bentuk tabel agar memudahkan dalam memantau cakupan layanan.



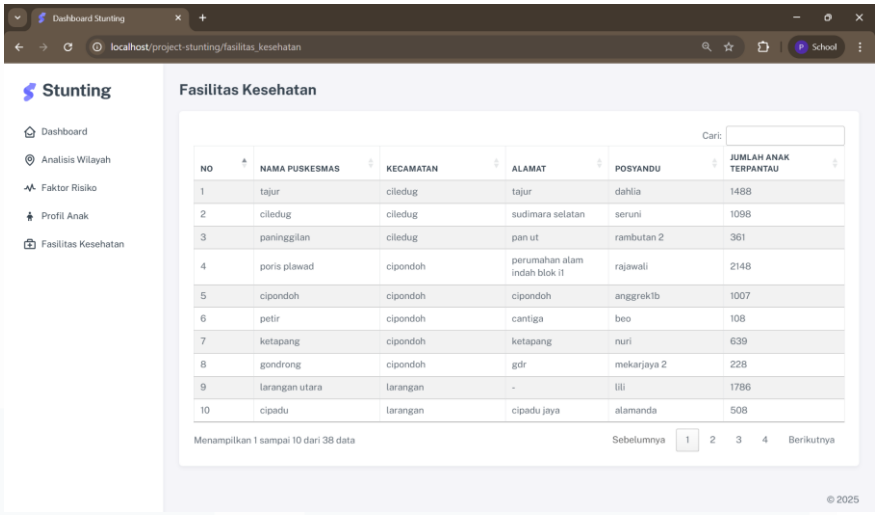
Gambar 3.96 Potongan Kode *controller*/fasilitas_kesehatan.php

Gambar 3.96 merupakan potongan kode *controller/fasilitas_kesehatan.php.* Kode ini berfungsi untuk mengambil data fasilitas kesehatan dari API lokal. Data yang diperoleh kemudian diubah menjadi array menggunakan json_decode. Selanjutnya, data akan dikirim ke view fasilitas_kesehatan.

```
<table id="dataPuskesmas" class="table table-bordered table-striped" style="display: none; width: 100%;">
  <thead>
    <tr>
      <th>No</th>
      <th>Nama Puskesmas</th>
      <th>Kecamatan</th>
      <?php if ($showAlamat): ?>
        <th>Alamat</th>
      <?php endif; ?>
      <?php if ($showPosyandu): ?>
        <th>Posyandu</th>
      <?php endif; ?>
      <th>Jumlah Anak Terpantau</th>
    </tr>
  </thead>
  <tbody>
</table>
```

Gambar 3.97 Potongan Kode *views/backend/*fasilitas_kesehatan.php

Gambar 3.97 menunjukkan potongan kode *views/backend/fasilitas_kesehatan.php.* Kode tersebut berfungsi untuk menampilkan tabel data fasilitas kesehatan yang diambil dari API. Tabel ini memiliki kolom seperti no, nama puskesmas, kecamatan, alamat, posyandu, dan jumlah anak terpantau.



Gambar 3.98 Tampilan Fasilitas Kesehatan

Gambar 3.98 menunjukkan tampilan dari fasilitas kesehatan. Dapat dilihat bahwa tampilan tersebut berisi tabel yang memuat informasi seperti no, nama puskemas, kecamatan, alamat, posyandu, dan jumlah anak terpantau. Tabel ini juga dilengkapi dengan fitur pencarian untuk memudahkan pengguna dalam menemukan data yang diinginkan.

3.2.4 Melaksanakan Pengujian Sistem, termasuk *User Acceptance Testing* (UAT) dan System Integration Testing (SIT) untuk Memastikan Kualitas dan Keandalan Aplikasi

Proses pengujian dilakukan ketika seluruh fitur dalam sistem telah selesai dikerjakan. Pengujian dilakukan dengan cara mencoba langsung halaman-halaman yang telah dibuat, seperti *dashboard,* analisis wilayah, faktor risiko, profil anak, dan fasilitas kesehatan, untuk memastikan tidak ada kesalahan pada tampilan maupun fungsionalitas. Pengujian dilakukan oleh lima orang *supervisor,* mentor, dan pemangku kepentingan yang terlibat dalam proyek ini. Pengujian dilakukan dengan mencoba sistem dari sisi pengguna untuk melihat apakah alur kerja sistem mudah dipahami, data yang ditampilkan jelas, dan fitur-fitur berjalan dengan baik. Dengan adanya pengujian ini, sistem diharapkan dapat memberikan manfaat yang maksimal.

No.	Skenario Uji	Langkah Uji	Hasil yang Diharapkan	Hasil Aktual	Status
1	Akses Awal Dashboard	Akses (http://localhost/project-stunting/) melalui browser	Halaman dashboard tampil, ada overview cards dan line chart	Sesuai	Berhasil
2	Buka Menu "Analisis Wilayah"	Klik menu "Analisis Wilayah" di sidebar	Menampilkan bar chart dan line chart yang menunjukkan prevalensi stunting	Sesuai	Berhasil
3	Buka Menu "Faktor Risiko"	Klik menu "Faktor Risiko" di sidebar	Pie chart, radar chart tampil dan menunjukkan faktor risiko	Sesuai	Berhasil
4	Buka Menu "Profil Anak"	Klik menu "Profil Anak" di sidebar	Data anak berhasil ditampilkan dan dapat dibaca dengan baik, fitur search dan pagination berhasil diterapkan	Sesuai	Berhasil
5	Buka Menu "Fasilitas Kesehatan"	Klik menu "Fasilitas Kesehatan" di sidebar	Informasi fasilitas kesehatan berhasil ditampilkan cukup lengkap, fitur search dan pagination berhasil diterapkan	Sesuai	Berhasil

Gambar 3.99 Pengujian Fungsional

Gambar 3.99 merupakan pengujian fungsional. Pengujian ini dilakukan untuk memastikan bahwa sistem *dashboard* analisis *stunting* berjalan sesuai dengan fungsionalitas. Pengujian dilakukan secara menual dengan cara mengakses setiap menu utama yang tersedia pada *sidebar.* Dari hasil pengujian, seluruh fitur utama aplikasi berhasil berjalan dengan baik, yang ditunjukkan dengan status "Berhasil" pada setiap skenario. Berikut adalah penjelasan dari masing-masing skenario:

1) **Akses awal *dashboard.***

Dashboard diakses melalui URL http://localhost/project-stunting/ dan pengguna langsung diarahkan ke halaman utama *dashboard.* Halaman ini menampilkan beberapa *overview cards* yang berisi informasi penting seperti total anak terpantau, anak risiko *stunting,* jumlah posyandu dan puskesmas aktif. Selain itu, ditampilkan juga grafik tren risiko *stunting* bulanan dalam bentuk *line chart* yang memudahkan pengguna untuk memantau perkembangan kasus dari waktu ke waktu.

2) **Buka menu "Analisis Wilayah".**

Analisis wilayah dibuka dengan cara melakukan klik menu "Analisis Wilayah" di *sidebar.* Menu ini menampilkan dua jenis visualisasi utama, yaitu *bar chart* dan *line chart. Bar chart* menampilkan 10 wilayah dengan prevalensi *stunting* tertinggi, sedangkan *line chart* menunjukkan tren naik/turun prevalensi *stunting* per bulan di masing-masing wilayah. Visualisasi berjalan

dengan baik dan dapat membantu pengguna dalam menganalisis wilayah yang memerlukan perhatian lebih dalam untuk penanganan *stunting.*

3) Buka menu "Faktor Risiko".

Faktor risiko dibuka dengan cara melakukan klik menu "Faktor Risiko" di *sidebar.* Menu ini menampilkan *pie chart* yang menggambarkan proporsi bayi lahir dengan berat badan (bb) dan tinggi badan (tb) rendah. Selain itu, *radar chart* disajikan untuk memperlihatkan faktor risiko *stunting* yang muncul di berbagai wilayah. Visualisasi ini memudahkan pengguna untuk mengidentifikasi penyebab utama *stunting* di tiap wilayah.

4) Buka menu "Profil Anak".

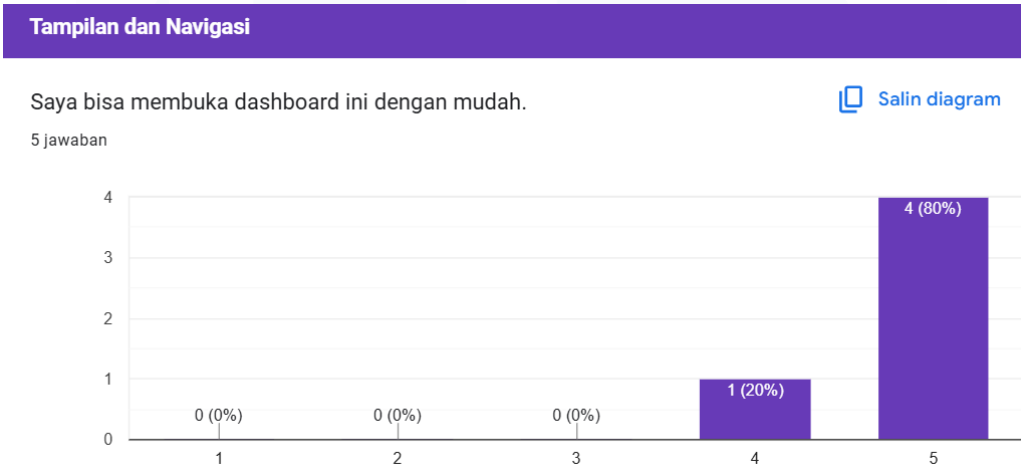
Profil anak dibuka dengan cara melakukan klik menu "Profil Anak" di *sidebar.* Halaman ini menyajikan tabel yang berisi data dari masing-masing anak. Data tersebut berisi nama, umur dalam bulan, dan status gizinya. Tabel dilengkapi dengan fitur pencarian *(search)* dan pemisah halaman *(pagination)* sehingga pengguna dapat mencari dan mengelola data dengan lebih cepat dan efisien. Seluruh data ditampilkan dengan jelas dan fungsi-fungsi pendukung dapat bekerja dengan baik.

5) Buka menu "Fasilitas Kesehatan".

Fasilitas kesehatan dibuka dengan cara melakukan klik menu "Fasilitas Kesehatan" di *sidebar.* Menu ini menampilkan tabel informasi fasilitas kesehatan seperti nama puskesmas, kecamatan, alamat, nama posyandu, dan jumlah anak yang terpantau di masing-masing lokasi. Tabel ini juga dilengkapi fitur pencarian dan *pagination.* Seluruh informasi berhasil ditampilkan dengan lengkap.

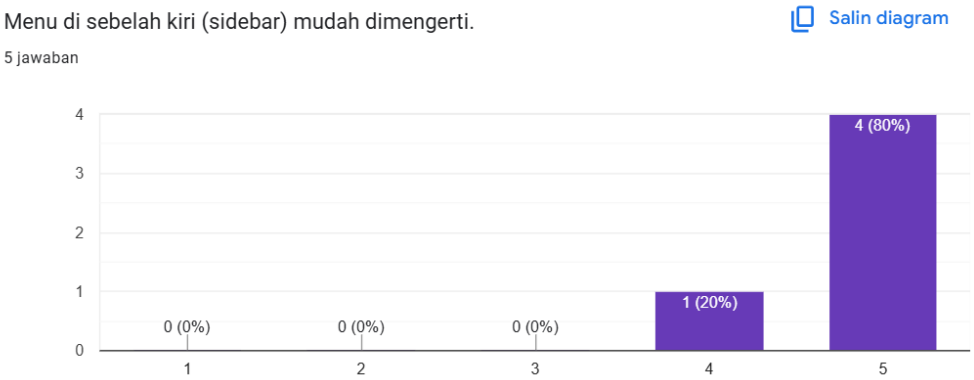
Pengujian *User Acceptance Testing* (UAT) dilakukan untuk mengetahui apakah *dashboard stunting* mudah digunakan dan dimengerti oleh pengguna. Pengujian dilakukan dengan menyebarkan kuesioner kepada 5 responden yang terdiri dari 12 pertanyaan. Setiap pertanyaan dijawab menggunakan skala penilaian 1 sampai 5, di mana angka 1 berarti sangat tidak setuju dan angka 5 berarti sangat setuju. Dari total 60 jawaban yang terkumpul (5 responden x 12 pertanyaan), sebanyak 48 jawaban (80%) memberikan nilai 5, dan 12 jawaban (20%)

memberikan nilai 4. Tidak ada jawaban dengan nilai di bawah 4, yang artinya semua responden merasa puas dengan *dashboard* yang sudah dikembangkan. Hasil ini menunjukkan bahwa *dashboard* sudah memenuhi ekspektasi pengguna dari segi kemudahan penggunaan, kejelasan data, dan tampilan.



Gambar 3.100 Penilaian Pengguna terhadap Kemudahan Akses *Dashboard*

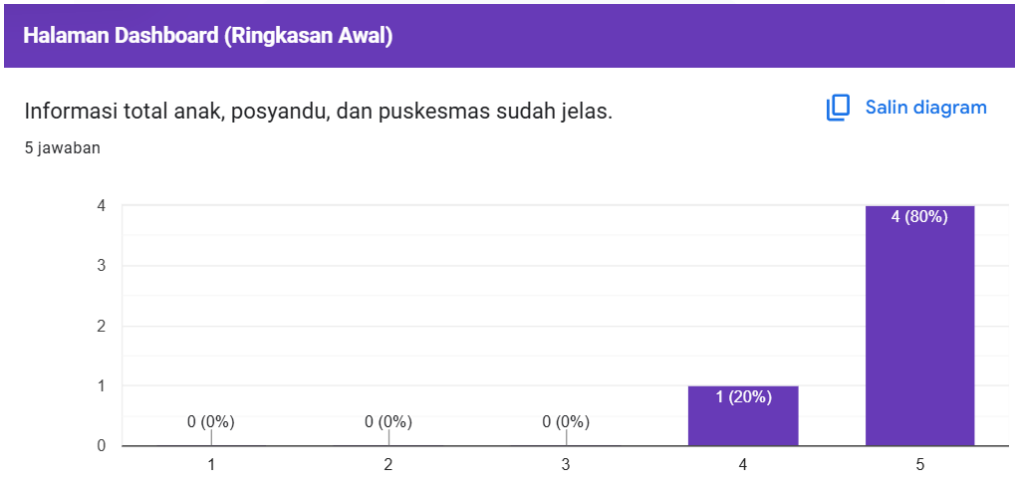
Gambar 3.100 menunjukkan penilaian pengguna terhadap kemudahan akses *dashboard.* Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa *dashboard* mudah diakses tanpa kendala, yang menandakan bahwa sistem sudah cukup baik dan tidak membingungkan.



Gambar 3.101 Penilaian Pengguna terhadap Kemudahan Menu *Sidebar*

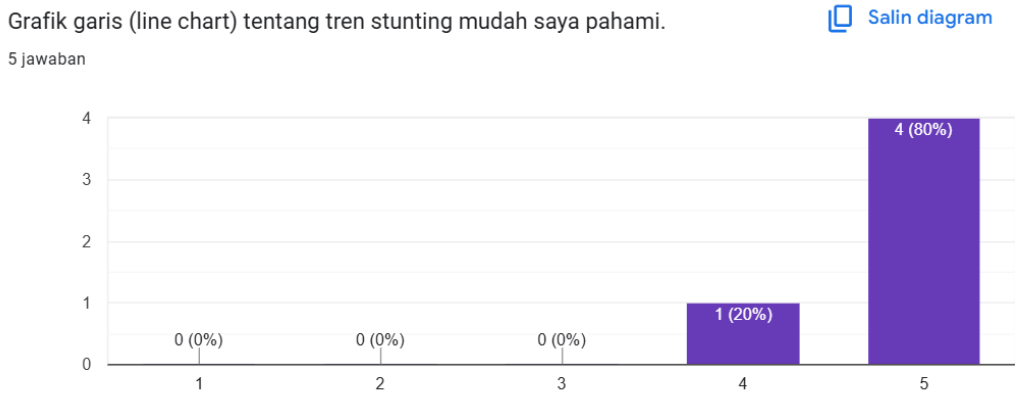
Gambar 3.101 menunjukkan penilaian pengguna terhadap kemudahan menu *sidebar.* Sebagian besar responden memberikan nilai 5, dan hanya satu yang

memberi nilai 4. Hal ini menandakan bahwa menu *sidebar* cukup jelas dan mudah dipahami pengguna saat berpindah halaman.



Gambar 3.102 Penilaian Pengguna terhadap Kejelasan Informasi Ringkasan

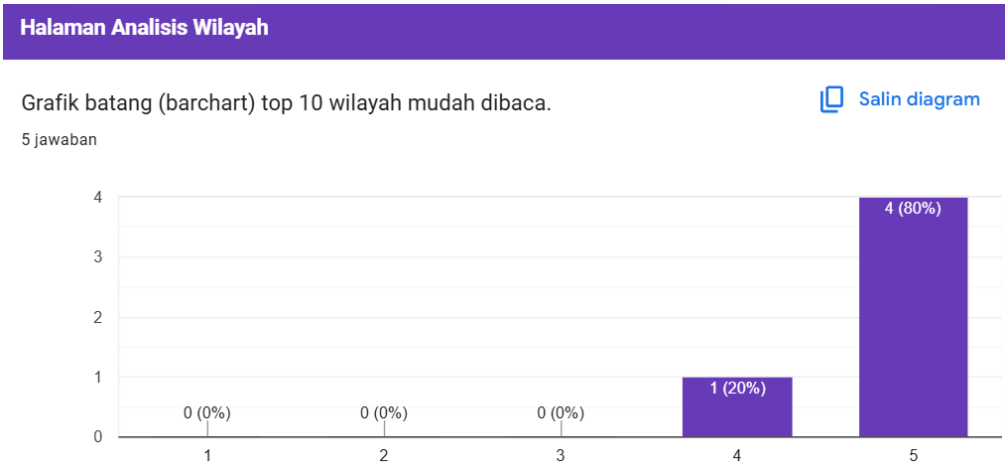
Gambar 3.102 menunjukkan penilaian pengguna terhadap kejelasan informasi ringkasan. Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa informasi utama di halaman awal *dashboard* disajikan dengan jelas dan mudah dipahami. Tampilan ringkasan ini membantu pengguna untuk mendapat gambaran awal secara cepat.



Gambar 3.103 Penilaian Pengguna terhadap Kemudahan Memahami Grafik Garis

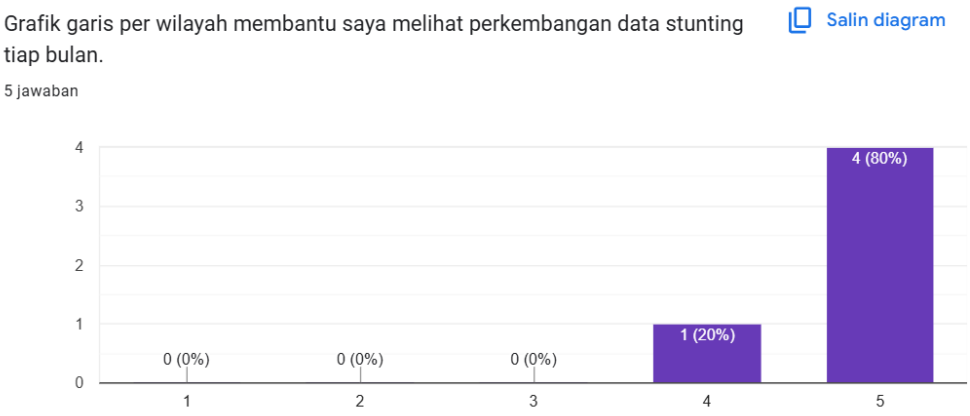
Gambar 3.103 menunjukkan penilaian pengguna terhadap kemudahan memahami grafik garis. Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa visualisasi tren stunting

melalui grafik garis cukup mudah dipahami oleh pengguna. Grafik ini berhasil menyampaikan informasi perubahan risiko stunting dari waktu ke waktu.



Gambar 3.104 Penilaian Pengguna terhadap Kemudahan Memahami Tren *Stunting*

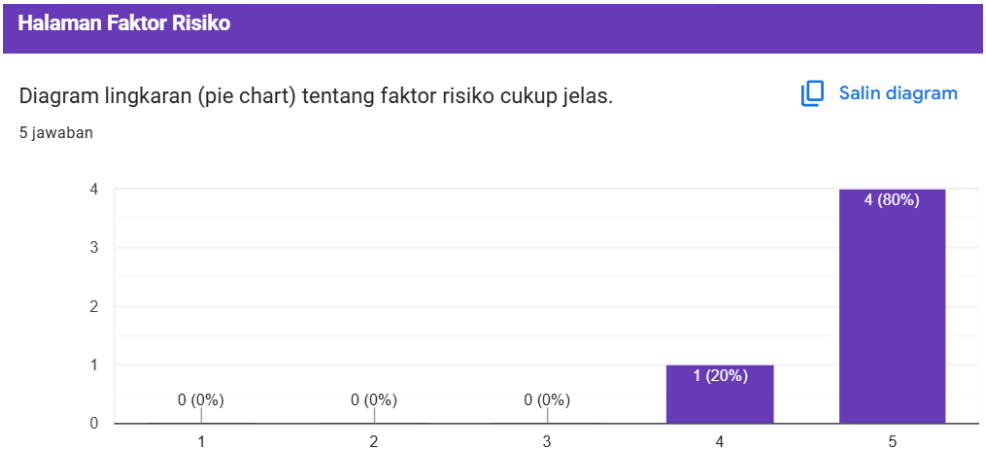
Gambar 3.104 menunjukkan penilaian pengguna terhadap kemudahan memahami tren stunting. Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa pengguna merasa visualisasi tren kenaikan dan penurunan stunting per wilayah sudah cukup jelas. Grafik garis membantu dalam memahami perubahan kondisi stunting dari waktu ke waktu.



Gambar 3.105 Penilaian Pengguna terhadap Kemudahan Kejelasan Grafik Batang

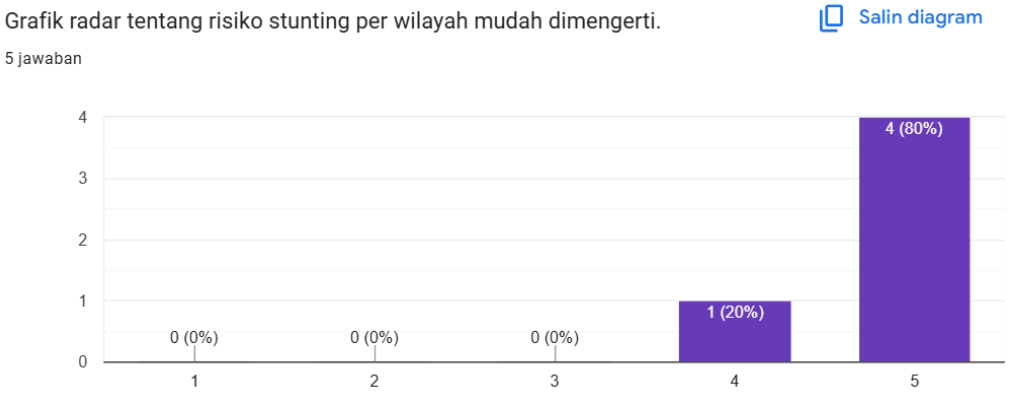
Gambar 3.105 menunjukkan penilaian pengguna terhadap kemudahan kejelasan grafik batang. Sebagian besar responden memberikan nilai 5, dan hanya

satu yang memberi nilai 4. Hal ini menunjukkan bahwa visualisasi ini dianggap mudah dipahami, karena berhasil untuk menunjukkan wilaya prioritas secara jelas.



Gambar 3.106 Penilaian Pengguna terhadap Kejelasan Diagram Lingkaran

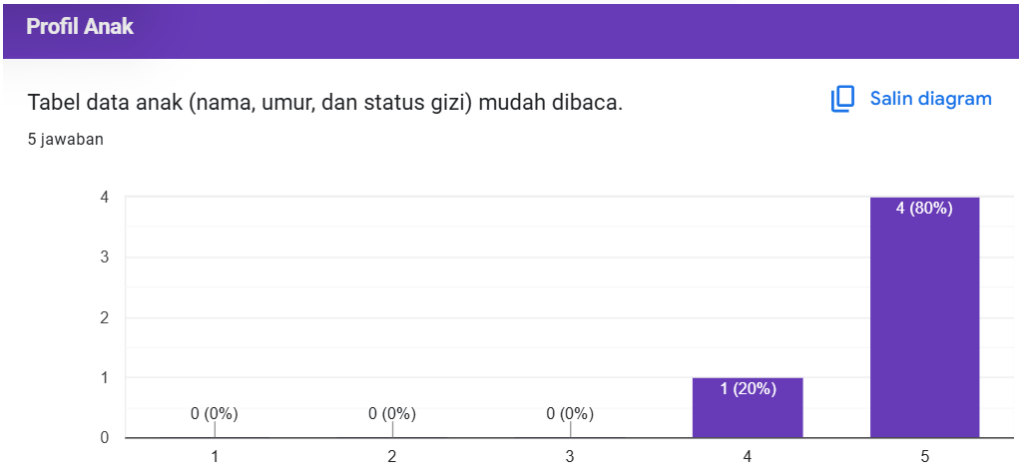
Gambar 3.106 menunjukkan penilaian pengguna terhadap kejelasan diagram lingkaran. Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa pengguna merasa informasi yang ditampilkan sudah cukup jelas dan mudah dipahami. Visualisasi tersebut berhasil untuk menyampaikan proporsi faktor risiko seperti berat badan lahir rendah.



Gambar 3.107 Penilaian Pengguna terhadap Kemudahan Memahami Grafik Radar

Gambar 3.107 menunjukkan penilaian pengguna terhadap kemudahan memahami grafik radar. Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa grafik radar cukup efektif

dalam menunjukkan perbandingan risiko antar wilayah. Visualisasi ini membantu pengguna untuk mengenali wilayah dengan tingkat risiko tertinggi.



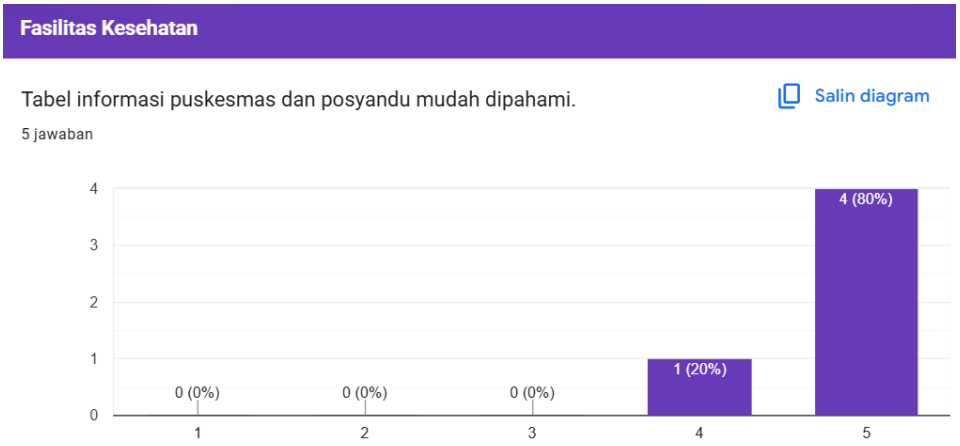
Gambar 3.108 Penilaian Pengguna terhadap Kemudahan Membaca Tabel Anak

Gambar 3.108 menunjukkan penilaian pengguna terhadap kemudahan membaca tabel anak. Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa tampilan tabel data anak yang berisi nama, umur, dan status gizi cukup jelas dan dapat dibaca dengan baik oleh pengguna. Informasi penting tersaji secara terstruktur sehingga memudahkan pengguna untuk menelusuri data anak.

Gambar 3.109 Penilaian Pengguna terhadap Kemudahan Mengetahui Informasi Anak

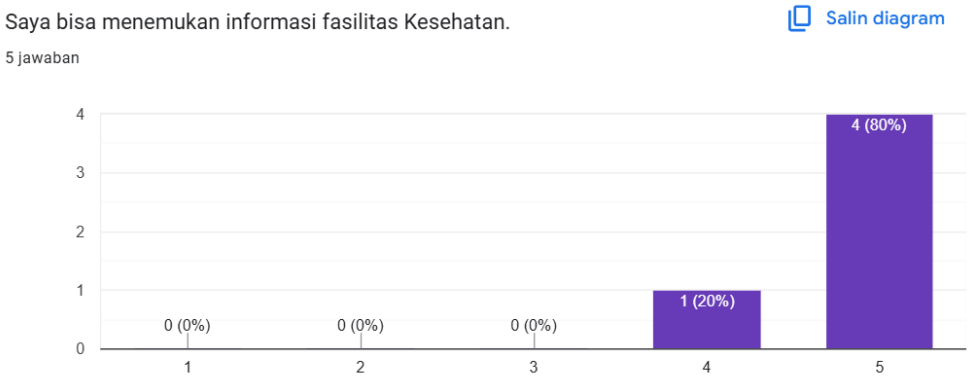
Gambar 3.109 menunjukkan pnilaian pengguna terhadap kemudahan mengetahui informasi anak. Sebagian besar responden memberikan nilai 5, dan

hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa tabel data anak dinilai berfungsi dengan baik dan membantu pengguna untuk mengerti status gizi anak.



Gambar 3.110 Penilaian Pengguna terhadap Kejelasan Tabel Informasi Puskesmas dan Posyandu

Gambar 3.110 menunjukkan penilaian pengguna terhadap kejelasan tabel informasi puskesmas dan posyandu. Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa tabel yang berisi data puskesmas, posyandu, alamat, dan jumlah anak terpantau ditampilkan dengan jelas dan terstruktur. Penggua dapat memahami isi tabel dengan mudah tanpa merasa bingung saat membaca informasi yang disediakan.



Gambar 3.111 Penilaian Pengguna terhadap Kemudahan Menemukan Informasi Fasilitas Kesehatan

Gambar 3.111 menunjukkan penilaian pengguna terhadap kemudahan menemukan informasi fasilitas kesehatan. Sebagian besar responden memberikan nilai 5, dan hanya satu yang memberi nilai 4. Hal ini menunjukkan bahwa pengguna

dapat menemukan informasi tentang fasilitas kesehatan tanpa kesulitan. Artinya, fitur pencarian atau tampilan data sudah cukup efektif membantu pengguna untuk menemukan informasi yang dibutuhkan.

3.2.5 Mematuhi Jadwal Kerja yang telah Disepakati dan Hadir di Kantor Sesuai dengan Ketentuan yang Berlaku

Selama pelaksanaan magang, jadwal kerja telah diikuti sesuai dengan ketentuan yang berlaku di instansi. Kegiatan magang dilakukan setiap hari kerja, yaitu Senin hingga Jumat. Sistem kerja berlangsung secara *work from office* (WFO) pada hari Senin sampai Kamis, sedangkan hari Jumat diberlakukan sistem *work from home* (WFH). Jam kerja dimulai pukul 08.00 dan berakhir pada pukul 17.00 WIB, dengan waktu istirahat selama satu jam dari pukul 12.00 hingga 13.00 WIB. Kehadiran selalu disesuaikan dengan ketentuan yang sudah disepakati bersama untuk menjaga kedisiplinan dan kelancaran dalam proses pelaksanaan kegiatan magang di Dinas Komunikasi dan Informatika Kota Tangerang.

3.3 Kendala yang Ditemukan

Kegiatan magang memberikan banyak pengalaman baru yang berguna untuk memahami dunia kerja secara langsung. Dalam prosesnya, tidak semua hal berjalan dengan mulus. Beberapa kendala muncul dan cukup menghambat jalannya proyek yang sedang dikerjakan. Situasi seperti ini perlu dihadapi dengan usaha dan penyesuaian agar pekerjaan tetap bisa diselesaikan dengan baik. Berikut adalah beberapa kendala utama yang dialami:

1) Kurangnya pengalaman menggunakan *framework CodeIgniter 3.*

   Framework CodeIgniter 3 adalah salah satu *framework* yang banyak digunakan dalam pengembangan aplikasi *web*. Namun, karena belum terbiasa menggunakan *framework* ini sebelumnya, banyak tantangan yang muncul selama proses pengembangan. Proses penyesuaian diri terhadap berbagai fungsi dan fitur di *CodeIgniter* memerlukan lebih banyak waktu seperti penggunaan *routing*, serta pengelolaan *view* dan *controller* yang

menjadi inti dari *framework* ini. Akibatnya, pekerjaan yang harus diselesaikan menjadi lebih lambat.

2) Data dari Dinas Komunikasi dan Informatika cenderung berantakan.

   Data yang diterima dari Dinas Komunikasi dan Informatika untuk proyek ini sering kali tidak terstruktur dengan baik. Banyak data yang memiliki format yang tidak konsisten, seperti kolom yang kosong, data yang duplikat, atau format angka dan teks yang tidak seragam. Hal ini mengharuskan untuk melakukan pembersihan data sebelum data bisa digunakan untuk analisis atau digunakan dalam aplikasi. Selain itu, ketidaklengkapan data juga menjadi kendala besar karena bisa memengaruhi akurasi analisis dan hasil dari prediksi yang ingin dicapai.

3) Kurangnya ketersediaan data di awal proyek dan kurangnya contoh referensi.

   Kurangnya ketersediaan data di awal proyek menjadi salah satu kendala utama dalam pengembangan proyek. Tidak adanya referensi atau contoh proyek sebelumnya menyulitkan dalam menentukan standar yang tepat terhadap hasil yang diharapkan. Salah satunya adalah ketika diberikan tugas untuk membuat visualisasi, namun data yang dibutuhkan tidak tersedia.

3.4 Solusi atas Kendala yang Ditemukan

Menghadapi berbagai kendala yang muncul selama proses magang, perlu memerlukan langkah-langkah yang benar untuk mengatasinya. Setiap masalah yang ditemukan memerlukan pendekatan yang sesuai agar pekerjaan tetap dapat berjalan dengan lancar dan tujuan proyek tercapai. Dengan menerapkan solusi yang tepat, diharapkan hambatan-hambatan tersebut dapat diminimalisir dan proyek dapat diselesaikan dengan hasil yang optimal. Berikut adalah beberapa solusi yang diterapkan untuk mengatasi kendala yang ditemukan:

1) Mencari referensi belajar dan tutorial tentang *CodeIgniter* secara mandiri.

   Solusi utama yang diterapkan untuk menghadapi tantangan kurangnya pengalaman dalam *CodeIgniter 3* adalah mencari berbagai

referensi belajar yang dapat membantu dalam memahami *framework* ini lebih cepat. Referensi ini dicari melalui berbagai *platform* contohnya *youtube*, sehingga bisa memahami cara menggunakan *CodeIgniter* secara mendalam. Selain itu, mencoba untuk melakukan eksperimen juga membantu dalam memberikan pengalaman. Karena dengan melakukan eksperimen langsung dengan kode bisa membantu dalam mempercepat proses pembelajaran, sehingga sedikit demi sedikit bisa mengerti. Dengan cara ini, pemahaman terhadap *framework* dapat diperoleh secara bertahap dan pekerjaan jadi lebih lancar.

2) Melakukan proses pembersihan data sebelum dipakai.

   Mengatasi masalah data yang berantakan dapat dilakukan dengan cara melakukan serangkaian langkah pembersihan data sebelum digunakan dalam analisis. Langkah pertama adalah memeriksa setiap kolom dan baris untuk memastikan tidak ada data yang kosong atau tidak relevan. Setelah itu, dilakukan penghapusan data duplikat yang dapat menyebabkan hasil analisis yang salah. Proses pembersihan data ini memakan waktu, karena untuk memastikan bahwa data yang digunakan dalam proyek adalah data yang akurat dan bersih, yang pada gilirannya akan meningkatkan kualitas hasil dari prediksi yang dilakukan. Dengan data yang bersih dan terstruktur, analisis dan aplikasi berjalan lebih lancar dan lebih dapat diandalkan.

3) Komunikasi yang jelas terkait data dan *output.*

   Masalah data yang tidak tersedia dapat diatasi dengan meminta data baru yang lebih lengkap. Namun, proses ini memakan waktu karena seluruh tahapan seperti analisis data harus diulang dari awal. Lalu, komunikasi yang lebih jelas antara pemberi tugas dan pelaksana sangat penting agar tidak terjadi kesalahpahaman yang dapat mengganggu kelancaran proyek.