

BAB 4

HASIL DAN PEMBAHASAN

4.1 Implementasi Sistem

Pada tahap implementasi sistem terdapat dua bagian utama, yaitu implementasi algoritma dan implementasi antarmuka pengguna (*interface*). Implementasi algoritma *Selection Sort* digunakan untuk mengurutkan data surat keluar berdasarkan kriteria tertentu seperti nomor surat atau tanggal, sehingga memudahkan pencarian dan pengelolaan data. Di sisi lain, implementasi antarmuka pengguna bertujuan untuk merancang tampilan web yang mudah digunakan oleh petugas kelurahan, memungkinkan mereka untuk memasukkan, mengelola, dan mengakses data surat keluar dengan lebih efisien melalui desain yang sederhana dan responsif. Kedua bagian ini bekerja bersama untuk memastikan sistem dapat berfungsi dengan baik dalam meningkatkan pengelolaan data surat keluar.

4.1.1 Implementasi Algoritma *Selection Sort*

```
$domisilis = Domisili::all();
$usahas = Usaha::all();
$bedanamas = Bedanama::all();
$hakpakais = HakPakai::all();
$ketpengantars = KetPengantar::all();

$allSurats = collect()
    ->merge($domisilis)
    ->merge($usahas)
    ->merge($bedanamas)
    ->merge($hakpakais)
    ->merge($ketpengantars)
    ->all();

$sortedSurats = $allSurats;
$n = count($sortedSurats);
```

```

for ($i = 0; $i < $n - 1; $i++) {
    $minIndex = $i;

    for ($j = $i + 1; $j < $n; $j++) {
        if($sortedSurats[$j]->noSurat< $sortedSurats
[$minIndex] ->noSurat) {
            $minIndex = $j;
        }
    }

    if ($minIndex != $i) {
        $temp = $sortedSurats[$i];
        $sortedSurats[$i] =
$sortedSurats[$minIndex];
        $sortedSurats[$minIndex] =
$temp;
    }
}

```

Gambar 4.1 Implementasi Algoritma *Selection Sort*

Gambar 4.1 menunjukkan implementasi algoritma *Selection Sort* untuk mengurutkan data surat keluar berdasarkan atribut `noSurat`. Data surat diambil dari lima tabel berbeda, yaitu `Domisili`, `Usaha`, `Bedanama`, `HakPakai`, dan `KetPengantar`, kemudian digabungkan menjadi satu koleksi menggunakan fungsi `collect()` dan `merge()`, lalu disimpan dalam variabel `\$allSurats`. Proses pengurutan dilakukan dengan mencari elemen terkecil pada setiap iterasi dan menukarnya dengan elemen di posisi awal yang sesuai. Panjang data dihitung menggunakan `count()` untuk menentukan jumlah iterasi, lalu dilakukan pencarian elemen terkecil dari indeks tertentu hingga akhir data, dan jika ditemukan elemen yang lebih kecil, dilakukan penukaran elemen tersebut. Setelah semua iterasi selesai, data dalam `\$sortedSurats` akan tersusun berdasarkan nilai `noSurat` dari kecil ke besar, sehingga mempermudah pengelolaan dan pencarian data dalam sistem.

4.1.2 Implementasi Koneksi *Database*

```
APP_NAME=Surat Keluar
APP_ENV=local
APP_KEY=base64:Xt3MqZU9TAqDyUnNCHdoZHTQsYldHHzcxl4KJ1y9
iUY=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=db_surat
DB_USERNAME=root
DB_PASSWORD=
BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

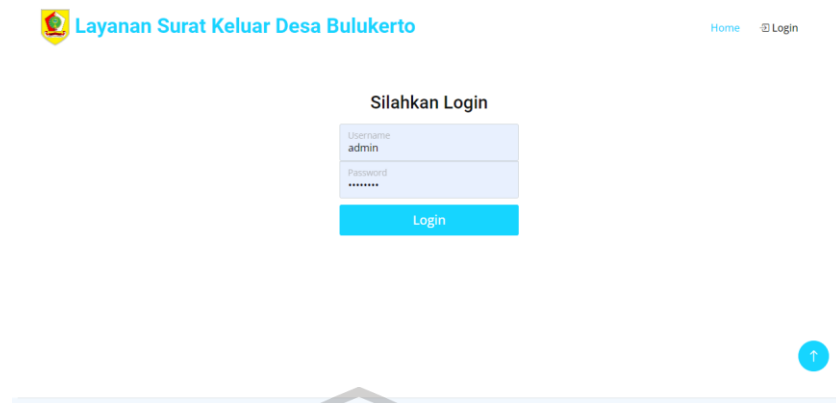
PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=mt1
```

Gambar 4.2 Koneksi *Database*

Pada gambar 4.2 menampilkan konfigurasi file `.env`, yaitu file konfigurasi utama dalam pengembangan aplikasi berbasis Laravel, yang berisi berbagai pengaturan lingkungan sistem untuk memastikan aplikasi berjalan dengan optimal. File ini mencakup pengaturan variabel `APP_NAME`, yang menentukan nama aplikasi, yaitu *Surat Keluar*, serta `APP_ENV`, yang menunjukkan bahwa aplikasi berjalan dalam mode *local* untuk keperluan pengembangan. Selain itu, file `.env` juga mengatur koneksi database dengan variabel `DB_CONNECTION=mysql` sebagai jenis koneksi yang digunakan, `DB_HOST=127.0.0.1` sebagai alamat host, `DB_DATABASE=db_surat` sebagai nama database, `DB_USERNAME=root` sebagai nama pengguna, dan password yang sesuai untuk menghubungkan aplikasi dengan database MySQL. Variabel tambahan, seperti `MAIL_MAILER`, digunakan untuk mengatur layanan pengiriman email, `CACHE_DRIVER` untuk menentukan metode caching yang digunakan, dan `SESSION_DRIVER` untuk mengelola penyimpanan sesi pengguna. Dengan konfigurasi ini, aplikasi dapat diatur sesuai kebutuhan, baik saat pengembangan lokal maupun untuk penerapan di lingkungan produksi, sehingga memberikan fleksibilitas dan keandalan dalam pengelolaan sistem.

4.1.3 Implementasi *Interface*

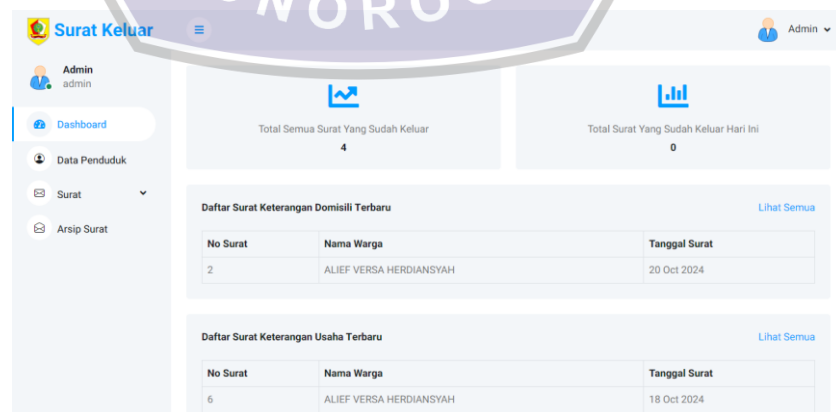
1) *Login*



Gambar 4.3 *Login*

Pada gambar 4.3 ditampilkan halaman login dari aplikasi "Layanan Surat Keluar Desa Bulukerto". Halaman ini digunakan untuk mengautentikasi pengguna sebelum mengakses sistem. Tampilan halaman login terdiri dari logo aplikasi di bagian kiri atas, judul aplikasi "Layanan Surat Keluar Desa Bulukerto" dengan warna biru, dan form login yang terletak di tengah halaman. Form ini memiliki dua input utama, yaitu *username* dan *password*, serta tombol login berwarna biru di bawahnya.

2) *Dashboard Admin*



No Surat	Nama Warga	Tanggal Surat
2	ALJEF VERSA HERDIANSYAH	20 Oct 2024

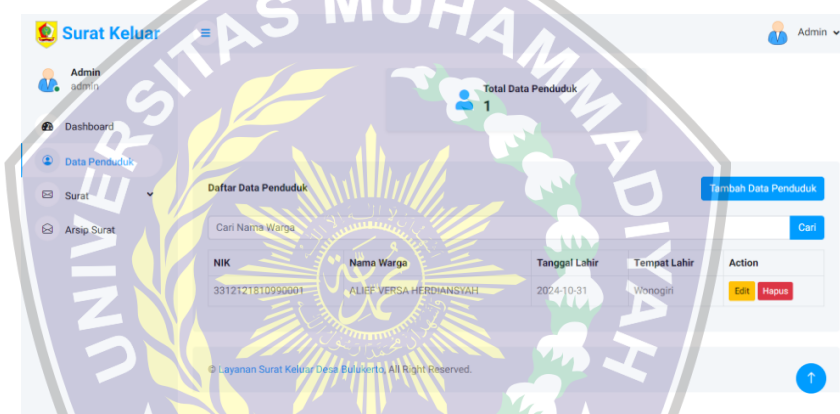
No Surat	Nama Warga	Tanggal Surat
6	ALJEF VERSA HERDIANSYAH	18 Oct 2024

Gambar 4.4 *Dashboard Admin*

Pada gambar 4.4 ditampilkan halaman *dashboard* admin dari aplikasi "Layanan Surat Keluar Desa Bulukerto". *Dashboard* ini

berfungsi sebagai pusat informasi bagi admin untuk memantau data surat keluar. Pada bagian atas halaman terdapat dua kotak informasi utama yang menampilkan total semua surat yang sudah keluar dan total surat yang keluar pada hari itu. Di bawahnya terdapat dua tabel yang masing-masing menampilkan daftar surat keterangan domisili terbaru dan daftar surat keterangan usaha terbaru, dengan informasi berupa nomor surat, nama warga, dan tanggal surat. Di sebelah kiri, terdapat menu navigasi berupa *dashboard*, data penduduk, surat, dan arsip surat untuk memudahkan akses ke fitur-fitur aplikasi.

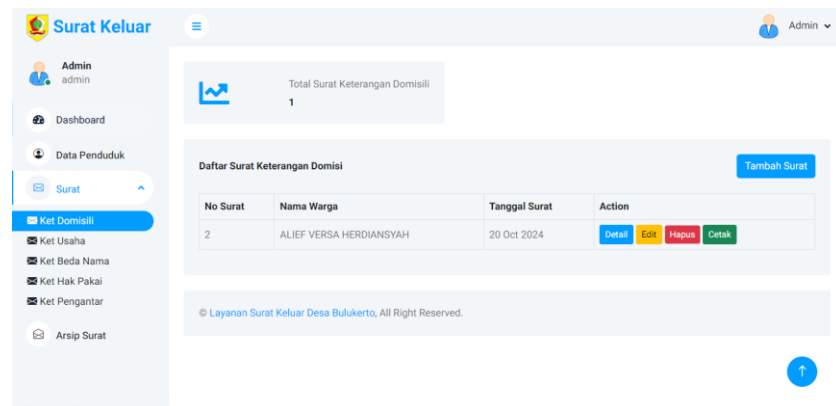
3) Tampilan Data Penduduk



Gambar 4.5 Data Penduduk

Pada Gambar 4.5 ditampilkan halaman dashboard admin dari aplikasi "Layanan Surat Keluar Desa Bulukerto". Dashboard ini berfungsi sebagai pusat informasi bagi admin untuk memantau data surat keluar. Pada bagian atas halaman terdapat dua kotak informasi utama yang menampilkan total semua surat yang sudah keluar dan total surat yang keluar pada hari itu. Di bawahnya terdapat dua tabel yang masing-masing menampilkan daftar surat keterangan domisili terbaru dan daftar surat keterangan usaha terbaru, dengan informasi berupa nomor surat, nama warga, dan tanggal surat. Di sebelah kiri, terdapat menu navigasi berupa dashboard, data penduduk, surat, dan arsip surat untuk memudahkan akses ke fitur-fitur aplikasi.

4) Tampilan Surat Keterangan Domisili



Gambar 4.6 Tampilan Surat Keterangan Domisili

Pada gambar 4.6, tampilan menunjukkan halaman Surat Keterangan Domisili dalam aplikasi layanan surat keluar desa, dengan informasi total surat yang telah dibuat sebanyak 1. Di bawahnya terdapat tabel daftar surat yang mencakup kolom No Surat, Nama Warga, Tanggal Surat, dan Action. Sebagai contoh, terdapat data surat dengan No Surat 2, Nama Warga ALIEF VERSA HERDIANSYAH, dan Tanggal Surat 20 Oktober 2024.

5) Tampilan Buat Surat

Gambar 4.7 Tampilan Buat Surat

Pada gambar 4.7, tampilan menunjukkan halaman Buat Surat dalam aplikasi layanan surat keluar desa. Halaman ini menyediakan formulir untuk mengisi data yang diperlukan dalam pembuatan

surat. Beberapa kolom yang tersedia meliputi kode surat, nomor surat, nama, jabatan, dan alamat.

6) Tampilan Arsip Surat Keluar



Gambar 4.8 Tampilan Arsip Surat Keluar

Pada gambar 4.8, tampilan menunjukkan halaman Arsip Surat Keluar dalam aplikasi layanan surat keluar desa. Halaman ini menampilkan informasi mengenai surat-surat yang telah dibuat dan diarsipkan. Di bagian atas terdapat dua panel yang menunjukkan total semua surat yang sudah keluar, yaitu 4, dan total surat yang sudah keluar hari ini, yaitu 0.

4.2 Pengujian Sistem

Pengujian sistem adalah tahap pengujian perangkat lunak di mana seluruh sistem atau aplikasi diuji sebagai satu kesatuan yang terintegrasi. Tujuan dari pengujian sistem adalah untuk memastikan bahwa semua komponen dan modul dari perangkat lunak berfungsi dengan baik bersama-sama dan memenuhi spesifikasi serta kebutuhan yang telah ditentukan. Berikut adalah penjelasan lebih rinci mengenai pengujian sistem :

1) Pengujian *Whitebox*

Pengujian unit adalah proses pengujian perangkat lunak yang berfokus pada pengujian masing-masing unit atau komponen terkecil dari suatu aplikasi secara terpisah untuk memastikan bahwa setiap unit berfungsi

sesuai dengan logika yang diharapkan. Dalam pengujian ini, *flow graph* digunakan untuk memvisualisasikan alur logika program, mempermudah identifikasi jalur eksekusi, dan memastikan bahwa setiap cabang kode telah diuji secara menyeluruh. Pengujian ini bertujuan untuk memverifikasi bahwa setiap unit perangkat lunak memenuhi kebutuhan spesifiknya sebelum diintegrasikan dengan unit lain, sehingga kesalahan dapat dideteksi lebih awal dalam proses pengembangan.

a) Pengujian Unit

Pengujian unit pada proses pengurutan data surat keluar menggunakan algoritma *Selection Sort* melibatkan beberapa langkah pengujian. Pertama, data surat dari tabel `Domisili`, `Usaha`, `Bedanama`, `HakPakai`, dan `KetPengantar` digabungkan menjadi satu koleksi menggunakan metode `collect()` dan `merge()`. Selanjutnya, pengujian memastikan bahwa proses pengurutan berjalan sesuai logika *Selection Sort*, dengan memeriksa setiap iterasi. Pengujian ini meliputi langkah-langkah seperti menetapkan variabel `\$minIndex`, membandingkan elemen berdasarkan atribut `noSurat`, dan memastikan bahwa elemen dengan nilai `noSurat` terkecil dipindahkan ke posisi yang sesuai. Setiap jalur logika dalam proses ini, termasuk kondisi saat tidak ada pertukaran yang diperlukan atau saat elemen ditukar, diuji secara menyeluruh. Melalui pengujian ini, diharapkan algoritma dapat mengurutkan data dengan benar, menghasilkan daftar surat yang terurut sesuai dengan nilai `noSurat`.

Tabel 4.1 *Pseudocode Method “\$allSurats = collect()” dari Class “Surat”*

Pseudocode	Node
<code>\$allSurats = collect()</code>	1
<code>->merge(\$domisilis)</code>	1
<code>->merge(\$usahas)</code>	1
<code>->merge(\$bedanamas)</code>	1
<code>->merge(\$hakpakais)</code>	1
<code>->merge(\$ketpengantars)</code>	1
<code>->all();</code>	1
<code>\$sortedSurats = \$allSurats;</code>	1

<code>\$n = count(\$sortedSurats);</code>	1
<code>for (\$i = 0; \$i < \$n - 1; \$i++)</code>	2
<code>{</code>	2
<code> \$minIndex = \$i;</code>	3
<code>for (\$j = \$i + 1; \$j < \$n; \$j++)</code>	4
<code>{</code>	4
<code> if (\$sortedSurats[\$j]->noSurat <</code>	5
<code> \$sortedSurats[\$minIndex]->noSurat)</code>	5
<code> {</code>	5
<code> \$minIndex = \$j;</code>	5
<code> if (\$minIndex != \$i)</code>	6
<code> {</code>	7
<code> \$temp = \$sortedSurats[\$i];</code>	7
<code> \$sortedSurats[\$i] = \$sortedSurats[\$minIndex];</code>	7
<code> \$sortedSurats[\$minIndex] = \$temp;</code>	7
<code> }</code>	7
<code>}</code>	7

b) *Flow graph*

Flow graph untuk pengujian unit dengan *white box testing*, seperti yang ditunjukkan pada Gambar 4.9, adalah representasi visual dari semua jalur logika dalam algoritma *Selection Sort* untuk proses pengurutan data surat keluar. *Flow graph* ini terdiri dari *node* dan *edges*, di mana setiap *node* mewakili satu atau lebih pernyataan dalam kode, seperti inisialisasi variabel, perbandingan elemen, atau pertukaran elemen. Setiap *edge* merepresentasikan aliran kontrol dari satu langkah logika ke langkah berikutnya. Dalam pengujian *white box* menggunakan teknik *basis path testing*, *flow graph* membantu mengidentifikasi semua jalur independen, seperti jalur ketika elemen terkecil ditemukan, jalur saat elemen perlu ditukar, atau jalur tanpa pertukaran. Dengan memastikan setiap jalur eksekusi dalam algoritma *Selection Sort* diuji, pengujian ini memberikan jaminan bahwa semua kondisi logis dan kontrol dalam kode telah terverifikasi, sehingga memperkecil risiko kesalahan atau jalur yang tidak teruji.

- *Cyclomatic Complexity* ($V(G)$)

Cyclomatic complexity adalah metrik yang digunakan untuk mengukur kompleksitas sebuah program dengan menghitung jumlah jalur independen dalam kode. Rumus yang digunakan adalah sebagai berikut:

Cyclomatic complexity dihitung dengan rumus:

- $V(G) = \text{Jumlah region} = 2$
- $V(G) = E - N + 2 = 8 - 7 + 2 = 3$

Keterangan :

E = Jumlah *Edge*

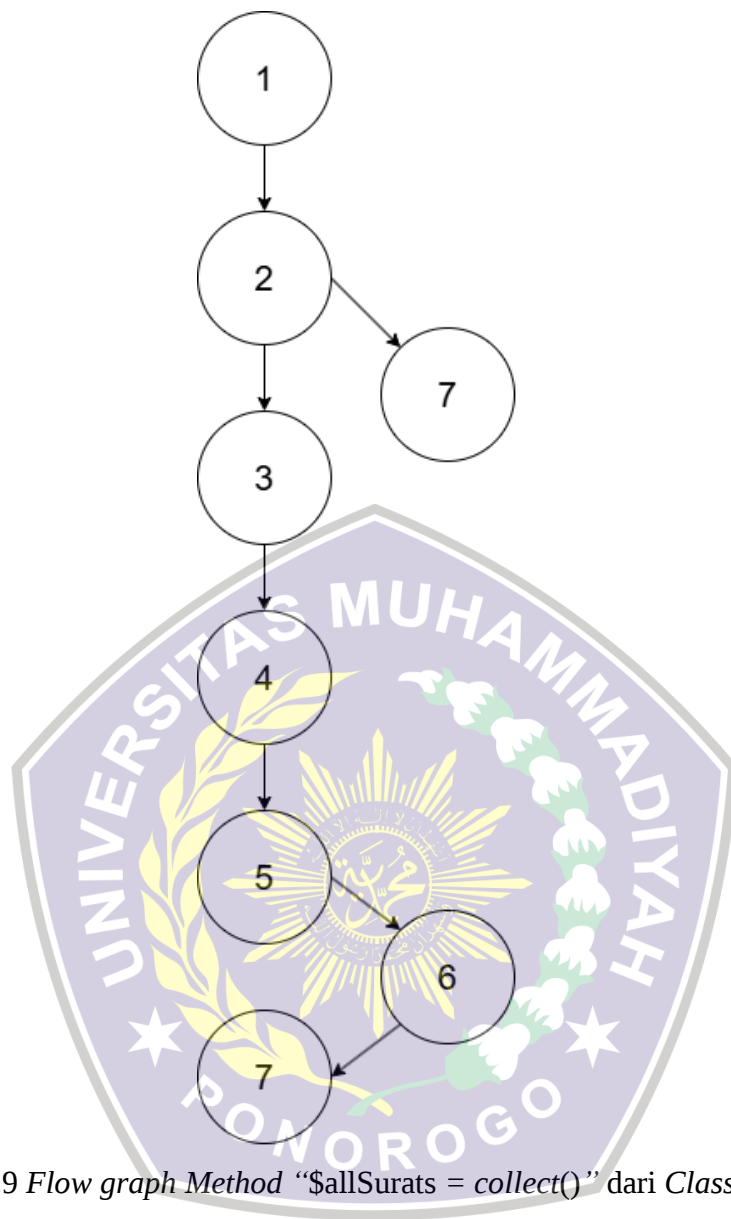
N = Jumlah *Node*

- $V(G) = P + 1 = 2 + 1 = 3$

Semua metode memberikan hasil yang konsisten yaitu $V(G) = 3$. *Cyclomatic complexity* sebesar 2 menunjukkan bahwa ada dua jalur independen yang perlu diuji untuk cakupan penuh dari metode tersebut.

- *Independent Path*

- Jalur 1: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7$
- Jalur 2: $1 \rightarrow 2 \rightarrow 7$
- Jalur 3: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$



Gambar 4.9 Flow graph Method “\$allSurats = collect()” dari Class “Surat”