BAB III

METODOLOGI DAN PERANCANGAN

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini antara lain adalah sebagai berikut.

1. Studi Literatur

   Melakukan studi mengenai teori-teori, konsep, serta penelitian sebelumnya yang berkaitan dengan pokok bahasan penelitian, antara lain mengenai kompresi data, algoritma LZ77, algoritma *Huffman*, algoritma *Deflate*, bahasa pemrograman PHP, dan basis data MySQL. Referensi-referensi yang digunakan dapat berupa buku, jurnal ilmiah, artikel, dan lain-lain yang terdapat pada media cetak maupun media internet.

2. Pengumpulan Kebutuhan

   Melakukan pengumpulan data mengenai hal-hal yang dibutuhkan yaitu website berbasis PHP dan basis data MySQL. Pengumpulan data dilakukan dengan mengajukan permohonan ke pemilik website.

3. Analisis dan Perancangan

   Melakukan analisis dan perancangan terhadap aplikasi yang dibangun meliputi analisis hasil pemahaman berdasarkan studi literatur dan kebutuhan, aplikasi yang dibuat, perancangan alur kerja aplikasi dalam bentuk *flowchart*, serta rancangan tampilan antarmuka aplikasi.

4. Implementasi

Melakukan pembangunan terhadap aplikasi dengan mengimplementasikan hasil rancangan yang telah didefinisikan sebelumnya dengan menggunakan bahasa pemrograman PHP.

5. Uji Coba dan Evaluasi

Melakukan uji coba terhadap aplikasi yang telah dibuat. Kemudian, melakukan evaluasi terhadap hasil uji coba tersebut.

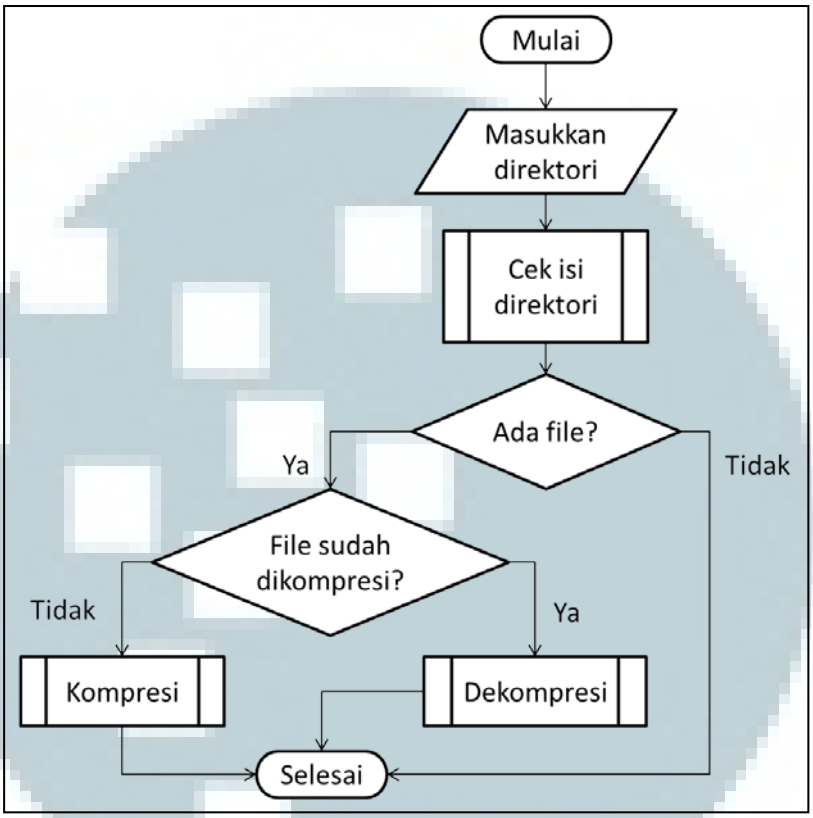
3.2 Perancangan

Aplikasi yang dirancang mengimplementasikan teknik kompresi data menggunakan algoritma *Deflate* ke website yang diinginkan. Aplikasi yang dirancang dapat melakukan proses kompresi juga dekompresi banyak file dalam suatu direktori. Aplikasi yang dirancang tidak memeriksa tipe blok karena kompresi dan dekompresi dilakukan dengan kode *Huffman* dinamis.

Masukan aplikasi berupa direktori website yang diimplementasikan. Keluaran aplikasi berupa file pada direktori tersebut sudah terkompres jika melakukan proses kompresi atau file pada direktori tersebut sudah terdekompres jika melakukan proses dekompresi. Setelah proses selesai, ditampilkan tabel yang berisi daftar file beserta ukuran sesudah dan sebelum proses.

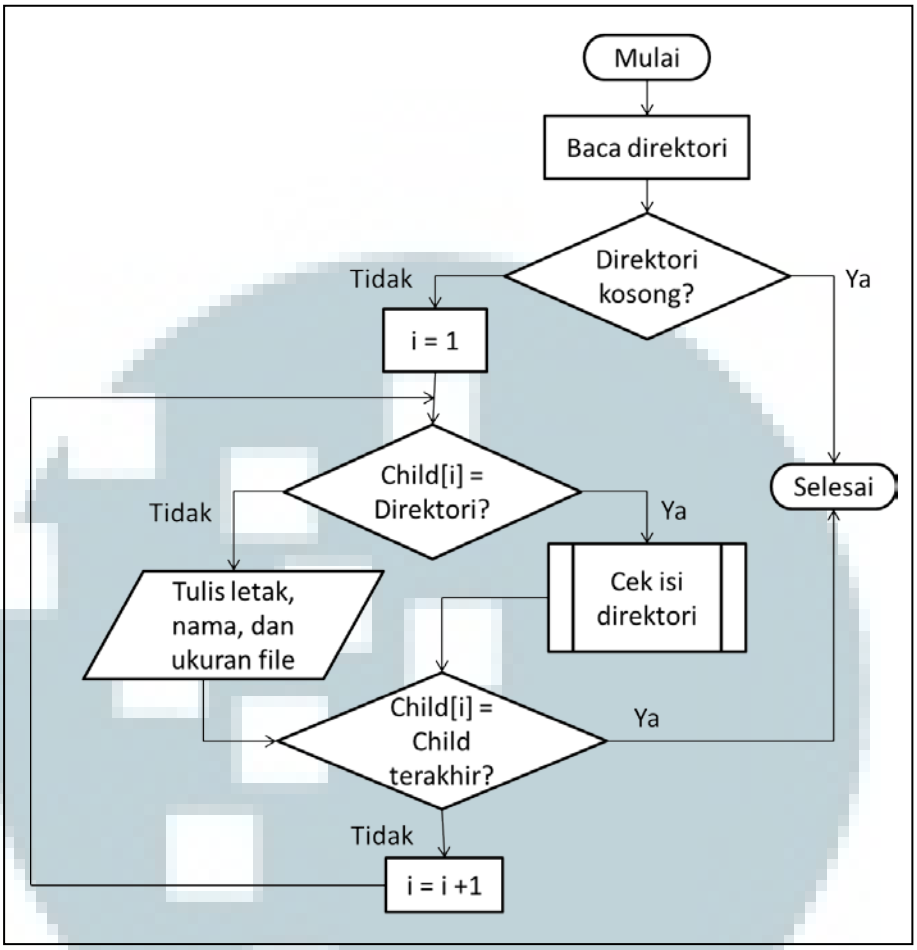
3.2.1 Perancangan Aplikasi

Rancangan aplikasi secara keseluruhan dapat dilihat pada gambar 3.1.



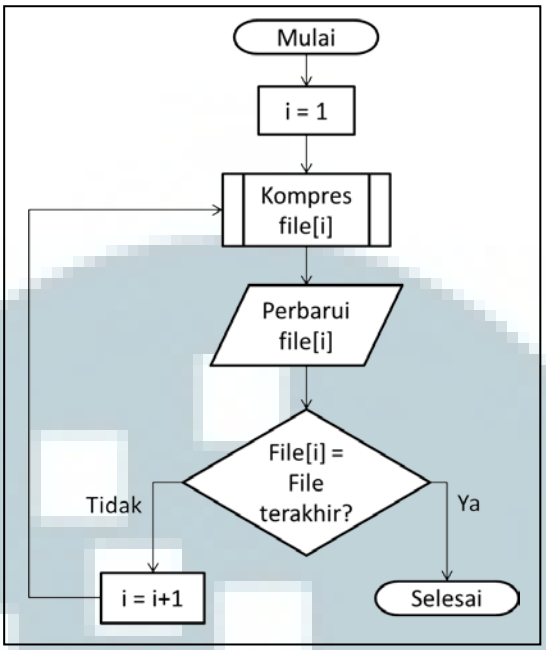
Gambar 3.1 *Flowchart* aplikasi

Pertama-tama aplikasi meminta masukan direktori website dari pengguna. Setelah itu aplikasi akan memeriksa isi direktori. Proses cek isi direktori akan dijelaskan lebih lanjut pada gambar 3.2. Jika tidak ada file di direktori, aplikasi selesai. Sebaliknya, jika ada file di direktori, pengguna akan diminta memilih ingin melakukan proses kompresi atau dekompresi file. Proses kompresi akan dijelaskan lebih lanjut pada gambar 3.3. Proses dekompresi akan dijelaskan lebih lanjut pada gambar 3.10. Setelah proses kompresi atau dekompresi selesai, aplikasi selesai.



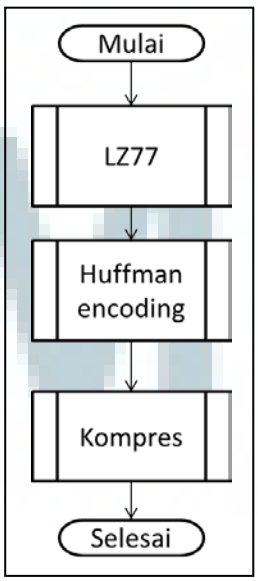
Gambar 3.2 *Flowchart* cek isi direktori

Pada proses cek isi direktori, pertama-tama aplikasi membaca direktori dan memeriksa direktori kosong atau tidak. Jika direktori kosong, proses selesai dan mengembalikan nilai *false*. Jika direktori tidak kosong, aplikasi memeriksa setiap *child* direktori. Jika *child* direktori adalah direktori, aplikasi memeriksa isi direktori tersebut. Jika *child* direktori adalah file, aplikasi menulis letak, nama, dan ukuran file tersebut. Setelah itu, aplikasi memeriksa apakah *child* ini merupakan *child* terakhir. Jika ya, proses selesai, dan mengembalikan nilai *true*. Jika tidak, periksa *child* selanjutnya.



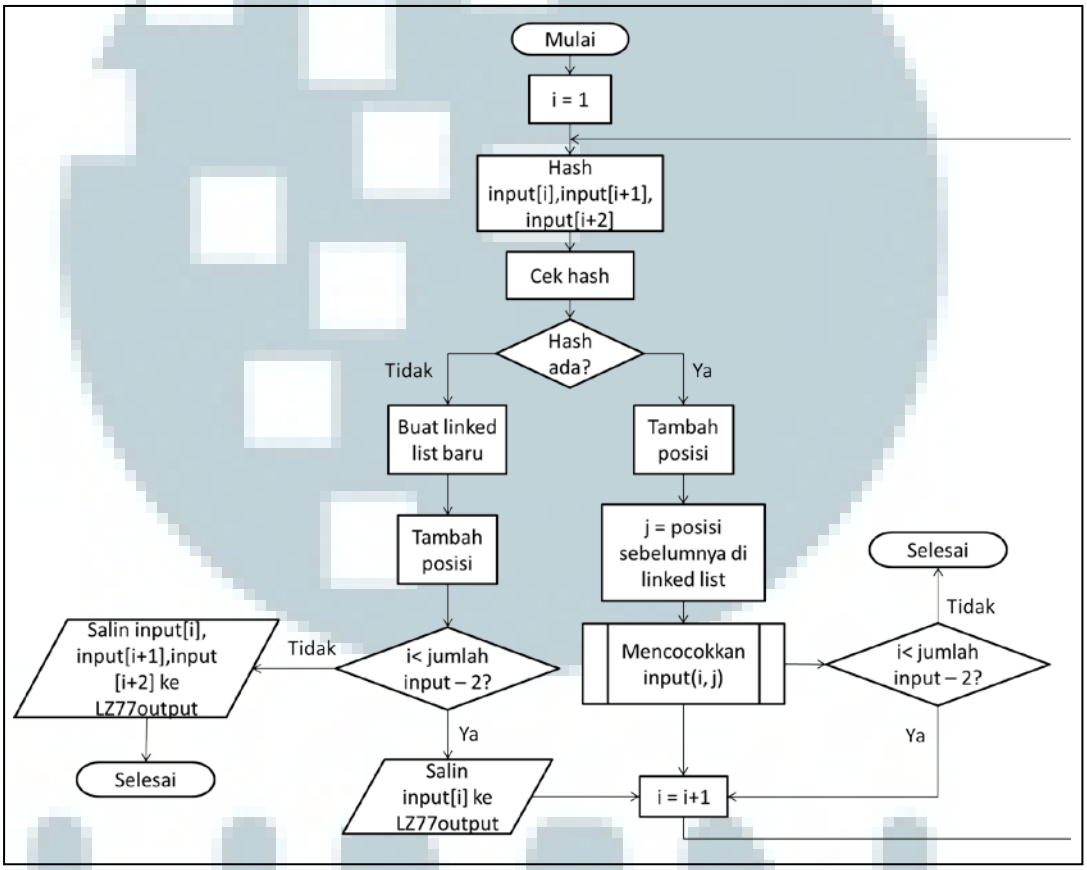
Gambar 3.3 *Flowchart* kompresi

Pada proses kompresi, aplikasi akan melakukan proses kompres file yang akan dijelaskan lebih lanjut pada gambar 3.4. Setelah melakukan proses kompres file, aplikasi akan memperbarui isi file. Setelah memperbarui isi file, aplikasi akan memeriksa file yang telah dikompres file terakhir atau bukan. Jika ya, proses selesai. Jika tidak, kompres dan perbarui file selanjutnya.



Gambar 3.4 *Flowchart* kompres file

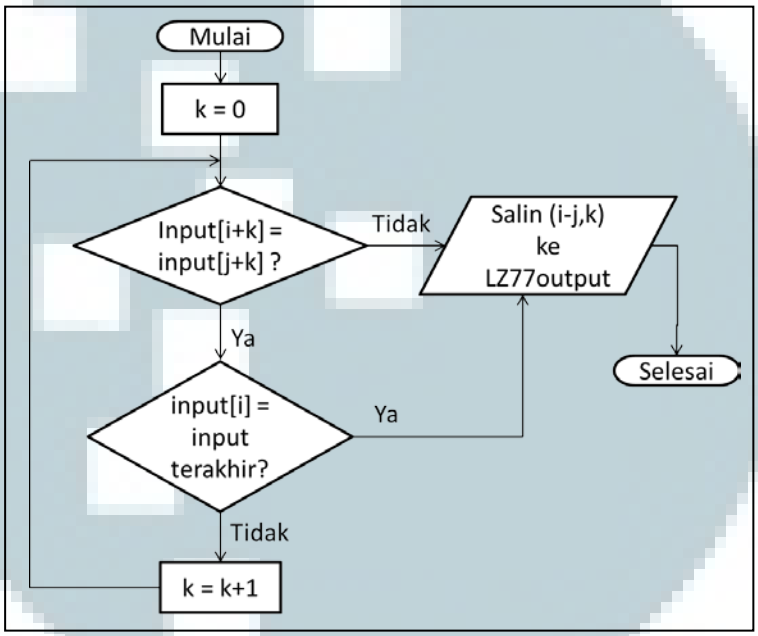
Pada proses kompres file, aplikasi melakukan 3 proses. Pertama, aplikasi mengkompres masukan menggunakan algoritma LZ77, akan dijelaskan lebih lanjut pada gambar 3.5. Setelah itu, masing-masing simbol hasil keluaran LZ77 dikompres menggunakan algoritma *Huffman*, akan dijelaskan lebih lanjut pada gambar 3.7. Terakhir, hasil keluaran LZ77 dikompres menggunakan hasil algoritma *Huffman*, akan dijelaskan lebih lanjut pada gambar 3.9.



Gambar 3.5 *Flowchart* LZ77

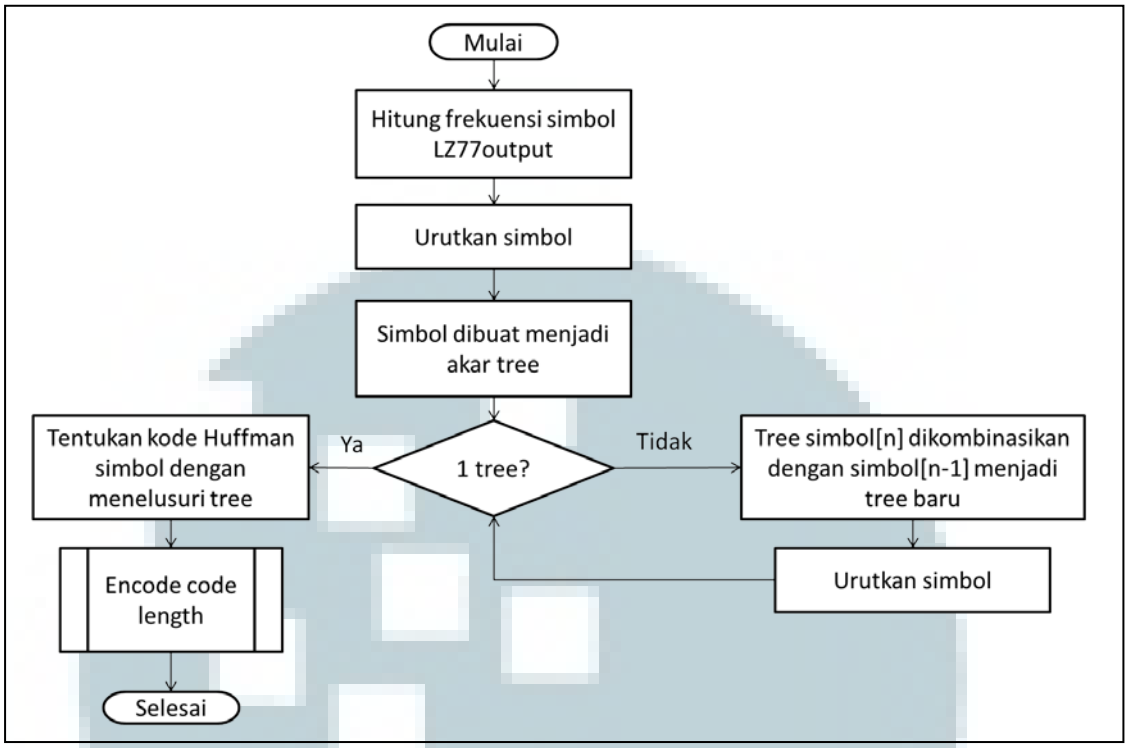
Pada proses LZ77, aplikasi melakukan *hash* 3 simbol dan memeriksanya. Jika *hash* tidak ada, aplikasi membuat *linked list* baru untuk *hash* tersebut, menambah posisi i, dan memeriksa apakah i = jumlah input – 2. Jika ya, aplikasi menyalin 3 simbol tersebut ke keluaran. Jika tidak, aplikasi hanya menyalin simbol pertama dan memeriksa *hash* simbol selanjutnya. Jika *hash* ada, aplikasi

menambah posisi i ke dalam *linked list*, j berisi posisi sebelumnya di *linked list*, lalu mencocokkan input dari posisi i dengan input dari posisi j, lalu memeriksa apakah i = jumlah input – 2. Jika ya, proses selesai. Jika tidak, aplikasi memeriksa *hash* simbol selanjutnya. Proses mencocokkan input akan dijelaskan lebih lanjut pada gambar 3.6.



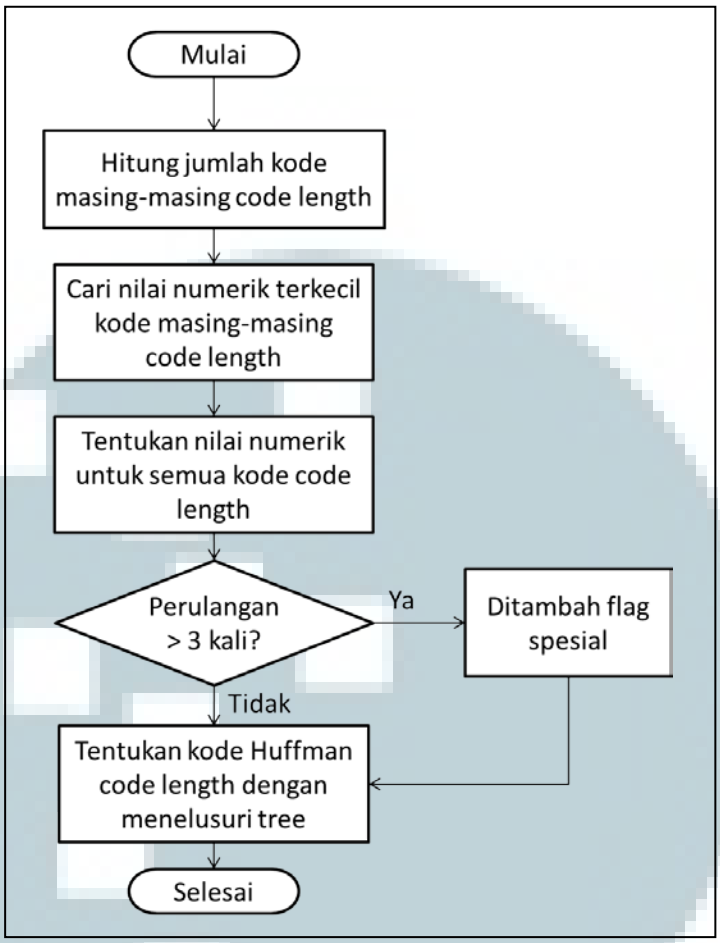
Gambar 3.6 *Flowchart* mencocokkan input

Pada proses mencocokkan input, aplikasi mencocokkan input dari posisi i dan input dari posisi j. Pertama, aplikasi menginisialisasi iterator k = 0. Setelah itu aplikasi mencocokkan tiap simbol input[i+k] dengan input[j+k]. Jika tidak cocok, aplikasi menyalin (i-j, k) ke keluaran. Jika cocok, aplikasi memeriksa apakah input pada posisi i merupakan input terakhir. Jika ya, aplikasi menyalin (i-j, k) ke keluaran. Jika tidak, aplikasi mencocokkan simbol selanjutnya.



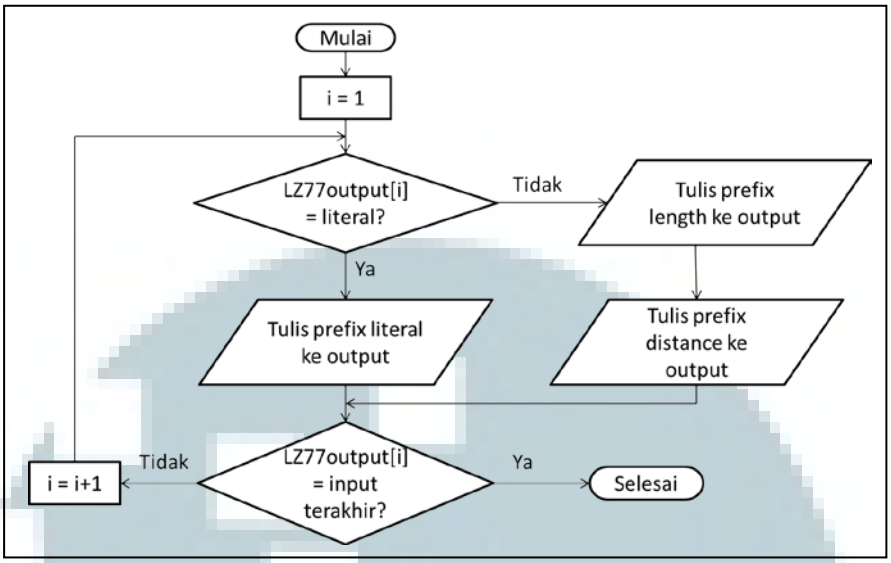
Gambar 3.7 *Flowchart Huffman encoding*

Pada proses *Huffman encoding*, aplikasi menghitung frekuensi masing-masing simbol hasil proses LZ77, lalu mengurutkannya berdasarkan frekuensi secara *descending* dan leksikografi secara *ascending*. Setelah simbol terurut, simbol-simbol tersebut diubah menjadi akar sebuah tree. Jika tree yang tersisa lebih dari 1, tree simbol di posisi terakhir dikombinasikan dengan tree simbol di posisi terakhir-1 menjadi tree baru, di mana frekuensi untuk tree baru ini total frekuensi dari kedua tree yang dikombinasikan. Kemudian, tree hasil kombinasi diletakkan di posisi terdepan untuk frekuensi yg sesuai. Jika tree yang tersisa 1, aplikasi menentukan kode *Huffman* simbol dengan menelusuri tree tadi. Terakhir, aplikasi melakukan proses *encode code length* dan proses selesai. Proses *encode code length* akan dijelaskan lebih lanjut pada gambar 3.8.



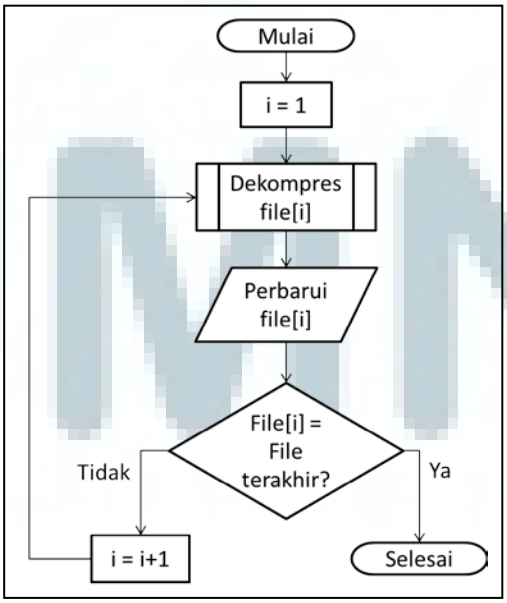
Gambar 3.8 *Flowchart encode code length*

Pada proses *encode code length*, pertama aplikasi menghitung jumlah kode untuk masing-masing *code length*. Setelah itu, aplikasi mencari nilai numerik terkecil kode masing-masing *code length*. Kemudian, aplikasi menentukan nilai numerik untuk semua kode *code length* menggunakan nilai berturut-turut semua kode dengan panjang yang sama dengan nilai dasar yang didapatkan sebelumnya. Ketika terjadi perulangan 3 kali untuk suatu *code length*, aplikasi menambahkan *flag* spesial 16 untuk *code length* selain 0 dan *flag* spesial 17 atau 18 untuk *code length* 0. Terakhir, aplikasi menentukan kode *Huffman code length* dengan menelusuri tree dan proses selesai.



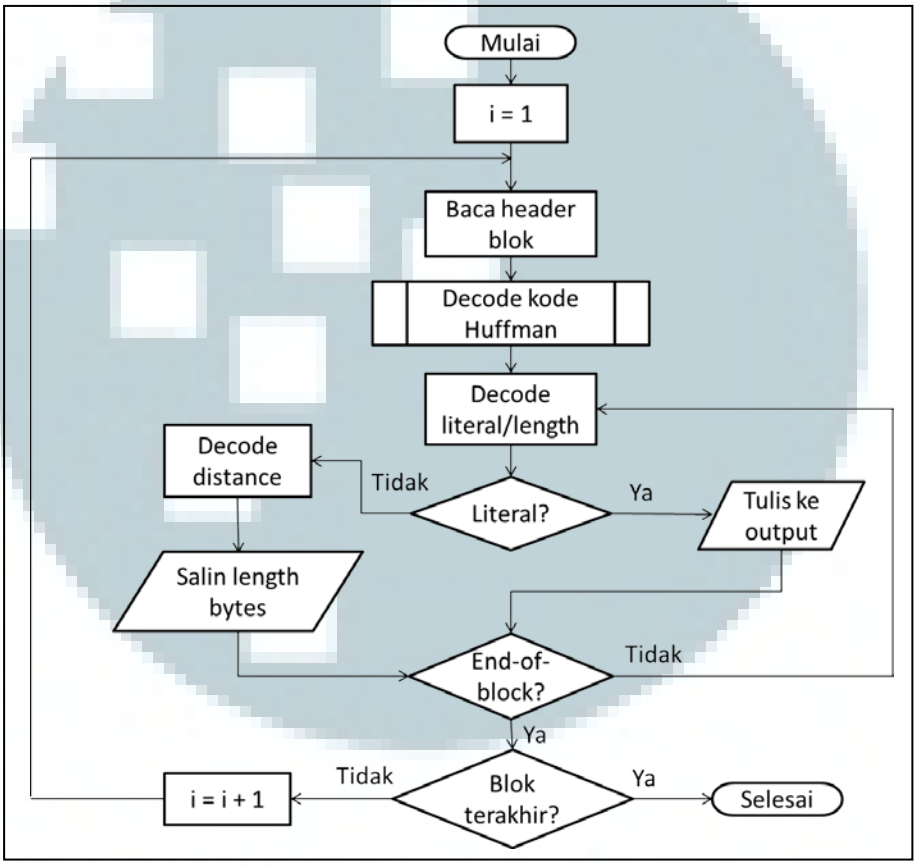
Gambar 3.9 *Flowchart* kompres

Pada proses kompres, pertama aplikasi memeriksa input yang merupakan hasil output LZ77. Jika input merupakan *literal*, aplikasi menulis kode *Huffman* untuk *literal*. Jika tidak, aplikasi menulis kode *Huffman* untuk *length*, kode *Huffman* untuk *distance*, dan kode *Huffman* untuk *literal*. Setelah itu aplikasi memeriksa apakah sudah mencapai input terakhir. Jika ya, proses selesai. Jika tidak, periksa input selanjutnya.



Gambar 3.10 *Flowchart* dekompresi

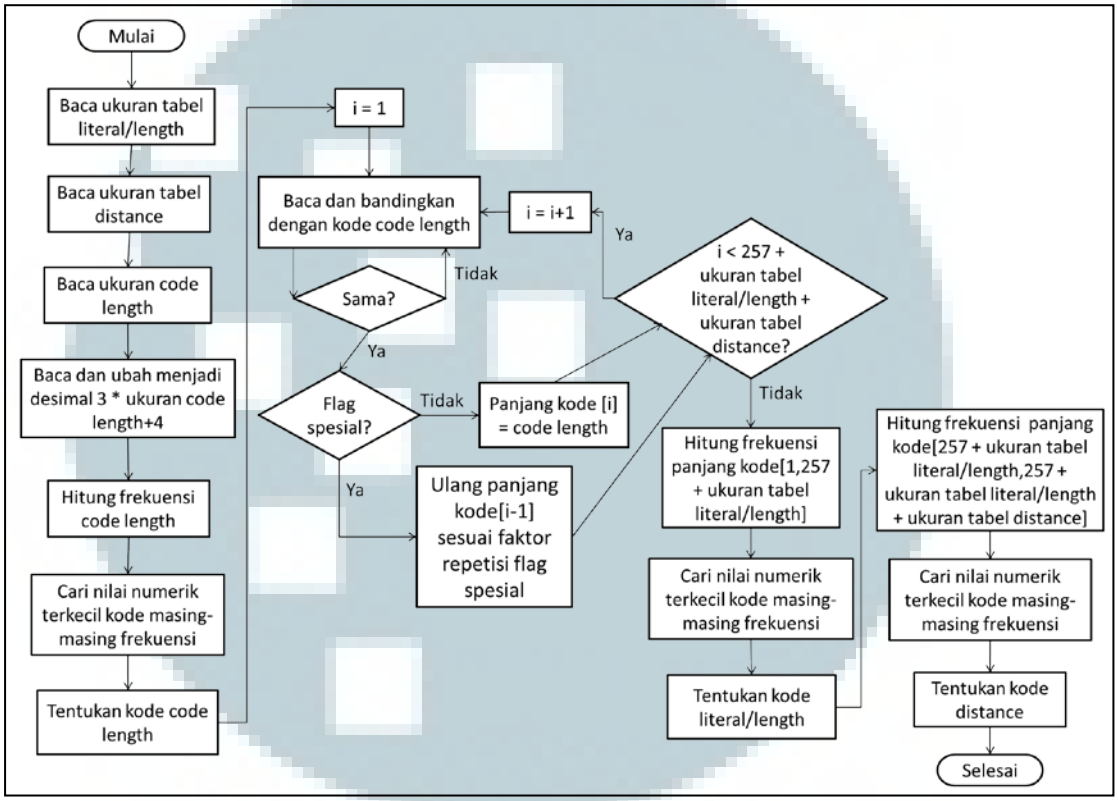
Pada proses dekompresi, aplikasi akan melakukan proses dekompres file yang akan dijelaskan lebih lanjut pada gambar 3.11. Setelah melakukan proses dekompres file, aplikasi akan memperbarui isi file. Setelah memperbarui isi file, aplikasi akan memeriksa file yang telah didekompres file terakhir atau bukan. Jika ya, proses selesai. Jika tidak, dekompres dan perbarui file selanjutnya.



Gambar 3.11 *Flowchart* dekompres file

Pada proses dekompres file, aplikasi mendekompres per blok. Pertama, aplikasi membaca *header* blok. Setelah itu, aplikasi melakukan *decode* kode *Huffman*. Proses *decode* kode *Huffman* akan dijelaskan lebih lanjut pada gambar 3.12. Setelah mendapatkan tabel *literal/length* dan *distance*, aplikasi melakukan *decode literal/length*. Jika *literal*, *literal* tersebut ditulis ke keluaran. Jika bukan *literal*, berarti *length*, aplikasi melakukan *decode distance* dan menyalin *length*

bytes ke keluaran. Kemudian, aplikasi memeriksa apakah sudah akhir blok. Jika tidak, *decode literal/length* selanjutnya. Jika ya, aplikasi memeriksa apakah sudah blok terakhir. Jika ya, proses selesai. Jika tidak, aplikasi membaca *header* blok selanjutnya.



Gambar 3.12 *Flowchart decode* kode *Huffman*

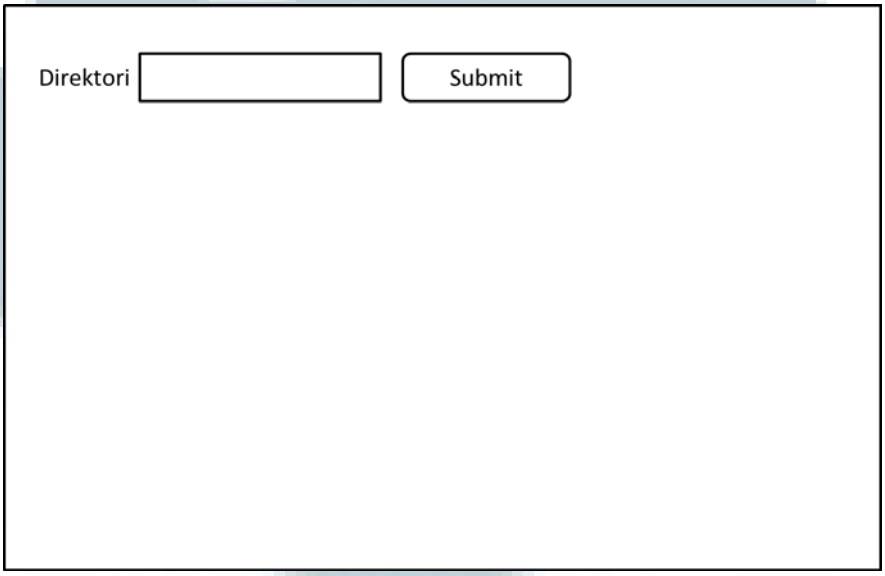
Pada proses *decode* kode *Huffman*, aplikasi membaca ukuran tabel *literal/length*, tabel *distance*, dan *code length* lalu melakukan *decode code length*, *decode* tabel *literal/length*, dan *decode* tabel *distance* untuk mendapatkan kode *literal/length* dan *distance*. Pertama, aplikasi membaca ukuran tabel *literal/length*, tabel *distance*, dan *code length*. Setelah itu, aplikasi membaca dan mengubah menjadi desimal 3 bit sesuai dengan ukuran *code length* ditambah dengan 4. Kemudian, aplikasi menghitung frekuensi *code length* dan mencari nilai numerik

terkecil kode masing-masing frekuensi untuk menentukan kode *code length*. Selanjutnya, aplikasi membaca bit per bit selanjutnya sesuai dengan panjang kode *code length* dan membandingkannya dengan kode *code length*. Jika bit per bit yang sesuai dengan panjang kode *code length* tidak sama dengan kode *code length*, bit per bit disesuaikan dengan panjang kode *code length* lainnya dan dibandingkan dengan kode *code length* tersebut. Jika bit per bit yang sesuai dengan panjang kode *code length* sama dengan kode *code length*, aplikasi memeriksa *code length* tersebut *flag* spesial atau bukan. Jika *code length flag* spesial, aplikasi mengulang panjang kode index i dikurang dengan 1 ke panjang kode index selanjutnya sesuai dengan faktor repetisi *flag* spesial. Jika *code length* bukan *flag* spesial, panjang kode index i adalah *code length*. Setelah itu, aplikasi memeriksa index i. Jika index i lebih kecil dari 257 ditambah dengan ukuran tabel *literal/length* ditambah dengan ukuran tabel *distance*, index i ditambah 1. Jika index i lebih besar sama dengan 257 ditambah dengan ukuran tabel *literal/length* ditambah dengan ukuran tabel *distance*, aplikasi menghitung frekuensi panjang kode index 1 sampai 257 ditambah dengan ukuran tabel *literal/length* dan mencari nilai numerik terkecil kode masing-masing frekuensi untuk menentukan kode *literal/length*. Terakhir, aplikasi menghitung frekuensi panjang kode index 257 ditambah dengan ukuran tabel *literal/length* sampai 257 ditambah dengan ukuran tabel *literal/length* ditambah dengan ukuran tabel *distance* dan mencari nilai numerik terkecil kode masing-masing frekuensi untuk menentukan kode *distance* dan proses selesai.

3.2.2 Perancangan Antarmuka

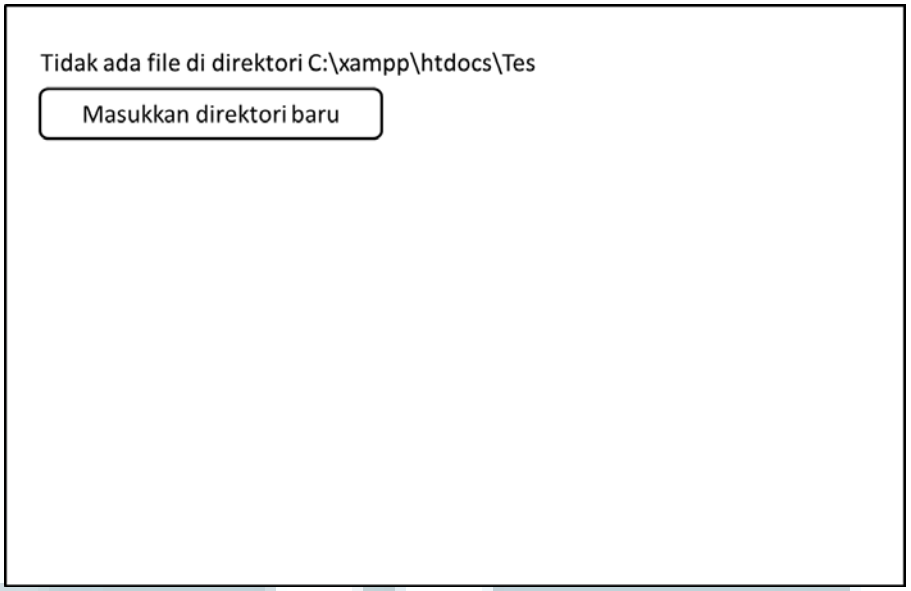
Desain tampilan antarmuka untuk aplikasi ini dibagi menjadi 7 yaitu halaman input direktori, halaman tidak ada file di direktori, halaman ada file di direktori, halaman proses kompresi, halaman proses kompresi selesai, halaman proses dekompresi, dan halaman proses dekompresi selesai.

Tampilan pertama yang ditampilkan saat menjalankan aplikasi adalah halaman untuk input direktori ditunjukkan pada gambar 3.13.



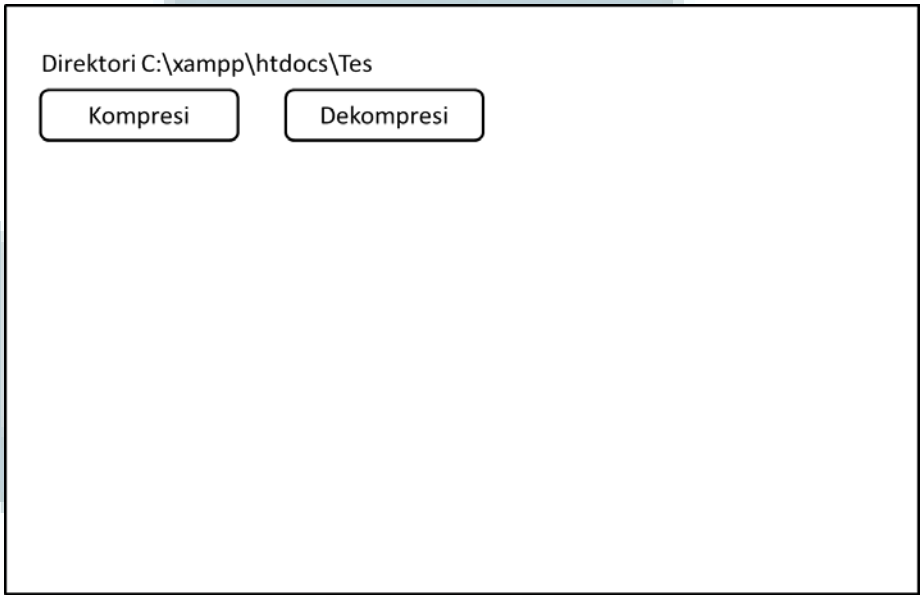
Gambar 3.13 Desain tampilan antarmuka halaman input direktori

Pada halaman input direktori, terdapat sebuah *text*, *text input*, dan sebuah *button*. *Text input* untuk memasukkan direktori website yang diinginkan misalnya "C:\xampp\htdocs\Tes". Di sebelah *text input*, ada sebuah *button* "Submit" yang ketika diklik akan memeriksa file di direktori tersebut ada atau tidak. Jika tidak ada, akan ditampilkan halaman tidak ada file di direktori ditunjukan pada gambar 3.14. Jika ada, akan ditampilkan halaman ada file di direktori ditunjukkan pada gambar 3.15.



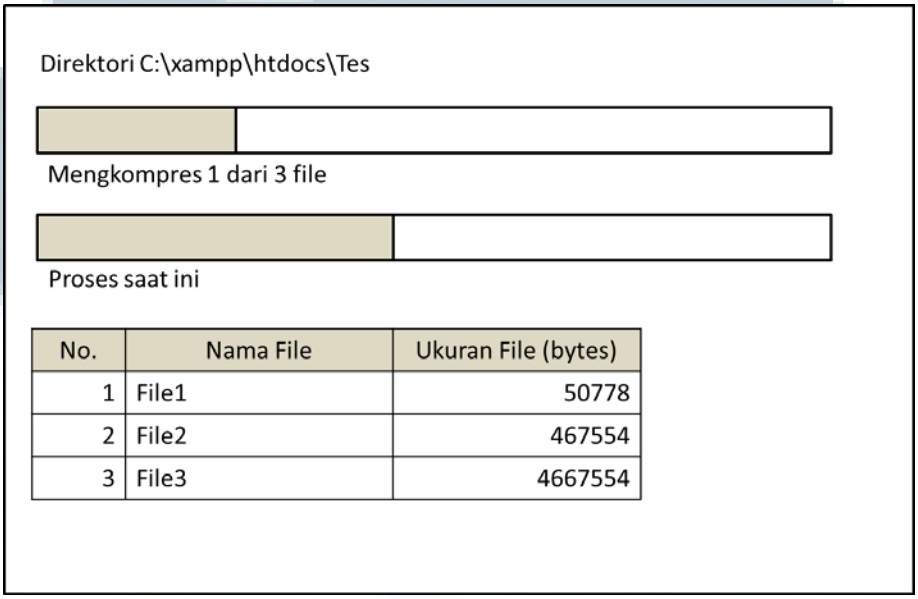
Gambar 3.14 Desain tampilan antarmuka halaman tidak ada file di direktori

Pada halaman tidak ada file di direktori, terdapat sebuah *text* dan *button*. *Text* memberitahukan bahwa tidak ada file di direktori yang dimasukkan. *Button* "Masukkan direktori baru", ketika diklik, akan menampilkan kembali halaman input direktori ditunjukkan pada gambar 3.13.



Gambar 3.15 Desain tampilan antarmuka halaman ada file di direktori

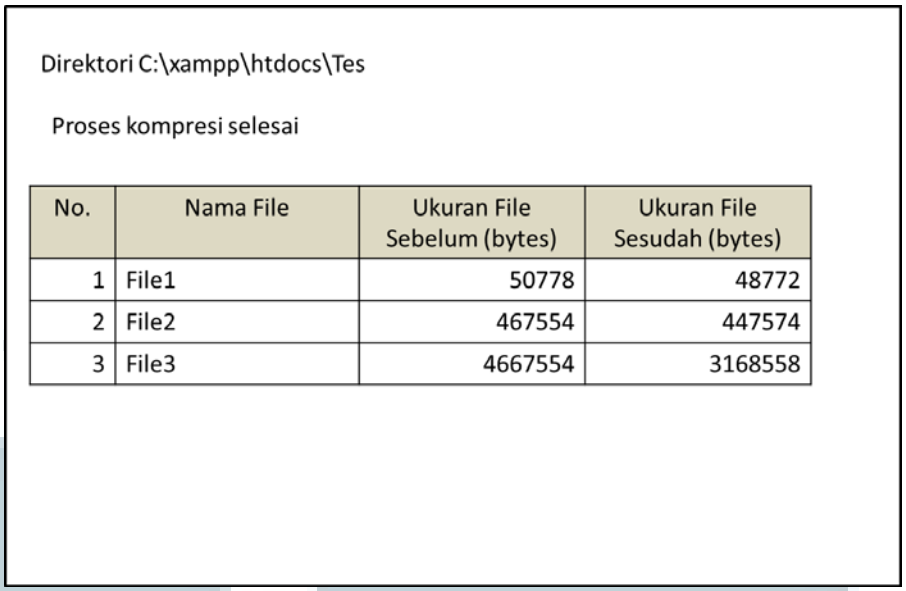
Pada halaman ada file di direktori, terdapat sebuah *text* dan dua buah *button*. *Text* memberitahukan bahwa ada file di direktori yang dimasukkan. *Button* "Kompresi", ketika diklik, akan menjalankan proses kompresi file pada direktori yang telah dimasukkan dan menampilkan halaman proses kompresi ditunjukkan pada gambar 3.16. *Button* "Dekompresi", ketika diklik, akan menjalankan proses dekompresi file pada direktori yang telah dimasukkan dan menampilkan halaman proses dekompresi ditunjukkan pada gambar 3.18.



Gambar 3.16 Desain tampilan antarmuka halaman proses kompresi

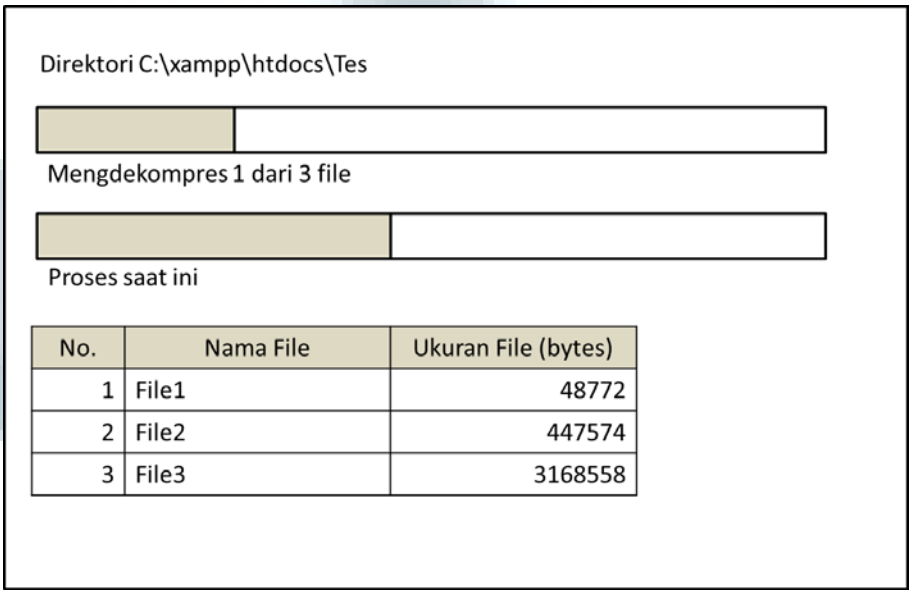
Pada halaman proses kompresi, terdapat tiga buah *text*, dua buah *progress bar*, dan sebuah tabel. *Text* pertama menampilkan direktori yang dimasukkan. *Text* kedua menampilkan file ke berapa yang sedang dan sudah dikompres. *Progress bar* pertama menampilkan *progress* proses kompresi semua file, sedangkan *progress bar* kedua menampilkan *progress* proses kompresi file yang sedang dikompresi. Tabel menampilkan daftar file yang dikompres beserta ukuran

sebelum proses kompresi. Setelah proses kompresi selesai, ditampilkan halaman proses kompresi selesai ditunjukkan pada gambar 3.17.



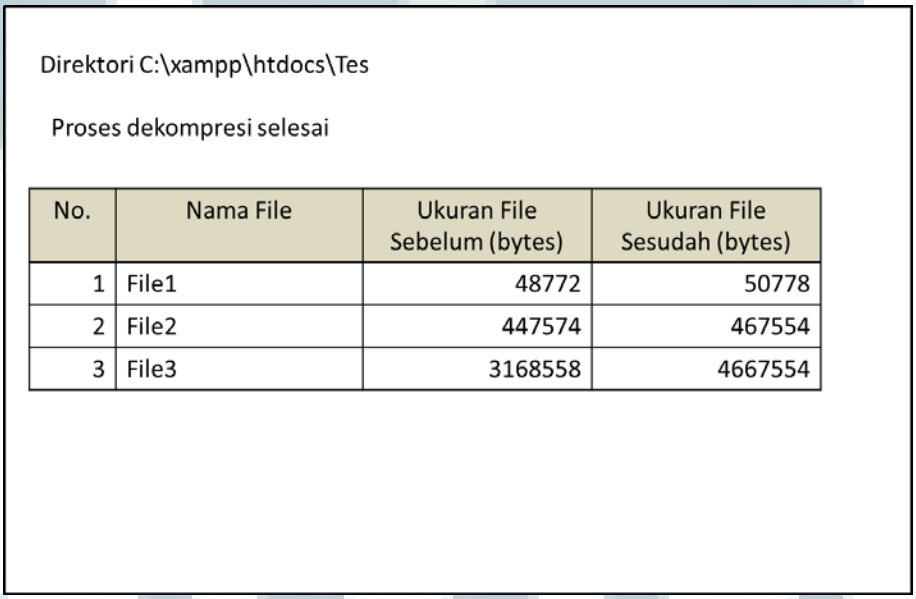
Gambar 3.17 Desain tampilan antarmuka halaman proses kompresi selesai

Pada halaman proses kompresi selesai, terdapat dua buah *text* dan sebuah tabel. *Text* pertama menampilkan direktori yang dimasukkan. Tabel menampilkan daftar file yang dikompres beserta ukuran sebelum dan sesudah proses kompresi.



Gambar 3.18 Desain tampilan antarmuka halaman proses dekompresi

Pada halaman proses dekompresi, terdapat tiga buah *text*, dua buah *progress bar*, dan sebuah tabel. *Text* pertama menampilkan direktori yang dimasukkan. *Text* kedua menampilkan file ke berapa yang sedang dan sudah didekompres. *Progress bar* pertama menampilkan *progress* proses dekompresi semua file, sedangkan *progress bar* kedua menampilkan *progress* proses dekompresi file yang sedang didekompresi. Tabel menampilkan daftar file yang didekompres beserta ukuran sebelum proses dekompresi. Setelah proses dekompresi selesai, ditampilkan halaman proses dekompresi selesai ditunjukkan pada gambar 3.19.



Gambar 3.19 Desain tampilan antarmuka halaman proses dekompresi selesai

Pada halaman proses dekompresi selesai, terdapat dua buah *text* dan sebuah tabel. *Text* pertama menampilkan direktori yang dimasukkan. Tabel menampilkan daftar file yang didekompres beserta ukuran sebelum dan sesudah proses dekompresi.