3. ANALISA DAN DESAIN SISTEM

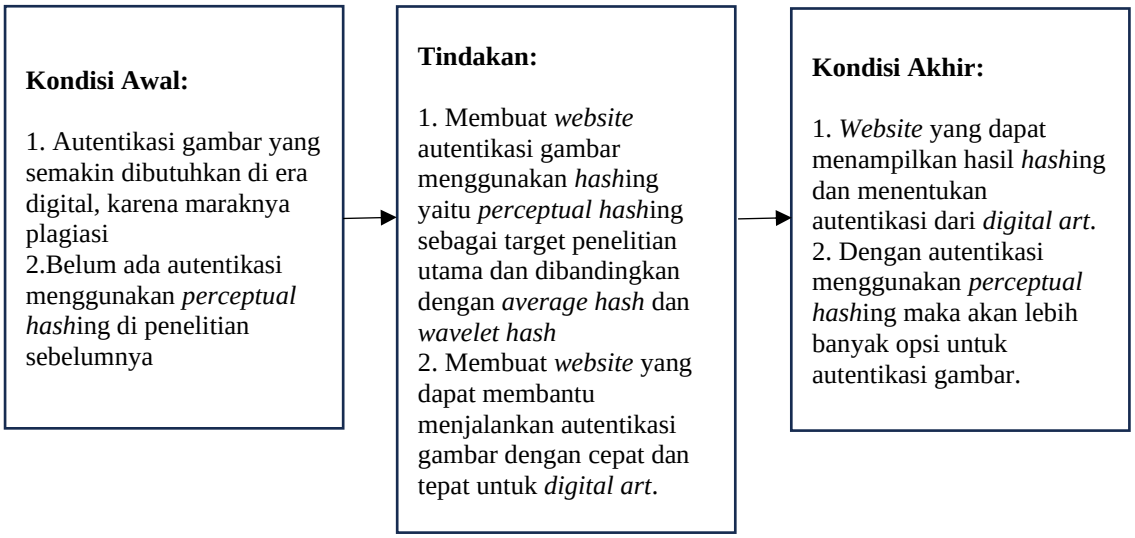
3.1 Kerangka Pemikiran

Buku skripsi ini mengusung eksplorasi tentang penggunaan ambang batas pada proses autentikasi *digital art* dengan memanfaatkan algoritma *perceptual hashing*, suatu metode yang masih jarang dieksplorasi dalam konteks ini. Penelitian sebelumnya belum banyak menggarap aspek ini, sehingga penelitian ini menjadi penting untuk mengisi celah pengetahuan tersebut. Fokus utama penelitian adalah untuk mengevaluasi dan membandingkan efektivitas *perceptual hashing* dengan metode autentikasi lain yang telah ada. Dalam proses ini, penelitian akan melibatkan serangkaian pengujian langsung terhadap algoritma *perceptual hashing*, yang kemudian akan dibandingkan dengan metode *hashing* lainnya yang umum digunakan dalam autentikasi *digital art*.

Dengan demikian, tujuan utama dari penelitian ini adalah untuk mendapatkan pemahaman yang lebih dalam tentang potensi dan keterbatasan *perceptual hashing* dalam konteks autentikasi *digital art*. *Output* yang diharapkan dari penelitian ini adalah identifikasi metode autentikasi yang paling efektif dengan memanfaatkan teknik *hashing*, dengan fokus pada keandalan dan keefektifan autentikasi karya *digital*. Melalui pengungkapan ini, diharapkan penelitian ini dapat memberikan kontribusi signifikan bagi pemahaman dan pengembangan praktis dalam bidang keamanan *digital art*. Selain itu, hasil dari penelitian ini juga diharapkan dapat memberikan pedoman bagi praktisi dan peneliti di bidang tersebut untuk meningkatkan metode autentikasi yang ada, sehingga dapat lebih efektif dalam melindungi keaslian karya *digital*.

Tabel 3.1

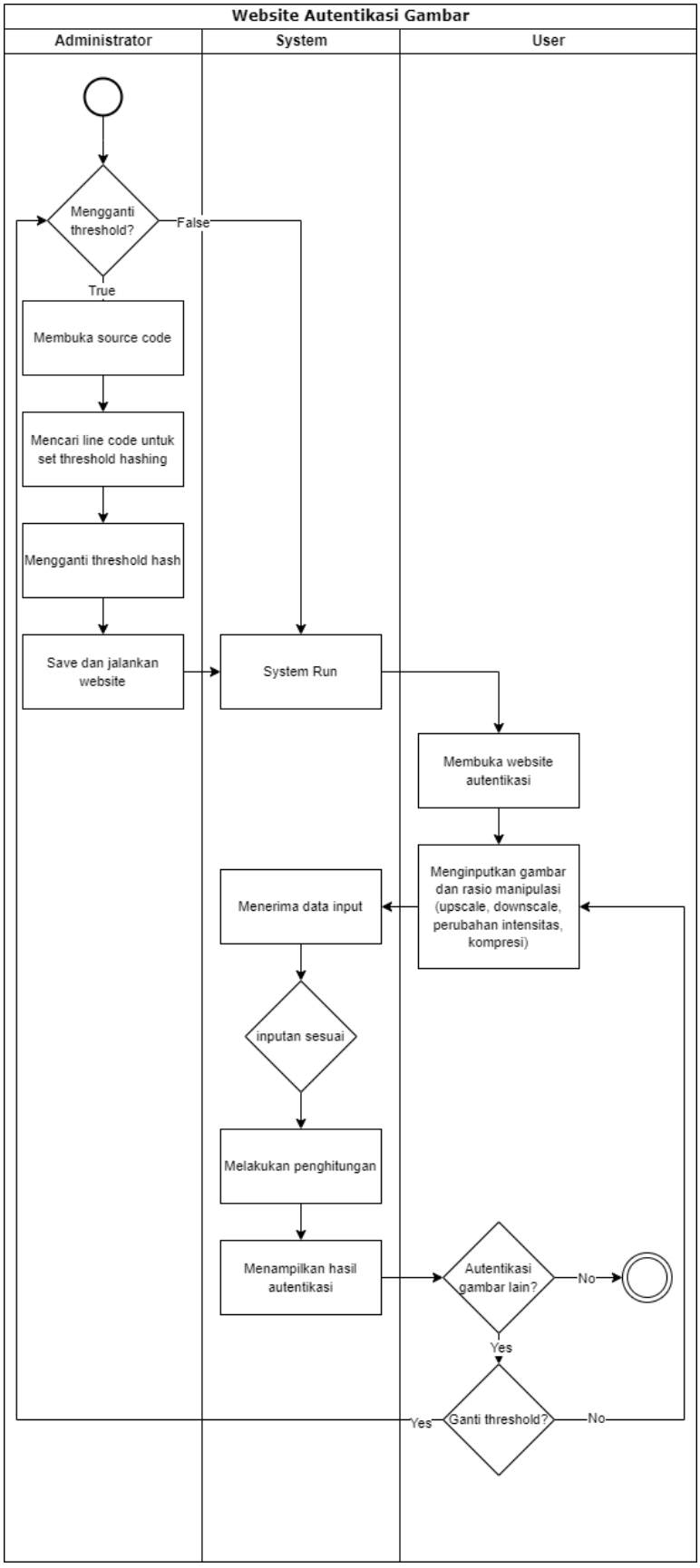
Penjelasan *Website*



3.2 Desain Sistem

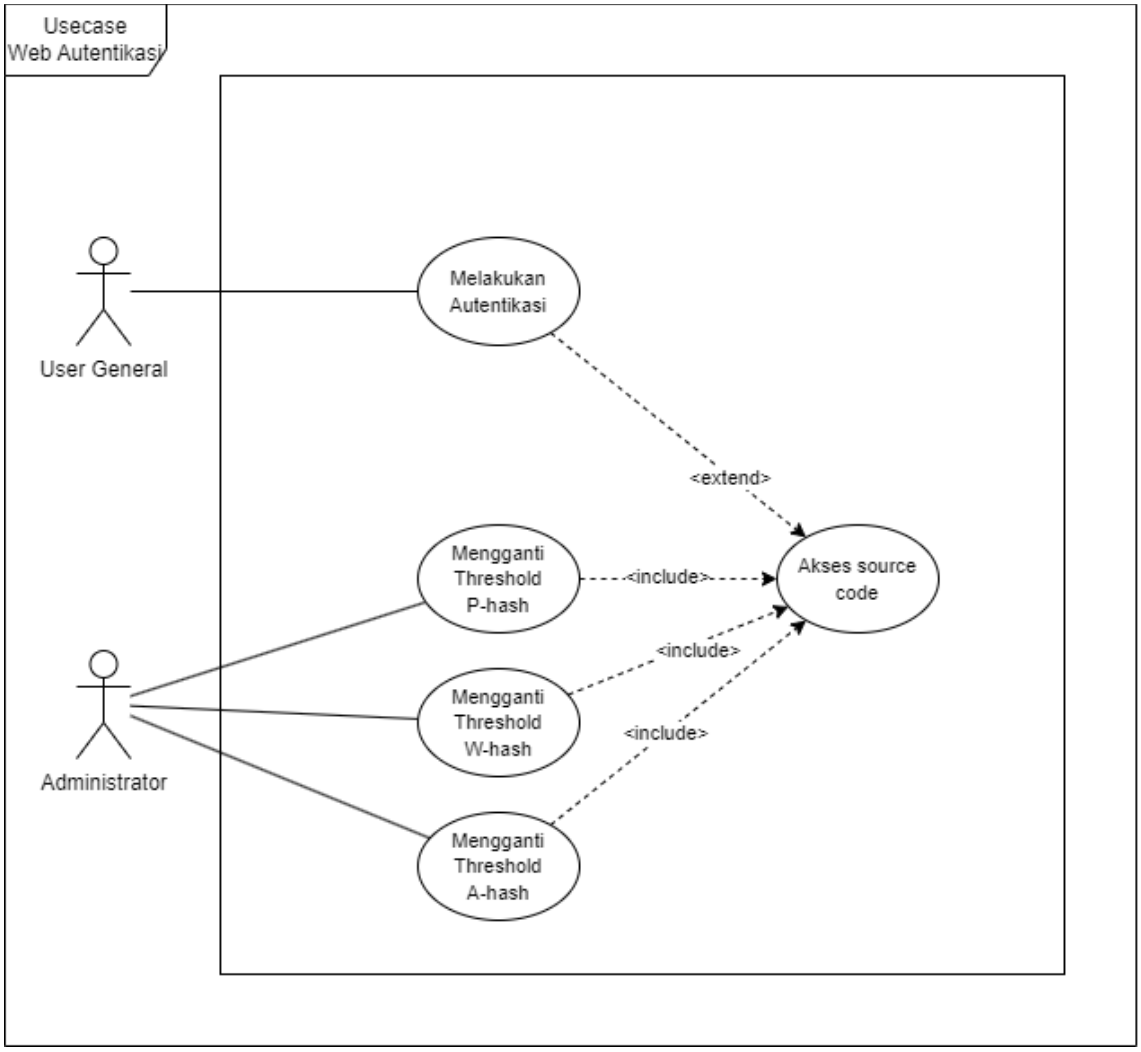
Dalam desain sistem aplikasi *Personal Digital Assistant* (PDA), digambarkan menggunakan *use case*, *flowchart*, dan *activity diagram*. *Use case diagram* yang disusun akan menjelaskan hubungan antara fitur yang dimiliki oleh pengguna. Dalam diagram ini, terdapat dua jenis relasi, yaitu "*Include*" dan "*Extend*". Relasi "*Include*" mengindikasikan bahwa suatu fungsi dapat dijalankan setelah fungsi yang berada di ujung anak panah relasi telah dilakukan. Sementara itu, relasi "*Extend*" menunjukkan bahwa fungsi tersebut merupakan perluasan dari fungsi utama.

Activity diagram digunakan untuk menggambarkan urutan aktivitas yang terjadi pada sistem dan pengguna saat menjalankan fungsi atau aktivitas tertentu. Diagram ini mencakup titik awal dan akhir sebagai state atau kondisi awal dan akhir dari sebuah proses. Dalam diagram tersebut, terdapat aktivitas atau fungsi yang dapat dijalankan oleh pengguna dengan peran sebagai lansia maupun caretaker.



Gambar 3.1 *Activity* Sistem

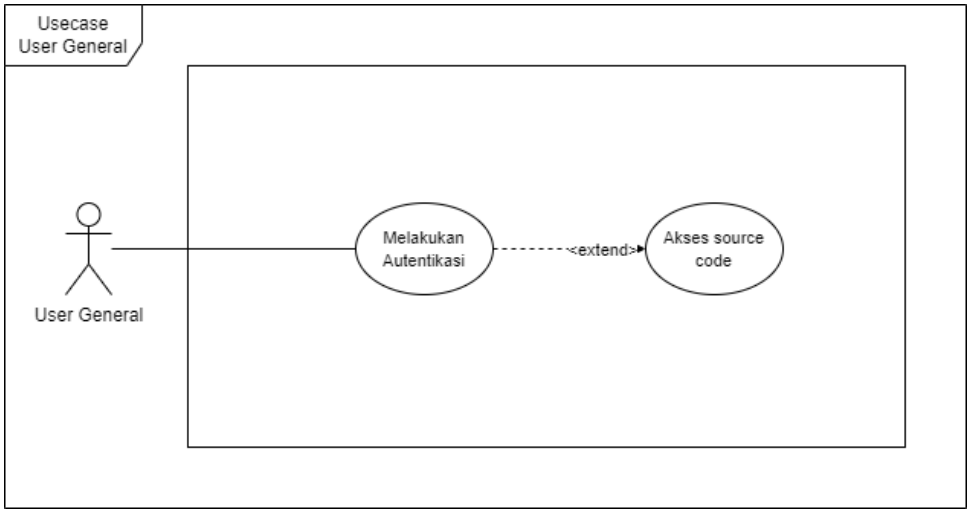
Use case diagram yang dirancang akan menampilkan fitur atau aktivitas yang dapat dilakukan oleh pengguna. Garis lurus dan panah menggambarkan kemampuan pengguna untuk menggunakan fitur yang ditunjukkan oleh anak panah. Sementara itu, garis putus dan tulisan "*extend*" menunjukkan bahwa fungsi tersebut dapat diperluas dari fungsi utama. Sedangkan untuk relasi "*include*", *art*inya fungsi tersebut dapat dijalankan setelah menjalankan fungsi yang ditunjukkan oleh anak panah garis putus.



Gambar 3.2 *Usecase Website*

- *User general*

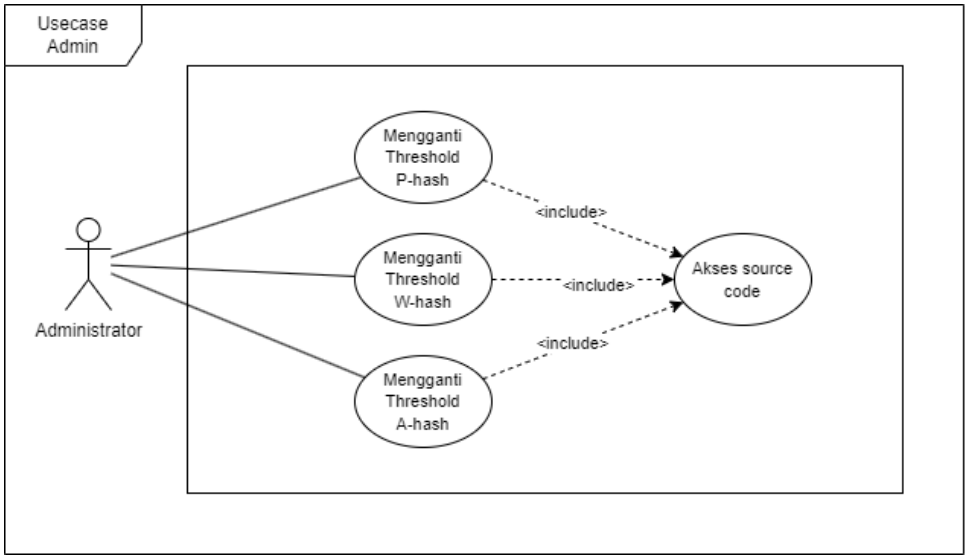
  User general atau pengguna secara umum (*user general*) dapat melakukan autentikasi dimulai dari *input* gambar pada *website* hingga memasukkan skala manipulasi pada *website* mulai dari *upscale*, *downscale*, perubahan intensitas, kompresi ukuran gambar. Setelah mengisi skala konversi *user* juga *user* selanjutnya bisa melakukan *submit* untuk memulai proses autentikasi gambar. Dalam *use case diagram User general* mempunyai relasi dengan fitur melakukan autentikasi untuk melakukan proses autentikasi dan "extend" kepada *source code* karena *user general* tidak harus memiliki akses ke *source code* untuk melakukan autentikasi.



Gambar 3.3 *Usecase User General*

- *Administrator*

  Administrator atau yang biasa dikenal sebagai admin memiliki akses terhadap *source code* dimana admin memiliki otoritas terhadap pengubahan *threshold* dari setiap metode autentikasi yang ada yaitu *perceptual hashing*, *wavelet hashing*, dan *average hashing*. Dimana setiap pengubahan *threshold* tersebut pada usecase dijadikan masing-masing satu fitur dengan relasi "include" kepada *source code* karena jika ingin mengganti atau menjalankan fitur tersebut wajib memiliki akses ke *source code*.



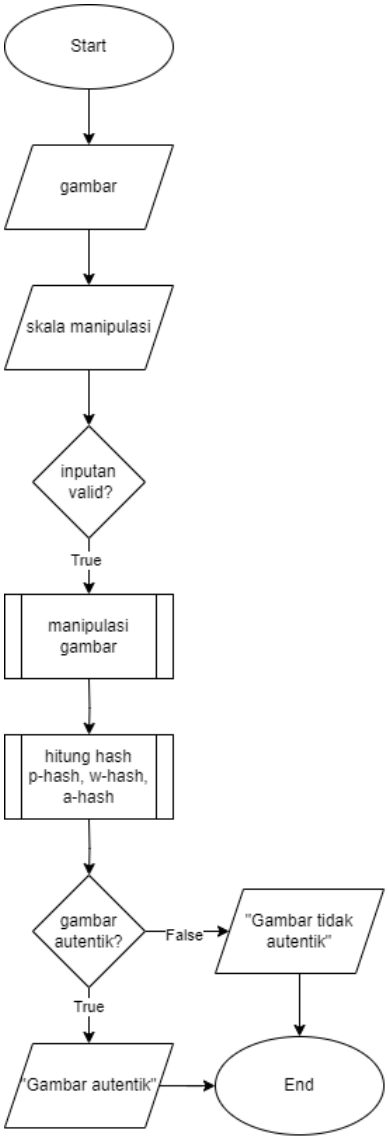
Gambar 3.4 *Usecase Administrator*

3.2.1 System *Website*

Website ini membuka peluang bagi pengguna untuk secara bebas mengunggah gambar dan mengautentikasi tampilannya sesuai keinginan. Dengan adanya fitur manipulasi seperti perubahan skala, penambahan *gamma*, dan kompresi gambar, pengguna dapat mengeksplorasi berbagai manipulasi gambar yang ingin mereka ciptakan. Misalnya, mereka dapat memperbesar atau memperkecil gambar untuk menyesuaikan ukurannya, menyesuaikan kecerahan dan kontras dengan menambah atau mengurangi nilai *gamma*, serta mengompresi gambar untuk mengurangi ukuran *file* dan mempercepat proses pengiriman atau penyimpanan.

Saat pengguna memberikan *input* mengenai manipulasi yang diinginkan, aplikasi akan secara otomatis menerapkan perubahan tersebut pada gambar yang diunggah. Setelah itu, untuk memastikan bahwa manipulasi yang dilakukan tidak merubah esensi atau keaslian gambar, *website* juga melakukan proses autentikasi menggunakan metode *hashing* seperti DCT, *wavelet*, dan *average hash*. Dengan demikian, pengguna dapat dengan percaya diri mengeksplorasi kreativitas mereka tanpa khawatir kehilangan identitas atau kualitas gambar asli.

Setelah proses manipulasi dan autentikasi selesai, pengguna akan diberikan laporan yang jelas mengenai hasilnya melalui antarmuka aplikasi web. Ini memberikan pengalaman pengguna yang lengkap, memberikan kemudahan dalam berkreasi sambil tetap memastikan integritas dan keaslian gambar yang dihasilkan. Dengan demikian, aplikasi ini tidak hanya menjadi alat praktis untuk manipulasi gambar, tetapi juga menjadi platform yang mempromosikan ekspresi kreatif dan perlindungan terhadap hak cipta dan keaslian karya.

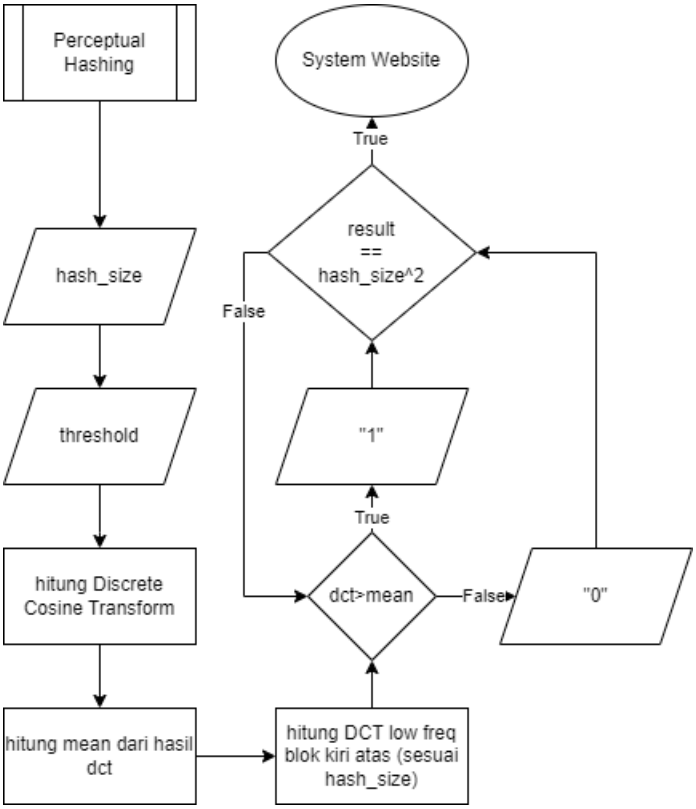


Gambar 3.5 *Flowchart Website*

3.2.1.1 *Perceptual hashing*

- *Input hash size*. Langkah pertama adalah pengguna memberikan *input* berupa ukuran *hash* (*hash size*). Ukuran *hash* ini menentukan seberapa besar blok gambar yang akan diolah dalam proses *hashing*, dan menentukan juga berapa besar *hash code* yang dihasilkan.
- *Input threshold*. Setelah ukuran *hash* ditentukan, pengguna memberikan *input* berupa nilai *threshold*. *Threshold* ini digunakan sebagai batas untuk membandingkan nilai *hash* gambar dengan gambar yang lain, *threshold* juga berguna untuk mengatur sensitivitas pada proses autentikasi gambar.

- Hitung DCT (*Discrete Cosine Transform*). Setelah ukuran *hash* dan *threshold* ditentukan, langkah selanjutnya adalah menghitung DCT dari citra. DCT adalah teknik transformasi yang digunakan untuk mengubah citra dari domain spasial menjadi domain frekuensi.
- Hitung mean dari DCT. Setelah DCT dari citra dihitung, langkah selanjutnya adalah menghitung nilai mean dari seluruh koefisien blok kiri atas DCT. Nilai mean ini kemudian akan digunakan sebagai acuan dalam menentukan apakah nilai DCT suatu blok akan dianggap besar atau kecil dan untuk dibandingkan dengan representatif gambar yaitu hasil koefisien DCT *low freq* blok kiri atas untuk menghasilkan hasil *hash*.
- Hitung DCT *low freq* blok kiri atas. Selanjutnya, citra dibagi menjadi blok-blok sesuai dengan ukuran *hash* yang telah ditentukan. Kemudian, DCT *low freq* dari blok kiri atas setiap blok dihitung. DCT *low freq* dari blok kiri atas digunakan karena blok tersebut memiliki informasi yang paling penting dalam representasi citra.
- *Output hash code*. Hasil DCT dari blok kiri atas dibandingkan dengan nilai mean yang telah dihitung sebelumnya. Jika nilai DCT lebih besar dari nilai median, maka keluaran (*output*) adalah 1, yang menandakan intensitas gambar tersebut dianggap besar. Namun, jika nilai DCT lebih kecil atau sama dengan nilai median, atau tidak melebihi nilai *threshold*, maka keluaran adalah 0, yang menandakan intensitas gambar tersebut dianggap kecil atau normal.

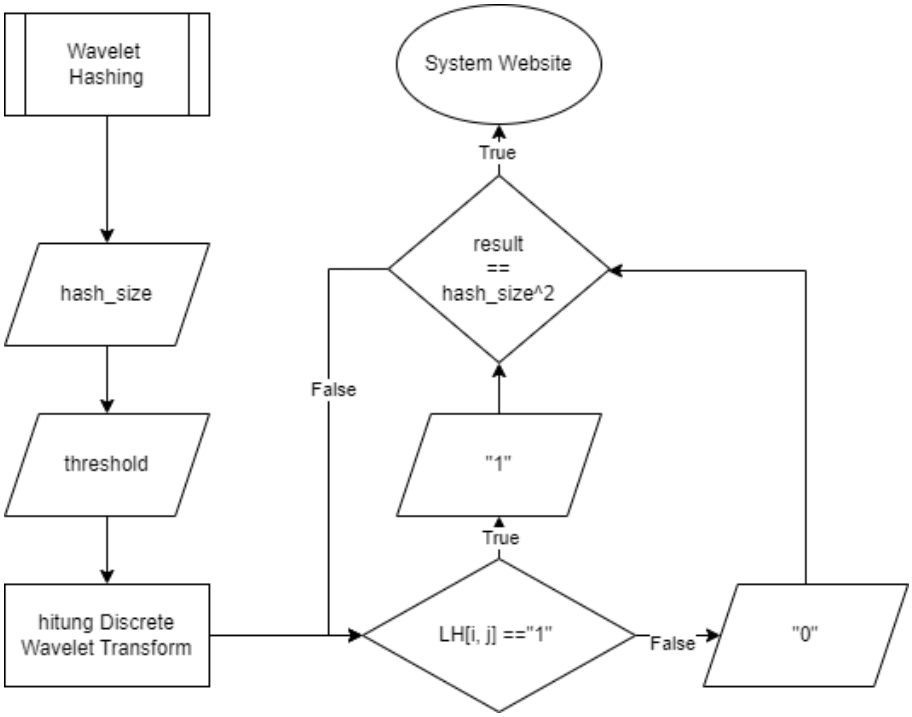


Gambar 3.6 *Flowchart Perceptual Hashing*

3.2.1.2 *Wavelet hashing*

- *Input hash size*. Pengguna memberikan *input* berupa ukuran *hash*, yang menentukan seberapa besar kode *hash* yang akan dihasilkan. Ukuran *hash* ini menentukan berapa banyak fitur citra yang akan dipertimbangkan dalam pembentukan *hash*. Semakin besar ukuran *hash*, semakin banyak informasi yang akan diwakili dalam *hash*, dan sebaliknya.
- *Input threshold*. *Input threshold*. Setelah ukuran *hash* ditentukan, pengguna memberikan *input* berupa nilai *threshold*. *Threshold* ini digunakan sebagai batas untuk membandingkan nilai *hash* gambar dengan gambar yang lain, *threshold* juga berguna untuk mengatur sensitivitas pada proses autentikasi gambar.
- Hitung DWT (*Discrete Wavelet Transform*). DWT merupakan proses transformasi untuk menguraikan sinyal atau citra menjadi komponen-komponen frekuensi yang berbeda. Dalam kode ini, citra diubah ke dalam skala keabuan dan kemudian diterapkan transformasi DWT menggunakan metode 'haar'. Proses ini menghasilkan koefisien *wavelet* yang digunakan untuk mengekstraksi fitur-fitur penting dari citra.

- Ekstraksi koefisien LH. Setelah DWT dari citra dihitung, langkah selanjutnya adalah mengekstraksi koefisien *wavelet* yang relevan. Dalam kasus ini, hanya koefisien *wavelet* LH (*Low-High*) yang diambil untuk pembentukan kode *hash*.
- *Output hash code*. Koefisien *wavelet* LH kemudian digunakan untuk menghasilkan kode *hash*. Setiap nilai koefisien LH dibandingkan dengan nilai koefisien LH dari blok tetangga. Jika nilai koefisien LH dari suatu blok lebih besar dari nilai koefisien LH dari blok tetangga, maka bit 1 akan ditambahkan ke kode *hash*. Namun, jika nilai koefisien LH tidak memenuhi kondisi tersebut, maka bit 0 akan ditambahkan ke kode *hash*.

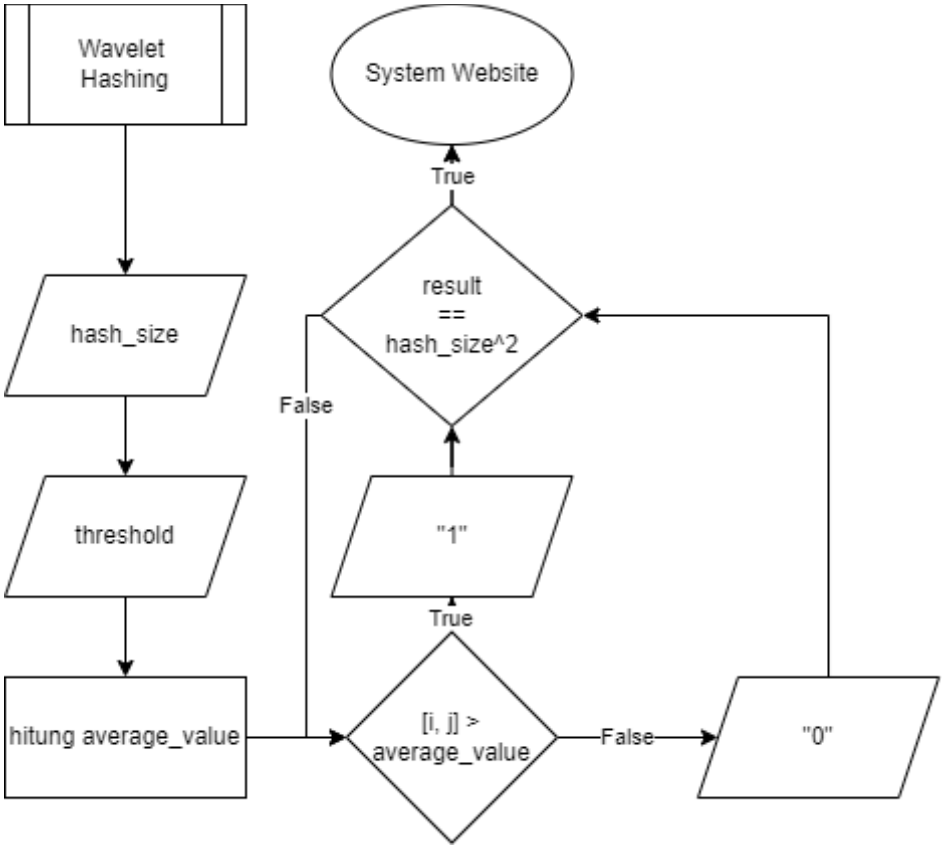


Gambar 3.7 *Flowchart Wavelet hashing*

3.2.1.3 *Average hashing*

- *Input hash size*. Pengguna memberikan *input* berupa ukuran *hash*, yang menentukan seberapa besar kode *hash* yang akan dihasilkan. Ukuran *hash* ini menentukan berapa banyak fitur citra yang akan dipertimbangkan dalam pembentukan *hash*. Semakin besar ukuran *hash*, semakin banyak informasi yang akan diwakili dalam *hash*, dan sebaliknya.
- *Input threshold*. *Input threshold*. Setelah ukuran *hash* ditentukan, pengguna memberikan *input* berupa nilai *threshold*. *Threshold* ini digunakan sebagai batas untuk membandingkan nilai *hash* gambar dengan gambar yang lain, *threshold* juga berguna untuk mengatur sensitivitas pada proses autentikasi gambar.

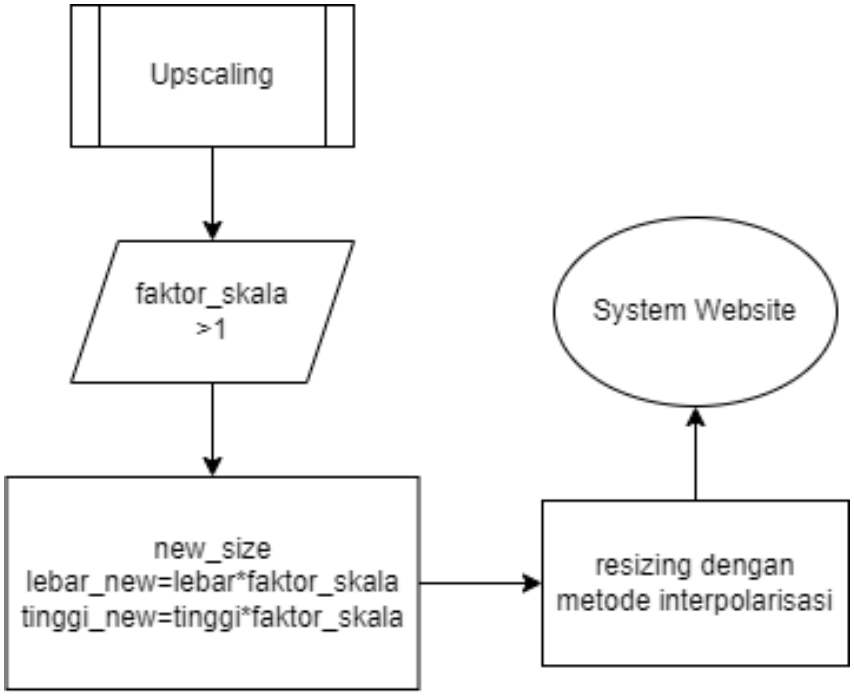
- Hitung *average value*. Citra diubah ukurannya menjadi ukuran *hash* yang telah ditentukan, dan kemudian diubah menjadi citra skala keabuan (*grayscale*). Selanjutnya, nilai rata-rata (*average value*) dari citra skala keabuan dihitung menggunakan fungsi np.mean(), untuk mendapat *average value* atau nilai rata-rata dari gambar tersebut.
- *Output hash code*. Setelah nilai rata-rata dihitung, setiap nilai piksel dalam citra skala keabuan dibandingkan dengan nilai rata-rata tersebut. Jika nilai piksel lebih besar dari nilai rata-rata, maka bit 1 akan ditambahkan ke kode *hash*. Namun, jika nilai piksel kurang dari atau sama dengan nilai rata-rata, maka bit 0 akan ditambahkan ke kode *hash*. Dengan demikian, setiap piksel dalam citra akan diwakili oleh satu bit dalam kode *hash*.



Gambar 3.8 *Flowchart Average hashing*

3.2.1.4 *Upscaling*

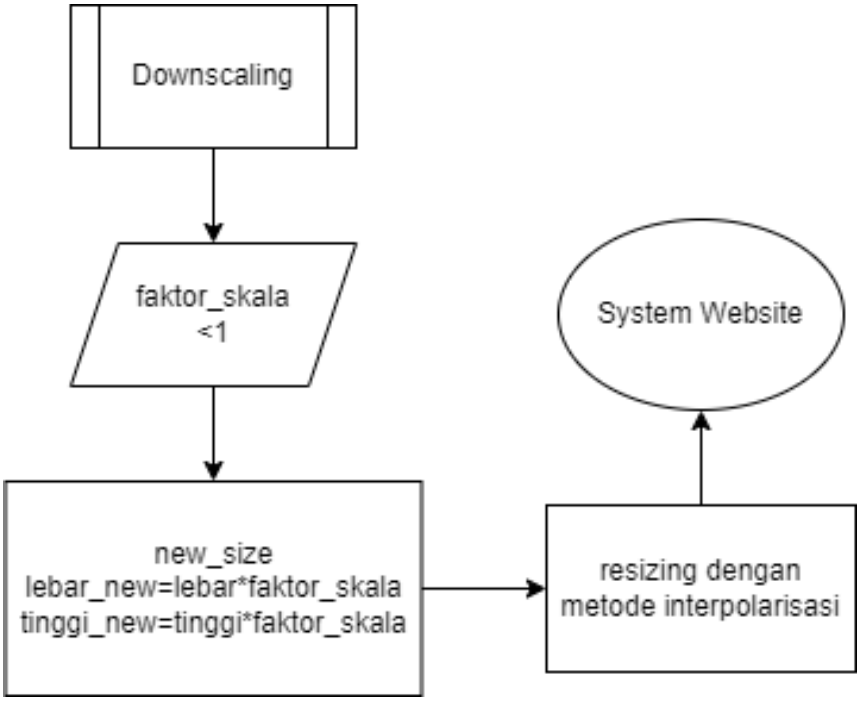
- Tentukan faktor skala. Langkah pertama adalah menentukan faktor skala, yang merupakan angka yang menentukan seberapa besar gambar akan diperbesar. Faktor skala ini bisa ditentukan oleh pengguna atau berdasarkan kebutuhan aplikasi. Misalnya, faktor skala 2.0 akan menggandakan ukuran gambar menjadi dua kali lipat dari ukuran aslinya.
- Hitung ukuran gambar baru. Setelah faktor skala ditentukan, langkah selanjutnya adalah menghitung ukuran gambar baru berdasarkan faktor skala tersebut. Ukuran gambar baru ini akan menjadi hasil *upscaling* dari gambar asli. Misalnya, jika ukuran gambar asli adalah 1000x1000 piksel dan faktor skala adalah 2.0, maka ukuran gambar baru akan menjadi 2000x2000 piksel.
- *Upscaling* dengan metode interpolarisasi. Setelah ukuran gambar baru dihitung, langkah terakhir adalah melakukan proses *upscaling* pada gambar menggunakan fungsi cv2.re*size* dari pustaka OpenCV. Fungsi ini akan mengubah ukuran gambar sesuai dengan ukuran baru yang telah dihitung sebelumnya. Untuk *upscaling*, metode interpolasi yang umumnya digunakan adalah cv2.INTER_LINEAR atau cv2.INTER_CUBIC. Metode ini akan melakukan interpolasi untuk memperkirakan nilai piksel baru berdasarkan piksel-piksel yang ada di sekitarnya, sehingga memberikan hasil yang lebih halus dan mendetail saat melakukan *upscaling*.



Gambar 3.9 *Flowchart Upscaling*

3.2.1.5 *Downscaling*

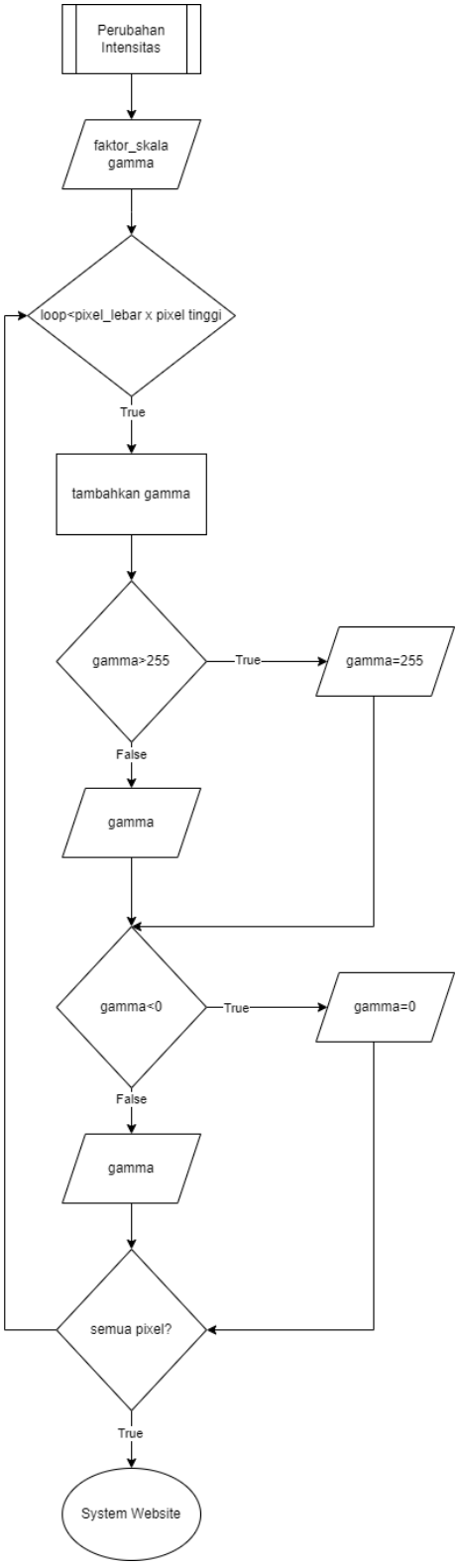
- Tentukan faktor skala. Langkah pertama adalah menentukan faktor skala, yang merupakan angka yang menentukan seberapa besar gambar akan diperkecil. Faktor skala ini bisa ditentukan oleh pengguna atau berdasarkan kebutuhan aplikasi. Misalnya, faktor skala 0.5 akan mereduksi ukuran gambar menjadi setengah dari ukuran aslinya.
- Hitung ukuran gambar baru. Setelah faktor skala ditentukan, langkah selanjutnya adalah menghitung ukuran gambar baru berdasarkan faktor skala tersebut. Ukuran gambar baru ini akan menjadi hasil *downsizing* dari gambar asli. Misalnya, jika ukuran gambar asli adalah 1000x1000 piksel dan faktor skala adalah 0.5, maka ukuran gambar baru akan menjadi 500x500 piksel.
- *Downscaling* dengan metode interpolarisasi. Setelah ukuran gambar baru dihitung, langkah terakhir adalah melakukan proses *downsizing* pada gambar menggunakan fungsi cv2.re*size* dari pustaka OpenCV. Fungsi ini akan mengubah ukuran gambar sesuai dengan ukuran baru yang telah dihitung sebelumnya. Selain itu, metode interpolasi cv2.INTER_AREA digunakan untuk memperoleh hasil yang optimal dalam penurunan ukuran gambar. Metode ini secara efektif menghilangkan piksel-piksel yang tidak diperlukan dan mempertahankan detail gambar yang lebih baik saat melakukan downsizing.



Gambar 3.10 *Flowchart Downscaling*

3.2.1.6 Perubahan Intensitas

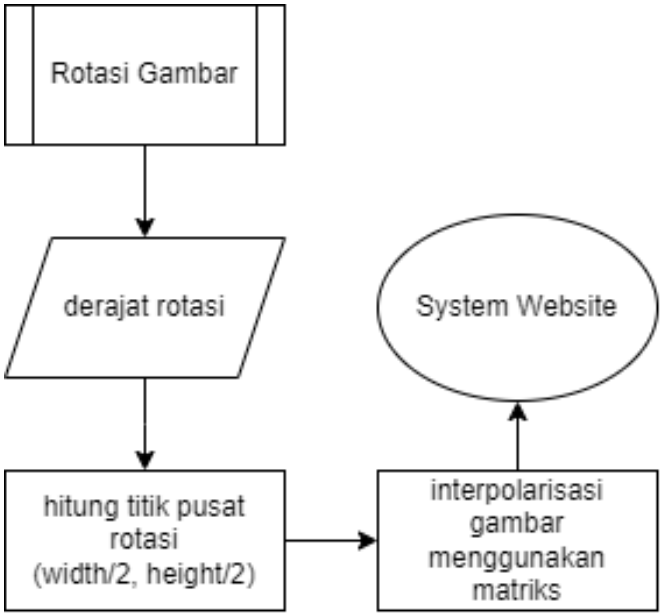
- *Input* faktor skala *gamma*. Langkah pertama adalah pengguna memberikan *input* berupa faktor skala *gamma*, yang merupakan nilai *integer* yang berkisar dari 0 hingga 100. Rentang nilai ini memberikan fleksibilitas dalam menentukan seberapa besar intensitas gambar akan diubah. Nilai *gamma* yang lebih rendah cenderung menghasilkan gambar yang lebih gelap, sementara nilai *gamma* yang lebih tinggi cenderung menghasilkan gambar yang lebih cerah. Pengguna dapat menyesuaikan nilai *gamma* sesuai dengan kebutuhan atau preferensi estetika.
- Tambahkan *gamma*. Setelah faktor skala *gamma* ditentukan, langkah selanjutnya adalah menambahkan nilai *gamma* tersebut ke setiap nilai piksel dalam gambar. Ini ber*art*i bahwa setiap komponen warna (misalnya, merah, hijau, dan biru) dari setiap piksel akan diperbesar atau diperkecil sesuai dengan nilai *gamma* yang telah ditentukan. Penambahan nilai *gamma* ini bertujuan untuk mengatur intensitas warna atau kecerahan gambar secara keseluruhan.
- Tambahkan ke setiap piksel. Setelah penambahan nilai *gamma*, langkah selanjutnya adalah menerapkan perubahan tersebut ke setiap piksel dalam gambar. Ini dilakukan dengan cara menambahkan nilai *gamma* ke setiap komponen warna (misalnya, merah, hijau, dan biru) dari setiap piksel. Dengan demikian, gambar secara keseluruhan akan mengalami penyesuaian intensitas yang seragam.
- Batasni nilai piksel (*Clipping*). Langkah terakhir adalah memastikan bahwa nilai piksel dalam gambar tetap berada dalam rentang yang valid, yaitu antara 0 hingga 255. Hal ini dilakukan untuk menghindari nilai piksel yang melebihi batas atas (255) agar tetap di range maksimal 255 atau melampaui batas bawah (0) agar tetap di range 0, yang dapat menghasilkan distorsi visual atau kehilangan detail. Dengan membatasi nilai piksel menggunakan fungsi np.clip, gambar yang dihasilkan akan tetap memiliki kualitas visual yang baik dan sesuai dengan representasi gambar yang benar.



Gambar 3.11 *Flowchart* Intensitas

3.2.1.7 Rotasi Gambar

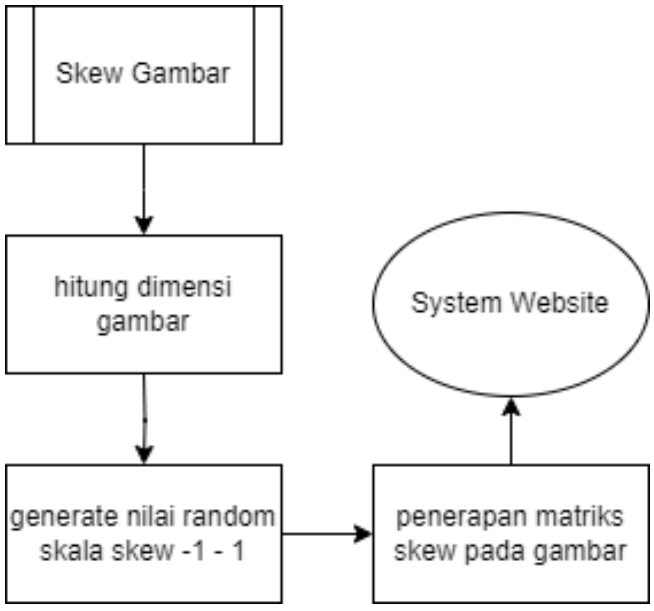
- *Input* derajat rotasi. Langkah ini melibatkan pengguna memberikan *input* berupa derajat rotasi yang diinginkan untuk gambar. Derajat rotasi ini menentukan seberapa jauh gambar akan diputar. Pengguna biasanya memasukkan nilai derajat rotasi melalui antarmuka pengguna, yang kemudian digunakan oleh program untuk melakukan rotasi.
- Hitung titik pusat. Setelah mendapatkan *input* derajat rotasi, program menghitung titik pusat rotasi. Titik pusat adalah titik di sekitar mana gambar akan diputar. Untuk melakukan ini, program mengambil setengah dari dimensi lebar dan tinggi gambar. Titik pusat yang dihasilkan menjadi acuan untuk proses rotasi selanjutnya, menggunakan rumus (lebar/2, tinggi/2).
- Interpolarisasi gambar menggunakan matriks. Setelah mendapatkan titik pusat dan derajat rotasi, program menggunakan matriks rotasi yang diperoleh dari fungsi cv2.get*Rotation*Matrix2D untuk melakukan rotasi gambar. Matriks rotasi ini merepresentasikan transformasi geometris yang diterapkan pada gambar. Selanjutnya, gambar hasil rotasi diinterpolasi menggunakan metode interpolasi linier (cv2.INTER_LINEAR). Proses interpolasi ini menghasilkan gambar yang telah dirotasi secara akurat sesuai dengan sudut yang diinginkan tanpa distorsi yang signifikan.



Gambar 3.12 *Flowchart* rotasi

3.2.1.8 *Skew* Gambar

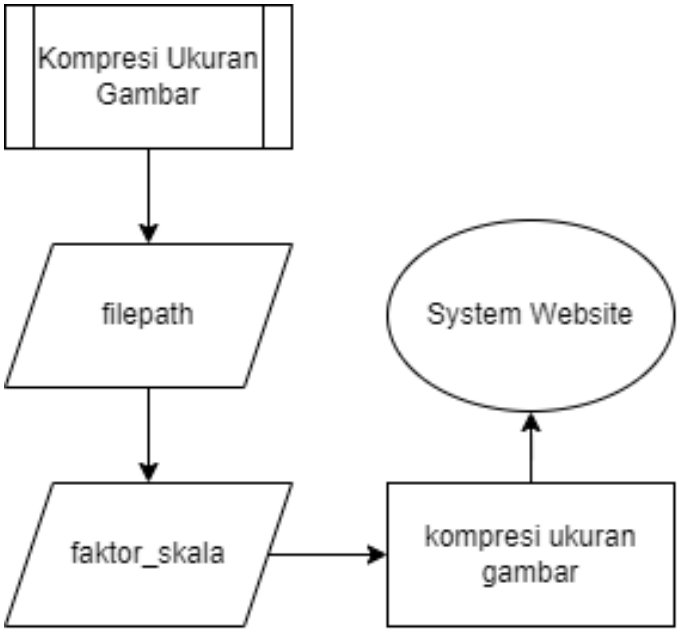
- Hitung dimensi gambar. Langkah pertama dalam proses adalah menghitung dimensi gambar, yaitu jumlah baris dan kolom piksel dalam gambar yang akan diubah *skew*. Informasi ini penting untuk menentukan ukuran gambar hasil setelah proses *skew*.
- Generate nilai random skala *skew* -1 hingga 1. Setelah mendapatkan dimensi gambar, langkah selanjutnya adalah menghasilkan nilai acak antara -1 hingga 1. Nilai ini mewakili seberapa jauh gambar akan diubah *skew*, dengan nilai negatif menghasilkan *skew* ke arah yang berlawanan. Sebagai contoh, nilai -1 akan menghasilkan *skew* penuh ke arah yang berlawanan, sedangkan nilai 0 akan menghasilkan gambar yang tidak mengalami *skew* sama sekali. Nilai ini kemudian dimasukkan ke dalam matriks *skew* yang akan digunakan dalam proses transformasi. Disini menggunakan random supaya tidak terlalu banyak *input*an yang perlu di*input*kan *user*.
- Penerapan matriks *skew* pada gambar. Setelah mendapatkan matriks *skew* dengan nilai random, langkah terakhir adalah menerapkan matriks *skew* tersebut pada gambar menggunakan fungsi cv2.warpAffine dari library OpenCV. Proses ini akan mengubah bentuk gambar sesuai dengan matriks *skew* yang telah dibuat, menciptakan efek *skew* yang diinginkan. Dengan demikian, gambar akan diubah secara *skew* berdasarkan nilai acak yang telah dihasilkan sebelumnya.



Gambar 3.13 *Flowchart skew*

3.2.1.9 Kompresi Ukuran Gambar

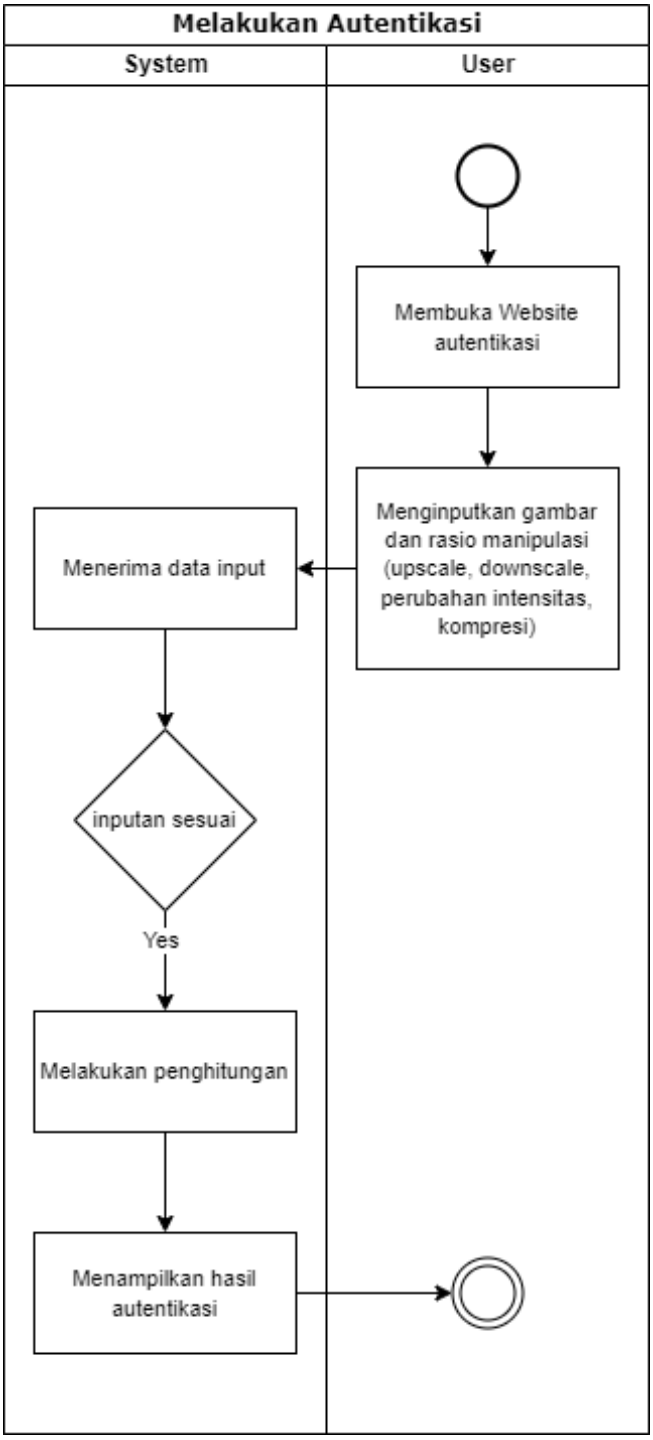
- *Input filepath*. Pengguna memberikan alamat *file* gambar yang akan dikompresi. Ini merupakan alamat atau path dari *file* gambar yang akan diolah oleh sistem. Misalnya, alamat *file* tersebut bisa berupa direktori dari sebuah gambar di dalam sistem pengguna.
- *Input* faktor skala. Pengguna menentukan faktor skala untuk proses kompresi gambar. Faktor skala ini menentukan seberapa besar gambar akan diperkecil. Faktor skala biasanya merupakan nilai numerik, seperti 50 untuk mereduksi ukuran gambar menggunakan skala 50. Pengguna memberikan nilai faktor skala ini sebagai parameter untuk menyesuaikan proses kompresi gambar.
- Kompresi ukuran gambar. Setelah gambar diambil dari *filepath*, sistem menghitung ukuran gambar baru berdasarkan faktor skala yang diberikan. Kemudian, gambar diperkecil sesuai dengan ukuran baru yang telah dihitung. Proses kompresi ini melibatkan operasi seperti *resize* pada cv2.



Gambar 3.14 *Flowchart* kompresi

3.2.2 *User general*

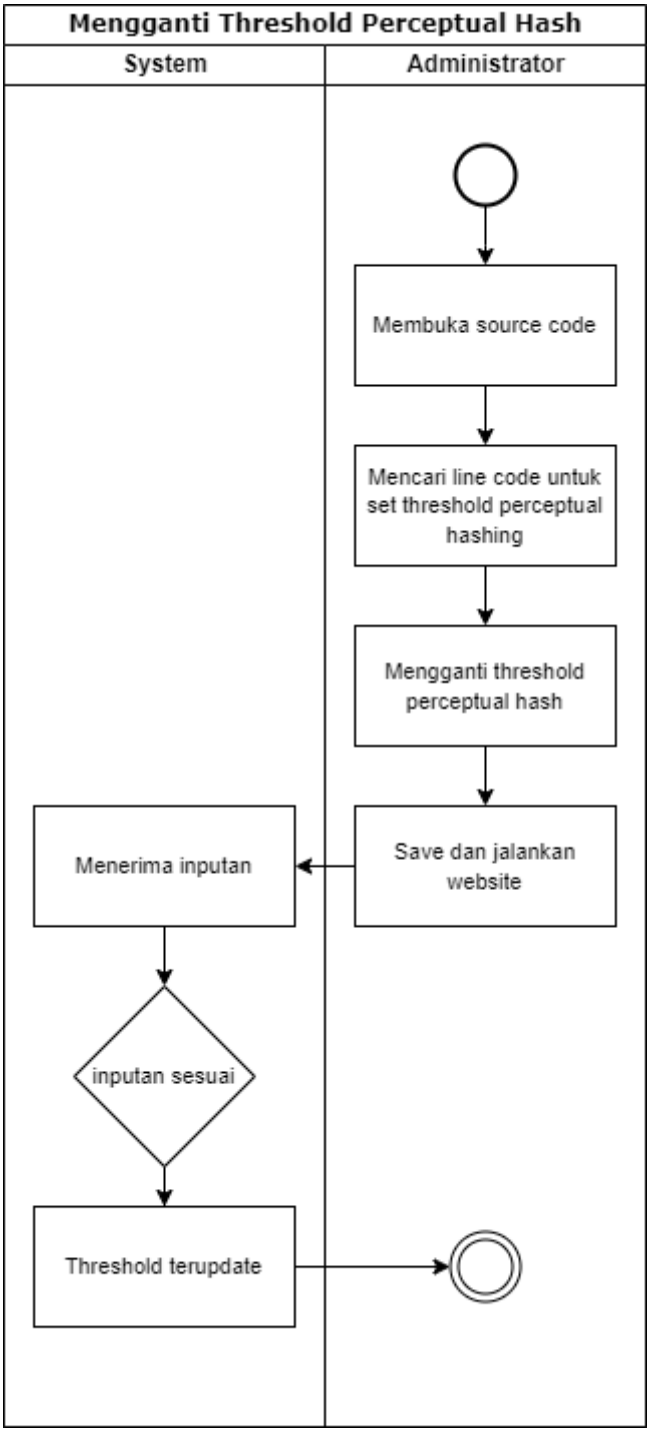
- Melakukan Autentikasi
    1. Membuka *Website* Autentikasi (Pengguna): Pengguna membuka *website* autentikasi gambar untuk memulai proses autentikasi.
    2. Meng*input*kan Gambar dan Rasio Manipulasi (Pengguna): Pengguna mengunggah gambar yang akan diverifikasi ke dalam sistem, serta memberikan informasi mengenai rasio manipulasi yang diinginkan (seperti skala pengecilan, pembesaran, *gamma*, dan kompresi).
    3. Menerima *Input*an (Sistem): Sistem menerima *input*an dari pengguna, berupa gambar yang diunggah dan rasio manipulasi yang diberikan.
    4. Melakukan Perhitungan Sistem (Sistem): Setelah menerima *input*an, sistem melakukan perhitungan yang diperlukan untuk memproses gambar sesuai dengan rasio manipulasi yang diinginkan. Hal ini meliputi penyesuaian skala, penambahan *gamma*, atau proses kompresi jika diperlukan.
    5. Menampilkan Hasil Autentikasi (Sistem): Setelah proses perhitungan selesai, sistem melakukan autentikasi terhadap gambar yang telah dimanipulasi. Hasil autentikasi, baik itu gambar masih otentik atau telah mengalami manipulasi, kemudian ditampilkan kepada pengguna melalui antarmuka *website*. Ini memberikan informasi kepada pengguna mengenai integritas gambar yang diunggah dan memastikan bahwa gambar tersebut tidak mengalami perubahan yang tidak diinginkan atau tidak diotorisasi.



Gambar 3.15 *Activity User General*

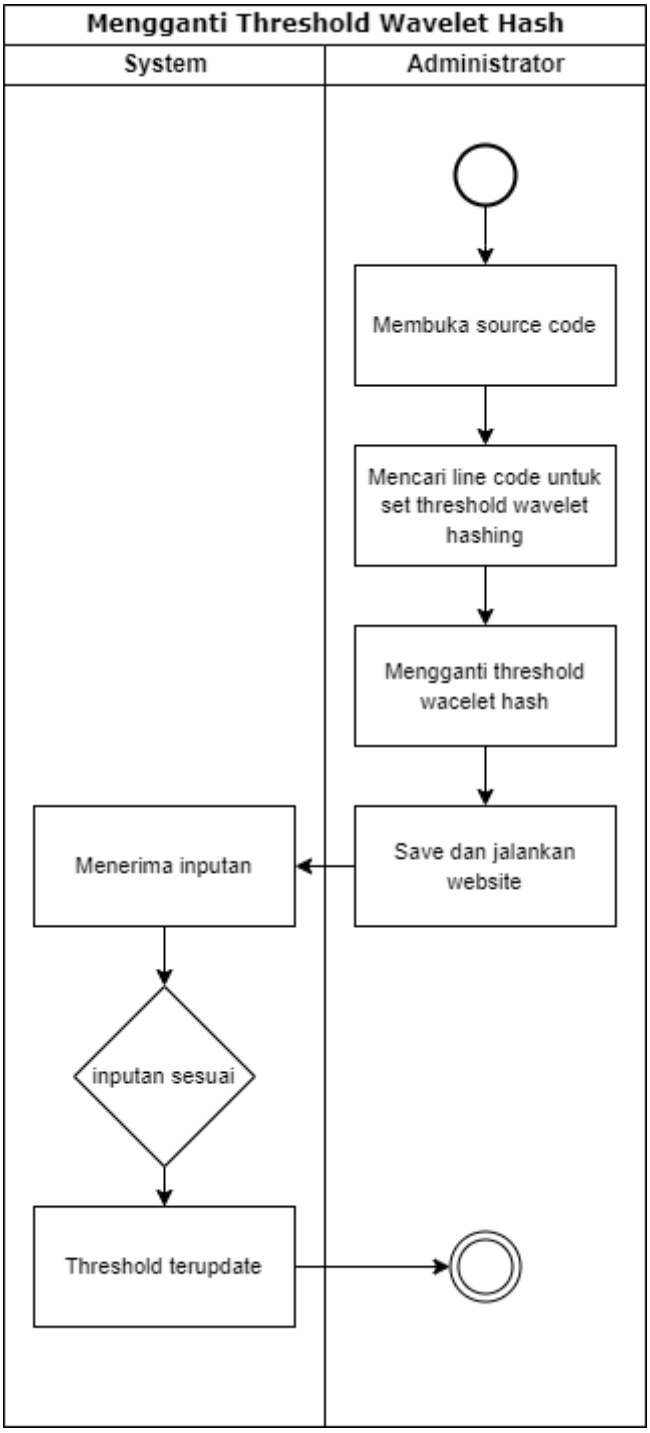
3.2.3 *Administrator*

- Mengganti *Threshold Perceptual Hash*
    1. Membuka *Source code* (Admin): *Administrator* membuka *source code* dari aplikasi *website* untuk memulai proses pengaturan *threshold*.
    2. Mencari Line *Code* untuk Set *Threshold Perceptual hashing* (Admin): *Administrator* melakukan pencarian pada *source code* untuk menemukan baris kode yang bertanggung jawab atas pengaturan *threshold* untuk *hashing perceptual*.
    3. Mengganti *Threshold Perceptual hashing* (Admin): Setelah menemukan baris kode yang tepat, *administrator* mengganti nilai *threshold* sesuai dengan preferensi atau kebutuhan tertentu.
    4. Save dan Jalankan *Website* (Admin): *Administrator* menyimpan perubahan yang telah dilakukan dan menjalankan ulang *website* untuk mengaktifkan pengaturan *threshold* yang baru.
    5. Menerima *Input*an (Sistem): Sistem menerima *input*an dari *administrator* yang mengindikasikan adanya perubahan pada *threshold* untuk *hashing perceptual*.
    6. Jika *Input*an Sesuai Maka *Threshold* Terupdate (Sistem): Setelah menerima *input*an, sistem memverifikasi apakah *input*an yang diberikan sesuai dengan format dan persyaratan yang ditentukan. Jika sesuai, sistem akan mengupdate nilai *threshold* sesuai dengan *input*an yang diberikan oleh *administrator*.



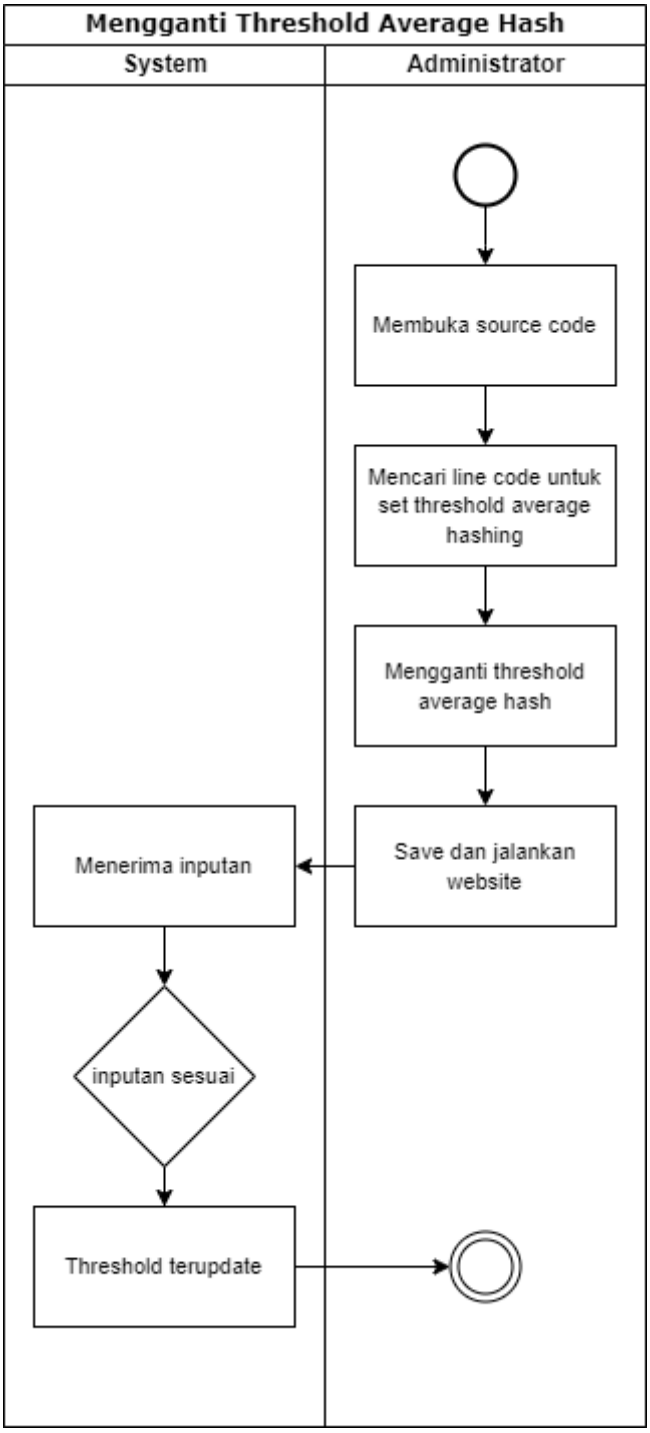
Gambar 3.16 *Activity Threshold Perceptual Hashing*

- Mengganti *Threshold Wavelet Hash*
    1. Membuka *Source code* (Admin): *Administrator* membuka *source code* dari aplikasi *website* untuk memulai proses pengaturan *threshold*.
    2. Mencari Line *Code* untuk Set *Threshold Wavelet hashing* (Admin): *Administrator* melakukan pencarian pada *source code* untuk menemukan baris kode yang bertanggung jawab atas pengaturan *threshold* untuk *hashing wavelet*.
    3. Mengganti *Threshold Wavelet hashing* (Admin): Setelah menemukan baris kode yang tepat, *administrator* mengganti nilai *threshold* sesuai dengan preferensi atau kebutuhan tertentu.
    4. Save dan Jalankan *Website* (Admin): *Administrator* menyimpan perubahan yang telah dilakukan dan menjalankan ulang *website* untuk mengaktifkan pengaturan *threshold* yang baru.
    5. Menerima *Input*an (Sistem): Sistem menerima *input*an dari *administrator* yang mengindikasikan adanya perubahan pada *threshold* untuk *hashing wavelet*.
    6. Jika *Input*an Sesuai Maka *Threshold* Terupdate (Sistem): Setelah menerima *input*an, sistem memverifikasi apakah *input*an yang diberikan sesuai dengan format dan persyaratan yang ditentukan. Jika sesuai, sistem akan mengupdate nilai *threshold* sesuai dengan *input*an yang diberikan oleh *administrator*.



Gambar 3.17 *Activity Threshold Wavelet hashing*

- Mengganti *Threshold Average Hash*
    1. Membuka *Source code* (Admin): *Administrator* membuka *source code* dari aplikasi *website* untuk memulai proses pengaturan *threshold*.
    2. Mencari Line *Code* untuk Set *Threshold Average hashing* (Admin): *Administrator* melakukan pencarian pada *source code* untuk menemukan baris kode yang bertanggung jawab atas pengaturan *threshold* untuk *hashing* rata-rata.
    3. Mengganti *Threshold Average hashing* (Admin): Setelah menemukan baris kode yang tepat, *administrator* mengganti nilai *threshold* sesuai dengan preferensi atau kebutuhan tertentu.
    4. Save dan Jalankan *Website* (Admin): *Administrator* menyimpan perubahan yang telah dilakukan dan menjalankan ulang *website* untuk mengaktifkan pengaturan *threshold* yang baru.
    5. Menerima *Input*an (Sistem): Sistem menerima *input*an dari *administrator* yang mengindikasikan adanya perubahan pada *threshold* untuk *hashing* rata-rata.
    6. Jika *Input*an Sesuai Maka *Threshold* Terupdate (Sistem): Setelah menerima *input*an, sistem memverifikasi apakah *input*an yang diberikan sesuai dengan format dan persyaratan yang ditentukan. Jika sesuai, sistem akan mengupdate nilai *threshold* sesuai dengan *input*an yang diberikan oleh *administrator*.



Gambar 3.18 *Activity Threshold Average hashing*

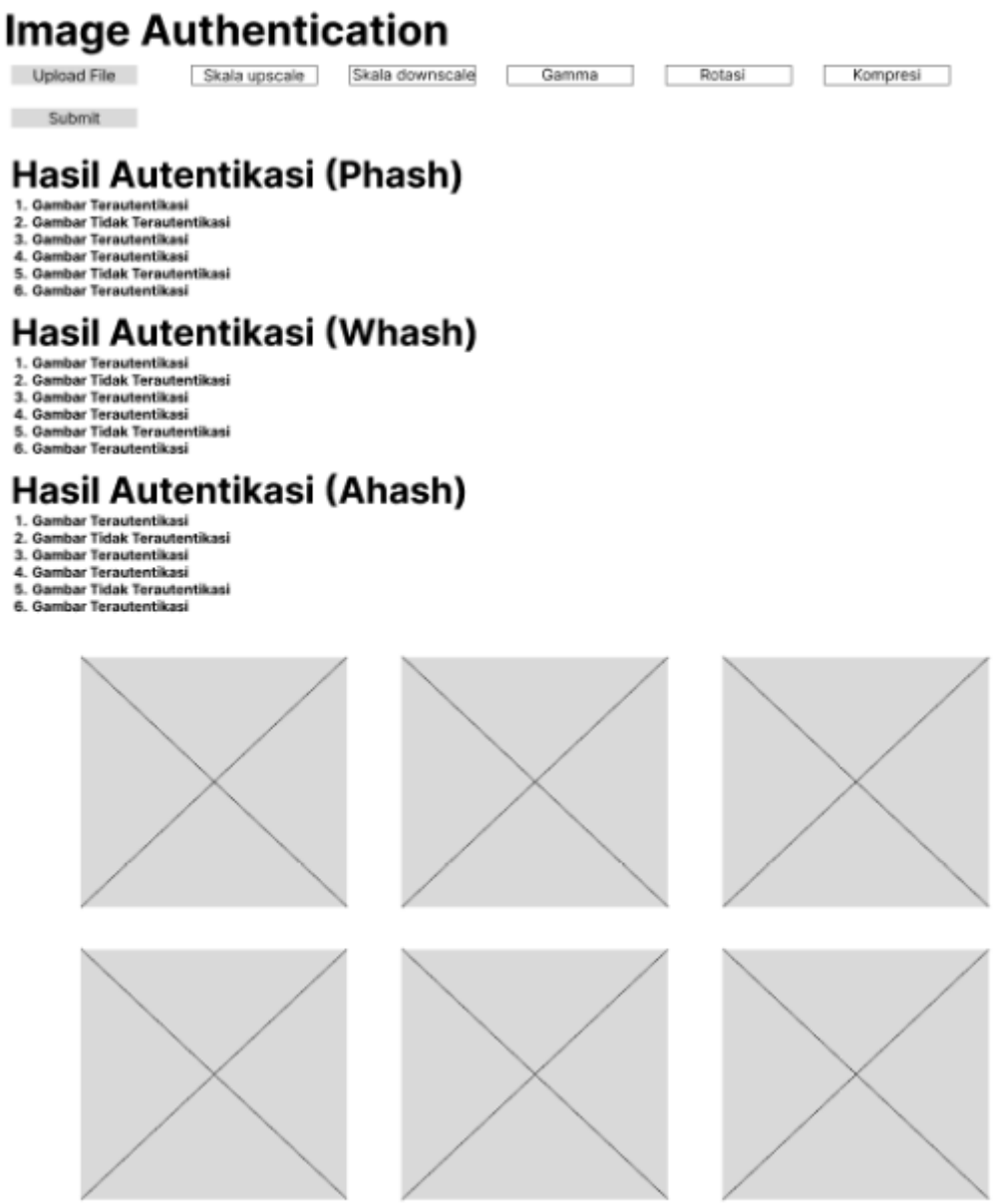
3.3 Desain *Website*

Halaman HTML yang disediakan menampilkan antarmuka pengguna untuk aplikasi autentikasi gambar. Ketika pengguna membuka halaman tersebut, mereka disajikan dengan judul "*Image Authentication*" yang memberi tahu mereka tentang tujuan aplikasi. Di bagian utama halaman, terdapat formulir yang memungkinkan pengguna untuk mengunggah gambar yang ingin mereka autentikasi. Formulir ini terdiri dari beberapa bidang *input* dan tombol "*Upload*". Pengguna dapat memilih gambar yang ingin diunggah dari sistem mereka dengan menggunakan *input file*. Selain itu, terdapat juga tiga *input* teks yang memungkinkan pengguna untuk memasukkan faktor skala untuk proses perbesaran dan perkecilan gambar, nilai *gamma* untuk penyesuaian *gamma* pada gambar, serta skala kualitas untuk proses kompresi gambar. Setelah mengisi semua bidang yang diperlukan, pengguna dapat mengklik tombol "*Upload*" untuk mengirimkan gambar dan parameter-parameter terkait ke *server*.

Gambar 3.19 *UI/UX Website 1*

Setelah pengguna mengunggah gambar dan mengklik tombol "*Upload*", hasil autentikasi akan ditampilkan di bawah form dalam tiga kategori yang berbeda: *Perceptual Hash*, *Wavelet Hash*, dan *Average Hash*. Setiap hasil autentikasi memberi tahu pengguna apakah gambar yang diunggah terautentikasi atau tidak menggunakan algoritma tertentu. Selain itu, jika

ada, gambar asli dan gambar yang dimanipulasi juga akan ditampilkan di bawah hasil autentikasi. Ini memungkinkan pengguna untuk membandingkan gambar asli dengan hasil manipulasi dan memahami apakah gambar tersebut telah dimanipulasi atau tidak. Keseluruhan, halaman ini memberikan antarmuka yang jelas dan intuitif untuk pengguna untuk mengunggah dan mengautentikasi gambar.



Gambar 3.20 *UI/UX Website 2*