CICES (*Cyberpreneurship Innovative and Creative Exact and Social Science*)
DOI: 10.33050/cices.v10i2.3405 ISSN: **2356-5209**
pp. 271~289 *Online* ISSN: **2655-3058**

Pengembangan Aplikasi Android Untuk Konversi Gambar ke Teks dengan *Flutter* dan OCR Menggunakan Metode *Jaro-Winkler*

Rifqi Aldy Al Hafizh Harahap[*1], Fauziah[2], Ratih Titi Komala Sari[3]
[1,2,3]program Studi Informatika, Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional
Email: **[*1]rifqialdy80@gmail.com**, [2]fauziah@civitas.unas.ac.id, [3]ratih.titi@civitas.unas.ac.id

ABSTRAK

Kebutuhan akan alat untuk konversi gambar ke teks yang efektif meningkat sebagai akibat dari ketergantungan yang meningkat pada aplikasi seluler untuk aktivitas sehari-hari dan kendala dalam pengembangan aplikasi lintas platform. Mengatasi permintaan ini dengan membuat aplikasi Android yang menggunakan *Flutter* dan Teknologi Pengenalan Karakter Optik (OCR) adalah fokus penelitian ini. Khususnya, metode *Jaro Winkler*, yang meningkatkan akurasi ekstraksi teks, adalah fokus utama penelitian ini. Metodologi penelitian menggunakan pendekatan sistematis, dimulai dengan tinjauan literatur untuk menciptakan fondasi teoretis dan menemukan perbedaan dalam teknologi saat ini. Pengembangan menggunakan kemampuan lintas *platform Flutter* dan memasukkan OCR untuk ekstraksi teks. Metode *Jaro Winkler* digunakan untuk meningkatkan akurasi dengan meningkatkan ukuran kesamaan antara teks yang diekstrak dan teks aktual, terutama untuk gambar yang tidak jelas. Meskipun ada masalah dengan gambar beresolusi rendah, aplikasi memiliki kemampuan untuk mengonversi gambar menjadi teks dengan akurat, seperti yang ditunjukkan oleh pengujian dan evaluasi aplikasi. Kinerja aplikasi menunjukkan kemungkinan bahwa kombinasi *Flutter*, OCR, dan metode *Jaro Winkler* dapat membantu membuat alat konversi gambar ke teks yang efektif. Tapi penelitian juga menemukan beberapa hal yang perlu diperbaiki, terutama dalam mengolah gambar berkualitas rendah dan penyempurnaan algoritme untuk akurasi yang lebih baik. Implementasi Algoritma *Jaro-Winkler* digunakan secara akurat dan signifikan. Ini terbukti dengan perbandingan dua kata sampel, "BRI" dan "BNI," yang menghasilkan hasil 0,7778, atau nilai persentase 77,3% melalui perhitungan manual. Berdasarkan perbandingan nilai pada aplikasi dan perhitungan menggunakan *phyton* dengan representasi *floating-point*, terjadi perbedaan kecil dalam hasil akhir, dengan perbedaan hanya 3%, yang menghasilkan hasil 80%.

Kata kunci: Konversi Gambar, *Flutter*, Teknologi Pengenalan Karakter Optik (OCR), Metode *Jaro Winkler*

ABSTRACT

The need for tools for effective image to text conversion is increasing as a result of increasing reliance on mobile applications for daily activities and constraints in cross-platform application development. Addressing this demand by creating an Android application that uses Flutter and Optical Character Recognition (OCR) technology is the focus of this research. In particular, Jaro Winkler's method, which improves the accuracy of text extraction, is the main focus of this research. The research methodology uses a systematic approach, starting with a literature review to create a theoretical foundation and discover differences in current technology. The development uses Flutter's cross-platform capabilities and includes OCR for text extraction. The Jaro Winkler method is used to improve accuracy by increasing the similarity measure

DOI: ISSN: **2356-5209**
pp. 271~270 *Online* ISSN: **2655-3058**

between extracted text and actual text, especially for unclear images. Despite issues with low-resolution images, the app has the ability to convert images to text accurately, as demonstrated by app testing and evaluation. The app's performance shows the possibility that a combination of Flutter, OCR, and Jaro Winkler's method can help create an effective image to text conversion tool. But the research also found some things that need improvement, especially in processing low-quality images and refining algorithms for better accuracy. Implementation of the Jaro-Winkler algorithm is used accurately and significantly. This is proven by a comparison of two sample words, "BRI" and "BNI," which produces a result of 0.7778, or a percentage value of 77.3% through manual calculation. Based on a comparison of the values in the application and calculations using Python with floating-point representation, there is a small difference in the final results, with a difference of only 3%, which results in a result of 80%..

Keywords: *Konversi Gambar, Flutter, Teknologi Pengenalan Karakter Optik (OCR), Metode Jaro Winkler*

1. PENDAHULUAN

Gambar tidak hanya menjadi produk yang indah, tetapi juga dapat diproses oleh komputer di era revolusi industri 4.0. Salah satu jenis pengolahan gambar adalah ekstraksi fitur gambar, yang merupakan teknik yang digunakan untuk mengekstraksi fitur gambar untuk proses klasifikasi dan pengenalan gambar (Napitupulu dkk, 2023). Ekstraksi karakter teks dari gambar berarti mengekstraksi objek teks dari sekumpulan gambar. Variasi ukuran, jenis, dan jenis teks membuat tugas ini sulit di berbagai bidang, seperti pembacaan otomatis plat nomor (Napitupulu dkk, 2023), pembacaan otomatis kartu nama (Retno dkk, 2019), pembacaan otomatis dokumen kwitansi dan faktur (Salim dkk, 2019; Sanjaya, 2020), dan pengenalan karakter *captcha (*Soedewi dkk, 2021). Meskipun tampaknya mudah bagi manusia untuk mengekstrak karakter teks, komputer menghadapi masalah yang rumit. Saat ini, ada banyak teknologi OCR yang dapat digunakan, seperti *Tesseract, Google Vission, PhotoScan* di *Windows*, dan lainnya.

Banyak gambar digital yang ada di sekitar kita saat ini, terutama di media sosial. Citra digital dapat berasal dari jepretan kamera, scan dokumen atau gambar, atau hasil dari aplikasi pengolah gambar. Namun, Munir mengatakan bahwa gambar digital adalah gambar yang dihasilkan dari proses digitalisasi gambar dua dimensi (Julisawaty dkk, 2020). Sementara itu, Putra mengatakan bahwa gambar digital adalah larik yang berisi nilai-nilai yang diwakili oleh deretan bit tertentu (Al-Faruq dkk, 2022). Banyak jenis gambar digital yang digunakan dalam kegiatan sehari-hari saat ini.

Normalisasi citra bertujuan untuk mendapatkan citra masukan yang lebih baik sehingga proses segmentasi dapat menghasilkan akurasi yang optimal. Ini adalah tahap awal proses normalisasi, di mana kemampuan metode dan teknik OCR sangat penting (Siregar, 2019; Phoenix, et al., 2021). Untuk mendapatkan hasil yang akurat, algoritma yang digunakan adalah *Jaro-Winkler*, yang merupakan salah satu algoritma yang digunakan untuk mengukur kesamaan antara dua buah string. Sangat penting untuk menggunakan algoritma ini untuk menambah fitur koreksi kata kunci dan memberikan usulan kata. Algoritma ini biasanya digunakan untuk mendeteksi duplikat. Semakin besar nilai *Jaro-Winkler* untuk dua string, semakin besar presentase kemiripan string tersebut.

Istilah pengenalan karakter optik (OCR) mengacu pada proses pengenalan karakter pada gambar atau gambar, seperti huruf, angka, atau simbol (Van Hoai et al., 2021). OCR adalah alat yang dapat digunakan untuk mengkonversi gambar yang mengandung teks dari hasil cetakan mesin atau printer menjadi file dokumen yang memiliki teks yang dapat dibaca dan diedit oleh aplikasi komputer. Anda dapat melihat foto tanpa mengetik ulang. Namun, meskipun teknologi OCR berkembang pesat, pengguna dan pemilik dokumen sering mengalami kesulitan mengidentifikasi karakter berupa huruf, angka, atau simbol pada gambar atau gambar. Selain

itu, memproses ulang gambar yang berisi teks atau dicetak dengan mesin adalah tantangan. Data foto juga masih diambil secara manual dengan mengetik. Selain itu, kesalahan pengetikan yang sering mungkin memerlukan pengeditan tambahan setelah dokumen diketik ulang dapat menyebabkan masalah saat mengubah gambar yang berisi teks atau hasil cetakan mesin dan printer menjadi file dokumen yang dapat dibaca dan diedit oleh aplikasi komputer. Hasilnya, sebuah aplikasi dikembangkan yang memiliki kemampuan untuk mengompresi foto dan mengubahnya menjadi file teks. Aplikasi yang direncanakan belum mendukung tulisan tangan dengan berbagai gaya dan karakter, menurut Chairunnas (2017).

Penelitian ini dilatarbelakangi oleh suatu permasalahan dimana suatu sistem yang membutuhkan pengembangan dalam hal teknologi untuk mendeteksi karakter pada citra gambar, salah satu contohnya adalah pendeteksian teks pada citra kutipan, karena sistem sebelumnya masih melakukan input teks secara manual. *Optical Character Recognition* telah banyak digunakan untuk mengekstrak karakter yang terdapat pada media citra digital (Qashlim et al., 2022; Pino, et al., 2021; Nazaruddin, 2022).

1.1 Tinjauan Pustaka

1.1.1 Aplikasi Android

Aplikasi berbasis Android telah banyak digunakan untuk melakukan aktivitas sehari-hari, seperti bermain game, berbelanja *online*, streaming video, dan menonton pertandingan olahraga. Aplikasi yang paling populer membantu pengguna memesan hotel, tiket pesawat, dan aktivitas wisata.

Akhir-akhir ini, orang lebih sering menggunakan media digital melalui ponsel dan aplikasi. Terlihat sulit untuk membuat aplikasi seluler. Dalam siklus hidup pengembangan aplikasi, kita akan melihat berbagai jenis aplikasi yang dapat dikembangkan serta tingkat kompleksitas aplikasi yang sedang dikembangkan. Kami akan membahas tujuan dan persyaratan aplikasi seluler yang harus ditentukan oleh pelanggan.

1.1.2 Flutter

Flutter, yang dipilih oleh *Google* sebagai kerangka kerja tingkat aplikasi untuk generasi berikutnya sistem operasi, diluncurkan untuk umum pada tahun 2016 dan tersedia di *Fuschia* selain di Android dan *iOS*. *Flutter* luar biasa karena menggunakan *widget* OiEM perangkat daripada tampilan web. Setiap elemen tampilan *Flutter* dibuat menggunakan mesingan Setelah pemuatan ulang panas *stateful*, kode sumber yang telah diperbarui dikirim ke *Dart Virtual Machine* (*Dart VM*) yang sedang berjalan tanpa mengubah struktur bagian aplikasi. Dengan demikian, transisi dan operasi aplikasi akan terjaga dengan baik setelah pemuatan ulang panas.

1.1.3 Optical Character Recognition (OCR)

Proses mengenali karakter dalam gambar atau gambar, seperti huruf, angka, atau simbol, disebut mengenali karakter optik (OCR). Ini dapat digunakan untuk mengubah gambar teks dari hasil cetakan mesin atau printer menjadi file dokumen yang memiliki teks yang dapat dibaca dan diedit oleh aplikasi komputer (Van Hoai et al., 2021).

Pengenalan karakter optik (OCR) adalah salah satu metode untuk mengubah gambar yang berisi teks menjadi karakter ASCII yang dapat dibaca komputer. Diantara metode yang dapat diterapkan untuk mengenali karakter optik adalah logika *fuzzy*, jaringan syaraf tiruan, pencocokan matriks, ekstraksi ciri, dan analisis struktur (Julisawaty dkk, 2020).

1.1.4 Google Cloud Platform (GCP)

Setelah diluncurkan pada tahun 2011, Google Cloud Platform (GCP) telah bekerja sama dengan perusahaan besar seperti *Airbus, Coca-Cola, Spotify, HTC, Equinix, Intel,* dan *Red Hat.* GCP juga menawarkan berbagai opsi penyimpanan, seperti penyimpanan, penyimpanan, database, jaringan, big data, pembelajaran mesin, alat manajemen, alat pengembang, dan

identitas dan keamanan. Database MySQL, *Cloud Datastore*, dan *Cloud Storage* dikenakan biaya terpisah per bulan berdasarkan kapasitas GB.

1.1.5 Dart

Dart, bahasa pemrograman yang digunakan secara luas oleh *Google*, digunakan untuk membuat setiap aplikasi *Flutter*. Ini terbukti memiliki kemampuan untuk membuat aplikasi web yang signifikan, seperti *AdWords*. Untuk menggantikan *JavaScript*, Dart memiliki sebagian besar fitur penting dari standar *JavaScript* berikutnya. Namun, sintaks Dart mirip dengan Java menarik pengembang yang tidak terbiasa dengan *JavaScript*. Pohon tampilan aplikasi Flutter diperbarui pada setiap frame baru, bahkan dalam kasus sistem lain yang menggunakan tampilan reaktif. Perilaku ini menghasilkan banyak objek yang dapat bertahan untuk satu frame. Dengan bantuan "Pengumpulan Sampah Generasi", *Dart* dioptimalkan untuk menangani situasi ini di tingkat memori.

1.1.6 Figma

Salah satu alat desain yang paling umum adalah Figma, yang dapat digunakan untuk membuat tampilan aplikasi *mobile*, desktop, website, dan lainnya. Dengan menggunakan internet, *Figma* dapat digunakan di sistem operasi *Windows*, *Linux*, dan *Mac*. Figma memiliki keunggulan, yaitu lebih dari satu orang dapat bekerja sama untuk tugas yang sama bahkan di tempat yang berbeda. Aplikasi figma memungkinkan banyak desainer UI/UX untuk membuat *prototype website* atau aplikasi dengan cepat dan efektif, yang membuatnya cocok untuk kerja kelompok.

1.1.7 Jaro-Winkler

JaroWinkler menyatakan bahwa algoritme *Jaro-Winkler* adalah versi metrik *Jaro Distance* yang lebih baik untuk string pendek dan biasanya digunakan untuk mencatat bidang tautan yang mengandung entri duplikat. Jika skornya 0 dan skor satu menunjukkan kesamaan yang sama, maka tidak ada kesamaan sama sekali. Untuk string pendek, algoritma ini memiliki kompleksitas runtime kuadrat yang baik. Namun, ini mungkin berhasil untuk string yang lebih panjang.

Jaro-Winkler adalah variasi dari *Jaro distance* metrik, sebuah algoritma yang biasanya digunakan untuk mendeteksi duplikat dan digunakan untuk mengukur kesamaan antara dua string. *String* yang lebih mirip semakin jauh jarak antara dua string. Untuk perbandingan singkat seperti nama orang, *Jaro-Winkler* menemukan jarak ideal. Skor normal 0 menunjukkan ketidaksamaan, sedangkan skor 2 menunjukkan kesamaan.

Meskipun memiliki kompleksitas waktu proses kuadrat dan kurang efektif pada *string* pendek, algoritma edit jarak dapat berjalan lebih lambat dibandingkan algoritma jarak *Jaro-Winkler*. Panjang string, jumlah karakter yang sama dalam dua *string*, dan jumlah transposisi adalah tiga komponen utama algoritma ini. Rumus algoritma *Jaro-Winkler* untuk menghitung jarak (dj) antara dua string, s1 dan s2, tersedia dalam persamaan (1) :

$$d_j=1/3\ X(m/s1+m/s2+(m-t)/s)$$

Di mana m adalah jumlah karakter yang sama persis, |s1| adalah panjang string 1, |s2| adalah panjang string 2, dan T adalah jumlah transposisi. Selain itu, dj adalah nilai jarak antara dua string yang dibandingkan.

Untuk memberikan tingkat penilaian yang lebih besar, *Jaro-Winkler* jarak menggunakan skala awalan (p) dan panjang awalan (l), yang merupakan panjang karakter yang sama dari string yang dibandingkan sampai ditemukan ketidaksamaan. *Jaro-Winkler* jarak (dw) adalah seperti dalam persamaan (2) jika string s1 dan s2 diperbandingkan :

$$dw = dj + (lp(1 - dj))$$

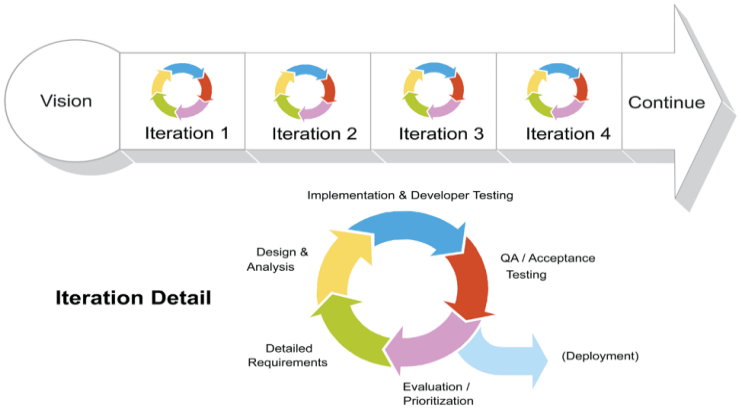
Dengan p=0,1, l adalah panjang awalan umum di awal string untuk nilai maksimum empat karakter (panjang karakter yang sama sebelum ditemukan ketidaksamaan empat), dan dw adalah nilai jarak *Jaro-Winkler*

1.1.8 Struktur Navigasi

Struktur navigasi aplikasi menggambarkan isi aplikasi secara keseluruhan; dalam penelitian ini, struktur navigasi adalah struktur hirarki. Karena alur aplikasi menampilkan gambar pada layer tertentu, pilihan bentuk struktur navigasi ini dibuat. Tampilan memiliki halaman percabangan pada menu halaman utama. Ini disebut halaman pendukung atau halaman pembantu. Anda akan dibawa ke halaman kedua dan seterusnya jika Anda memilihnya. Struktur navigasi ini bertujuan untuk menjelaskan hubungan dan rantai kerja seluruh komponen yang digunakan dalam aplikasi.

Sistem navigasi dalam arsitektur informasi harus seimbang dan fleksibel sesuai konteksnya. Jika sistem navigasi tidak fleksibel, terlalu banyak cabang, atau terlalu kompleks, seperti banyak *hyperlinks*, pengguna akan tersesat dan kesulitan menemukan konten yang mereka butuhkan. Akibatnya, sistem navigasi pada aplikasi ponsel menjadi tidak efisien.

2. METODE PENELITIAN



Gambar 1 iterational detail

Metode penelitian ini menggunakan algoritma *Jaro Winkler*. Ekstraksi Teks menggunakan OCR Untuk mengekstraksi teks dari gambar atau dokumen berbasis gambar, gunakan perangkat lunak OCR seperti *Tesseract* atau *Google Cloud Vision*. Teks *Preprocessing* untuk membersihkan dan normalisasi teks yang baru diekstrak. Penghapusan karakter non-alfanumerik, konversi ke huruf kecil, dan tindakan tambahan untuk meningkatkan kecocokan mungkin termasuk dalam hal ini. Implementasi Algoritma *Jaro-Winkler*, gunakan algoritma ini pada teks yang telah diekstraksi dan ingin dibandingkan. Implementasi algoritma ini tersedia dalam beberapa bahasa pemrograman. Pengambilan Keputusan untuk menetapkan ambang batas, atau threshold, untuk menentukan seberapa mirip string-string harus untuk dianggap cocok. Ambang batas yang dipilih bergantung pada kasus yang digunakan.

2.1 Fokus Penelitian

Fokus penelitian pengembangan aplikasi konversi gambar ke teks dapat mencakup beberapa aspek penting. mengembangkan algoritma pengenalan gambar yang akurat dan efisien. Ini membutuhkan pemrosesan gambar untuk menemukan pola dan teks dalam gambar. Untuk mendukung pengenalan teks pada berbagai jenis gambar dengan berbagai kualitas, perbaikan atau peningkatan teknologi OCR yang sudah ada. Teknik segmentasi gambar, yang dapat memisahkan teks dari latar belakang atau objek lainnya, memungkinkan deteksi teks yang lebih

baik. mengoptimalkan kinerja aplikasi untuk mengatasi masalah kecepatan dan skala, terutama ketika digunakan dalam lingkungan waktu nyata atau dengan volume gambar yang besar.

2.2 Sumber Data

Penelitian saat ini sedang dilakukan, dan data dikumpulkan dari karya peneliti sebelumnya yang terkait untuk digunakan sebagai referensi. Pada akhirnya, data ini akan digunakan untuk membantu membuat aplikasi.

2.3 Teknik Pengumpulan Data

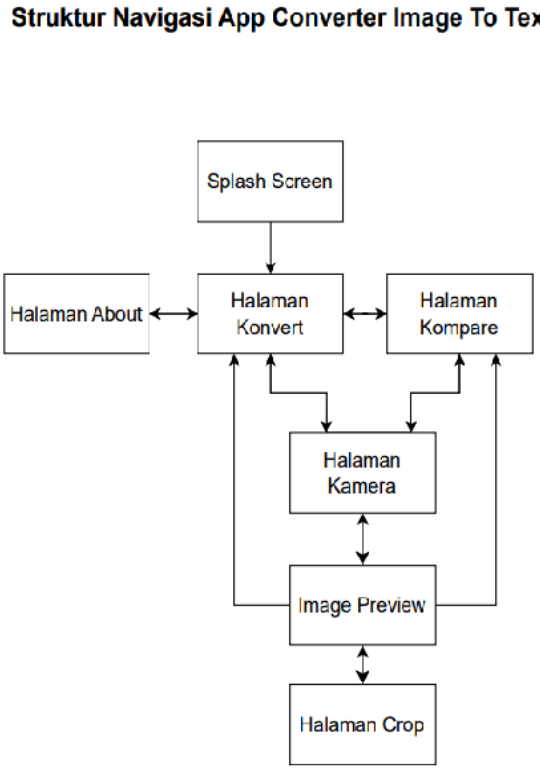
Data yang dikumpulan untuk pengembangan aplikasi android konversi gambar ke teks ini diambil dari berbagai artikel artikel dan jurnal jurnal terdahulu untuk mendapatkan berbagai informasi yang ada.

2.4 Struktur Aplikasi

Pada gambar struktur navigasi dibawah terdapat tata cara sebagai berikut :

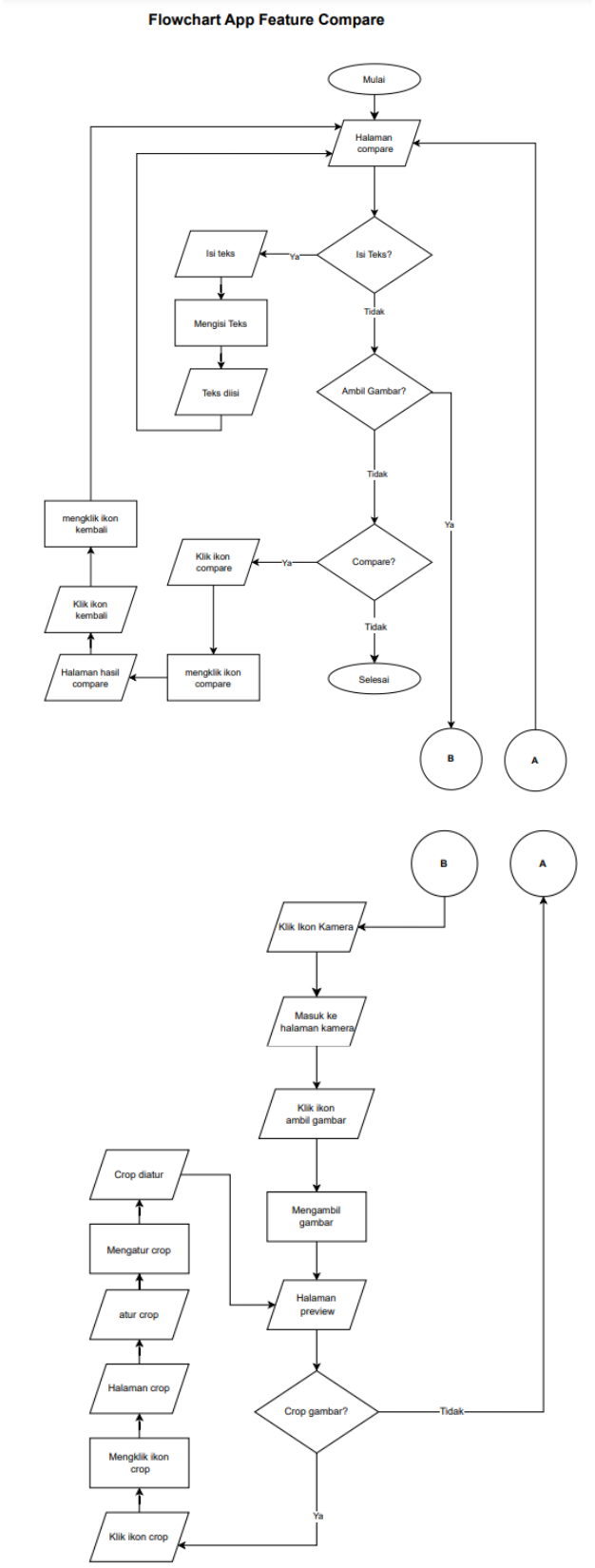
Aplikasi dimulai dari *Splash Screen* sebagai halaman awal. Setelah *Splash* Sicreen, *user*/pengguna secara default masuk ke halaman homie, yang memiliki fitur utama yaitu :

a. *Convert image to teks*
b. Hasil teks
c. *Select image*
d. *Compare teks*
e. Hasil *compare teks*

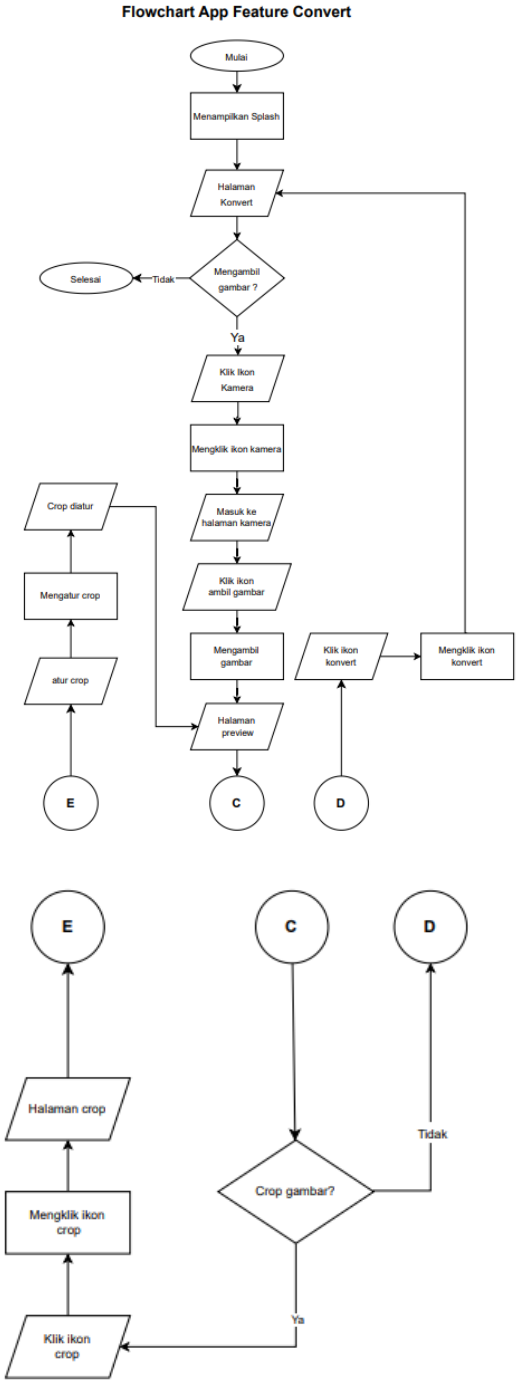


Gambar 2 Struktu Navigasi

DOI: ISSN: **2356-5209**
pp. 271~270 *Online* ISSN: **2655-3058**



Gambar 3 *Flowchart App Feature Compare*



Gambar 4 *Flowchart App Feature Convert*

Algoritma *Flowchart App Converter Image to Text:*

1. Aplikasi dimulai pada halaman *Splash Screen* (Halaman awal)
2. Setelah melalui *Splash Screen*, *user* akan langsung menuju pada halaman *Home*. Pada halaman home terdiri dari beberapa fitur, yaitu:
   a) *Text Field* digunakan untuk mengisi teks yang akan di *convert*
   b) *Button Take a Picture* digunakan untuk mengambil gambar yang akan di *convert*
   c) *Camera* digunakan untuk mengambil gambar yang akan di *convert*
   d) *About* digunakan untuk memberikan informasi terkait aplikasi

3. *User* dapat memulai untuk *convert* dengan *text to image* atau i
4. Jika *user* memilih metode *convert text to image*, maka:
   a) *User* dapat memasukkan teks pada fitur *Text Field*
   b) Kemudian user melakukan aksi dengan mengklik *button* (*Take a Picture*)
   c) *User* akan menuju ke halaman Camera untuk mengambil gambar yang akan di *convert*
   d) *User* masuk ke halaman *Image Preview* untuk melihat hasil dari pengambilan gambar. Jika gambar tidak sesuai, user akan masuk ke halaman *Cropper*. Dan jika gambar sesuai maka user akan menuju halaman *Image Preview.*
   e) Apabila gambar telah sesuai, user akan menuju pada halaman *Preview*, kemudian *user* melakukan aksi dengan klik *button* untuk *convert*
   f) *User* akan diarahkan ke halaman Perbandingan untuk melihat hasil dari keakuratan dari text dan gambar.
5. Jika *user* memilih metode *convert image to text*, maka :
   a) *User* mengklik fitur Camera
   b) *User* akan menuju ke halaman Camera untuk mengambil gambar yang akan di *convert*
   c) *User* masuk ke halaman *Image Preview* untuk melihat hasil dari pengambilan gambar. Jika gambar tidak sesuai, *user* akan masuk ke halaman *Cropper*. Dan jika gambar sesuai maka user akan menuju halaman *Image Preview.*
   d) Apabila gambar telah sesuai, *user* akan menuju pada halaman *Preview*,
   e) Kemudian user memasukan teks pada fitur Text Field di halaman Preview
   f) Apabila teks dan gambar telah sesuai, maka *user* melakukan aksi dengan klik *button* untuk *convert*
   g) *User* akan diarahkan ke halaman Perbandingan untuk melihat hasil dari keakuratan dari gambar dan text.
6. Hasil dari *convert text to image* atau *image to text* akan muncul pada halaman perbandingan
7. Jika *user* ingin melihat informasi mengenai aplikasi, maka user dapat mengklik fitur *about*
8. Aplikasi selesai dan user dapat melakukan tindakan lain sesuai dengan kebutuhan.

Metode penelitian ini menggunakan algoritma *Jaro Winkler*. Ekstraksi Teks menggunakan OCR Untuk mengekstraksi teks dari gambar atau dokumen berbasis gambar, gunakan perangkat lunak OCR seperti *Tesseract* atau *Google Cloud Vision*. Teks *Preprocessing* untuk membersihkan dan normalisasi teks yang baru diekstrak. Penghapusan karakter non-alfanumerik, konversi ke huruf kecil, dan tindakan tambahan untuk meningkatkan kecocokan mungkin termasuk dalam hal ini. Implementasi Algoritma *Jaro-Winkler*, gunakan algoritma ini pada teks yang telah diekstraksi dan ingin dibandingkan. Implementasi algoritma ini tersedia dalam beberapa bahasa pemrograman. Pengambilan Keputusan untuk menetapkan ambang batas, atau *threshold*, untuk menentukan seberapa mirip string-string harus untuk dianggap cocok. Ambang batas yang dipilih bergantung pada kasus yang digunakan.

3. HASIL DAN PEMBAHASAN

3.1 Hasil Penelitian

Penelitian ini menganalisis kinerja aplikasi Konversi Gambar ke Teks. Beberapa metrik yang diukur termasuk kecepatan, akurasi, dan kemampuan aplikasi untuk menangani Konversi Gambar ke Teks. Hasil pengujian menunjukkan bahwa aplikasi ini dapat mengkonversi gambar ke teks. Meskipun aplikasi dapat mengkonversi gambar ke teks, terdapat beberapa kesalahan saat mengkonversi gambar yang kurang jelas. Adanya ketidaktepatan dalam mengkonersi gambar ke teks jika gambar yang ingin di konversi kurang jelas.

Evaluasi ini bertujuan untuk mengukur tingkat akurasi dan kebehasilan dalam mengkonversi gambar ke teks. Meskipun ada beberapa masalah aplikasi konversi gambar ke teks, potensi aplikasi untuk mengkonversi gambar ke teks tetap akurat dan signifikan. Masalah

yang ada tergantung dari resolusi kamera user, semakin bagus dan tinggi resolusi kamera user semakin akurat dan signifikan dalam mengkonversi gambar ke teks.

3.2 Tahap Analisis

Analisis fungsional dan nonfungsional termasuk dalam tahap analisis, yang mencakup pengumpulan data yang diperlukan untuk pembuatan aplikasi.

1) Analisis Fungsional

Analisis fungsional merupakan analisis terhadap fungsi-fungsi yang diperlukan pada aplikasi konversi gambar ke teks antara lain:

a. Aplikasi dapat mengaktifkan flash untuk mengkonversi gambar ke teks dalam pencahayaan yang kurang memadai
b. Aplikasi dapat memotong atau *cropper* gambar yang ingin dikonversi menajadi teks
c. Aplikasi dapat mengkompare atau membandingkan teks dengan gambar yang ingin dikonversi menggunakan algortima *jaro winkler* yang terdapat pada aplikasi
d. Aplikasi dapat melakukan salin dan tempel, user dapat menyalin gambar yang sudah dikonversi menjadi teks.

2) Analisis Nonfungsional

Analisis untuk konten program, perangkat keras, dan perangkat lunak diperlukan selama pengembangan aplikasi. Analisis konten program membahas materi yang akan disajikan, sementara analisis perangkat keras membahas spesifikasi perangkat keras yang dibutuhkan. Ini mencakup semua perangkat keras yang diperlukan untuk menjalankan lingkungan pengembangan seperti Android Studio dan Visual Studio Code. Berikut adalah perangkat keras minimal yang diperlukan untuk Android Studio:

a. Microsoft Windows 7/8/10
b. 3 GB RAM
c. 2 GB hard disk space

Untuk Visual Studio Code, spesifikasi minimal adalah sebagai berikut:

a. Microsoft Windows 7
b. *Intel Core i3* atau AMD Athlon 64
c. 2 GB RAM d) 2 GB *hard disk space* Untuk *hardware* yang dapat mengjalankan aplikasi *Transform Pixel* ialah *smartphone* dengan sistem operasi Android versi 4.0.x atau lebih baru.

3.3 Desain Aplikasi

Pada titik ini, desain aplikasi konversi gambar ke tesk dibuat dengan teliti. Ini juga mempermudah user yang ingin mengambil teks yang ada digmabar tanpa harus melakukan penginputan manual atau pengetikaan.

1) Halaman *Splash Screen*

Splash screen, atau halaman awal aplikasi. Aplikasi ini bernama "*Transform Pixel*" yang berarti mengubah gambar menjadi kata atau teks. Berikut gambar halaman *Splash Screen* :

Gambar 5 Halaman Splash Screen

2) Halaman Home

Halaman home adalah halaman yang menyediakan fitur konversi gambar ke teks, dihalaman home terdapat beberapa fitur yaitu :

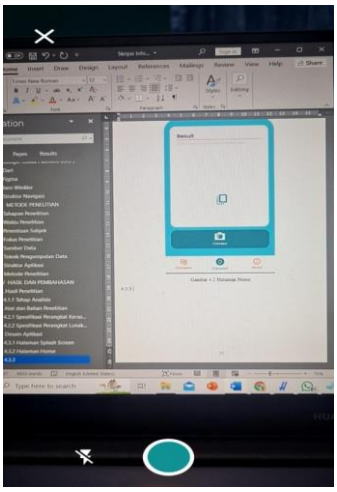
a. Hasil dari konversi
b. Fitur salin
c. Kamera yang digunakan untuk mengambil gambar yang akan dikonversi ke tesk
d. Fitur *about*
e. Fitur *compare*

Gambar 6 Halaman Home

3) Halaman Kamera

Halaman Kamera digunakan untuk mengambil gambar yang ingin dikonversi menjadi teks, terdapat beberapa fitur yang ada dikamera yaitu:

a. Fitur *Flash* yang digunakan untuk memberi pencahayaan yang lebih terang dan jelas
b. Fitur menngambil gambar
c. Fitur silang (X) untuk kembali ke halaman *home*

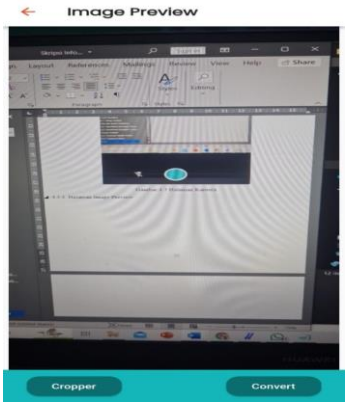


Gambar 7 Halaman Gambar

4) Halaman *Image Preview*

Halaman *image preview* adalah halaman setelah gambar yang sudah di ambil dihalaman kamera untuk mengetahui gambar yang di ambil sudah jelas dan benar. Ada beberapa fitur yang ada pada halaman image preview yaitu :

a. Fitur *cropper*
b. Fitur *convert* untuk mengkonversi gambar yang akan dan sudah di ambil oleh *user* ke teks.

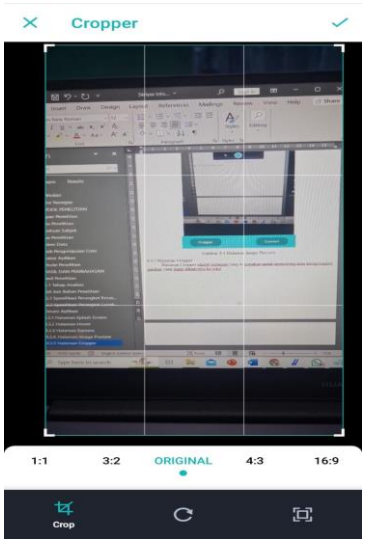


Gambar 8 Halaman *Image Preview*

5) Halaman *Cropper*

Halaman *Cropper* adalah halaman yang di gunakan untuk memotong atau mengcropper gambar yang akan dikonversi ke teks. Ada beberapa fitur yang ada di halaman *Cropper* yaitu:

a. Fitur *Crop* untuk memotong gambar, terdapat juga ukuran yang dapat user gunakan untuk memotong gambar yang akan di konversi ke tesk, dan dapat mengatur besar, kecilnya gambar yang akan di potong atau di *crop*
b. Fitur *Rotate* untuk merotate gambar yang akan di konversi ke teks, terdapat juga ukuran yang akan digunakan untuk gambar yang sudah diambil
c. Fitur *Scale* untuk memperbesar dan memperkecil gambar yang akan dikonversi ke teks, terdapat juga berbagai skala yang akan digunakan user.
d. Fitur ceklis (✓) untuk gambar yang telah selesai di *crop* dan kembali ke halaman image *preview* untuk dikonversi menjadi teks



Gambar 9 Halaman *Cropper*

6) Halaman *About*

Halaman *About* adalah halaman penjelasan tentang aplikasi yang dibuatnya halaman ini membahas tentang profil pembuat dan penjelasan singkat apa yang dibuatnya.

Gambar 10 Halaman *About*

7) Halaman *Compare*

Halaman *Compare* adalah halaman yang digunakan untuk mengcompare teks yang didapat dari hasil konversi dengan gambar lain, untuk mengetahui kemiripan antara teks yang didapat dari gambar yang sudah di konversi dengan teks yang ada digambar lain. Ada beberapa fitur yang terdapat pada halaman *convert* yaitu:

a. Fitur kamera untuk mengambil gambar lain yang ingin dikonver dengan teks hasil konversi dari gambar sebelumnya
b. Fitur *compare* untuk mengetahui seberapa persen kemiripan teks yang didapat dari hasil konversi gambar sebelumnya dengan gambar lain.

Gambar 11 Halaman *Compare*

3.4 Tahap Implementasi

Proses menerapkan suatu rencana atau kebijakan disebut sebagai tahap implementasi. Ini adalah langkah penting dalam siklus pengembangan atau perubahan di berbagai situasi, termasuk dalam pengembangan perangkat lunak. Pada tahap implementasi, kode yang telah dirancang dan diuji akan diimplementasikan dalam proses produksi atau dimasukkan ke dalam produk yang sebenarnya. Instalasi, konfigurasi, dan pengujian perangkat lunak adalah bagian dari proses ini untuk memastikan bahwa itu berfungsi dengan baik dan memenuhi kebutuhan pengguna.

3.5 Implementasi Pemrograman

Jika perintah tidak digunakan, aplikasi tidak dapat dijalankan. perintah yang dimaksudkan untuk memungkinkan aplikasi menjalankan fungsinya. Supaya fungsi dapat dijalankan, bahasa pemrograman diperlukan. Android menggunakan *Dart* sebagai bahasa pemrograman default. File yang digunakan dalam aplikasi sinaubasa dibagi ke dalam empat paket: paket *utilities*, *view*, dan main.

3.6 Implementasi Algoritma Jaro-Winkler

Pada Aplikasi *Transform* Pixel menggunakan algortima *Jaro-Winkler* yang bertujuan untuk mencompare teks yang didapat pada hasil konversi gambar. Algoritma *Jaro-Winkler* bertujuan untuk mencari kesamaan antara teks hasil konversi dengan gambar baru yang ingin dicompare.

```
double jaroWinklerSimilarity(String a, String b) {
  double jaroSimilarityValue = jaroSimilarity(a, b);

  // Winkler modification
  double prefixScale = 0.1; // Winkler scaling factor (0.1 is a common value)
  int commonPrefixLength = 0;

  for (int i = 0; i < min(4, min(a.length, b.length)); i++) {
    if (a[i] == b[i]) {
      commonPrefixLength++;
    } else {
      break;
    }
  }

  double similarityPercentage = (jaroSimilarityValue +
          (commonPrefixLength * prefixScale * (1 - jaroSimilarityValue))) *
      100;

  return similarityPercentage;
}
```

```
double jaroSimilarity(String a, String b) {
  if (a == b) {
    return 1.0;
  }

  int matchingCharacters = 0;
  int transpositions = 0;

  int maxLength = max(a.length, b.length);
  int matchDistance = (maxLength ~/ 2) - 1;

  List<bool> aMatched = List.filled(a.length, false);
  List<bool> bMatched = List.filled(b.length, false);

  for (int i = 0; i < a.length; i++) {
    int start = max(0, i - matchDistance);
    int end = min(i + matchDistance + 1, b.length);

    for (int j = start; j < end; j++) {
      if (!bMatched[j] && a[i] == b[j]) {
        aMatched[i] = true;
        bMatched[j] = true;
        matchingCharacters++;
        break;
```

```
if (matchingCharacters == 0) {
  return 0.0;
}

int k = 0;
for (int i = 0; i < a.length; i++) {
  if (aMatched[i]) {
    while (!bMatched[k]) {
      k++;
    }
    if (a[i] != b[k]) {
      transpositions++;
    }
    k++;
  }
}

double jaroSimilarityValue = (matchingCharacters / a.length +
        matchingCharacters / b.length +
        (matchingCharacters - transpositions / 2) / matchingCharacters) /
    3.0;

return jaroSimilarityValue;
```

Gambar 12 *Sorce Code Algoritma Jaro Winkler*

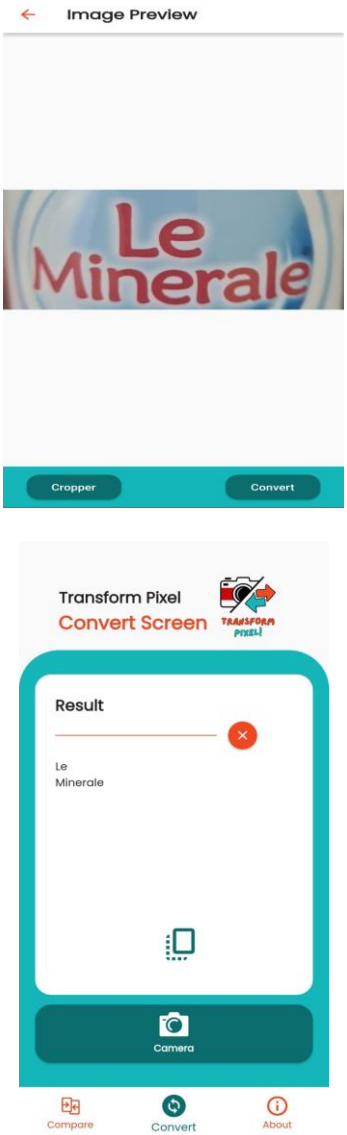
Algoritma *Jaro-Winkler* memiliki perhitungannya sendiri untuk menentukan kemiripan atau kesamaan dari teks hasil konversi dengan teks yang ada pada gambar digital.

3.7 Uji Coba Aplikasi Transform Pixel

Uji coba aplikasi adalah proses di mana sebuah aplikasi atau perangkat lunak diuji untuk menemukan masalah, memastikan kinerja yang diharapkan, dan memvalidasi fungsionalitasnya sebelum dirilis secara resmi kepada pengguna akhir. Ini dilakukan untuk memastikan bahwa aplikasi tersebut berfungsi dengan baik dan memenuhi kebutuhan dan tujuan yang ditetapkan. Uji coba biasanya terdiri dari sejumlah pengujian dasar hingga pengujian kinerja dan beban yang lebih mendalam. Pengujian ini dapat membantu pengembang menyelesaikan masalah, meningkatkan kinerja, dan memastikan bahwa aplikasi siap untuk digunakan secara luas.

3.8 Uji Coba Konversi Gambar ke Teks

Berikut gambar uji coba konversi gambar ke teks :



Gambar 13 Uji Coba Konversi Gambar Ke Teks

3.9 Uji Coba Algoritma Jaro-Winkler

Peneliti melakukan uji coba algoritma *Jaro-Winkler* dengan melakukan perhitungan manual dengan contoh kata "BRI" dan "BNI" sebagai berikut :

1) Menghitung *Jaro Similarity*:
   a. Jumlah karakter yang sama (m) antara kedua *string*: 2 (yaitu B dan I).
   b. Jumlah karakter yang berbeda (t) di kedua *string*: 1 (yaitu R dan N).
   c. Jumlah karakter yang cocok pada setengah jarak maksimum (c): 0 (karena tidak ada karakter yang cocok pada setengah jarak maksimum).
   d. Panjang *string* pertama (m1) = 3 (BRI)
   e. Panjang *string* kedua (m2) = 3 (BNI)

Jaro Similarity (JS) dihitung menggunakan rumus berikut:

JS = (m / m1 + m / m2 + (m - c) / m) / 3
JS = (2/3 + 2/3 + (2-0)/2) / 3
JS = (0.6667 + 0.6667 + 1) / 3
JS = 2.3334 / 3

JS ≈ 0.7778

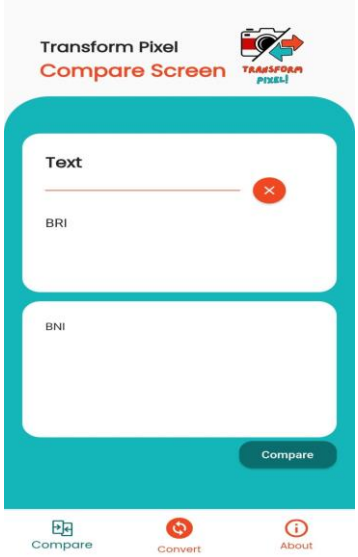
2) Menghitung *Jaro-Winkler Similarity*:
   a. *Jaro Similarity* yang telah dihitung sebelumnya: JS ≈ 0.7778
   b. Nilai p (p = 0.1) yang ditetapkan untuk menghitung *Jaro-Winkler Similarity*.

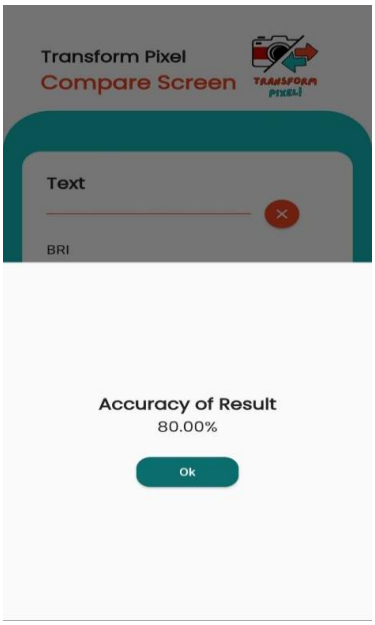
Jaro-Winkler Similarity (JWS) dihitung menggunakan rumus berikut:

JWS = JS + (l * p * (1 - JS))

Di mana l adalah panjang awal *string* yang cocok dalam *string* yang sama (dalam kasus ini, tidak ada karakter yang cocok pada posisi awal).
JWS = 0.7778 + (0 * 0.1 * (1 - 0.7778))
JWS = 0.7778 + (0 * 0.1 * 0.2222)
JWS = 0.7778 + 0
JWS ≈ 0.7778

Jadi, nilai *Jaro-Winkler Similarity* antara "BRI" dan "BNI" adalah sekitar 0.7778 Sekitar 77%. Semakin tinggi nilai *Jaro-Winkler Similarity*, semakin mirip kedua string tersebut. Dalam contoh ini, kedua string memiliki tingkat kesamaan yang cukup tinggi.





Gambar 14 Uji Coba Algoritma *Jaro Winkler* Pada Aplikasi

4. KESIMPULAN

Aplikasi *Transform Pixel* ini memiliki kemampuan yang baik untuk mengkonversi teks yang ada pada gambar digital menjadi teks yang bisa di salin dan bisa mencompare teks hasil konversi dari gambar digital dengan gambar digital lainnya. Aplikasi ini juda dapat menganalisa tulis tangan, angka-angka, simbol-simbol, berbagai tipe huruf pada kata dan bisa melakukan *cropper* atau memotong gambar maupun dapat melakukan zoom in dan zoom *out* pada gambar digital yang akan dikonversi dan dicompare. Implementasi Algoritma *Jaro-Winkler* pada aplikasi ini akurat dan signifikan karena dalam mengcompare 2 kata sampel yaitu "BRI" dan "BNI" dengan hasil 0,7778 atau dalam nilai persentase 77% melalui perhitungan manual. Berdasarkan perbandingan nilai pada aplikasi dan perhitungan menggunakan *phyton* dengan representasi *floating-point*, menyebabkan perbedaan kecil dalam hasil akhir dengan sesilih 3% yang hasilnya menjadi 80%.

DAFTAR PUSTAKA

[1] Agustina Julisawaty, E., & Munich Heindari Ekasari, dan. (2020). Struktur Navigasi. Universitas Gunadarma Jl. Margonda Raya, 4(1), 16424.

[2] Al-Faruq, M. N. M., Nur'aini, S., & Aufan, M. H. (2022). FIGMA. Walisongo Journal of Information Technology, 4(1), 43–52. https://doi.org/10.21580/wjit.2022.4.1.12079

[3] Napitupulu, R. I., Fikri Mauludin, M., Gianadevi, F., & Rumambi, T. (2023). Rancang Bangun Aplikasi Konversi File Berformat Image Menjadi File Berformat Txt.

[4] Novie Tri Lestari, I., Iskandar Mulyana, D., & Cipta Karya Informatika, S. (2022). Implementation Of Ocr (Optical Character Recognition) Using Tesseract In Detecting Character In Quotes Text Images. In Journal of Applied Engineering and Technological Science (Vol. 4, Issue 1).

[5] Penerapan Teknologi Informasi dan Komunikasi, J., Bryan Prasetio, F., & Wellem, T. (n.d.). ANDROID JURNAL.

[6] Prasetyo, A., Baihaqi, W. M., & Had, I. S. (2018). Algoritma Jaro-Winkler Distance:

Fitur Autocorrect dan Spelling Suggestion pada Penulisan Naskah Bahasa Indonesia di BMS TV. Jurnal Teknologi Informasi Dan Ilmu Komputer, 5(4), 435–444. https://doi.org/10.25126/jtiik.201854780

[7] Puspita Sari, R., Rahmayuda, S., Sistem Informasi, J., Mipa, F., Tanjungpura Jalan ProfDrH Hadari Nawawi, U., & Telp, P. (2022). Coding : Jurnal Komputer dan Aplikasi Implementasi Framework Flutter Pada Sistem Informasi Perpustakaan Masjid (Studi Kasus: Masjid di Kota Pontianak).

[8] Retno, A., Ririd, T. H., Saputra, P. Y., Sastri, A. M., Informatika, T., Informasi, T., & Malang, P. N. (2019). Sistem Koreksi Kesalahan Pengetikan Kata Kunci dalam Pencarian Artikel Menggunakan Algoritma Jaro-Winkler.

[9] Salim, R. A., Septian, M. R. D., Suhartini, S., Anggraini, D., & Qomariyah, Q. (2021). Aplikasi Pendeteksi Kesamaan Dokumen Dengan Menggunakan Algoritma Jarak Jaro Winkler Dan Levenshtein. Sebatik, 25(1). https://doi.org/10.46984/sebatik.v25i1.1309

[10] Sanjaya, A. (2020). Optimasi Pencarian Data Menggunakan Text Filtering Dan Algoritma Jaro Winkler. In Jurnal Ilmiah NERO (Vol. 5, Issue 1).

[11] Soedewi, S., Swasty, W., Mustikawan, A., & Naufalina, F. E. (2021). Struktur Navigasi (2). Jurnal Bahasa Rupa, 5(1), 22–34. https://doi.org/10.31598/bahasarupa.v5i1.848

[12] Tashildar, A., Shah, N., Gala, R., Giri, T., & Chavhan, P. (1262). Application Development Using Flutter. In International Research Journal of Modernization in Engineering Technology and Science @International Research Journal of Modernization in Engineering. www.irjmets.com

[13] Tinaliah, T., & Elizabeth, T. (2018). Perbandingan Hasil Deteksi Plagiarisme Dokumen dengan Metode Jaro-Winkler Distance dan Metode Latent Semantic Analysis. Jurnal Teknologi Dan Sistem Komputer, 6(1), 7–12. https://doi.org/10.14710/jtsiskom.6.1.2018.7-12

[14] Wankhede, P., Talati, M., & Chinchamalatpure, R. (2020). GCP. In International Journal of Research in Engineering and Applied Sciences ISSN (Vol. 05).

[15] Wibawa, C., & Anggraeni, D. T. (2023). Comparison Of Image Segmentation Method In Image Character Extraction Preprocessing Using Optical Character Recogniton. Jurnal Teknik Informatika (Jutif), 4(3), 583–589. https://doi.org/10.52436/1.jutif.2023.4.3.956

[16] Zulhida Putri, D., Setiawan, Y., Supratman, J. W., Limun, K., & Bengkulu, K. (2018). Konversi Citra Kartu Nama Ke Teks Menggunakan Teknik OCR Dan Jaro-Winkler Distance. Jurnal TEKNOINFO, 12(1), 1–6. http://code.google.com/p/tesseract-ocr