Perancangan Website Text Scanner Untuk Konversi Handwritten ke Teks Digital Dengan Menggunakan Optical Character Recognition

1st Hafizh Ghiyats Ash Shiddiq
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
aldianadiwijaya@student.telkomuniversity.ac.id

2nd Muhammad Iqbal
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
miqbal@telkomuniversity.ac.id

3rd Aris Hartaman
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
arishartaman@telkomuniversity.ac.id

***Abstrak* — Pesatnya perkembangan teknologi saat ini menuntut kita untuk mengembangkan cara yang lebih efisien dan efektif dalam mengkonversi tulisan tangan maupun teks dokumen dalam bentuk *hardfile* menjadi teks digital. Oleh karena itu, dibutuhkan sebuah platfrom yang menggunakan teknologi *Optical Character Recognition* (OCR) untuk melakukan scan tulisan tangan maupun teks dokumen dalam bentuk *hardfile* dan mengubahnya menjadi teks digital yang dapat diedit dan disimpan Dengan demikian, dibuatkannya *website* yang dapat melakukan scan tulisan tangan dan mengubahnya menjadi teks digital. *Website* ini dirancang untuk memudahkan pengguna dalam mengkonversi tulisan tangan ke dalam format digital tanpa perlu melakukan proses manual. Pengguna hanya perlu mengunggah gambar yang terdapat tulisan tangan tersebut ke dalam *website*. OCR akan melakukan pemindaian pada gambar dan mengubahnya menjadi teks digital. Dalam perancangan *website* ini, diperhatikannya faktor-faktor penting seperti kecepatan dan akurasi pengenalan karakter oleh OCR. *Website* ini juga dilengkapi dengan fitur pengeditan teks sehingga pengguna dapat mengoreksi kesalahan pengenalan karakter oleh OCR. Perancangan ini melibatkan pengujian *text recognition* oleh tesseract.js sebanyak 10 kali untuk gambar tulisan tangan dan 10 kali untuk dokumen *hardfile* dan pengujian performa *website*. Hasil pengujian menunjukkan bahwa tingkat akurasi rata-rata pengujian OCR untuk dokumen *hardfile* adalah 99,69% dan 91,27% untuk gambar tulisan tangan.**

***Kata kunci*— *website*, OCR, optical character recognition, teks digital, tesseract.js**

I. PENDAHULUAN

Perkembangan teknologi dan inovasi telah memberikan kemajuan pada berbagai aspek kehidupan, termasuk dalam bidang digitalisasi dokumen. Salah satu inovasi yang cukup dominan adalah teknologi teks recognition untuk tulisan tangan maupun teks dokumen dalam bentuk *hardfile*. Teknologi pengenalan teks, atau yang lebih dikenal dengan *Optical Character Recognition* (OCR) [1], telah menjadi bagian integral dalam proses digitalisasi dokumen. OCR memungkinkan kita untuk mengubah dokumen *hardfile* atau tulisan tangan menjadi teks digital yang dapat diedit dan disimpan.[2]

Namun, penting juga untuk mempertimbangkan kemudahan penggunaan dan aksesibilitas teknologi ini. Salah satu solusi yang mungkin adalah dengan mengintegrasikan teknologi OCR ke dalam sebuah platfrom online, seperti *website*, yang dapat diakses oleh pengguna kapan saja dan dimana saja.[3] *Website* ini dirancang untuk kemudahan pengguna dalam mengkonversi tulisan tangan ataupun *hardfile* dokumen lainnya ke dalam format digital tanpa perlu melakukan proses manual. Pengguna hanya perlu mengunggah gambar yang terdapat teks ataupun tulisan.[4]

Dalam perancangan *website* ini, diperhatikan beberapa faktor penting seperti kecepatan dan akurasi pengenalan karakter oleh OCR.[5] Selain itu, *website* ini juga dilengkapi dengan fitur pengeditan teks sehingga pengguna dapat mengoreksi kesalahan pengenalan karakter yang dilakukan oleh OCR. Pada implementasinya, ada beberapa kebutuhan yang dapat diproses oleh teknologi OCR, salah satu contohnya adalah laporan keuangan Usaha Mikro Kecil Menengah (UMKM) yang ditulis dengan menggunakan tangan.

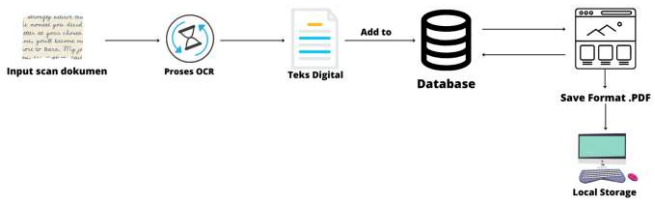
Pada perancangan sebelumnya, implementasi OCR ini tidak mengintegrasikan teknologi OCR kedalam sebuah platfrom online seperti *website*. Karena itu, dengan adanya *website* dan kemajuan teknologi serta inovasi ini, diharapkan dapat mempercepat dan membuat proses digitalisasi dokumen menjadi lebih efisien dan efektif, terutama untuk dokumen yang ditulis tangan. *Website* ini tidak hanya dapat membantu mengurangi kesalahan yang mungkin terjadi saat melakukan pengetikan manual, tetapi juga membantu individu dan organisasi dalam mengelola informasi dan data mereka dengan lebih baik.

II. KAJIAN TEORI

Sistem *Website Text scanner*

Website text scanner sebagai sistem digitalisasi dokumen menggunakan OCR. Dokumen yang sudah di *scan* dan menjadi gambar digital, lalu di input kedalam *website*, gambar akan memalui proses ke tahap selanjutnya yaitu *image preprocessing*. Setelah itu, OCR akan melakukan

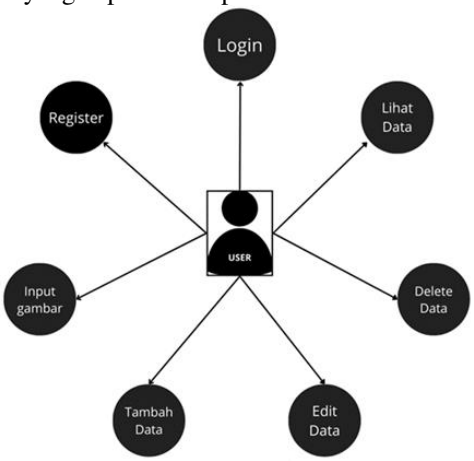
pengenalan karakter dan proses penerjemahan lalu hasilnya akan ditampilkan dalam bentuk teks digital lalu di *submit* kedalam *database* ataupun dapat disimpan di *local storage* dan riwayat teks digital dapat dilihat didalam *history website*. Berikut merupakan ilustrasi pada Sistem *Website Text scanner* yang dapat dilihat pada Gambar 1 dibawah ini



Gambar 1. Gambaran Sistem *Website Text scanner*

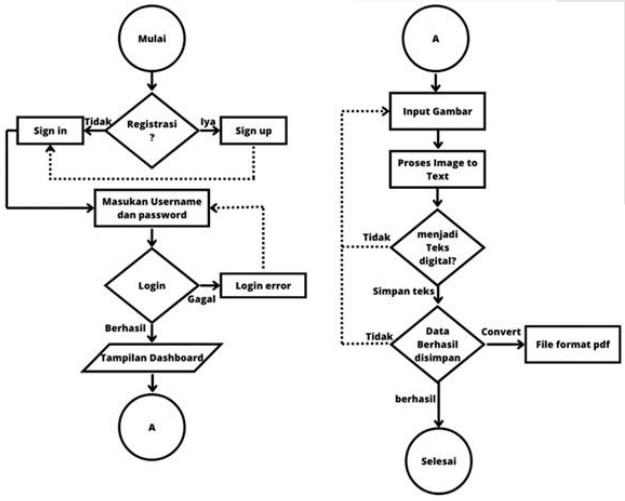
***Use Case* Diagram**

Pembuatan diagram use case berdasarkan data yang telah dikumpulkan. Diagram *use case* adalah salah satu jenis diagram UML (*Unified Modelling Language*) yang menggambarkan interaksi antara sistem dan aktor. Dalam perancangan *website text scanner*, terdapat sistem untuk pengguna yang dapat dilihat pada Gambar 2.



Gambar 2. Use Case Diagram

Alur Kerja Sistem



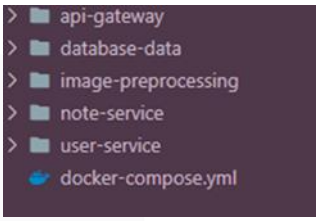
Gambar 3. Alur Kerja Sistem

Pada Gambar 3 menunjukkan alur kerja sistem dimana pada proses diatas dimulai dengan langkah pertama pengguna melakukan *login* atau registrasi, namun apabila belum terdaftar pengguna akan diminta untuk melakukan registrasi terlebih dahulu, registrasi mencakup dalam pembuatan akun. Jika pengguna sudah registrasi, pengguna akan diminta untuk memasukan *username* dan *password*, jika *login* berhasil pengguna akan diarahkan ke *dashboard*, jika gagal pengguna diarahkan untuk mencoba lagi, dari *dashboard* pengguna dapat memasukan gambar untuk diproses menjadi teks, jika gambar tidak bisa diproses maka pengguna akan diminta untuk mencoba lagi, jika gambar berhasil diproses menjadi teks maka hasilnya akan disimpan dan apabila perlu hasilnya bisa dikonversi menjadi *file* PDF.

III. METODE

Koding dan Pengembangan

Tahap penulisan kode dan pengembangan, pada tahap ini *website text scanner* berbasis *microservices* mulai diimplementasikan, Proses pengkodean dimulai dengan pembuatan user service terlebih dahulu lalu dilanjutkan dengan note service, image preprocessing, Api gateway dan inisiasi *backend*, yang menghasilkan kerangka kerja atau struktur direktori seperti yang ditunjukkan pada Gambar 4



Gambar 4. Struktur Direktori *Backend*

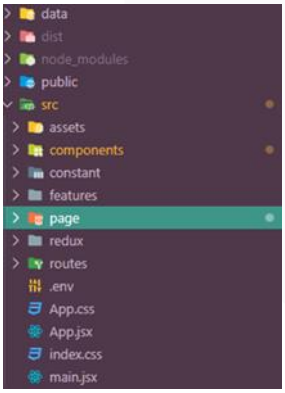
salah satu contoh kodingan yaitu pada *folder* "api-gatway", didalam *folder* tersebut terdapat *file* 'index.js' yang berfungsi sebagai titik akses tunggal ke beberapa layanan lainnya. Kode sumber dari dokumen 'index.js' dapat dilihat pada Gambar 5 berikut.

```
const express = require("express");
const cors = require("cors");
const { createProxyMiddleware } =
require("http-proxy-middleware");
const app = express();
const port = process.env.PORT;
app.use(cors());
app.use("/user", createProxyMiddleware({
target: "http://user-service:8081",
changeOrigin: true }));
```

```
app.use("/note", createProxyMiddleware({
target: "http://note-service:8082",
changeOrigin: true }));
app.use("/preprocess",
createProxyMiddleware({ target:
"http://image-preprocessing:8084",
changeOrigin: true }));
app.listen(port, () => {
  console.log(`API Gateway is running on
port ${port}`);
});
```

Pada source code Gambar 5 Kode ini mendefinisikan sebuah API gateway menggunakan Node.js dan Express.js. API Gateway ini berfungsi sebagai titik akses tunggal ke beberapa layanan lainnya. Menggunakan *middleware* proxy HTTP, permintaan ke "/user", "/note", dan "/preprocess" diteruskan ke layanan yang sesuai. Layanan ini berjalan pada port yang ditentukan dalam variabel lingkungan PORT. CORS diaktifkan untuk mengizinkan permintaan lintas asal.

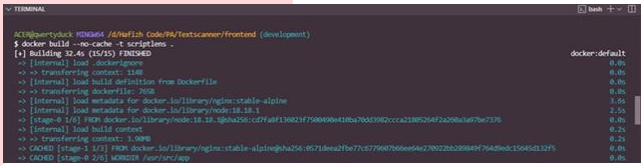
Setelah selesai membangun server API atau bagian *backend*, langkah berikutnya dalam eksekusi kode adalah mengembangkan bagian *frontend*. Anda dapat merujuk pada Gambar 6 berikut untuk melihat struktur *folder* atau kerangka kerja yang digunakan.



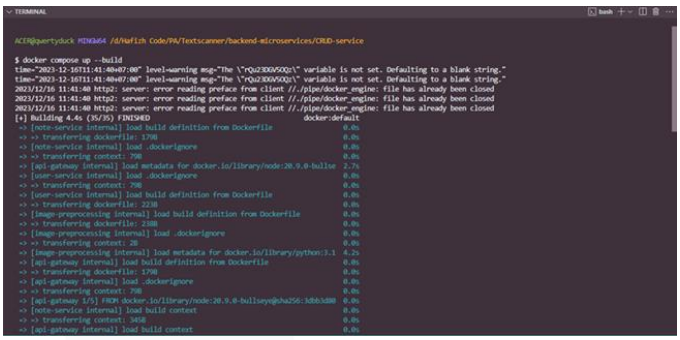
Gambar 6. Struktur Direktori *Frontend*

Pada Gambar 6, kerangka kerja sisi *frontend* terdapat beberapa *file* dan *folder* yang tersusun. Sama hal nya dengan sisi *backend*, pada sisi ini juga terdapat beberapa *file* dan *folder* utama dan kongfigurasi. Untuk mengkases API yang telah dibuat sebelumnya pada sisi *backend*, sisi *frontend* terdapat *folder* redux yang berfungsi menjalankan request ke endpoint pada sisi *backend*. Redux sendiri adalah sebuah perpustakaan JavaScript sumber terbuka yang digunakan untuk mengatur state dalam aplikasi web. Redux beroperasi dengan membuat sebuah store yang berisi semua state dalam aplikasi web. Saat ada perubahan state, Redux akan membuat sebuah aksi yang akan diproses oleh reducer sehingga store akan berubah sesuai dengan perubahan state.

Selanjutnya yang terakhir, men-deploy seluruh komponen *website* kedalam docker. Proses ini melibatkan pembuatan Docker*file* dan Docker Compose yang mendefinisikan lingkungan runtime aplikasi. Setelah Docker*file* dan Docker Compose dibuat, kita dapat menjalankan perintah 'docker composer up --build' untuk membuat *image* docker bagian *backend* dan perintah 'docker build --no-cache -t scriptlens .' Untuk membuat *image* docker bagian *frontend*, command ini dapat dilihat pada Gambar 7 dan Gambar 8 berikut.

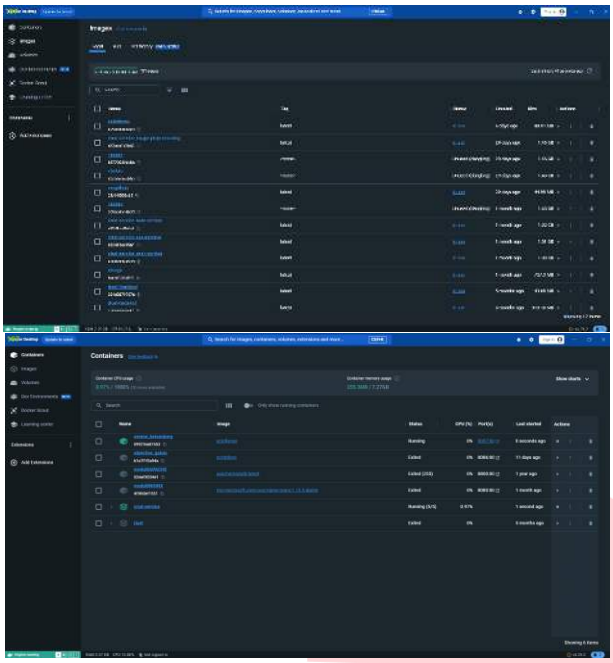


Gambar 7. Membuat Image Docker Untuk *Frontend*



Gambar 8. Membuat Image Docker Untuk *Frontend*

Dapat dilihat pada gambar 7 dan gambar 8. Sistem sedang membuat *image* docker untuk setiap *service* pada *backend* dan seluruh komponen serta dependensi pada *frontend*, dapat dilihat hasil dari pembuatan *image* docker pada Gambar 9.

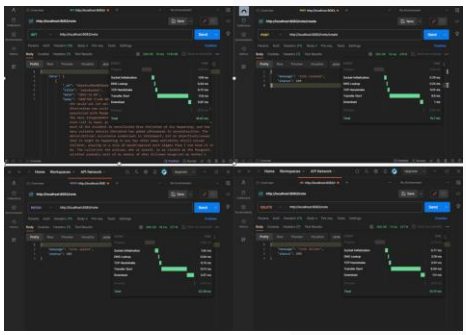


Gambar 9. Docker Image dan Docker Container

IV. HASIL DAN PEMBAHASAN

Pengujian dan Analisis respones times dari requestmethode (get, create, Update, dan delete)

Pengujian ini bertujuan utuk meneliti respon waktu setiap *methode* pada *backend* yang menggunakan arsitektur *microservices*. Pengujian ini menggunakan Postman, Pengujian setiap methode dapat dilihat pada Gambar 10 dan Table 1.



Gambar 10. Performa *Get, Create, Update, Delete*

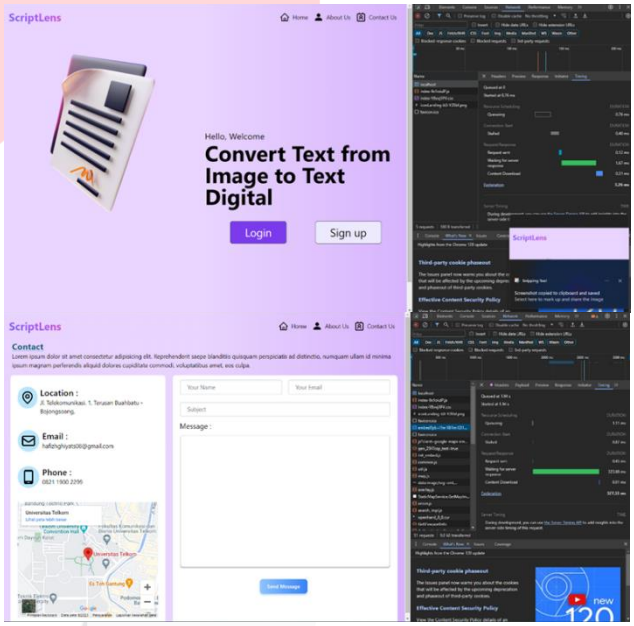
Table1. Hasil Pengujian *Respones Time*

Methode	Total *respones time*
Get	18,47 ms
Post	15,7 ms
Patch	23,8 ms
Delete	13,72 ms

Dapat dilihat pada Gambar 10 dan *table* 1merupakan catatan waktu performa CRUD data dengan API. pengujian performa ini dilakukan dengan menggunakan aplikasi Postman. Dengan performa *create* data adalah 15,7 ms, *read* data 18,47 ms, *update* data 23,8 ms dan *delete* data 13,72 ms. Berdasarkan kecepatan performa tersebut dapat disimpulkan bahwa kecepatan yang didapat dari implementasi arsitektur *microservice* ini terbilang sangat cepat, bahkan tidak sampai menghabiskan waktu 1 detik dibandingkan waktu pemuatan *web* yang tidak normal menurut Kinsta adalah diatas 4 detik[9]. Meski hanya sebagian dari proses pemuatan data *website*, pemuatan data ini tetap memiliki pengaruh signifikan terhadap durasi dan kecepatan pemuatan *website* secara keseluruhan.

Pengujian dan Analisis Performa *Website*

Untuk memahami kinerja *website* secara komprehensif, penulis melakukan pengecekan menggunakan *developer tools* dari *browser* pada tab performa dan *network*. Inspeksi ini dilakukan pada transaksi data CRUD pada *website scanner* teks dan SPA (*Single Page Aplication*) saat berpindah antara halaman, berdasarkan permintaan dan respons yang dikirim ke server. Hasil dari evaluasi performa tersebut dapat dilihat sebagai berikut:.



Gambar 11. Performa *website* pertama kali dimuat dan Perpindahan Halaman Dengan Basis SPA

Perbedaan dapat dilihat pada Gambar 10 menunjukkan saat pertama kali *website* dimuat dengan semua sumber daya yang menyusun *website* tersebut. Ini menghasilkan 5 permintaan ke *server* untuk mendapatkan sumber daya dan data yang diinginkan, dengan waktu pemuatan sebesar 3,26 ms. Namun, jika kita bandingkan gambar tersebut menunjukkan perpindahan dari halaman *Home* ke halaman *Contact*. Pada perpindahan halaman berbasis SPA tidak ada pemuatan ulang secara keseluruhan, melainkan hanya pemuatan data yang dibutuhkan saja. Hal ini terlihat pada Gambar 10. Ketika perpindahan halaman terjadi, program akan mengambil data yang dibutuhkan dari server, dalam hal ini data halaman yang menjadi tujuan perpindahan, yaitu halaman Contact. Sehingga, jumlah permintaan data ke server menjadi 46 permintaan, ini dapat dilihat penambahan dari 5 permintaan menjadi 51 permintaan ke server. Pada SPA, hanya 46 permintaan tersebut yang dimuat ulang, sedangkan 5 lainnya sudah dimuat saat pertama kali mengunjungi *web*. Meskipun banyaknya permintaan yang dimuat setelah perpindahan halaman, waktu yang dibutuhkan untuk

memuatnya adalah 327,33 ms. Hal ini masih sangat cepat mengingat pemuatan halaman yang terdiri dari 46 permintaan.

Pengujian Image Pre-Processing

Untuk menguji kinerja dari image pre-processing penulis melakukan pengujian terhadap gambar yang akan diinputkan yaitu satu gambar tulisan tangan dan gambar yang terdapat pada dokumen hardfile. Hasil pengujian dapat dilihat pada tabel 2.

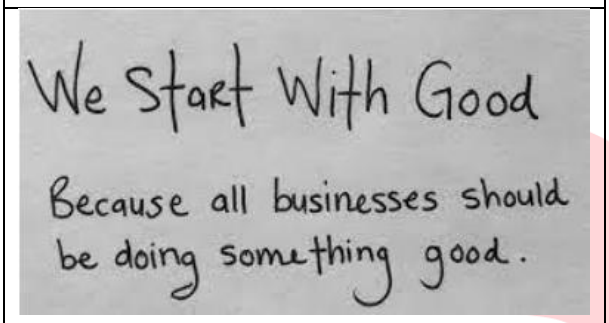
Sebelum dilakukan image pre-processing

sesudah dilakukan image pre-processing


Table 2. pengujian image pre-procecssing

Dari tabel 2 pengujian image pre-processing, proses pre-processing telah meningkatkan kualitas dan kejelasan teks dalam gambar. Teks menjadi lebih mudah dibaca dan jelas setelah dilakukan image pre-processing. Sebagai contoh, frase bahasa Inggris di bagian atas "We Start With Good Because all businesses should be doing something good." tampak lebih jelas dan mudah dibaca setelah pre-processing. Demikian juga dengan beberapa baris teks di bawahnya. Jadi, secara umum, image pre-processing telah berhasil meningkatkan legibilitas teks dalam gambar.

4 Pengujian dan Analisis OCR

Pada pengukuran tingkat akurasi OCR dan *execution time*, penulis melakukan pengujian dengan cara mencoba fitur OCR pada *website* dengan sampel 10 tulisan tangan dan 10 dokumen *hardfile*. Berikut rumus untuk mencari presentase *error*.

$$\text{Percent error } (\%) = \left|\frac{a-b}{a}\right| X\ 100\%$$

Keterangan : a: jumlah karakter ; b: jumlah karakter tidak *error*.

Table 3. Pengujian tingkat akurasi OCR dan execution time dokumen *hardfile*

Pengujian ke	OCR		Ukuran *file*	execution time	Selisih	Error Rate (%)	Akurasi (100%-Error rate)
	Jumlah karakter	Jumlah karakter tidak error					
1	2106	2095	100KB	5,8 s	11	0,5	99,5%
2	1844	1843	423KB	3,5 s	1	0,05	99,95%
3	1289	1284	73,9KB	4,5 s	5	0,3	99,7%
4	1590	1585	78KB	4,6 s	5	0,3	99,7%
5	1804	1800	939KB	8,6 s	4	0,2	99,8%
6	1829	1821	893KB	5,1 s	8	0,4	99,6%
7	2022	2009	562KB	6.4 s	13	0.6	99,2%
8	1858	1851	783KB	6,04s	7	0,3	99,7%
9	2376	2369	857KB	7,5 s	7	0,2	99,8%
10	1744	1743	884KB	5.7 ss	1	0,05	99,95%
Rata-rata					**6**	**0,26**	**99,69%**

Table 4. Pengujian tingkat akurasi OCR dan execution time dokumen tulisan tangan

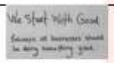
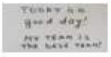
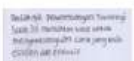
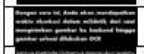
Pengujian	OCR		Ukuran *file*	execution time	Selisih	Error Rate (%)	Akurasi (100%-Error rate)
	Jumlah karakter	Jumlah karakter tidak error					
	19	19	17,6KB	0,6 s	0	0	100%
	73	71	56,4KB	0,6 s	2	2,7	97,3,%
	47	46	41KB	0,7 s	1	2,1	97,9%
	146	143	32,5KB	0,9 s	3	2,05	97,95%
	116	107	1,44MB	1,5 s	9	7,7	92,3%
	11	10	58,3KB	0,6 s	1	9,9	90,1%
	110	17	202KB	1,5 s	17	15,45	84,55%
	11	7	62,8KB	0,7 s	4	36,3	63,7%
	53	50	254KB	1,5 s	3	6	94%
	98	93	205KB	1,4 s	5	5,1	94,9%
Rata-rata					**4,5**	**8,73**	**91,27%**

Table 5. Pengujian gambar background hitam

Pengujian	OCR		Ukuran file	execution time	Selisih	Error Rate (%)	Akurasi (100%-Error rate)
	Jumlah karakter	Jumlah karakter tidak error					
	231	231	67KB	2,4 s	0	0	100%
	6	5	35KB	0,6 s	1	16,6	83,4%
	19	19	73,9KB	4,5 s	0	0	100%
	146	145	48KB	1,7 s	1	0,6	99,4%
	53	53	20KB	0,8s	0	0	100%
Rata-rata					0.4	3,44	96,56%

Berdasarkan hasil pengujian OCR, dapat kita ketahui bahwa tingkat akurasi pembacaan karakter yang ditunjukan pada tabel diatas terutama tabel 4.1 yang mana merupakan perbandingan antara hitungan jumlah karakter dan jumlah karakter yang tidak error dengan hitungan manual menghasilkan presentase yang sangat baik yaitu dengan rata-rata presentase 99,69% dan untuk kecepatan eksekusi gambar relatif cepat mengingat jumlah karakter terbanyak pada gambar yang diuji adalah 2376 karakter dengan kecepatan eksekusinaya adalah 7,6 detik.

Namun dapat dilihat pada tabel 4.2 untuk pembacaan karakter tulisan tangan walaupun rata-rata presentase menghasilkan 91,27% diperlukan lagi pemodelan OCR dan proses image preprocessing yang lebih canggih, dikarenakan ada beberapa pengujian yang mendapati pembacaan OCR yang akurasinya kurang baik. Selain itu, menurut Docsumo juga menyebutkan bahwa untuk kasus yang kompleks yang melibatkan teks tulisan tangan dengan konten yang sangat heterogen dan di luar kosakata, nilai CER (Character Error Rate) sebesar sekitar 20% (yaitu akurasi 80%) dapat dianggap memuaskan[9]

Berdasarkan tabel 4.5 pengujian OCR pada gambar dengan latar belakang hitam, dapat disimpulkan bahwa OCR memiliki akurasi yang cukup tinggi dalam mengenali teks pada gambar dengan latar belakang hitam. Sebagian besar hasil pengujian menunjukkan tingkat kesalahan yang rendah dan akurasi di atas 83%. Tabel tersebut menampilkan hasil pengujian pada beberapa gambar spesifik, seperti "RESUME" dan "SORRY WE'RE CLOSED", dengan akurasi pengenalan karakter yang bervariasi dari 83,4% hingga 100%. Rata-rata, tingkat kesalahan pengujian adalah 3,44% dan akurasi adalah 96,56%. Ini menunjukkan bahwa meskipun ada beberapa kasus di mana OCR tidak sempurna, secara umum, performanya cukup baik.

Melihat hasil tersebut, jelas bahwa ada ruang untuk peningkatan, terutama dalam hal pembacaan karakter tulisan tangan. Untuk mencapai ini, beberapa strategi dapat diadopsi. Misalnya, penggunaan teknik pemrosesan gambar yang lebih canggih dapat membantu dalam meningkatkan kualitas gambar sebelum diteruskan ke mesin OCR.

V. KESIMPULAN

Berdasarkan analisis dari hasil pengujian *respones times* dari *request methode (get, create, update, dan delete)* arsitektur *microservices* berbasis docker, kecepatan yang didapat ini terbilang sangat cepat bahkan tidak sampai menghabiskan waktu 1 detik meski pengujiannya hanya pada bagian *backend* dengan menggunakan postman. Performa *website* secara menyeluruh yang telah dilakukan pada *website* scanner teks dan SPA saat berpindah antara halaman. waktu yang dibutuhkan untuk memuat semua itu adalah 327,33 ms. Hal ini masih terbilang sangat cepat mengingat pemuatan halaman yang cukup banyak setelah perpindahan halaman. analisis dari hasil pengujian OCR dapat disimpulkan bahwa teknologi OCR telah menunjukkan kinerja yang sangat baik dalam pengujian, dengan tingkat akurasi rata-rata 99,69% dan kecepatan eksekusi yang cepat. Namun, ada tantangan dalam membaca karakter tulisan tangan, di mana tingkat akurasi rata-rata adalah 91,27% dan itupun terdapat salah satu pengujian yang mendapati presentase akurasi dibawah 80%. Meskipun demikian, tingkat akurasi rata – rata ini masih dianggap memuaskan untuk kasus yang kompleks.

Pengembangan lebih lanjut guna meningkatkan performa *website* dan OCR tersebut dapat menggunakan arsitektur pengembangan *website* yang lebih baik seperti micro*frontend* dan Untuk meningkatkan akurasi lebih lanjut, peningkatan pada image preprocessing diperlukan, serta penggunaan teknik pemrosesan gambar yang lebih canggih. Kesimpulannya, meskipun teknologi OCR telah menunjukkan hasil yang mengesankan, masih ada ruang untuk peningkatan, terutama dalam hal pembacaan karakter tulisan tangan

REFERENSI

[1] F. O. Rahmalisty, “Perancangan Language Translator Image To Text Menggunakan Metode *Optical Character Recognition* Berbasis Pengolahan Citra Language Translator Image To Text Design Using Optical Character Tecognition Method Based On Image Processing.”2023

[2] H. Nindya Murwato, S. Aulia, and A. Novianti, “PERANCANGAN TRANSLATOR IMAGE TO TEXT DENGAN MENGGUNAKAN METODE *OPTICAL CHARACTER RECOGNITION* BERBASIS MATLAB IMAGE TO TEXT TRANSLATOR DESIGN USING MATLAB BASED *OPTICAL CHARACTER RECOGNITION* METHOD.” 2020

[3] H. D. Aldi , “Pemanfaatan Teknologi *Optical Character Recognition* pada Pembacaan Kartu Tanda Penduduk Artikel Ilmiah,” 2019.

[4] J. Memon, M. Sami, R. A. Khan, and M. Uddin, “Handwritten *Optical Character Recognition* (OCR): A Comprehensive Systematic Literature Review (SLR),” *IEEE Access*, vol. 8. Institute of Electrical and Electronics Engineers Inc., pp. 142642–142668, 2020. doi: 10.1109/ACCESS.2020.3012542.

[5] Setiawan A, “Implementasi *Optical Character Recognition* (OCR) pada Mesin Penerjemah Bahasa Indonesia ke Bahasa Inggris,” *Jurnal Sistem dan Teknologi Informasi (JUSTIN)*, vol. 5, pp. 135–141, 2017.

[6] A. N. Rahmawati, S. A. Wibowo, and U. Sunarya, “ANALISIS SISTEM *OPTICAL CHARACTER RECOGNITION* (OCR) PADA DOKUMEN DIGITAL MENGGUNAKAN METODE

TESSERACT PERFOMANCE ANALYSIS OF *OPTICAL CHARACTER RECOGNITION* (OCR) SYSTEM ON DIGITAL DOCUMENTS USING TESSERACT METHOD." 2021

[7] J. Stastny, J. Šastný, and M. Minaík, "A Brief Introduction to Image Pre-Processing for Object Recognition," 2007. [Online]. Available: http://www.fme.vutbr.cz/

[8] S. Supriadi, "Simple Handwriting Calculator Application Using *Optical Character Recognition* (OCR)," *Buletin Ilmiah Sarjana Teknik Elektro*, vol. 2, no. 3, p. 163, Jan. 2021, doi: 10.12928/biste.v2i3.3348.

[9] Nasution, "IMPLEMENTASI MONGO DB, EXPRESS JS, REACT JS DAN NODE JS (MERN) PADA PENGEMBANGAN APLIKASI FORMULIR, KUIS, DAN SURVEI ONLINE" 2022.

[10] F. Rifandi, Tri Viqi Adriansyah, and Rina Kurniawati, "*Website* Gallery Development Using Tailwind CSS Framework," *Jurnal E-Komtek (Elektro-Komputer-Teknik)*, vol. 6, no. 2, pp. 205–214, Dec. 2022, doi: 10.37339/e-komtek.v6i2.937.

[11] M. Fihri, R. M. Negara, and D. D. Sanjoyo, "IMPLEMENTASI & ANALISIS PERFORMANSI LAYANAN WEB PADA PLATFORM BERBASIS DOCKER IMPLEMENTATION & ANALYSIS OF WEB SERVICE PERFORMANCE BASED ON DOCKER PLATFORM."

[12] T. A. Diajukan, M. Salah, S. Persyaratan, and M. Derajat, "IMPLEMENTASI ARSITEKTUR MICROSERVICE PADA APLIKASI WEB PENGAJARAN AGAMA ISLAM HOME PESANTREN," 2020.