Analisis Penerapan *Code Splitting Library React* pada Aplikasi Penjualan Mebel Berbasis *Website*

http://dx.doi.org/10.28932/jutisi.v7i2.3493

Riwayat Artikel
Received: 04 Maret 2021 | *Final Revision*: 29 Juni 2021 | *Accepted*: 06 Juli 2021

Daniel Tanudjaja[✉ #1], Radius Tanone[*2]

[#]*Jurusan Teknik Informatika, Universitas Kristen Satya Wacana*
Jl. Diponegoro no. 52-60, Kota Salatiga
[1]daniel.tndj@gmail.com

[*]*Jurusan Teknik Informatika, Universitas Kristen Satya Wacana*
Jl. Diponegoro no. 52-60, Kota Salatiga
[2]radius.tanone@staff.uksw.edu

Abstract — **Mebel Murah is a shop that sells furniture products located in Semarang, Central Java, Indonesia. Due to Covid-19, the sales revenue have decreased due to limited access for buyers. Therefore, an innovation is needed to overcome this, namely by creating a website that can support especially the marketing and sales of furniture products to the public. The aims of this research is to focus on testing the rendering speed of components through the React Profiler to analyze the comparison of component rendering times on the client side of the code breaking features of the React library. The result of this research is to produce a website by utilizing the code splitting feature to display information content related to products sold by Mebel Murah stores.**

Keywords— **Code Splitting; Furniture; Online Shopping; React; Reusable Components.**

I. PENDAHULUAN

Toko Mebel Murah adalah toko yang menjual produk mebel yang berlokasi di Semarang, Jawa Tengah. Berdasarkan hasil wawancara dengan salah satu pelaku Usaha Mikro Kecil Menengah (UMKM) yang bidang usahanya terdampak *Covid-19* yaitu pemilik Toko Mebel Murah menyatakan bahwa omzet penjualan dan jumlah pembeli produk mebel mengalami penurunan sejak adanya pandemi *Covid-19*. Penurunan itu disebabkan oleh keterbatasan akses pembeli ke Toko Mebel Murah dikarenakan adanya *Covid-19* dan area pemasaran hanya di Kota Semarang dan sekitarnya. Oleh karena itu, Toko Mebel Murah ingin menerapkan penggunaan aplikasi *website* dalam kegiatan jual beli mebel.

Proses transaksi di Toko Mebel Murah biasanya dilakukan dengan pembeli yang datang langsung ke toko untuk memilih produk yang tersedia di toko. Dikarenakan pandemi Covid-19, tingkat kunjungan pembeli ke toko semakin menurun. Untuk mengatasi masalah tersebut diperlukan aplikasi yang dapat mempermudah transaksi antara penjual dan pembeli, membantu penjual dalam memberikan informasi terkait produk mebel yang tersedia, dan memberikan informasi terkait status pemesanan barang yang dibeli oleh.

Berdasarkan permasalahan yang dialami pembeli ataupun penjual maka dibangun sistem berupa *website* penjualan produk mebel yang dapat mengakomodasi segala kebutuhan *user*, seperti: memilih dan melakukan pembelian secara *online* terhadap produk mebel dan fitur notifikasi *email* bagi *user* dapat mengetahui status pesanan. Untuk membangun *website* yang memiliki fitur yang kompleks dibutuhkan teknologi yang mendukung proses pengembangan *website*. Oleh karena itu, dalam pengembangan sistem digunakan teknologi *MongoDB*, *Express*, *React*, dan *Node.js* untuk pembuatan *website* penjualan Toko Mebel Murah.

Pengembangan *user interface* dari *website* sering dilakukan dengan menulis kode berulang-ulang dengan pola yang sama. Dengan hal itu membuat kerugian bagi proses pengembangan, yaitu waktu yang diperlukan menjadi lebih lama. Berdasarkan sebuah penelitian disimpulkan bahwa rata-rata 48% dari ukuran *file* dari kode aplikasi berasal dari *user interface*. Semakin besar ukuran *file* yang perlu diproses oleh *browser* maka akan semakin membebani waktu *rendering* di sisi klien [1].

Dalam proses pengembangan *user interface website*, dimanfaatkan *library React*. *React* merupakan *library* dari JavaScript yang digunakan untuk membangun *user interface* yang sudah mendukung *reusable component*. *Reusable component* dapat mengurangi penulisan kode yang repetitif dengan pola yang sama. Sehingga, pembuatan aplikasi menjadi lebih cepat dan terstruktur [2].

Rumusan masalah dari penelitian ini yaitu, bagaimana efektivitas penggunaan *code splitting* terhadap *rendering* komponen di sisi klien. Adapun tujuan dari penelitian ini adalah mengetahui efektivitas kecepatan *rendering* komponen dengan menggunakan fitur *code splitting* terhadap kecepatan *rendering* tanpa menggunakan fitur *code splitting*. Manfaat dari penelitian ini adalah diharapkan menghasilkan sistem yang memiliki performa yang baik sehingga dapat membantu proses penjualan produk mebel kepada pembeli. Batasan masalah dari penelitian ini adalah pengembangan *front-end* dari *website* menggunakan *library React* dan pengujian kecepatan *rendering* komponen menggunakan *React profiler* yang terdapat dalam *React Developer Tools*.

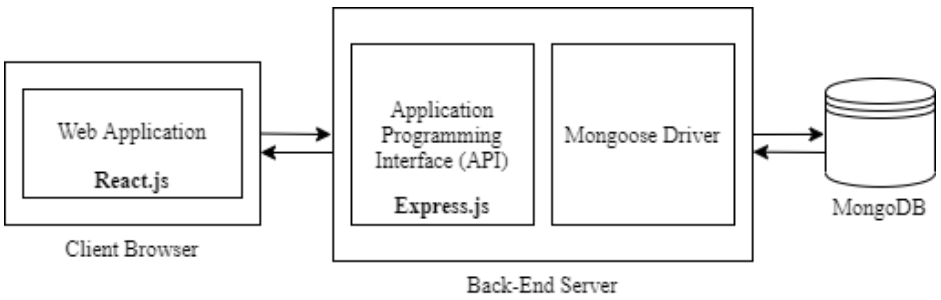
Pada penelitian terdahulu pernah mengangkat topik serupa dengan penelitian ini sehingga dapat dijadikan acuan dalam pelaksanaan penelitian ini. Penelitian pertama dengan judul "*End-to-end E-commerce Web Application, a Modern Approach using MongoDB, Express, React, and Node.js* (MERN) *stack*" yang menerapkan penggunaan dari setiap teknologi yang digunakan yaitu *MongoDB, Express, React,* dan *Node.js.* Pada penelitian ini, *website* yang dikembangkan memiliki fitur yang kompleks sehingga konsep *reusable component* digunakan untuk mengurangi waktu pengembangan *front-end*, desain aplikasi lebih konsisten, dan memudahkan *maintenance* kode ke depannya. Hasil dari penelitian berupa *website* penjualan buku berbasis *website* yang bertujuan memperluas target pasar dari toko buku dan mempermudah proses transaksi pembelian buku [3].

Penelitian selanjutnya berjudul "*Optimizing the Critical Rendering Path for Decreased Website Loading Time*" membahas tentang bagaimana mengurangi waktu *rendering* sebuah *website* di sisi klien. Pada penelitian ini ditemukan bahwa faktor-faktor yang mempengaruhi kecepatan *rendering* yaitu kecepatan koneksi internet, ukuran *file* (HTML, CSS, dan JS), urutan dalam *rendering* konten, dan *client browser*. Peneliti menggunakan beberapa strategi yang digunakan untuk mengurangi *loading time*, antara lain: mengurangi jumlah request dari *client* ke *server*, *minify* dan *compress resources*, mengurangi ukuran *file* gambar yang diupload oleh klien, dan *caching* data. Tujuan dari penelitian ini adalah menganalisis penerapan berbagai strategi untuk mengurangi *loading time* sebuah *website*. Hasil dari penelitian yaitu strategi-strategi yang digunakan terbukti mengurangi waktu *loading* dari sebuah *website* [4].

Menurut penelitian yang berjudul "*Present Day Web-Development using React.js*" didapatkan simpulan bahwa *React* adalah *library* yang memiliki performa yang cepat dalam *rendering* konten ke sisi klien. Hal tersebut disebabkan *React* memanfaatkan *Virtual Document Object Model* (VDOM) untuk membandingkan perubahan data yang terjadi pada *Document Object Model* (DOM) *tree* dengan DOM pada *browser*. *React* hanya akan melakukan *rendering* ulang terhadap komponen yang mengalami perubahan [5].

Berdasarkan penelitian-penelitian yang telah dipaparkan di atas, penelitian ini berfokus menguji strategi *code splitting* dengan membandingkan waktu *rendering* komponen. Pengujian dilakukan dengan menggunakan *React Profiler*. *React Profiler* merupakan *tools* yang berfungsi mengidentifikasi performa *website* dalam menampilkan konten.
Tingkat *rendering website* yang lambat dapat mempengaruhi *user experience* dalam mengunjungi *website*. Hal itu didukung oleh penelitian yang menunjukkan bahwa 53% pengguna akan meninggalkan *website* yang memiliki *rendering time* di atas 3 detik [6]. Sehingga, berdasarkan penelitian tersebut dapat disimpulkan bahwa batas waktu yang wajar dalam *rendering* konten *website* adalah 3 detik.

Pengembangan aplikasi menggunakan arsitektur *MongoDB, Express, React,* dan *Node.js* (MERN). *MongoDB* akan digunakan sebagai *database*, *Express* digunakan sebagai *back-end framework*, *React* digunakan sebagai *front-end library*, dan *Node.js* berperan sebagai *environment* untuk menjalankan *JavaScript* di sisi *server*.



Gambar 1**.** Arsitektur Aplikasi

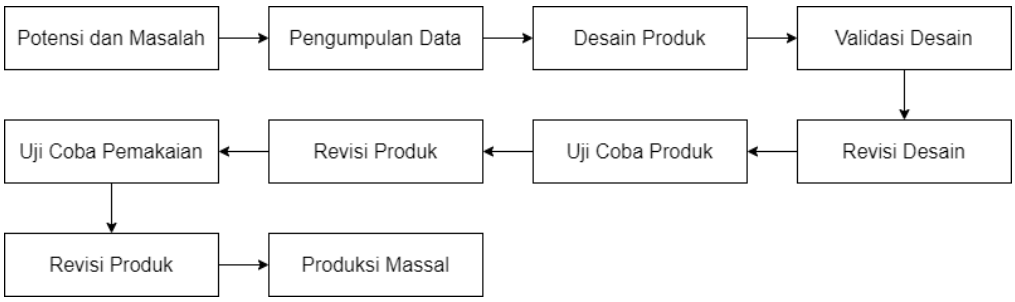
Pada gambar 1, menggambarkan arsitektur aplikasi menggunakan MERN. Langkah pertama adalah membuat API dengan menggunakan *Express*, dilanjutkan konfigurasi koneksi ke *MongoDB* menggunakan Mongoose *driver* dan membuat skema dan model yang diperlukan. Ketika *back-end server* sudah selesai dibangun, *client* melakukan HTTP *request* ke *server* melalui *website* yang dibangun menggunakan *library React* dan *server* akan mengirimkan file yang dibutuhkan untuk di proses oleh *browser* [3].

React merupakan *library JavaScript* yang bersifat *open-source* yang digunakan untuk membangun *interfaces* yang menerapkan *Single Page Applications* (SPA) [7]. Keuntungan penggunaan *React* adalah penerapan *reusable component* sehingga kode yang dibangun akan menjadi lebih terstruktur, mudah dikelola dan dapat mempercepat proses pengembangan. Pembuatan komponen *React* dapat dilakukan dengan 2 cara, yaitu dengan menggunakan *functions* atau *class*. Secara umum, *developer* disarankan untuk menggunakan *functions* dikarenakan penulisan kode menjadi lebih efisien [8].

React mendukung *Virtual Document Object Model* (*VDOM*) [9]. VDOM adalah suatu konsep untuk merepresentasikan struktur dari suatu dokumen HTML dan bagaimana dokumen HTML dapat diakses serta dimodifikasi [10]. VDOM menyimpan representasi dari *user interface* di dalam memori dan disinkronkan dengan DOM pada *browser* oleh *library* seperti *React*DOM. Jadi, ketika ada *update* data pada *state* yang akan ditampilkan pada komponen, VDOM akan mencari perbedaan struktur pada DOM di *browser*. Proses ini disebut dengan *DOM diffing* atau *reconciliation of state and view* [5].
Dengan menggunakan *React*, semakin berkembangnya aplikasi maka ukuran *file* yang dihasilkan akan semakin besar. Hal itu akan membuat kecepatan *rendering* di sisi klien akan lambat. Untuk menghindari masalah ukuran *file* yang besar, *React* memiliki fitur *code splitting* yang bekerja dengan memecah *file* yang akan di proses untuk ditampilkan di *browser*. *Code splitting* akan melakukan *lazy-load* yang memuat komponen *website* yang sedang dibutuhkan oleh pengguna saja [11]. Dengan adanya *code splitting* diharapkan menjadi salah satu strategi dalam meningkatkan performa *website*.

II. METODE PENELITIAN

Dalam penelitian ini menggunakan metode penelitian *Research and Development* (RnD). Metode penelitian RnD merupakan tahapan dalam menghasilkan sebuah sistem yang berdasarkan kebutuhan pengguna [12]. Langkah-langkah penelitian dapat dilihat pada Gambar 2.

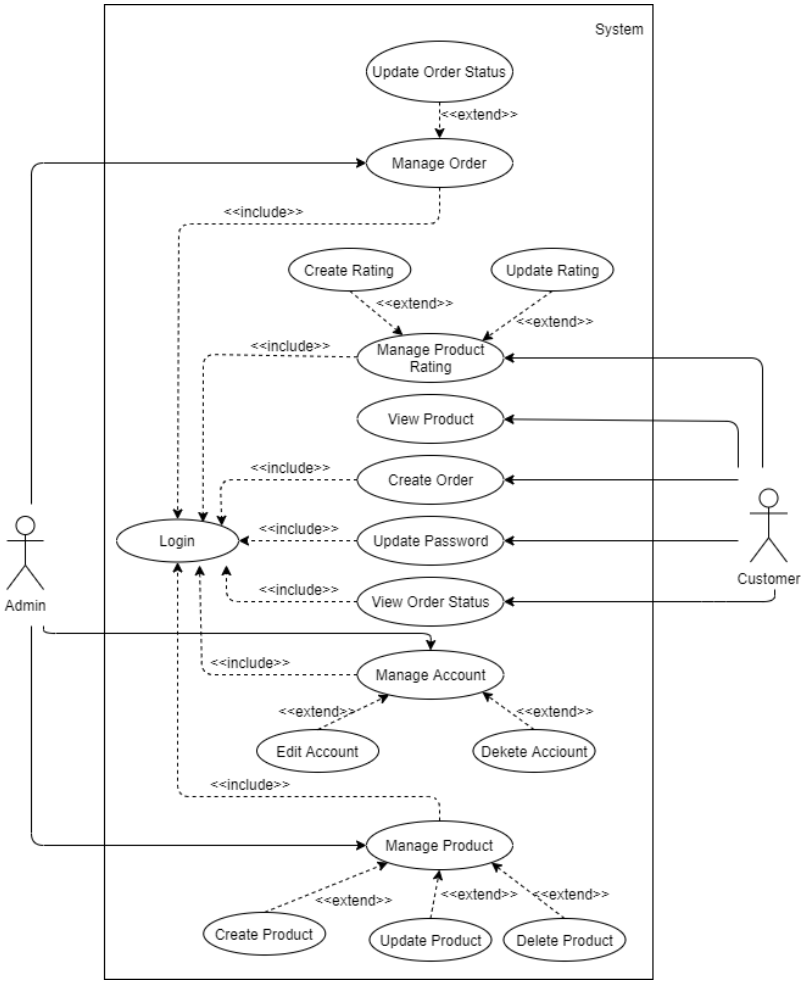


Gambar 2. Tahapan Penelitian

Pada penelitian ini tahapan penelitian dilakukan sampai pada tahap revisi produk dan tidak dilakukan sampai pada tahap produksi masal. Tahapan penelitian pada Gambar 2 dapat dijelaskan sebagai berikut:

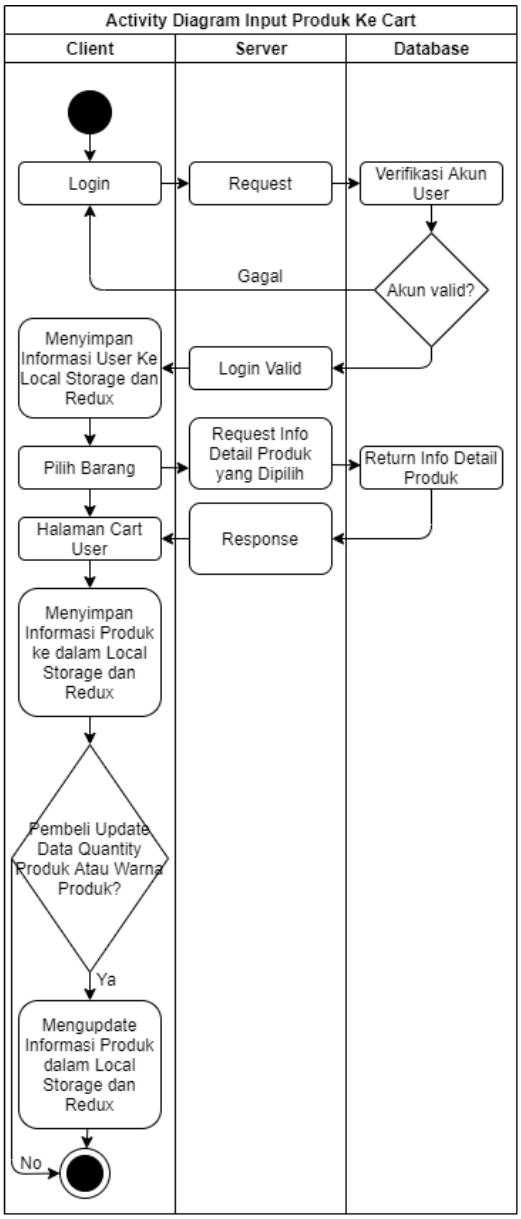
1. Tahap pertama adalah potensi dan masalah. Dalam tahap ini dilakukan identifikasi permasalahan berdasarkan hasil wawancara dengan pemilik Toko Mebel Murah. Permasalahan yang ditemukan yaitu menurunnya omzet toko sehingga diperlukan sistem yang dapat mempermudah proses transaksi.
2. Tahap kedua adalah pengumpulan data. Pada tahapan ini dilakukan pengumpulan data yang akan digunakan untuk pembuatan sistem. Data yang diperlukan nama dan jumlah karyawan, data barang, harga barang, jumlah barang, dan proses bisnis. Data tersebut diperoleh dengan metode wawancara dengan pemilik Toko Mebel Murah.
3. Tahap ketiga adalah desain produk. Pada langkah ini, peneliti membuat desain produk dalam bentuk diagram UML.
4. Tahap keempat adalah validasi desain. Pada tahapan ini dilakukan penilaian terhadap desain produk dengan kebutuhan.
5. Tahap kelima adalah revisi desain. Pada tahap ini, peneliti melakukan revisi atas desain produk yang telah dibuat sebelumnya sesuai dengan kebutuhan.
6. Tahap keenam adalah uji coba produk. Pada tahapan ini dilakukan pengujian oleh *developer* untuk menemukan *bug* yang terdapat dalam sistem untuk diperbaiki.

7. Tahap ketujuh adalah revisi produk. Dalam tahap ini dilakukan perbaikan pada sistem. Hasil akhir dari revisi produk merupakan produk yang diuji coba.
8. Tahap kedelapan adalah uji coba pemakaian. Pada tahapan ini, *user* melakukan pengujian *beta* terhadap fungsionalitas dari sistem.
9. Tahap kesembilan adalah revisi produk. Jika dalam tahap uji coba pemakaian ditemukan *bug* pada fungsionalitas website oleh pengguna, maka sistem akan direvisi kembali oleh *developer*.
10. Tahap kesepuluh adalah produksi masal. Pada tahapan ini sistem yang dihasilkan sudah bisa digunakan oleh pengguna jika sistem yang dibuat lulus dari tahap pengujian oleh *developer* dan *user*.



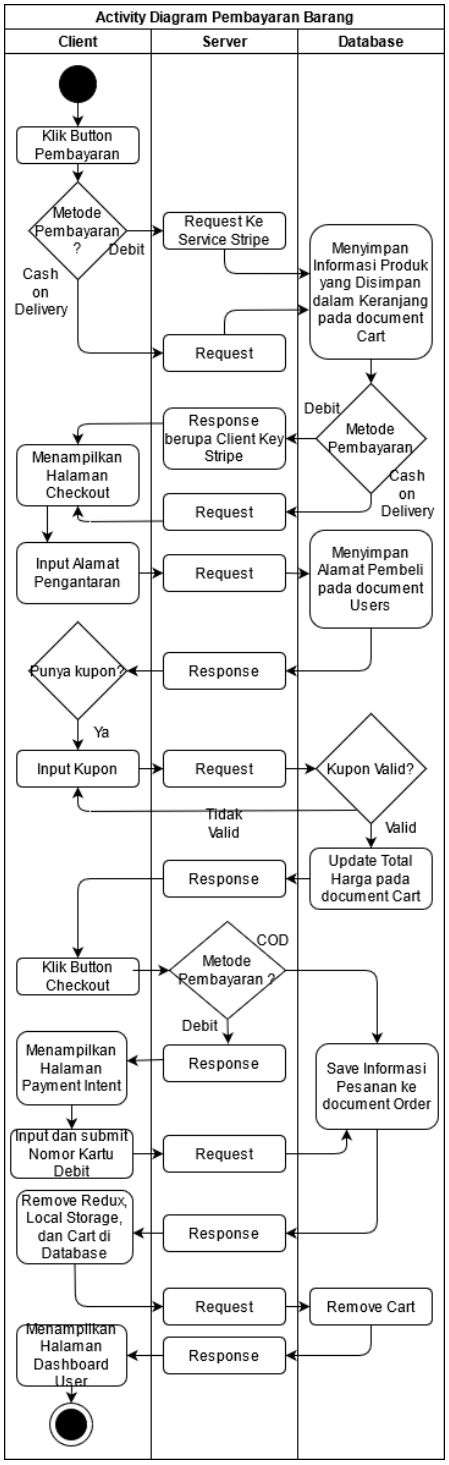
Gambar 3. *Use Case Diagram*

Pada Gambar 3 merupakan desain *use case* sistem Toko Mebel Murah. Terdapat 2 aktor dalam sistem ini yaitu *admin* dan *customer*. Kedua aktor memiliki hak akses yang berbeda-beda untuk setiap fungsi yang ada dalam sistem. *Admin* dapat melakukan menginput atau *update* produk yang akan dijual, mengelola akun pengguna, dan memperbaharui status pemesanan. *Customer* dapat melihat produk yang dijual, memberikan *rating* produk, *update password*, melakukan pemesanan produk, dan melihat status pesanan



Gambar 4. *Activity Diagram Input* Produk ke dalam Keranjang

Pada Gambar 4 merupakan *activity diagram* yang menggambarkan interaksi antara *client*, *server*, dan *database* jika *customer* memilih barang. *Customer login* terlebih dahulu, kemudian *database* akan validasi akun, jika valid maka informasi *login user* akan disimpan dalam *Local Storage* dan Redux. Kemudian, jika *customer* memilih barang maka akan masuk ke halaman *cart user* dan menampilkan informasi detail produk yang dibeli. Untuk produk yang dimasukkan ke dalam *cart* secara otomatis juga akan masuk ke dalam *Local Storage* dan *Redux*. *Redux* digunakan untuk menyimpan dan mengelola data antar komponen. Pada halaman *cart*, *customer* dapat melakukan *update* pada setiap produk pada bagian *quantity* dan warna.



Gambar 5. *Activity Diagram Pembayaran*

Pada Gambar 5 menunjukkan *activity diagram* yang menjelaskan interaksi antara *client*, *server*, dan *database*. Pada halaman *cart, user* dapat memilih metode pembayaran yang diinginkan yaitu metode debit atau *cash on delivery*. Jika *customer* memilih metode debit maka akan dilakukan proses *request* ke *service Stripe* untuk memperoleh *client key*, setelah itu *customer* akan masuk ke halaman *checkout*, *customer* dapat menginput alamat dan kupon. Jika kupon yang diinput *valid*, maka akan langsung *update* total harga yang harus dibayarkan disesuaikan dengan besar diskon yang terdapat dalam kupon. Setelah itu, *customer* dapat melanjutkan proses *checkout* dan akan masuk ke halaman *payment intent* yang terdapat *form input* untuk nomor kartu debit. Selanjutnya, *customer* melakukan *submit form* dan akan menyimpan informasi *order* ke

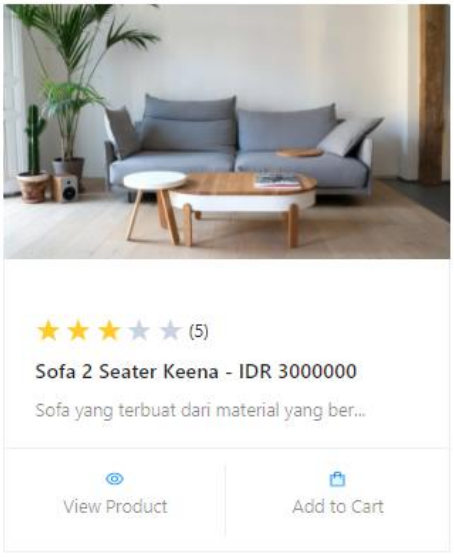
document Order pada *MongoDB*. Jika *customer* memilih metode *cash on delivery*, maka informasi order akan langsung disimpan ke dalam *document Order*.

III. HASIL DAN PEMBAHASAN

Hasil dan pembahasan meliputi *website* penjualan produk mebel dengan menerapkan teknologi *MongoDB, Express, React,* dan *Node. js* (MERN) yang dapat digunakan sesuai dengan analisis kebutuhan yang telah dilakukan. *Website* ini berfungsi membantu pemilik toko dalam memasarkan produknya, memudahkan pembeli dalam melakukan pemesanan, dan mendapat informasi terkait status pesanan pembeli.

Pada saat pertama kali mengakses *website*, *customer* akan berada pada halaman *home* yang berisi daftar produk-produk mebel yang dijual. Pada bagian menu navigasi, *customer* dapat melakukan *register* dengan memasukkan *email* dan *password* dan bagi *customer* yang sudah memiliki akun dapat *login* menggunakan *email.*

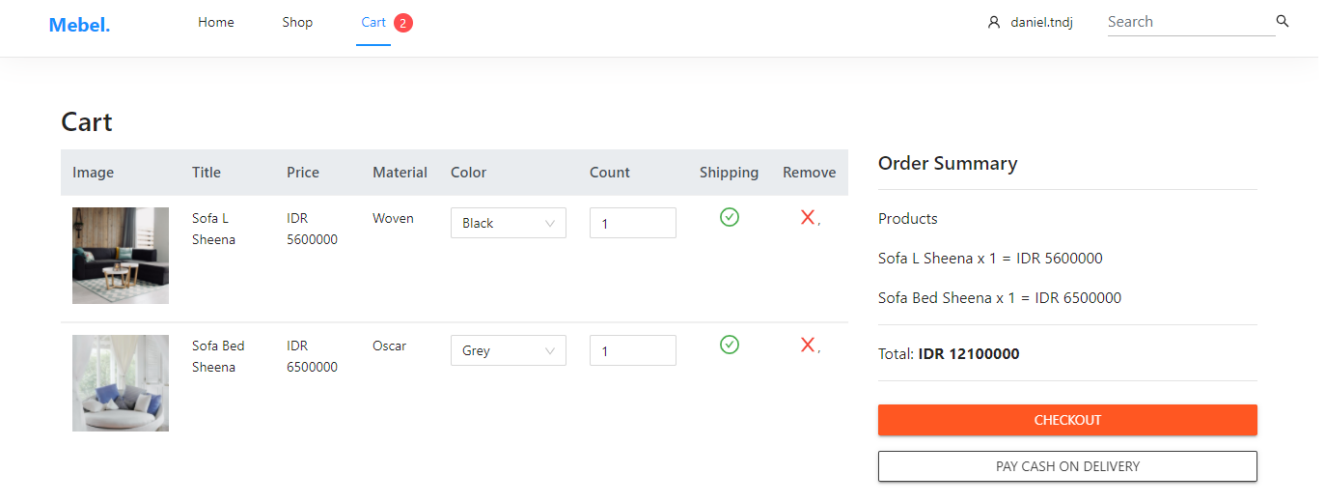
Setelah *login*, *customer* akan masuk ke halaman *dashboard* untuk *customer*. Pada *dashboard customer* terdapat informasi *history* pesanan yang dilakukan oleh *customer*, halaman untuk penggantian *password customer,* dan halaman untuk menampilkan *wishlist* produk. Jika *login* sebagai *admin*, maka akan masuk ke halaman *dashboard* untuk *admin.* Pada *dashboard admin* terdapat halaman untuk menampilkan informasi pesanan yang masuk, halaman untuk menampilkan daftar produk yang dijual, halaman untuk mengelola produk, halaman untuk mengelola kategori dan sub kategori, dan halaman untuk menambah dan menghapus kupon.



Gambar 6. Komponen *Product Card*

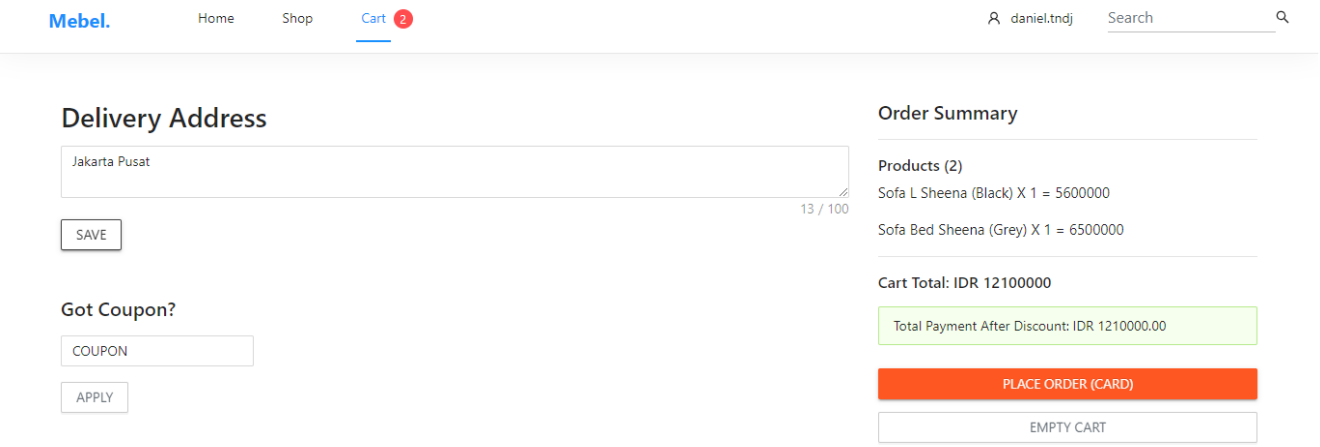
Pada Gambar 6 adalah komponen *product card*. Komponen ini berisi informasi produk yang terdiri dari gambar, penilaian, nama, harga, deskripsi, *button* untuk melihat detail produk, dan *button* untuk menambahkan produk ke dalam keranjang. Pembeli dapat membeli produk mebel dengan melakukan klik pada *button Add to Cart*. Setelah itu, pembeli akan diarahkan ke halaman *Cart*.

Dalam pengembangan *front-end*, diterapkan konsep *reusable component*. Penerapan tersebut dapat diketahui dalam komponen *product card* pada Gambar 6 yang diterapkan pada halaman *home* dan halaman detail produk. Keuntungan dari penerapan *reusable component* yaitu kode yang ditulis menjadi lebih singkat sehingga mempercepat pengembangan dikarenakan komponen dapat menerima *props* atau data dari *parent component* sehingga produk yang ditampilkan dapat bersifat dinamis.



Gambar 7. Halaman *Cart*

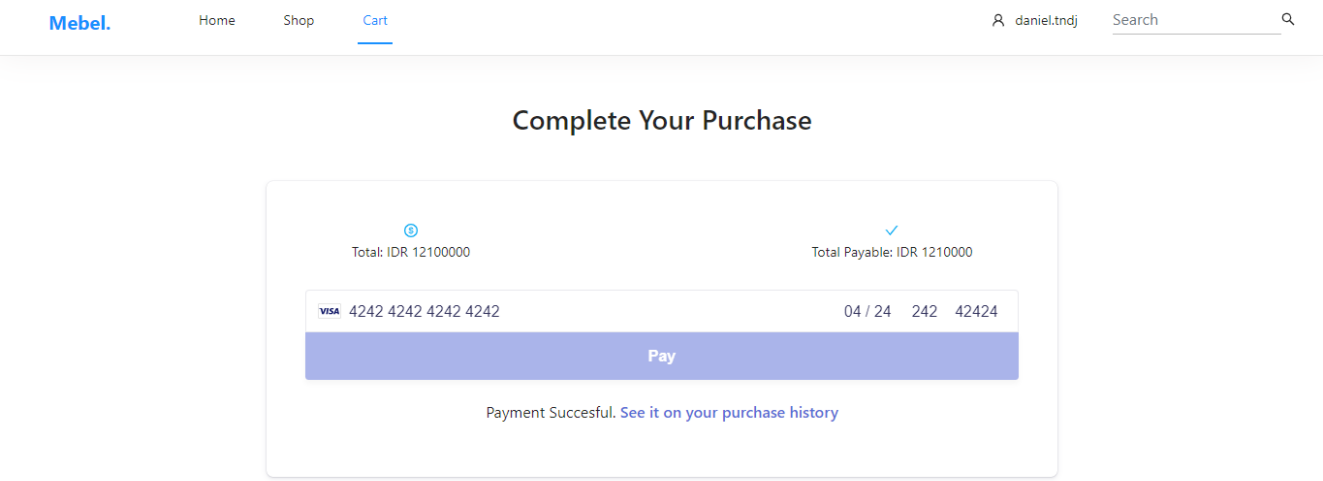
Pada Gambar 7 merupakan halaman *Cart*, terdapat tabel yang menjelaskan detail produk yang dibeli beserta harga total dari produk. Pembeli dapat memilih metode pembayaran yang tersedia yaitu metode debit atau *cash on delivery*. Jika pembeli memlih metode debit maka pembeli harus menginput nomor kartu. Setelah memilih metode pembayaran, pembeli akan masuk ke halaman *checkout*.



Gambar 8. Halaman *Checkout*

Pada Gambar 8, pembeli diwajibkan memasukkan alamat tujuan pengiriman dan pembeli dapat menginput kode kupon jika memilikinya. Setelah menginput, pembeli dapat melakukan proses pembayaran dengan melakukan klik pada *button Place Order*.

p-ISSN : 2443-2210
e-ISSN : 2443-2229



Gambar 9. Halaman *Input* Nomor Kartu Debit

Setelah itu pada Gambar 9, pembeli menginput nomor kartu untuk melakukan pembayaran dan pembeli akan diarahkan ke halaman *dashboard customer*.

Setelah melakukan transaksi, *admin* akan melakukan *update* status pesanan pada halaman *dashboard admin*. Status pesanan terdiri diri, antara lain:

1. *Not Processed*
   Jika pembayaran dilakukan dengan metode debit maka status pesanan akan otomatis menjadi *Not Processed*.
2. *Cash on Delivery*
   Jika pembayaran dilakukan dengan metode debit maka status pesanan akan otomatis menjadi *Cash on Delivery*.
3. *Processing*
   Jika barang tersedia dan alamat pengantaran berada dalam jangkauan maka status pesan akan diubah menjadi *Processing*.
4. *Dispatched*
   Jika barang dalam proses pengiriman menuju alamat pembeli maka status pesan akan diubah menjadi *Dispatched*.
5. *Cancelled*
   Jika barang yang dipesan tidak tersedia atau alamat pengantaran di luar jangkauan maka status pesanan akan diubah menjadi *Cancelled*.
6. *Completed*
   Jika barang sudah diterima pembeli maka status pesanan akan diubah menjadi *Completed*.

Untuk setiap pergantian status pesanan maka secara otomatis akan mengirim notifikasi ke *email* pembeli yang terdaftar, sehingga pembeli dapat mengetahui status pesanan.

```
1. import Home from "./pages/Home";
2. import Login from "./pages/auth/Login";
3. import Register from
4. "./pages/auth/Register";
5. import Header from
6. "./components/nav/Header";
```

Kode Program 1. *Import* Komponen Tanpa Menggunakan *Code Splitting*

Pada Kode Program 1 terdapat kode program yang berfungsi melakukan *import* komponen tanpa penerapan *code splitting*. *Import* merupakan salah satu fitur dari ES2015 *Module JavaScript* yang digunakan untuk *import function*, objek, atau variabel yang sudah di *export* pada modul lain.

```
1. const Home = lazy(() =>
2. import("./pages/Home"));
3. const Login = lazy(() =>
4. import("./pages/auth/Login"));
5. const Register = lazy(() =>
6. import("./pages/auth/Register"));
7. const Header = lazy(() =>
8. import("./components/nav/Header"));
9. const SideDrawer = lazy(() =>
10. import("./components/drawer/SideDrawer"))
```

Kode Program 2. *Import* Komponen dengan Menggunakan *Code Splitting*

Pada Kode Program 2 terdapat kode program yang berfungsi melakukan *import* komponen dengan menggunakan *function lazy*. *Lazy* berfungsi untuk melakukan *render* komponen ketika komponen itu dibutuhkan sehingga ketika halaman pertama kali diakses oleh pengguna maka tidak semua komponen akan diproses sehingga akan mengurangi waktu *render* dari suatu komponen.

Pengujian *user interface* dilakukan dengan menggunakan fitur *React Profiler* yang tersedia pada *React Dev Tools* yang berfungsi menguji waktu *rendering* komponen pada saat halaman *website* pertama kali diakses. Pada Tabel 1 menunjukkan hasil rata-rata total waktu yang diperlukan untuk *rendering* komponent pada *client side* dalam satuan waktu *milisecond* antara penerapan *code splitting React* dan tanpa menggunakan *code splitting React*. Pada keseluruhan komponen yang terdapat pada halaman *website* yang diuji menunjukkan pengurangan waktu dalam *rendering*. Sebelum penerapan *code splitting*, seluruh komponen akan ditampilkan langsung oleh *browser* sehingga dibutuhkan waktu yang lebih banyak dibandingkan setelah menerapkan code splitting yang akan *render* komponen yang dibutuhkan oleh *user*.

TABEL 1
PERBANDINGAN *RENDER DURATION* KOMPONEN

Komponen	***Render Duration (milisecond)***	
	Tanpa *Code Splitting*	**Menggunakan *Code Splitting***
Home	94.4	64.3
Shop	237	90.4
Detail Product	283	96.7
Cart	141	97.9
Checkout	98.3	81.2
User Dashboard	73.4	62.1
User Password	67.7	47.3
User Wishlist	118	59.1
Admin Dashboard	110.5	68.4
Admin Create Product	91.1	75.9
Admin Manage Product	99.2	54.3
Admin Manage Category	150	52.2
Admin Manage Sub Category	81.5	44.8
Admin Manage Coupon	65.8	47.3
Rata-rata	122.20	67.28

Pengujian aplikasi dilakukan dengan menguji fungsi-fungsi dari aplikasi yang telah dibuat untuk mencari *bug* pada sistem. Pengujian aplikasi dilakukan agar sistem yang dibuat berjalan sesuai dengan yang diharapkan dan dapat memenuhi kebutuhan pengguna. Pengujian aplikasi ini menggunakan teknik pengujian yaitu pengujian *alpha* dan *beta*. Pengujian

Alpha adalah pengujian yang dilakukan oleh pihak *developer*. Tabel 2 adalah hasil pengujian *alpha* dari aplikasi yang telah dilakukan.

Berdasarkan hasil pengujian *alpha* yang ditampilkan dalam Tabel 2 menunjukkan masing-masing fungsionalitas dari *website* penjualan mebel berjalan dengan baik dan sesuai dengan harapan pengguna.

TABEL 2
HASIL PENGUJIAN ALPHA

Fungsi yang Diuji	Kondisi	Hasil	Status Pengujian
Register	Mengirim *email* ketika *button register* diklik	Sukses mendapat *email* untuk memasukkan *password* dan berhasil memasukkan *password*	*Valid*
Login	*Login* menggunakan *email* dan *Google account*	Sukses *login*	*Valid*
Produk	*Admin* melakukan *Create, update*, dan *delete*, produk	Sukses *create, update*, dan *delete* produk	*Valid*
Menambah produk ke *cart*	*Customer* dapat menambahkan satu atau lebih produk ke dalam *cart*	Tidak ada duplikasi produk jika user sudah menambahkan produk yang sama sebelumnya	*Valid*
Update* produk pada halaman *cart	*Customer* dapat melakukan *update quantity* dan warna produk	Sukses melakukan *update*	*Valid*
Pembayaran	Pembayaran dengan debit atau *cash on delivery*	*Customer* berhasil melakukan transaksi	*Valid*
***Update* status pesanan**	*Admin* dapat melakukan *update* status pesanan	Status pesanan berhasil diperbaharui	*Valid*

Untuk pengujian *beta* yang ditunjukkan pada Tabel 3 terdapat kuesioner yang berisi pertanyaan-pertanyaan yang ditujukan untuk memperoleh tanggapan dari responden. Pengguna terlebih dahulu mencoba *website* kemudian pengguna diminta memberikan tanggapan melalui kuesioner. Kuesioner terdiri dari 5 pertanyaan yang bertujuan untuk mengetahui pendapat pengguna terkait *website* penjualan Toko Mebel Murah.

TABEL 3
HASIL PENGUJIAN BETA

No	Pertanyaan Kuesioner	STS	TS	N	S	SS
1.	Apakah *website* menampilkan konten informasi di bawah 3 detik saat pertama kali diakses?	0	0	2	9	9
2.	Apakah *website* dapat mempermudah dalam pemesanan produk mebel?	0	1	1	8	10
3.	Apakah *website* dapat memberikan informasi dari setiap produk mebel?	0	0	4	8	8
4.	Apakah dengan *website* ini mempermudah dalam *tracking* status pesanan?	0	1	7	7	5
5.	Apakah tampilan *website* mudah dipahami?	0	0	1	10	9

Jumlah responden yang terlibat dalam pengisian kuesioner adalah 20 orang yang terdiri dari pemilik dan karyawan dari toko Mebel Murah dan *customer*. Untuk menghitung indeks dari setiap jawaban kuesioner maka digunakan skala *Likert* [13]. Pada Tabel 4 menunjukkan jawaban dari kuesioner yang terdiri atas 5 variasi dan masing-masing diberi nilai.

TABEL 4
PENILAIAN BERDASARKAN JAWABAN

Jawaban	Nilai
Sangat Tidak Setuju (STS)	1
Tidak Setuju (TS)	2
Netral (N)	3
Setuju (S)	4
Sangat Setuju (SS)	5

Untuk mengetahui indeks dari setiap jawaban maka dilakukan perhitungan dengan menggunakan rumus 1:

$$\boldsymbol{Indeks}\ (\%) = \frac{Total\ Nilai}{Jumlah\ Responden\ \times\ Nilai\ Maksimum} \times 100\% \quad (1)$$

Untuk menyimpulkan hasil perhitungan skala *Likert* maka dapat dilakukan dengan analisis interval nilai. interval nilai dihitung dengan menggunakan rumus 2:

$$\boldsymbol{Interval\ Nilai} = \frac{Jumlah\ Responden}{Jumlah\ Nilai} \quad (2)$$

Berdasarkan rumus 1 dan rumus 2 maka kriteria interval nilai dapat ditunjukkan pada Tabel 5.

TABEL 5
INTERVAL PENILAIAN INDEKS

Interval Nilai	Kategori
Indeks 0% - 19,99%	Sangat Tidak Setuju (STS)
Indeks 20% - 39,99%	Tidak Setuju (TS)
Indeks 40% - 59,99%	Netral (N)
Indeks 60% - 79,99%	Setuju (S)
Indeks 80% - 100%	Sangat Setuju (SS)

.

Dari Tabel 6 menunjukkan hasil perhitungan indeks dari masing-masing pertanyaan kuesioner. Dari pertanyaan 1 dapat disimpulkan bahwa *website* dapat menampilkan informasi secara cepat di sisi klien karena sudah memanfaatkan fitur *code splitting*. Dari pertanyaan 2 dapat disimpulkan bahwa *website* dapat mempermudah pembeli dalam memesan produk mebel. Dari pertanyaan 3 dapat disimpulkan bahwa *website* ini dapat memberikan informasi yang jelas terhadap pembeli terkait informasi produk mebel yang dijual. Dari pertanyaan 4 dapat disimpulkan bahwa pembeli dapat mendapatkan informasi terkait status pesanan melalui *email*. Dari pertanyaan 5 dapat disimpulkan bahwa *user interface website* mudah dimengerti oleh pengguna.

TABEL 6
HASIL PERHITUNGAN INDEKS

Pertanyaan	Total Nilai	Indeks	Kategori
1.	87	87%	Sangat Setuju (SS)
2.	87	87%	Sangat Setuju (SS)
3.	84	84%	Sangat Setuju (SS)
4.	76	76%	Setuju (S)
5.	88	88%	Sangat Setuju (SS)

IV. SIMPULAN

Berdasarkan penelitian dan pembahasan di atas maka dapat ditarik simpulan bahwa pengembangan *front-end* dari sistem penjualan produk mebel dapat menerapkan konsep *reusable component* untuk menghindari pengulangan kode sehingga dapat mempercepat pengembangan *website* serta kode yang dihasilkan menjadi lebih terstruktur dan mudah dalam pengembangan ke depannya. Dalam mempersingkat waktu *rendering* di sisi klien dapat disimpulkan bahwa dengan menggunakan strategi *code splitting* yang terbukti efektif berdasarkan dari hasil pengujian yang dilakukan dengan menggunakan *React Profiler*.

Berdasarkan tahapan penelitian yang telah dilakukan maka dapat disimpulkan juga bahwa *website* ini memiliki fungsionalitas yang baik dimana pembeli dapat dipermudah dalam melakukan pembelian produk mebel secara *online* dan membantu pembeli dalam mengetahui status pesanan melalui notifikasi *email*. Saran untuk pengembangan penelitian ini pada masa yang akan datang adalah melalukan *unit testing* pada kode dalam komponen *React* yang sudah dibuat untuk memastikan bahwa kode tersebut berjalan sesuai dengan harapan dan fitur *chat* dalam *website* yang membantu *admin* dan pembeli dalam berkomunikasi secara langsung.

DAFTAR PUSTAKA

[1] M. Bajammal, D. Mazinanian, and A. Mesbah, "Generating reusable web components from mockups," *ASE 2018 - Proc. 33rd ACM/IEEE Int. Conf. Autom. Softw. Eng.*, pp. 601–611, 2018, doi: 10.1145/3238147.3238194.
[2] Christopher. (2020). How To Maximize Reusability For Your *React* Components. [Online]. Available: https://jsmanifest.com/how-to-maximize-reusability-for-your-*React*-components/.
[3] H. Nguyen, "End-to-end E-commerce web application, a modern approach using MERN stack," no. May, 2020.
[4] G. Kivilohkare, "Optimizing the Critical *Rendering* Path for Decreased Website Loading Time," 2020.
[5] A. Bhalla, S. Garg, and P. Singh, "Present Day Web-Development Using *React*js," no. May, pp. 1154–1157, 2020.
[6] P. Speed, "Mobile-Page-Speed-New-Industry-Benchmarks," no. February, 2017.
[7] Facebook. (2020). A JavaScript library for building user interfaces. [Online]. Available: https://*React*js.org/.
[8] E. Wohlgethan, "Bachelorarbeit Comparing Three Major JavaScript Frameworks :," 2018.
[9] Facebook, (2020). Components and Props. [Online]. Available: https://*React*js.org/docs/components-and-props.html.
[10] A. Kumar and R. K. Singh, "Comparative analysis of angularjs and *React*js," *Int. J. Latest Trends Eng. Technol.*, vol. 7, no. 4, pp. 225–227, 2016, doi: 10.21172/1.74.030.
[11] Facebook. (2020). Code Splitting. [Online]. Available: https://id.*React*js.org/docs/code-splitting.html.
[12] Hanafi, "Konsep Penelitian R & D Dalam Bidang Pendidikan," *Saintifika Islam. J. Kaji. Keislam.*, vol. 4, no. 2, pp. 129–150, 2017.
[13] T. Nempung, T. Setiyaningsih, and N. Syamsiah, "Otomatisasi Metode Penelitian Skala Likert Berbasis Web," no. November, pp. 1–8, 2015.