Perancangan dan Pengimplementasian Sistem Backend Aplikasi Cafeasy (Studi Kasus: Kafe daerah Bandung)

1st I Komang Danda Priyowittesa
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
dandakomang@student.telkomuniversity.ac.id

2nd Dana Sulistyo
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
danakusumo@telkomuniversity.ac.id

3rd Nungki Selviandro
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
nselviandro@telkomuniversity.ac.id

***Abstrak* — Pada era teknologi yang semakin maju, memanfaatkan teknologi dalam berbagai bidang sangat penting, seperti bidang kuliner terutama kafe. Manajemen kafe yang manual tentu saja sering terjadi kesalahan pada saat pembukuan dan juga sistem pemesanan masih antre dikasir tentunya akan membuat manajemen kafe dan pelayanan konsumen menjadi kurang efektif. Teknologi pemindaian kode QR sebagai sarana pemesanan, serta pembukuan otomatis akan membantu dalam pengelolaan data pada kafe. Hal tersebut membuat aspek manajemen kafe, dan pelayanan konsumen semakin efektif. Karena manajemen kafe otomatis dapat menghindari kesalahan saat pembukuan, serta konsumen tentunya tidak perlu antre dikasir untuk pemesanan. Aplikasi Cafeasy diperkenalkan sebagai solusi untuk mendukung pelanggan memesan makanan di kafe dengan memindai kode QR. Selain itu, Cafeasy juga menyediakan fitur pembukuan secara otomatis, yang berarti fitur pada Cafeasy sejalan dengan manfaat penggunaan teknologi pada kafe. Pengembangan Cafeasy menggunakan MERN(MongoDB, Express JS, React JS, Node JS) Stack dan berfokus pada sistem backend. MongoDB sebagai database, dan Express JS sebagai controller yang membantu dalam pengimplementasiannya. Express JS digunakan dalam pembuatan endpoint API dengan beberapa metode. Endpoint API yang dibuat berjalan sesuai dengan product backlog, dan performa pemuatan data juga tidak menghabiskan waktu sampai 1 detik. Namun performa pemuatan data tersebut juga bergantung dengan koneksi internet perangkat.**

***Kata kunci*— mern stack, express js, mongodb, node js, backend, api.**

I. PENDAHULUAN

A. Latar Belakang

Pada masa perkembangan teknologi yang semakin maju, pemanfaatan aplikasi dalam berbagai bidang sangat penting, salah satunya dalam bidang kuliner, namun pemanfaatan teknologi pada bidang usaha kuliner masih cukup minim[1]. Berdasarkan hasil riset yang telah dilakukan pada 9 kafe didaerah bandung yang termasuk kedalam kategori *medium* kafe, terdapat 6 kafe yang masih melakukan transaksi secara *offline,* serta pembukuan secara manual. Selain itu pada riset lainnya, terdapat 31 responden secara keseluruhan, yang secara garis besar bahwasanya transaksi secara *online* lebih efektif dibandingkan transaksi yang dilakukan masih dengan cara *offline.*

Jika dilihat dari sudut pandang kafe tersebut, tentunya kurang efektif dari segi proses pencarian data, dan validasi data karena masih dilakukan secara manual[2]. Selain itu karena proses transaksi juga masih dilakukan secara *offline,* maka potensi penumpukan antrean yang terjadi akan cukup meningkat, walaupun pihak kafe memperluas area, dan menyediakan tambahan lain, tentunya hal ini tidak sejalan dengan penambahan tenaga kerja kafe tersebut yang dapat mengakibatkan masalah baru[3].

Cafeasy merupakan sebuah aplikasi yang digunakan untuk memudahkan pelanggan melakukan pemesanan di *café* atau restoran melalui scan kode QR, serta aplikasi ini juga bertujuan memudahkan dalam proses pembukuan *café* atau restoran. Aplikasi ini banyak melakukan pengelolaan data dan penyimpanan data. Oleh karena itu, pada tugas akhir ini akan dilakukan perancangan dan pengimplementasian sistem *backend* pada aplikasi Cafeasy menggunakan basis data MongoDB. Metode perancangan sistem mengacu pada metode Scrum yang merupakan sebuah metode yang mudah dikontrol, memuat strategi pengembangan menyeluruh dimana seluruh tim bekerja sebagai satu unit untuk mencapai goal yang sama[4]. Perancangan ini juga menggunakan teknologi MERN Stack (MongoDB, Express JS, React JS, Node JS), karena performa yang cepat, penyimpanan data yang fleksibel oleh MongoDB, dan kemudahan dalam pengembangan[5], serta popularitas yang diberikan Express JS[6], sehingga aplikasi Cafeasy dapat meningkatkan efisiensi penjualan, dan pembukuan bagi *café* atau restoran yang menggunakannya.

B. Rumusan Masalah

1. Bagaimana perancangan dan pengimplementasian sistem *backend* aplikasi cafeasy menggunakan MongoDB, Express JS, dan Node JS.

2. Bagaimana pengimplementasian sistem *backend* dengan Rest API(*Application Programming Interface*) menggunakan Express JS pada aplikasi Cafeasy admin dan pelanggan.

C. Topik dan Batasan

Topik dari tugas akhir ini adalah Perancangan dan Pengimplementasian Sistem Backend Aplikasi Cafeasy (Studi Kasus: Kafe Daerah Bandung), dengan menggunakan metode Scrum dan Teknologi MERN Stack. Bentuk tugas akhir ini juga merupakan tugas akhir Capstone (group project). Adapun batasan dari tugas akhir ini adalah sebagai berikut:

1. Mengerjakan sistem backend menggunakan MongoDB sebagai basis data yang akan mengelola data pelanggan dan admin.
2. Membuat API (Application Programming Interface) untuk melakukan beberapa metode pengelolaan basis data.
3. Tugas akhir ini hanya melakukan pembuatan sistem backend dengan pengelolaan data pada MongoDB.
4. Perancangan dan pengimplementasian aplikasi Cafeasy dikerjakan dalam bentuk tugas akhir Capstone(Group Project).

Dengan adanya batasan tersebut, diharapkan tugas akhir ini dapat berfokus dengan perancangan dan pengimplementasian sistem backend pada aplikasi Cafeasy menggunakan basis data MongoDB, dengan menggunakan metode yang tepat.

D. Tujuan

Tujuan dari tugas akhir berbentuk Capstone ini adalah:

1. Merancang sistem backend untuk aplikasi Cafeasy yang dapat melakukan pengelolaan data pelanggan dan admin, menggunakan teknologi MERN Stack.
2. Membuat API dengan beberapa metode pengelolaan data, yang dapat berjalan dengan baik sesuai metode dan teknologi yang digunakan.

Dengan mencapai tujuan tersebut, diharapkan hasil dari tugas akhir ini dapat memberikan kontribusi dalam pengembangan aplikasi Cafeasy yang lebih baik bagi penggunanya. Selain itu, diharapkan juga dapat memberikan kontribusi dalam perancangan dan pengimplementasian sistem backend aplikasi Cafeasy yang sesuai dengan kegunaan dan tujuan tugas akhir ini.

II. KAJIAN TEORI

A. Software Development Life Cycle: Scrum

Scrum adalah metode atau kerangka kerja struktural untuk mengembangkan produk yang kompleks[4]. Scrum terdiri dari beberapa fase, yaitu *Product Bakclog*¸ *Sprint Planning*, *Sprint, Sprint Review*, dan *Retrospective*. Scrum memberikan transparansi dalam komunikasi dan menciptakan lingkungan akuntabilitas kolektif, dan peningkatan keberlanjutan[7]. *Product Backlog* berisi aktivitas untuk membuat daftar pekerjaan yang harus diselesaikan. *Sprint Planning* meliputi pembagian tugas dan apa yang perlu dilakukan, yang hasilnya adalah *sprint backlog*. Fase *sprint* mengerjakan tugas-tugas yang telah dipecah oleh *sprint backlog*. *Sprint Review* adalah kegiatan untuk *review output* dari *sprint backlog*, yaitu produk-produk yang siap dirilis. Dan terakhir adalah *retrospective*, yaitu evaluasi dari proses sebelumnya.

B. Node JS

Perangkat lunak Node JS digunakan dalam pengembangan aplikasi berbasis web dan ditulis dalam sintaks bahasa pemrograman JavaScript. Ditulis dalam JavaScript dan berjalan di Windows, Mac OS X, dan Linux tanpa mengubah kode program. Node JS memiliki perpustakaan server HTTP sendiri sehingga memungkinkan untuk menjalankan server web tanpa menggunakan program server web seperti Apache atau lainnya[8]. Node JS sendiri memiliki keunggulan yaitu teknik *non-blocking* memungkinkan operasi dilakukan oleh sistem secara paralel tanpa harus menunggu operasi sebelumnya selesai, sehingga memungkinkan memungkinkan beberapa permintaan diproses secara paralel[9]. Selain itu, Node JS menggunakan teknik *nonblocking* untuk mempercepat proses[10]. Karena hasil yang baik dalam hal kecepatan, dan penggunaan sumber daya, Node JS telah digunakan oleh banyak perusahaan terkenal seperti Netflix, Linkedin, dan juga Paypal[6]. Node JS juga memiliki banyak *package* yang tersedia didalam NPM (*Node Package Manager*) yang tentunya dapat mempermudah dalam pembuatan dan pengembangan aplikasi.

C. Express JS

Express JS adalah *framework* atau kerangka kerja yang termasuk dalam Node JS yang mudah dikembangkan untuk pengembangan aplikasi web, layanan API, perutean, dan keamanan. Express JS adalah kerangka kerja dimana pengguna memiliki kebebasan untuk menentukan metode mana yang harus digunakan untuk menjalankan perintah yang diberikan. Express JS juga jauh lebih dinamis dan bertenaga karena banyak modul tersedia di dalam NPM (*Node Package Manager*) yang dapat dipetakan dan diintegrasikan langsung ke dalam Express JS[11]. Express JS sebagai *unopinionated framework* lebih fleksibel, sehingga dapat memasukkan beberapa komponen seperti *Middleware* untuk menangani kebutuhan tertentu. Artinya Express tidak memiliki ketentuan tertentu dan memiliki batasan yang lebih sedikit. Selain itu, Express JS juga merupakan *framework* yang sangat populer saat ini. Mengenai kepopuleran ini, kepopuleran suatu *framework* menjadi penting karena dapat dijadikan sebagai indikator apakah *framework* tersebut akan terus eksis dan dipertahankan, atau sebaliknya. Kemudian semakin populer *framework* juga akan memiliki sumber dokumentasi yang baik.

D. MongoDB

MongoDB sebagai *database non-relational* biasanya digunakan untuk membangun sebuah *website*. MongoDB memakai format file JSON (*JavaScript Object Notation*). Penggunaan *database* berbasis NoSQL(*Not Only SQL*) dan *database* berbasis SQL(*Structured Query Language*) memiliki arsitektur sangat berbeda. *Database* NoSQL adalah *database* yang menyimpan data dalam format selain tabel

relasional[12], NoSQL ini juga tidak menggunakan relasi dalam pembuatan basis data[13]. Penggunan MongoDB dapat berbanding lurus dengan peningkatan permintaan untuk jenis data yang akan disimpan, karena dalam kasus tipe data ini, *database* SQL memiliki sifat yang terbatas pada kumpulan tipe data yang mendasarinya, misalnya seperti integer, string, dan tanggal, tentu saja akan menimbulkan masalah ketika data membutuhkan penyimpanan yang tidak teratur atau dinamis. Misalnya, tipe data array yang tentu saja tidak teratur atau dinamis tergantung pada kebutuhan yang diinginkan. Tipe data array tidak ada dalam *database* SQL. *Database* NoSQL juga merupakan model data yang fleksibel, skema seperti ini memudahkan untuk memodifikasi *database* saat kondisi berubah[6]. MongoDB memiliki format yang cepat karena menyimpan model data berbasis dokumen dan *cache*, berupa file JSON yang disebut BSON[14]. MongoDB dapat secara otomatis membuat struktur tabelnya sendiri saat melakukan Insert. Oleh karena itu skemanya lebih fleksibel, sebagai *database* NoSQL. MongoDB juga memiliki kapasitas penyimpanan data yang lebih besar dan tidak mahal. Di MongoDB hanya ada koleksi dan dokumen. Dokumen dalam MongoDB dapat memiliki atribut yang berbeda dengan dokumen lain meskipun berada dalam satu koleksi yang sama. Hal ini tidak mungkin terjadi pada RDBMS(*Relational Database Management System*), dimana suatu baris dalam sebuah tabel tidak boleh memiliki kolom yang berbeda dengan baris lainnya jika berada dalam sebuah tabel, sehingga teknologi ini sangat bermanfaat bagi pengguna dalam pengelolaan data yang efisien[15]. Dalam penggunaannya, diharuskan membuat skema model untuk ke *database* terlebih dahulu yang bertujuan untuk melakukan pemetaan terhadap koleksi MongoDB, dan *document* didalamnya sesuai dengan skema yang sudah dibuat[16].

E. MERN Stack

MERN stack merupakan salah satu teknologi yang terdiri dari MongoDB, Express JS, ReactJS, dan Node JS[8]. Istilah MERN Stack disusun dari komponen-komponen sumber terbuka yang menyediakan kerangka kerja *end-to end* untuk pengembangan aplikasi berbasis web yang komprehensif dan memudahkan peramban untuk terhubung dengan basis data[17]. Oleh karena itu pada dasarnya memiliki kesamaan yang cukup banyak antara komponen-komponen tersebut seperti MongoDB yang datanya berformat JSON(*JavaScript Object Notation*), lalu Express JS yang merupakan *framework open source* dari JavaScript, React JS yang juga merupakan *library opensource* dari JavaScript, dan yang terakhir Node JS yang merupakan *runtime* JavaScript yang dapat digunakan diluar *browser*. Semua komponen memiliki dasar yang sama yaitu JavaScript, sehingga untuk melakukan pertukaran data dari *model, controller, dan view* akan cukup mudah dan fleksibel. Pemilihan teknologi ini juga berdasarkan faktor keunggulan teknologi itu sendiri, baik dalam hal performa, kemudahan, popularitas, dan keberlanjutan teknologi untuk beberapa tahun kedepan.

F. API(*Application Programming Interface*)

API adalah sebagai antarmuka digunakan untuk penghubung bagi *client* dan server. *Client* dapat melakukan *request,* akan diterima ke server melalui API lalu server akan mengirimkan *response* sesuai dengan *request* yang diminta[18]. API sebagai penghubung antara *client* dengan fungsi yang telah dikerjakan sesuai dengan kebutuhan yang telah ditentukan[19]. API membuat pengembang dapat memakai fungsi yang sudah dibuat, sehingga untuk proses permintaan dan penerimaan data tidak perlu dilakukan secara manual, karena data yang diminta dan diterima didapatkan melalui *request* HTTP yang dilakukan sesuai kebutuhan. API mempercepat proses pengembangan aplikasi, dan juga membuat fungsi yang tersedia menjadi lebih struktural karena API yang dibuat terpisah[18].

G. Postman

Perangkat lunak postman digunakan dalam pengujian API. Postman cukup populer dalam pengembangan dan pengujian API, karena pengembang dapat membuat dan menerima permintaan HTTP beserta dengan respon yang diberikan terhadap permintaan tersebut. Pengembang dapat membuat koleksi di postman untuk pemanggilan API, koleksi ini berfungsi untuk membuat setiap permintaan yang dibuat agar lebih struktural dan tidak tercampur dengan API pada projek lain[20] . Dalam pemanggilan API menggunakan postman juga tersedia performa catatan waktu pemuatan data. Pengaruh postman terhadap API sangat membantu dalam pengembangan sistem *backend*, karena postman sebagai *client* yang menggantikan *frontend* pada pengetesan ini terhubung ke server melalui HTTP[21].

H. MongoDB *Collection*

Koleksi pada MongoDB sebagai *database* NoSQL, merupakan *tables* jika diumpamakan pada *database* SQL[22]. Namun perbedaan yang terdapat antara keduanya bukan hanya dari penamaan, akan tetapi dari sistem penggunaannya. Jika pada koleksi MongoDB dapat memasukkan sebuah data dokumen dengan struktur berbeda pada koleksi yang sama, maka *tables* pada *database* SQL tidak dapat melakukan hal tersebut, karena data yang dimasukkan harus sesuai dengan atribut yang telah ditentukan pada *tables*. Berdasarkan hal tersebut, hal ini berarti MongoDB tidak memerlukan pemodelan yang bersifat relasional seperti halnya SQL, karena MongoDB bersifat NoSQL maka hanya memerlukan pemodelan dalam bentuk pembuatan koleksi saja[23].

III. METODE

A. Scrum

Pengimplementasian *scrum* dilakukan pada *platform* Trello. *Product backlog* yang telah didapatkan dari *user story*, lalu dilakukan *sprint planning* untuk menentukan *backlog* yang mana saja yang akan dikerjakan, serta siapa yang akan bertanggung jawab terhadap *backlog* tersebut. Pengerjaan *sprint* dilaksanakan kurang lebih dalam estimasi 2 minggu. *Sprint review* dilakukan pada akhir *sprint*¸ untuk mengecek bagaimana perfoma pekerjaan setiap anggota tim pada *sprint* tersebut, lalu dilakukan *retrospective* untuk mengevaluasi apa yang masih kurang pada *sprint* sebelumnya agar dapat diperbaiki pada *sprint* selanjutnya.

TABEL 1
(Beberapa Product Backlog Fokus Tugas Akhir Ini)

Product Backlog* Fokus Sistem *Backend	
Backlog ID	**Backlog**
PM-02	Sebagai pelanggan cafe, saya ingin melakukan sign in dengan menuliskan nama saja, sehingga

	saya dapat melakukan pemesanan.
PM-07	Sebagai pelanggan cafe, saya ingin melihat harga, dan status / ketersediaan dari setiap makanan dan minuman, sehingga saya dapat meyimpulkan makanan atau minuman apa yang enak atau sering dipesan.
PM-21	Sebagai pelanggan cafe, saya ingin melihat status pada setiap menu, sehingga saya dapat melakukan pemesanan pada menu yang tersedia.

B. Desain Teknologi Arsitektur

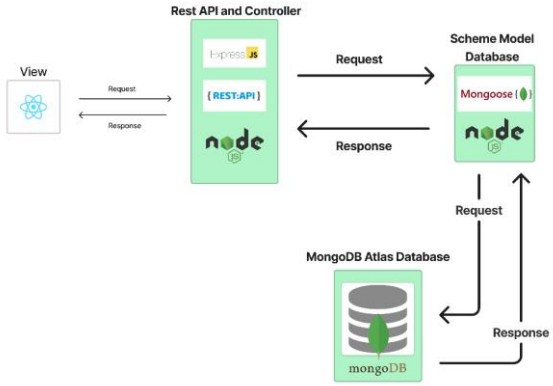
Aplikasi Cafeasy dalam perancangannya MERN Stack, sehingga untuk penjabaran teknologi apa saja yang digunakan, dicantumkan pada table berikut.

TABEL 2
(Tech Stack Cafeasy)

Tools	Name
Database	MongoDB
Programming Language	Node JS
Versioning	Git(Github)
Service API Framework	Express JS
Library Object Data Model	Mongoose
Library Frontend	React JS

Berdasarkan Tabel 1(Tech Stack Cafeasy), kombinasi dari MongoDB, Express JS, React JS, dan Node JS merupakan implementasi dari MERN pada komponen teknologi yang digunakan pada Cafeasy. React JS berfungsi sebagai kerangka kerja *frontend*, dan berkomunikasi dengan server, dengan mengirimkan permintaan *request* ke server dan menerima *response* dari server. Express JS yang bertugas menerima *request*, mengirimkan *response*, dan juga mengembangkan sisi *backend* agar dapat diakses oleh *frontend*. MongoDB memiliki performa yang baik, dan kemudahan dalam pengolahan data, karena data yang diolah berupa JSON. Selain itu *versioning* sistem aplikasi Cafeasy akan menggunakan GitHub, serta pada *backend* menggunakan *Library Object Data Model* yang bernama mongoose yang akan sangat membantu dalam membuat skema model yang akan langsung terhubung ke MongoDB Atlas *Database*.

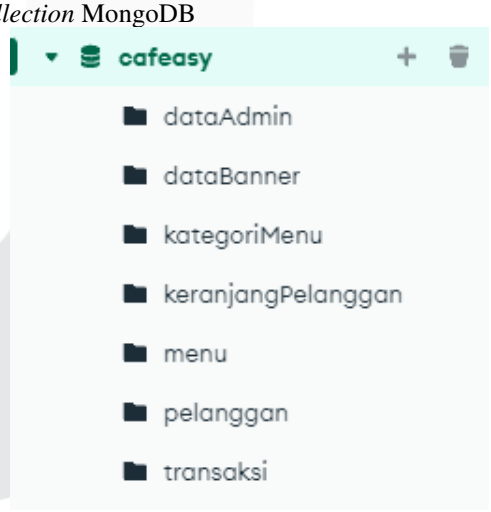
C. Diagram Arsitektur



GAMBAR 1
(Diagram Arsitektur Cafeasy)

Gambar 1(Diagram Arsitektur Cafeasy) merupakan diagram yang menggambarkan implementasi MERN Stack terhadap Diagram Arsitektur Cafeasy yang telah difokuskan pada bagian *backend*. Aplikasi Cafeasy menggunakan Express JS dan Node JS digunakan untuk membuat API dan pengolahan data yang berinteraksi dengan MongoDB sebagai *database*. React JS akan mengirimkan permintaan HTTP *request* ke server *backend*, lalu server akan menerima permintaan *request* dan melakukan pengambilan data ke *database* yang kemudian akan dikirimkan dalam bentuk *response* sesuai dengan data yang di *request*. Kemudian respon juga akan diperbarui berdasarkan data yang dikembalikan.

D. *Collection* MongoDB



GAMBAR 2
(Koleksi Database MongoDB)

Pembuatan *collection database* pada MongoDB dilakukan berdasarkan *product backlog* yang sudah dibuat sebelumnya. *Collection* ini nantinya akan berisikan dokumen-dokumen yang diperlukan. Lalu perancangan skema sebagai model mengikuti penamaan sesuai dengan masing-masing *collection* yang akan digunakan. Pembuatan *collection* ini menggunakan MongoDB Compass yang sebelumnya sudah dikonfigurasi melalui atlas *user interface* yang sudah disediakan oleh MongoDB secara *online*.

IV. HASIL DAN PEMBAHASAN

A. Skema Model *Database*

Skema Model *Database* pada aplikasi Cafeasy dibuat sesuai dengan *database* yang digunakan yaitu MongoDB. MongoDB merupakan *database* NoSQL yang berbasis dokumen, oleh karena itu pemodelan *database* MongoDB antar dokumen tidak menggunakan tabel untuk memodelkannya. Pemodelan MongoDB yang berbasis dokumen dengan data berbentuk JSON hanya menggunakan baris-baris kode yang dibuat, sesuai dengan *modelling* MongoDB[24]. Sehingga dalam pemodelan MongoDB hanya menggunakan skema saja, yang dibuat menyesuaikan dengan kebutuhan data yang diperlukan.

Pembuatan skema ditujukan untuk setiap *collection* yang telah dibuat sebelumnya, untuk itu skema yang dibuat sesuai dengan *collection* dan memiliki tipe data yang berbeda-beda sesuai dengan kegunaan. Salah satu skema yang telah dibuat adalah skema untuk *collection* keranjangPelanggan, berikut merupakan hasil dari pembuatan skema yang telah dilakukan.

```
    idKeranjang: {
        type: String,
        required: true
    },
    idPelanggan: {
        type: String,
        required: true
    },
    namaPelanggan: {
        type: String,
        required: true
    },
    dataPesanan: {
        type: Array,
        required: true
    },
    noMeja: {
        type: Number,
        required: false
    }


}, { versionKey: false, timestamps: true });
```

GAMBAR 3
(Skema Koleksi Keranjang)

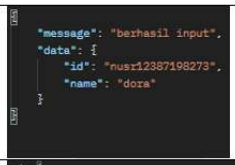
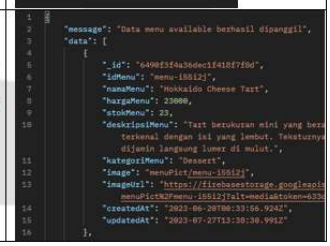
Pada Gambar 3(Skema Koleksi Keranjang) merupakan contoh salah satu skema yang telah dibuat untuk *collection* keranjang. Skema pada *collection* keranjang memiliki beberapa atribut yaitu, idKeranjang, idPelanggan, namaPelanggan, dataPesanan, noMeja. Pemilihan atribut ini berdasarkan analisis pada *product backlog* yang ada, sehingga ditentukan atribut apa yang akan dibutuhkan sesuai dengan kondisi dalam *product backlog* tersebut. Sebagai sampel, atribut pada *collection* keranjang yang telah dirancang untuk memenuhi kondisi pada *product backlog* PM-22, PM-08, dan PM-09.

Atribut-atribut pada setiap skema tentunya berbeda-beda menyesuaikan dengan kebutuhan data pada *database*. Skema ini juga dibuat menggunakan bantuan dari *package* Node JS yaitu Mongoose sebagai *Library Object Data Model*.

B. *Application Programming Interface*(API)

API merupakan sebuah penghubung antara klien dan server, untuk melakukan kelola data tanpa adanya penambahan data secara manual ataupun secara *hardcoded*. API ini dibuat menggunakan Express JS yang merupakan *framework* oleh Node JS yang ditulis juga menggunakan bahasa javascript. Express JS berfungsi untuk mengatur fungsionalitas seperti *routing, session, request http*, dan juga pertukaran data di server. Express JS yang digunakan pada Cafeasy memiliki arsitektur berupa MVC (*Model*, *View*, *Controller*), yang berarti data pada model dihubungkan oleh *controller* lalu akan ditampilkan ke *view* setelah melewati proses *routing*. API yang telah dibuat dites menggunakan postman sebagai sebuah *platform* yang berguna untuk mendokumentasikan, sekaligus menguji API yang telah dibuat. Berikut adalah sampel dari beberapa API yang telah dibuat, dan didaftarkan ke *routing* terlebih dahulu, dijabarkan kedalam tabel.

API Cafeasy Pelanggan

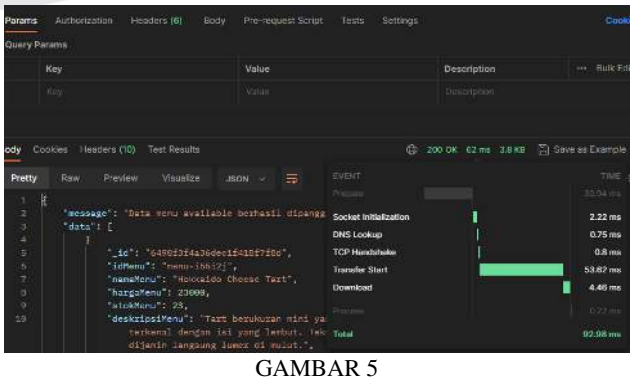
Test Case ID	Method	Endpoint https://cafeasy.shop/api/	Product Backlog ID - Deskripsi	Output
TCP -1	POST	/customer Input: Body:{ id, name }	PM-02 – Melakukan sign in hanya dengan nama	
TCP -2	GET	/ListMenu Input: Tidak ada	PM-07, PM-21 - Melihat harga, dan status / ketersediaan dari setiap makanan dan minuman	

GAMBAR 4
(Sampel Endpoint API Cafeasy Pelanggan)

Pada Gambar 4(Sampel Endpoint API Cafeasy Pelanggan) terdapat beberapa *endpoint* API yang telah dibuat. Sampel pada tabel tersebut bersumber dari API Pelanggan yang telah dirancang pada *routes*, yang akan digunakan untuk menerima permintaan atau pengiriman respon data terhadap *frontend*.

C. Performa *Endpoint* API pada Postman

Pengujian terhadap seluruh *Endpoint* API yang telah dibuat menggunakan Express JS, dilakukan menggunakan postman sebagai *platform* tes performa ini. Tes ini dilakukan tanpa pengecualian, yang berarti dilakukan pada seluruh *endpoint* API *customer* atau pelanggan, dan juga fitur admin. Performa dari setiap *endpoint* API, diberikan sebuah sampel yang tertampil sebagai berikut.



GAMBAR 5
(Performa Get data available menu: 62ms – pelanggan)

Pada gambar 5(Performa Get data available menu: 62ms – pelanggan) merupakan salah satu contoh dari hasil catatan waktu performa Express JS dalam melakukan pengelolaan data menggunakan *endpoint* API Cafeasy. Berdasarkan kecepatan performa dari hasil tes tersebut, kecepatan yang didapat dari implementasi Express JS terhadap Cafeasy tidak menghabiskan waktu sampai 1 detik. Walaupun hal ini hanya dalam performa pemuatan data, tentu saja akan berdampak besar terhadap jumlah waktu yang dibutuhkan dalam proses pemuatan *website*, karena pemuatan *website* memerlukan pemuatan data. Namun hal ini tentu saja berkaitan dengan koneksi internet pada masing-masing perangkat yang digunakan, karena MongoDB yang digunakan berupa *database* yang dikoneksikan secara online. Pengujian ini juga hanya dilakukan dengan kasus akses pengguna berjumlah 1 saja.

D. Analisis Hasil Pengujian

Berdasarkan hasil pengujian *endpoint* API pada postman, dan juga performa yang sudah dijabarkan sebelumnya, dapat dinilai *endpoint* API Cafeasy yang telah dibuat dapat bekerja sesuai dengan *product backlog* yang ada. Meskipun pengujian ini hanya berupa pemuatan data saja, akan tetapi mempunyai dampak yang besar nantinya terhadap jumlah waktu untuk pemuatan *website*. Selain itu karena Express JS merupakan *framework* yang tidak memiliki aturan dalam penggunannya, maka hal ini tentu saja memberikan kebebasan kepada pengembang untuk melakukan dan menentukan metode yang akan digunakan untuk mengeksekusi suatu perintah seperti *request*, *response*, atau *next*. Hal tersebut termasuk kedalam aspek fleksibilitas, yang memudahkan pengembang dalam pengembangan sisi *backend*.

Pengaruh Express JS dalam pengimplementasian MERN juga berperan besar, Express JS digunakan sebagai *Controller* pada konsep MVC, yang dimana dapat dilihat pada diagram arsitektur Cafeasy, bahwasanya Express JS menerima *request* dari *client* yaitu React JS. *Request* yang dikirimkan dari *client*, akan diterima Express JS, lalu akan memberikan *response* sesuai *request* yang diminta. Dalam hal ini, performa kecepatan waktu dalam penerimaan *request* dan pengiriman *response* berhubungan dengan pemuatan data yang sudah diuji sebelumnya.

E. Analisis Implementasi MongoDB, Expres JS, dan Node JS

Pada implementasi MongoDB dibantu oleh mongoose sebagai *library object data model* pada saat membuat skema. Selain itu pada implementasi MongoDB, pada beberapa koleksi dapat menggunakan tipe data array sesuai dengan kebutuhan tipe data yang dinamis karena bersifat NoSQL yang berarti tidak terstruktur. Hal ini juga berkaitan dengan fleksibilitas kedepannya jika terjadi modifikasi skema, tentunya akan lebih memudahkan perubahan saat kondisi berubah, karena jika terjadi perubahan pada skema koleksi maupun tipe data, dapat langsung diubah melalui model skema yang telah dibuat.

Pembuatan Rest API aplikasi Cafeasy menggunakan Express JS dengan konsep MVC(*Model*, *View*, *Controller*). Express JS berperan sebagai *controller*, melakukan *routing*, dan membuat middleware yang dapat disesuaikan dengan kebutuhan karena bersifat *unopinionated framework*. Hal ini tentunya memudahkan dalam pengembangan karena dapat menyesuaikan dengan kondisi kebutuhan sehingga dapat menjadi indikator fleksibilitas yang baik. Selain itu juga dalam pengembangan dimudahkan karena memanfaatkan dokumentasi yang baik dari Express JS.

Node JS sebagai *runtime environment* memiliki peran yang sangat penting, karena selain dapat menjalankan JavaScript diluar *browser*, Node JS juga dapat menginstal *framework* dan *library*. Hal ini sangat penting, karena dalam pengembangan Cafeasy pada sisi *backend* memerlukan instalasi seperti Express JS yang digunakan untuk proses pembuatan *endpoint* API, Mongoose sebagai *package* pendukung untuk melakukan koneksi ke MongoDB, dan juga beberapa *package* lainnya. Hal ini dapat disimpulkan kemudahan perancangan dan pengimplementasian sisi backend Cafeasy adalah dampak dari implementasi Node JS yang terfokus terhadap aspek *backend*, yang dapat dilihat Node JS sebagai inti utama karena merupakan *runtime environment* yang bertugas untuk menjalankan, dan instalasi *package*.

V. KESIMPULAN

A. Kesimpulan

Berdasarkan hasil dan pengujian, serta proses dalam pengembangan dan implementasi MERN terhadap sisi *backend* Cafeasy, dapat dilihat implementasi ini berjalan dengan baik pada sisi *backend* yang dibuat. Penggunaan MERN dalam implementasi sisi *backend* ini berarti terfokus dengan aspek-aspek *backend* seperti MongoDB, Express JS, dan Node JS. Hasil dari *endpoint* API admin berjumlah 24, dan *endpoint* pelanggan berjumlah 14. Semua *endpoint* API yang dibuat dapat berjalan dengan baik.

Dengan adanya tes *endpoint* API, dapat disimpulkan *endpoint* API Cafeasy yang dibuat dalam perancangan dan pengimplementasian MERN dapat bekerja sesuai dengan *product backlog*, serta tes performa pemuatan data memiliki hasil yang baik, tidak menghabiskan waktu sampai 1 detik. Namun tidak menutup kemungkinan pemuatan data ini akan berjalan dengan waktu lebih dari total waktu yang didapatkan pada pengujian sebelumnya, karena MongoDB dikoneksikan secara online, sehingga sangat bergantung dengan kecepatan internet perangkat, dan juga ukuran dari data yang dimuat, karena semakin rendah kecepatan internet perangkat atau semakin besar ukuran dari data yang dimuat maka kemungkinan besar akan menghabiskan waktu lebih lama.

B. Saran

Saran pada proses pengembangan selanjutnya adalah untuk membuat *endpoint* API yang lebih tertata lagi seperti menggunakan v1 atau v2 yang berguna untuk menunjukkan versi *endpoint* API, sehingga dimasa yang akan datang akan lebih mudah dalam mengetahui *endpoint* API yang lama dan yang terbaru. Selain itu membuat penanganan kesalahan dalam menjalankan Cafeasy. Penanganan kesalahan yang dimaksud berupa penanganan agar tidak terjadinya server yang berhenti ketika mendapatkan sebuah *error* dalam menjalankannya.

REFERENSI

[1] J. F. Rusdi, N. A. Abu, N. Agustina, and S. Dewi, "Sofware Development Stages of Mobile Computing Implementation in Restaurant Food Ordering," *SciTech Framework*, vol. 1, no. 1, pp. 24–33, 2019.

[2] L. B. A. Pambudi, A. Rahagiyanto, and G. E. J. Suyoso, "Implementasi QR code untuk efisiensi waktu pemesanan menu makanan dan minuman di restoran maupun kafe," *BIOS: Jurnal Teknologi Informasi dan Rekayasa Komputer*, vol. 1, no. 1, pp. 35–39, 2020.

[3] S. Nadia, "IMPLEMENTASI BACK-END PADA DIGITALISASI PEMESANAN MENU MAKANAN DAN MINUMAN BERBASIS WEBSITE (STUDI KASUS: TOKO MANSURE)," 2023.

[4] W. Warkim, M. H. Muslim, F. Harvianto, and S. Utama, "Penerapan Metode SCRUM dalam Pengembangan Sistem Informasi Layanan Kawasan," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 6, no. 2, Aug. 2020, doi: 10.28932/jutisi.v6i2.2711.

[5] Moch. A. Maulana, H. Haryoko, B. Santoso, and L. Lukman, "Penerapan Teknologi Stack MERN pada Aplikasi Service Manajemen Bengkel Berbasis Web," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 6, no. 3, p. 1536, Jul. 2022, doi: 10.30865/mib.v6i3.4147.

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

[7] N. Hadinata dan Muhammad Nasir, N. Hadinata, M. Nasir, U. Bina Darma, and J. Jenderal Ahmad Yani No, "IMPLEMENTASI METODE SCRUM DALAM RANCANG BANGUN SISTEM INFORMASI PENJUALAN (STUDY KASUS : PENJUALAN SPERPART KENDARAAN)," 2017.

[8] D. Nurul Huda and A. Saputra, "Perancangan Aplikasi IT Help Desk Menggunakan Platform Node.js Pada Mittasys," *Bangkit Indonesia*, vol. IX, no. 01, 2020.

[9] M. A. Solahudin, I. Kadek, and D. Nuryana, "RANCANG BANGUN SISTEM INFORMASI STAYCATION BERBASIS WEB DENGAN IMPLEMENTASI TEKNOLOGI MERN STACK."

[10] A. Mubariz *et al.*, "Perancangan Back-End Server Menggunakan Arsitektur Rest dan Platform Node.JS (Studi Kasus: Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang)," 2020.

[11] H. A. Purnama, "PENGEMBANGAN DAN MAINTENANCE APLIKASI KESEHATAN PADA PT. GLOBAL URBAN ESENSIAL," 2020.

[12] A. Septia Maharani *et al.*, "PERANCANGAN DATA BASE KASIR DAN PERSEDIAAN BARANG MENGGUNAKAN MONGODB," 2022.

[13] A. Aan, S. Daroini, and W. Yustanti, "PERBANDINGAN PENGGUNAAN NOSQL MONGODB DAN MYSQL PADA BASIS DATA FORUM KOMUNIKASI," 2016.

[14] B. Cahyo Santoso, Y. Natasya, S. Willian, and F. Alfando, "Tinjauan Pustaka Sistematis terhadap Basis Data MongoDB."

[15] U. Bina Darma and D. Kurniawan, "Seminar Hasil Penelitian Vokasi (SEMHAVOK) PENERAPAN MONGODB PADA SISTEM INFORMASI MANAGEMENT ACADEMIC SDN 13 BANYUASIN".

[16] A. L. Pisa, H. N. Palit, and J. Andjarwirawan, "PEMBUATAN APLIKASI AUDIENCE RESPONSE SYSTEM BERBASIS WEB DAN ANDROID," *Jurnal Informatika*, vol. 13, no. 1, Feb. 2016, doi: 10.9744/informatika.13.1.25-32.

[17] M. P. Widodo, J. Informatika, F. T. Industri, and A. B. Cahyono, "Pengembangan Aplikasi Pelaporan Progress-Plan-Problem untuk Manajemen Tugas dan Penentuan OKR di Krafthaus Indonesia."

[18] I. Kurniawan and F. Rozi, "REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android," *JITSI: Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 1, no. 4, pp. 127–132, 2020.

[19] U. Rahardja, Q. Aini, and N. P. L. Santoso, "Pengintegrasian YII Framework Berbasis API pada Sistem Penilaian Absensi," *Sisfotenika*, vol. 8, no. 2, pp. 140–152, 2018.

[20] M. Shershneu and A. Oskin, "POSTMAN PLATFORM FOR API DEVELOPMENT IN THE MOBILE APPLICATION 'MUSICIANS OF RUSSIA.'"

[21] S. Arshad Busro Cahyono *et al.*, "Bulletin of Information Technology (BIT) Rancangan Pembuatan Api Website Data Tanaman Obat Dan Langka Kabupaten Kediri," vol. 3, no. 4, pp. 255–260, 2022, doi: 10.47065/bit.v3i1.

[22] Ankita Saini, "MongoDB – Database, Collection, and Document," *https://www.geeksforgeeks.org/mongodb-database-collection-and-document/*, Aug. 13, 2021.

[23] Institute of Electrical and Electronics Engineers, *2017 International Conference on Networks & Advances in Computational Technologies (NetACT) : 20-22 July 2017, Trivandrum, Kerala, India.*

[24] Joe Karlsson, "MongoDB Schema Design Best Practices," *https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/*, Jan. 11, 2022.