Perancangan dan Pengembangan Backend Aplikasi Cafeasy Berbasis Website Terintegrasi Google Sheets (Studi Kasus: Cafe Daerah Bandung)

1st Rizky Naufal Alghifari
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
rizkyalghifari@student.telkomuniversity.ac.id

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

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

***Abstrak* — Pemanfaatan teknologi di industri layanan, terutama di kafe, dapat meningkatkan kualitas pelayanan mereka. Salah satu cara efektif adalah dengan memanfaatkan Google Sheets untuk memanipulasi dan mengelola data pembukuan kafe, serta menyediakan penggunaan basis data secara gratis. Sebelum menerapkan teknologi Google Sheets, kafe melakukan pembukuan secara manual oleh pegawainya. Pendekatan manual ini meningkatkan risiko kesalahan dalam pembukuan harian kafe. Untuk mengatasi tantangan ini, memanfaatkan Google Sheets, sehingga pembukuan menjadi otomatis dan mengurangi risiko kesalahan manusia. Hal ini menghasilkan pembuatan "Cafeasy", sebuah solusi yang memanfaatkan Google Sheets API dan disajikan dalam bentuk situs web. Dalam mengembangkan aplikasi Cafeasy, teknologi seperti NodeJS dan ExpressJS digunakan dengan mengikuti metodologi Scrum. Proses pengembangan didukung oleh pengujian API untuk memastikan fungsionalitas aplikasi sesuai dengan harapan yang diinginkan. Terdapat delapan test case dengan semua test case berstatus pass yang menandakan fungsionalitas backend aplikasi sesuai harapan. Dengan begitu, aplikasi Cafeasy memiliki fungsionalitas yang sesuai pada product backlog atau harapan calon pengguna aplikasi ini. Namun, terdapat dua API yang tidak dilakukan pengujian dikarenakan fungsi API hanya sebagai redirect page API login akun Google. Sedangkan pada sisi request time, waktu yang dibutuhkan pada API yang tidak terhubung dengan Google Sheet API dibawah 600 milidetik atau proses API cukup cepat.**

***Kata kunci*— nodejs, expressjs, pembukuan, google sheets, google apis, scrum**

I. PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi dan informasi semakin memudahkan manusia terutama pelayanan pada rumah makan, *cafe*, ataupun tempat penyedia makanan dan minuman. Salah satu teknologi yang digunakan oleh sebagian besar kafe adalah penyediaan kode QR yang menampilkan menu kafe untuk kenyamanan pelanggan mereka [1]. Namun, teknologi tersebut masih terdapat kekurangan yaitu dalam sistem pembayarannya. Teknologi tersebut hanya menampilkan menu – menu yang ada pada *cafe* dan tidak terdapat fitur transaksi di dalamnya. Dalam riset pada 9 cafe dengan kategori medium cafe di daerah Bandung, 6 diantaranya masih melakukan transaksi secara manual dengan pelanggan mengatri di kasir. Sedangkan dalam proses wawancara kepada 31 responden, menyebutkan bahwa transaksi pada cafe lebih efektif dilakukan secara online. Permasalahannya adalah pelanggan masih harus mengantri untuk melakukan pembayaran di kasir yang akan membuat antrian semakin panjang. Hal tersebut dapat mengurangi kualitas pelayanan *cafe*, membuat pelanggan kurang nyaman.

Selain permasalahan tersebut, sistem penjualan secara konvensional pada *cafe* memiliki permasalahan pada pembukuan harian. Sistem penjualan konvensional sudah tidak dapat diterapkan lagi karena dapat meningkatkan risiko kesalahan pembukuan dan berisiko terjadinya kesalahan dalam perhitungan data serta risiko kehilangan data pembukuan [2].

Dalam mengatasi permasalahan mengenai fitur transaksi online dan pembukuan aplikasi Cafeasy hadir sebagai solusi. Cafeasy merupakan aplikasi berbasis website yang memiliki fitur pembukuan secara otomatis berdasarkan transaksi yang masuk dan juga menyediakan menu – menu cafe melalui scan QR code serta fitur pembayaran di dalamnya. Dengan begitu, antrian panjang pada kasir dapat dikurangi dan meningkatkan kualitas pelayanan cafe. Aplikasi ini terintegrasi dengan Google Spreadsheet untuk pencatatan yang lebih terstruktur. Selain dapat menampilkan menu, aplikasi ini juga menyediakan fitur pembayaran dengan berbagai metode pembayaran.

B. Topik dan Batasannya

Topik dari tugas akhir ini adalah melakukan perancangan dan pengembangan backend pada aplikasi Cafeasy dalam bentuk tugas akhir Capstone (*group project*), dengan metodologi scrum dan pemanfaatan teknologi Google Sheet API, Midtrans Payment Gateway, Google APIs dan NodeJS. Batasan pada penelitian ini sebagai berikut:

1. Tugas akhir ini hanya merancang dan mengembangkan sistem backend aplikasi dengan terintegrasi Google Spreadsheet, Midtrans Payment Gateway, dan Google APIs.

2. Tugas akhir ini hanya menguji sistem backend aplikasi yang telah terintegrasi terintegrasi Google Spreadsheet, Midtrans Payment Gateway, dan Google APIs dengan menggunakan API testing.

C. Rumusan Masalah

Berdasarkan latar belakang di atas, terdapat beberapa rumusan masalah sebagai berikut:

1. Bagaimana mengembangkan backend aplikasi Cafeasy yang dapat melakukan transaksi pesanan Cafe dan dapat melakukan pembukuan harian Cafe secara otomatis?
2. Apakah backend aplikasi Cafeasy dapat melakukan fungsi transaksi pesanan pada cafe dan pembukuan harian cafe secara otomatis?

D. Tujuan

Adapun tujuan penelitian ini adalah:

1. Merancang dan mengembangkan sistem backend aplikasi Cafeasy berupa API yang terintegrasi Google Spreadsheet, Midtrans Payment Gateway, dan Google Account.
2. Menguji keberhasilan dan kualitas setiap fungsi API aplikasi Cafeasy dengan melakukan API testing.

Tujuan dari tugas akhir ini adalah merancang dan mengembangkan backend aplikasi Cafeasy agar dapat berjalan sesuai fungsionalitas yang diharapkan. Dalam mencapai tujuan ini, pengembangan dilakukan menggunakan NodeJS dan ExpressJS serta metode scrum. Dengan begitu, diharapkan dapat menghasilkan produk Cafeasy sesuai fungsionalitas yang dibutuhkan bagi pengguna dan berkontribusi terhadap pemahaman aplikasi yang terintegrasi Google Spreadsheet, Midtrans Payment Gateway, dan Google APIs.

II. KAJIAN TEORI

A. Scrum

Scrum adalah framewok untuk pengembangan perangkat lunak inkremental dan iteratif dalam mengelola pengembangan produk. Metode scrum menantang asumsi "pendekatan tradisional dan berurutan" untuk pengembangan produk, dan mendorong suatu tim untuk mengatur sendiri kolaborasi online dan komunikasi tatap muka antara anggota tim secara disiplin dalam proyek [3]. Prinsip utama scrum adalah fleksibilitas dalam perubahan pada requirement selama proses produksi, hal ini tidak dapat dengan mudah diatasi dengan pendekatan tradisional karenanya metodologi scrum digunakan[3]. Dalam pengembangan menggunakan scrum terdapat beberapa proses yang dilalui seperti product backlog, sprint planning, sprint backlog & restrospective, dan testing.

B. NodeJS

NodeJS adalah framework *open-source* yang menggunakan javascript sebagai runtime digunakan untuk aplikasi serverside [4]. Sistem manajemen paket Node, Node Package Manager atau npm, dibundel dengan instalasi Node yang memberikan akses ke ratusan ribu paket Node yang dapat digunakan oleh pengembang di seluruh dunia dan memiliki ekosistem *open-source library* terbesar di dunia[5]. Arsitektur NodeJS digerakkan oleh *event-based* dan menggunakan *thread* tunggal dalam menjalankan aplikasi. Tidak ada paralelisme aplikasi karena berjalan pada *thread* tunggal, yang berarti semua permintaan dieksekusi oleh *thread* itu[6]. Dengan demikian, fungsi kompleks dan pemblokiran permintaan I/O (*intput/output*) dilakukan di dalam *thread event,* menyebabkan aplikasi tidak dapat memproses permintaan lainnya. Untuk mengatasi hal tersebut, NodeJS menggunakan I/O asinkron dengan pola yang sama untuk memblokir operasi I/O dalam permintaan seperti ke database atau sumber lain[6]. Dengan arsitektur NodeJS yang memiliki proses tersebut, memudahkan pengembangan NodeJS untuk menangani *error* dan penggunaan sumber daya sistem yang lebih efisien.

C. ExpressJS

ExpressJS adalah library pada node.js yang digunakan untuk perutean dengan beberapa metode untuk membantu operasi CRUD (Create, Read, Update, Delete) seperti put, get, post, dan delete request[4]. ExpressJS dapat digunakan untuk perutean API dan memungkinkan menjembatani semua middleware untuk menangani request, sehingga sangat fleksibel digunakan[5].

Express dapat digunakan untuk API di serverside, file statis server ke pengguna, membatasi akses ke aset dengan integrasi otentikasi, mengendalikan eror, dan menambahkan paket middleware apapun yang akan memperluas fungsionalitas aplikasi web sesuai kebutuhan [5].Fungsi penting dalam aplikasi web adalah sistem penyimpanan data. Express tidak membatasi dalam pengintegrasian database untuk aplikasi web, sehingga pengembang memiliki fleksibilitas untuk memilih database apapun baik RDBMS maupun NoSQL [5].

D. Google Sheets API

Google Sheets API adalah antarmuka RESTful yang dapat membaca dan memanipulasi data spreadsheet dengan berbagai fungsi, termasuk membuat spreadsheet, membaca dan menulis nilai sel, memperbarui format spreadsheet dan mengelola spreadsheet yang ditautkan[7]. API ini juga sudah memiliki dokumentasi yang lengkap untuk bahasa pemrograman Java, JavaScript, dan Python[7]. Google Sheets API memungkinkan data disimpan dan diproses secara *real time* serta penyajian data yang dapat diformat sedemikian rupa. Ini memudahkan pengguna untuk memonitoring data yang ada[8]. Google Sheets API menawarkat tingkat keamanan yang cukup tinggi berkat otorisasi OAuth Google, yang memungkinkannya untuk memberikan akses ke spreadsheet sesuai kebutuhan dengan membuat sebuah fungsi skrip dan juga ringan[9].

E. Midtrans

Midtrans adalah payment gateway yang memiliki fitur untuk mempermudah melakukan pengujian pembayaran dan payment gateway bertindak sebagai jembatan antara pemilik website dengan institusi keuangan dalam melakukan proses transaksi[10]. Midtrans Payment Gateway adalah layanan transaksi secara *online* dengan berbagai metode pembayaran dan mengesahkan informasi pada transaksi sesuai kebijakan yang telah diatur oleh provider[10].

F. Github

GitHub adalah layanan hosting *online* untuk pengembangan perangkat lunak dan kontrol versi menggunakan Git. Git menawarkan kontrol versi terdesentralisasi serta kontrol akses, pelacakan bug, fitur pemrograman, manajemen tugas, integrasi berkelanjutan, dan wiki untuk setiap proyek[11].

GitHub adalah situs hosting kode kolaboratif yang dibangun di atas dari sistem kontrol versi git. GitHub memperkenalkan model "fork & pull" di mana pengembang membuat sendiri salinan repositori dan mengirimkan permintaan *pull* saat ingin menarik perubahan mereka ke dalam cabang utama. Selain hosting kode, kode kolaboratif tinjauan, dan pelacakan masalah terintegrasi, GitHub memiliki fitur sosial terintegrasi[11].

G. Docker

Docker adalah platform yang menyediakan kemampuan untuk mengotomatisasi, mengirim, dan mengemas aplikasi ketika telah diterapkan pada docker *container*. Desain docker bertujuan untuk memberikan lingkungan aplikasi yang telah diterapkan menjadi lebih cepat dan ringan[12]. Dengan menggunakan Docker, aplikasi yang telah diterapkan atau *deployed* dapat berjalan dengan konsisten di lingkungan atau perangkat lain tanpa perlu menghadapi perbedaan lingkungan aplikasi yang dapat menimbulkan masalah pada aplikasi saat dijalankan[12].

H. Postman

Postman adalah perangkat lunak yang dijelaskan di situs webnya sebagai platform API yang dapat digunakan untuk membuat dan menyebarkan API. Selain itu, Postman dapat digunakan untuk menguji API Postman dan API lainnya[13]. Aplikasi ini memungkinkan pengembang untuk membuat, membagikan, menguji, dan mendokumentasikan API. Selain itu, Postman dapat membuat berbagai skenario pengujian yang memungkinkan dalam memperluas cakupan *source code* aplikasi[14].

I. API Testing

API *testing* melibatkan perangkat lunak khusus yang mengirimkan pesan ke *encpoint* API yang ditentukan dalam antarmuka pengguna, menangkap *output*, dan mencatat serta menganalisis responsnya. *Platform* yang digunakan untuk menguji API harus dilakukan secara terprogram untuk menghasilkan pesan dengan berbagai kombinasi parameter input[15]. Eksekusi pengujian memerlukan pengiriman nilai yang dinamis ke berbagai parameter input dalam skenario pengujian API. Dalam eksekusi pengujian memiliki waktu eksekusi dari setiap *endpoint* API dalam *log debug* atau jejak *debug* eksekusi aliran pesan. Penting untuk melihat waktu eksekusi karena dapat membantu mengidentifikasi potensi hambatan kinerja di awal pengujian dan mengurangi upaya yang diperlukan untuk memecahkan masalah latensi API nanti di pengujian kinerja[15].

III. METODE

A. Metodologi Scrum

Dalam pengembangan aplikasi Cafeasy menggunakan metodologi atau *software development life cycle* Scrum, yang diimplementasi pada *platform* Trello board untuk setiap sprint. Setiap sprint dilakukan dalam waktu kurang lebih 2 minggu dengan diawali *sprint planning* dan diakhiri *sprint retrospective*. *Report* proses dalam pengerjaan tiap *sprint* atau *daily sprint* dilakukan pada aplikasi MS Team. Pada Tabel 1. (Jadwal Scrum), merupakan waktu pelaksanaan sprint pada pengembangan aplikasi dengan metode Scrum.

TABEL 1
(Jadwal Scrum)

Sprint	Bulan	Waktu Pelaksanaan	Goals
1	Februari	± 2 minggu	Customer dapat melakukan scan qr dan membuka website yang menampilkan menu utama Cafeasy
2	Maret	± 2 minggu	Customer dapat melihat detail menu, memesan menu, riwayat pesanan dan edit pesanan
3	April	± 2 minggu	Customer dapat melihat menu bantuan, status pembayaran, status menu
4	Mei	± 2 minggu	Customer dapat melakukan pemesanan, pembayaran, status pembayaran dan melihat status menu
5	Juni	± 2 minggu	Menyelesaikan Web App Cafeasy (customer side). Admin Dapat Melakukan login, sign up, mengelola data menu dan mengelola inventori
6	Juni	± 2 minggu	Admin dapat melakukan sign up, sign in, CRUD pada sub menu Data menu, kategori, dan baner pada admin cafeasy web
7	Juli	± 2 minggu	Deployment aplikasi Cafeasy

Dalam implementasi backend aplikasi Cafeasy menyesuaikan atau merujuk pada *product backlog*. *Product backlog* yang dikembangkan akan menjadi bentuk *endpoint API* untuk diakses melalui frontend aplikasi Cafeasy. Adapun daftar *product backlog* yang dikembangkan pada penelitian ini pada Tabel 2 (Daftar Product Backlog).

TABEL 2
(Daftar Product Backlog)

Product Backlog ID	Product Backlog	Pemetaan pada Arsitektur Backend
PM-01	Sebagai pengguna, saya ingin melakukan sign in dengan menggunakan akun google saya, sehingga saya dapat melakukan pemesanan.	Dapat dilihat pada. • View: menampilkan login Google • Controller: menghubungkan pada Google APIs
PM-11	Sebagai pelanggan cafe, saya ingin melakukan pembayaran melalui bank transfer atau E-Wallet, sehingga saya tidak perlu melakukan pembayaran di kasir.	Dapat dilihat pada. • View: menampilkan pop up Snap JS Midtrans • Controller: menghubungkan pada Midtrans Payment Gateway • Model: parameter input transaksi • Database: menyimpan data transaksi
PM-14	Sebagai pelanggan cafe, saya ingin melihat status pembayaran menggunakan Bank Transfer / E-Wallet dari pesanan yang saya lakukan, sehingga saya dapat memastikan bahwa	Dapat dilihat pada , • View: menampilkan status pembayaran, • Controller: menghubungkan pada Midtrans • Model: parameter order id

	pesanan yang saya lakukan sudah terbayar ataupun belum.	
KF-12	Sebagai pemilik dan karyawan cafe, saya ingin mencetak pembukuan dari transaksi per-hari ke dalam google sheets, sehingga saya dapat menyimpannya.	Dapat dilihat pada , • View: menampilkan Google Spreadsheet • Controller: menghubungkan pada Google Sheet API • Model: input parameter Google Spreadsheet • Database: Google Spreadsheet

Daftar *product backlog* merupakan hasil dari perancangan fungsionalitas aplikasi Cafeasy yang akan dibangun dalam bentuk *product backlog*. Daftar *product backlog* tersebut menjadi acuan dalam pengembangan aplikasi Cafeasy. Dalam implementasi *product backlog* pada sisi backend aplikasi Cafeasy, dibangun sebuah API agar aplikasi dapat berjalan sesuai fungsionalitasnya.

B. Sistem Backend

Dalam perancangan sistem *backend* aplikasi Cafeasy menggunakan beberapa *tools* untuk membangun aplikasi, seperti pada Tabel 3(Tech Stack). Penggunaan MVC (*Model, View, Controller*) mudah untuk diimplementasikan bersamaan dengan *framework* yang akan dibangun NodeJS dan ExpressJS karena berbasis *routing*.

TABEL 3
(Tech Stack)

Tools	**Name**
Architecture pattern	MVC
Programming language	NodeJS
Versioning code	Github
Service API Framework	ExpressJS
Third Party APIs	Midtrans Payment Gateway Google APIs Google Sheets API
Database	Google Sheets
Service Product Deploy	Docker

Aplikasi ini membutuhkan metode pemabayaran secara *online* dengan memanfaatkan teknologi dari Midtrans sebagai *payment gateway*. Transaksi yang telah dilakukan akan dicetak menjadi sebuah pembukuan suatu *cafe* yang akan tersimpan pada *database* (Google Spreadsheet). Dalam masa pengembangan aplikasi dibutuhkan *version control* dan teknologi yang dapat membantu tim bekerja sama. Pemanfaatan Github sebagai media kolaborasi tim pengembang.

C. Google APIs

Dalam implementasi aplikasi memiliki fitur login yang dimana *user* dapat memilih untuk *login* menggunakan akun google atau nama *user* saat akan menggunakan aplikasi. Dibutuhkan konfigurasi Google Cloud Console untuk mendapatkan *credentials* saat akan menggunakan servis ini.

Dalam melakukan konfigurasi dibutuhkan host url aplikasi dari aplikasi Cafeasy. Sehingga dalam melakukan *redirecting* url aplikasi akan berfungsi saat menggunakan fitur login akun Google. Implementasi code dalam aplikasi menggunakan ExpressJS dan PassportJS untuk membuat service atau API yang akan diintegrasi pada frontend aplikasi. Pemanfaatan ExpressJS pada aplikasi untuk membuat RESTful API sebagai penghubung frontend aplikasi dan Google APIs. Sedangkan PassportJS dimanfaatkan untuk menyimpan session dan autentikasi pada Google Account, sehingga aplikasi dapat menggunakan akun Google yang telah tersimpan pada perangkat user.

D. Google Sheets API

Selain Google Sign In, diperlukan konfigurasi service account untuk menggunakan API Google Sheets. Dalam konfigurasi ini akan diberikan client id, client secret dan email serta key untuk digunakan saat membuat service pada backend aplikasi atau yang biasa disebut *credentials*. *Credential* digunakan dalam menghubungkan aplikasi pada API service Google Sheet untuk mendapatkan akses dalam pemanfaatan teknologinya.

Service atau API yang akan dibuat pada backend aplikasi menggunakan *third party APIs* dengan *scope https://www.googleapis.com/auth/spreadsheets* yang dimana *scope* tersebut dapat memberikan akses POST, GET, CREATE, dan UPDATE pada Google Spreadsheet. Google spreadsheet membutuhkan tambahan konfigurasi yang dibutuhkan sesuai format pembukuan cafe dan memberikan akses pada email service account. Aplikasi akan terhubung dengan spreadsheet yang telah dihubungkan melalui service backend dan dapat mengelola data pada spreadsheet tersebut

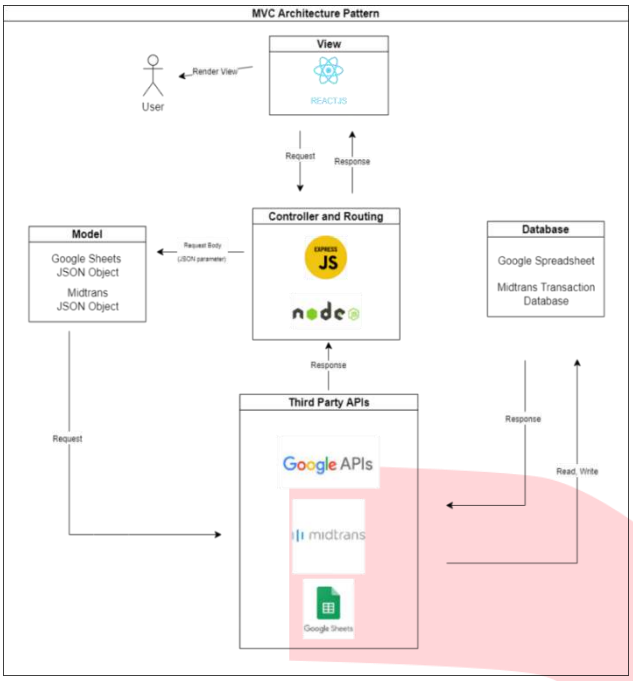
E. Midtrans Payment Gateway

Dalam pengimplementasian pembayaran secara *online* menggunakan Midtrans Payment Gateway. Dengan mendaftarkan akun *developer* untuk mendapatkan dan menggunakan API keys yang diberikan setelah membuat akun pada Midtrans. Midtrans memiliki dua tipe yaitu *production* dan *sandbox* untuk membedakan aplikasi dalam tahap pengembangan atau telah diproduksi.

Dalam tahap pengembangan digunakan tipe *sandbox* untuk melakukan *testing payment* dari servis API Midtrans. Servis *payment online* menggunakan *library* yang dimiliki *Node Package Manager* (NPM) yaitu *midtrans-client* agar dapat melakukan simulasi *payment*. Dengan pemanfaatan teknologi *snap* pada midtrans dapat melakukan simulasi pembayaran.

F. Arsitektur Backend

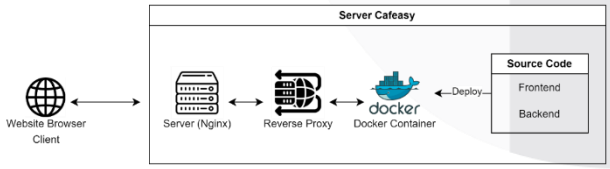
Arsitektur dalam pengembangan aplikasi backend Cafeasy menggunakan MVC (Model, View, Controller). Alur berjalannya aplikasi backend Cafeasy dimulai dari *user* melakukan request dengan fungsi CRUD pada website Cafeasy. ReactJS merupakan view dari sistem Cafeasy. Request tersebut akan berupa JSON sehingga server akan berjalan lebih cepat. Penggunaan ExpressJS dan NodeJS sebagai controller untuk melakukan request ke Google Sheets API, Midtrans Payment Gateway, dan Google APIs. *Request body (JSON parameter)* layaknya seperti data model yang telah ditentukan oleh *third party api* sebagai parameter untuk melakukan *request* ke *database*. Adapun alur dari MVC *architecture pattern* diilustrasikan pada Gambar 1 (MVC Architecture Pattern).



GAMBAR 1
(MVC Architecture Pattern)

G. Server Arsitektur

Dalam produksi aplikasi Cafeasy agar dapat digunakan pada berbagai perangkat sehingga tidak menimbulkan masalah ataupun *error*, produk Cafeasy *deployed* pada Docker Container. Docker Container menyimpan *website resource* yang terdiri dari backend dan frontend aplikasi Cafeasy yang telah diterapkan sebagai *container*. Dengan menggunakan Docker Container aplikasi dapat dijalankan secara otomatis yang akan ditangkap oleh server untuk setiap proses aplikasi. *Reverse proxy* berperan sebagai server yang berada di depan dalam meneruskan permintaan *user* ke *website resource*. Implementasi *reverse proxy* dapat meningkatkan keamanan, kinerja, dan keandalan server Cafeasy. Sedangkan dalam berlajannya server dikendalikan dengan nginx. Nginx berfungsi agar aplikasi dapat dikases secara online. Adapun ilustrasi berjalannya aplikasi Cafeasy dari perspektif web server disajikan pada Gambar 2(Server Arsitektur Diagram).



GAMBAR 2
(Server Arsitektur Diagram)

IV. HASIL DAN PEMBAHASAN

A. Service Endpoint API

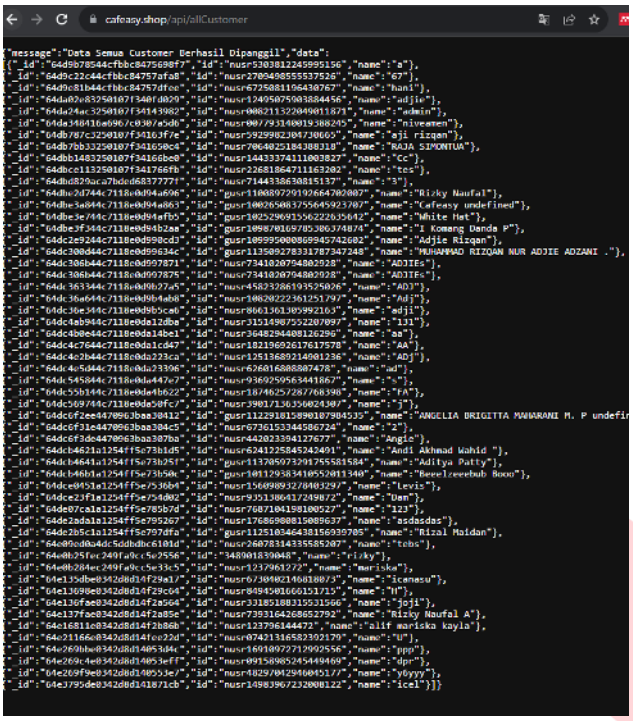
Dalam implementasi API aplikasi Cafeasy mengacu pada Tabel 2(Daftar Product Backlog) yang berisikan daftar *product backlog* aplikasi Cafeasy. Sehingga, pengembangan backend aplikasi Cafeasy mengacu pada daftar tersebut untuk membangun *service endpoint API.* Adapun daftar *service endpoint API* yang telah dibangun pada Tabel 4(Service Endpoint API).

TABEL 4
(Service Endpoint API)

No	Product Backlog ID	Method	Service Endpoint API	Deskripsi
1	PM-01	GET	/auth/google/callback	Mendapatkan akun google list yang tersimpan pada device user
2		GET	/auth/login	Melakukan *authentication* login Google Account
3		GET	/login/success	Melakukan redirect ke halaman dashboard jika login berhasil
4		GET	/login/failed	Melakukan redirect ke halaman login jika login gagal
5		GET	/auth/logout	Melakukan logout google akun
6	PM-11	POST	/midtransPayment	Membuat token transaksi untuk melakukan *payment*
7	PM-14	GET	/getTransactionStatus/:idOrder	Mendapatkan status transaksi berdarasarkan order id
8	KF-12	GET	/getSpreadsheet	Mendapatkan data dari google spreadsheet
9		POST	/writeSpreadsheet	Melakukan inputan sesuai data yang diinput ke Google Spreadsheet
10		POST	/createSpreadsheet	Membuat sheet baru pada Google Spreadsheet

Pada Tabel 4 (Service Endpoint API) merupakan daftar *service endpoint API* sebagai bentuk implementasi dari *product backlog*. *Service endpoint API* dibangun pada aplikasi Cafeasy agar memiliki fungsionalitas yang sesuai *product backlog*. *Service endpoint API* tersebut sebagai *controller* aplikasi yang dibangun menggunakan ExpressJS untuk menghubungkan pada *third party APIs*. Frontend aplikasi Cafeasy mengirim permintaan pada *service endpoint* dan *response* dari *third party APIs* akan dikirim ke frontend aplikasi untuk ditampilkan pada pengguna. *Service endpoint API* yang berjenis *method* GET akan mengirim data permintaan pengguna sesuai fungsinya. Sedangkan *method* POST membutuhkan input dari pengguna untuk mengirimkan data yang diinginkan oleh pengguna.

API Url backend aplikasi Cafeasy dapat diakses dari mana saja dengan perangkat apapun dikarenakan *resource* backend aplikasi telah disimpan pada Docker Container dan dijalankan pada server VPS Google Cloud Platform. Format data yang dikirim pada frontend Cafeasy berupa JSON sebagai *response* dari permintaan *user*. Dapat dilihat pada Gambar 3 (API Url Cafeasy Customer) dan Gambar 4 (API Url Cafeasy Admin) merupakan *sample* akses API pada aplikasi Cafeasy.



GAMBAR 3
(API Url Cafeasy Customer)

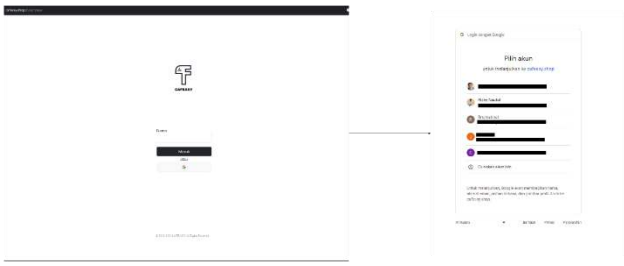
GAMBAR 4
(API Url Cafeasy Admin)

B. Login Akun Google

Fungsionalitas login akun Google pada aplikasi Cafeasy diimplementasi pada endpoint Api /auth/google/callback. Pada Gambar 5 (Akses Endpoint API /auth/google/callback) merupakan akses endpoint API untuk mendapatkan data akun Google yang tersimpan pada browser pengguna untuk melakukan login. Response pada endpoint API tersebut berupa tampilan dari Google yang didapat pada Google APIs.

GAMBAR 5 (Akses Endpoint API /auth/google/callback)

Saat *endpont API* mendapatkan *request* dari frontend aplikasi, maka akan memunculkan *pop up list* Google *account* untuk *login*. List Google Account tersebut didapatkan dari akun yang tersimpan pada *browser* Chrome. Adapun ilustrasi dari penggunaan *endpoint API* ketika digunakan oleh *user* pada Gambar 6 (Login Google Account).



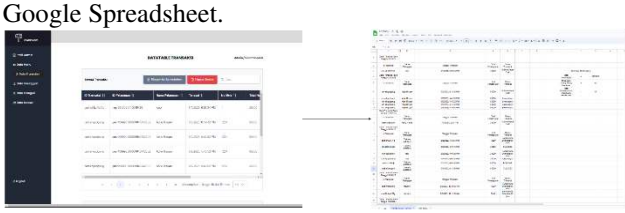
GAMBAR 6
(Login Google Account)

C. Pembukuan Otomatis Google Sheets

Dalam implementasi aplikasi Cafeasy untuk melakukan pengolahan data pembukuan pada Google Sheet memanfaatkan Google APIs. ExpressJS sebagai penghubung Google APIs salah satunya Google Sheet API yang diimplementasikan menjadi endpoint API. Endpoint API akan diakses melalui frontend Cafeasy dan mengirimkan data dengan format JSON. Pemanfaatan teknologi tersebut dapat menghasilkan fitur pembukuan secara otomatis untuk aplikasi Cafeasy. Adapun sample penggunaan *endpoint API* untuk melakukan pembukuan secara otomatis pada Gambar 7 (Sample Endpoint API /writeSpreadsheet).

GAMBAR 7
(Sample Endpoint API /writeSpreadsheet)

Dapat dilihat pada Gambar 7 (Sample Endpoint API /writeSpreadsheet) merupakan akses dari backend endpoint API aplikasi Cafeasy https://admin.cafeasy.shop/api/writeSpreadsheet dari *platform* Postman untuk melakukan pembukuan pada Google Sheet API. Pada Gambar 8 (Pembukuan Otomatis pada Cafeasy) merupakan hasil dari endpoint API /writeSpreadsheet dengan data transaksi yang tersimpan pada aplikasi Cafeasy dan tersimpan pada Google Spreadsheet sebagai data pembukuan. Dengan begitu, pengguna hanya perlu meng-klik tombol “export ke spreadsheet” untuk melakukan pembukuan secara otomatis dan tersimpan pada Google Spreadsheet.

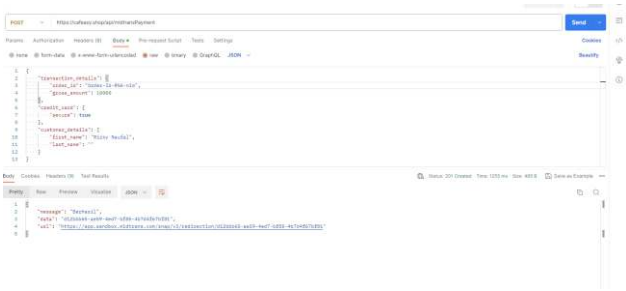


GAMBAR 8
(Pembukuan Otomatis pada Cafeasy)

D. Pembayaran Online

Dalam melakukan transaksi aplikasi menggunakan *endpoint API* /midtransPayment. Untuk mencoba API tersebut menggunakan *platform* Postman dengan parameter

input data pada Postman didapat dari keranjang pelanggan saat melakukan pemesanan pada frontend Cafeasy. Adapun gambaran penggunaan *endpoint API* di *platform* Postman pada Gambar 9 (Akses Endpoint API Midtrans Payment).



GAMBAR 9
(Akses Endpoint API Midtrans Payment)

Data *response* dari *endpoint* API ini terdapat url yang digunakan sebagai link pembayaran dengan *pop up* Snap Midtrans. SnapJS Midtrans akan menampilkan berbagai metode pembayaran. Ketika pengguna selesai melakukan pembayaran, transaksi tersebut akan masuk ke database Midtrans dan uang pengguna akan tersimpan pada Midtrans. Midtrans memiliki layanan *withdraw* pada dashboard aplikasinya untuk pengguna melakukan penarikan uang dari transaksi pada Cafe mereka. *Pop up* Snap Midtrans dan ilustrasi selesai pembayaran dapat dilihat pada Gambar 10 (Midtrans Payment Cafeasy).

GAMBAR 10
(Midtrans Payment Cafeasy)

E. API Testing

Testing aplikasi Cafeasy pada bagian backend yang menggunakan *third apis party* memakai metode API testing. *Test case* dibuat sebagai *variable* yang menentukan servis *endpoint* telah berjalan sesuai dengan fungsional yang diinginkan. *Test Case* dilakukan untuk menjadi parameter keberhasilan dari suatu *endpoint* dengan menginputkan data yang diinginkan dengan *actual output* akan sama dengan *expected output.*

Dalam melakukan pengujian menggunakan *platform* Postman untuk mengakses *endpoint* API dan melampirkan hasil test yang dilakukan. *Endpoint* API yang digunakan pada *test case* menggunakan *service endpoint* API yang telah dijalankan oleh server Cafeasy. Sehingga, akses *endpoint* API dapat dilakukan secara *online* yang membutuhkan koneksi internet. Adapun pada Tabel 5 (Daftar Test Case) merupakan daftar *test case* yang dibuat berdasarkan *product backlog* dan telah diimplementasi dalam bentuk *service endpoint* API.

TABEL 5
(Daftar Test Case)

No	Test Case ID	Test Case Description	Test Case Endpoint API	Test Case Endpoint API Input dan params
1	TCs - 1	API dapat mengirim metode login menggunakan Google Account.	https://cafeasy.shop/api/auth/google/callback	-
2	TCs-2	API dapat mengirim data logged Google Account	https://cafeasy.shop/api/auth/login	-
3	TCs-3	API dapat melakukan logout Google Account dan mengirim data akun yang dilogout	https://cafeasy.shop/api/auth/logout	-
4	TCs-4	API dapat memilih metode pembayaran menggunakan e-wallet, dan e-banking serta melakukan payment.	https://cafeasy.shop/api/midtransPayment	Input : { "transaction_details":{ "order_id":"Order-Id-RNA-1302194098", "gross_amount": 10000 }, "credit_card": { "secure": **true** }, "customer_details": { "first_name": "Rizky Naufal", "last_name": "" } }
5	TCs-5	API dapat mengirim status transaksi.	https://cafeasy.shop/api/getTransactionStatus/:idOrder	Params:{ idOrder : Order-Id-RNA-1302194098 }
6	TCs-6	API dapat melakukan *read* data dengan Google Spreadsheet	https://admin.cafeasy.shop/api/getSpreadsheet	-
7	TCs-7	API dapat melakukan input data dengan Google Spreadsheet	https://admin.cafeasy.shop/api/writeSpreadsheet	Input:{8 "data": [["Transaksi"], ["Sosis", "Harga : 4000 "]]}
8	TCs-8	API dapat melakukan *create* sheets baru pada Google Spreadsheet	https://admin.cafeasy.shop/api/createNewSpreadsheet	Input: {"sheetName": "Harga Menu", "data": [["nama Menu", "harga menu"], ["Sosis", "23000"], ["Sosis", "23000"], ["Sosis", "23000"], ["Sosis", "23000"]] }

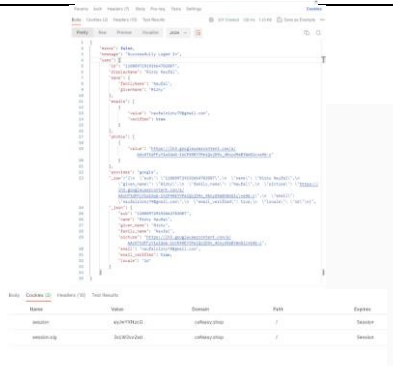
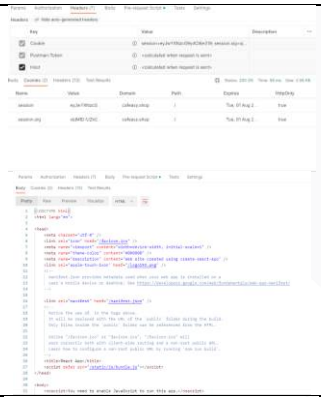
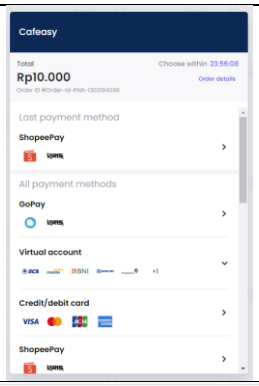
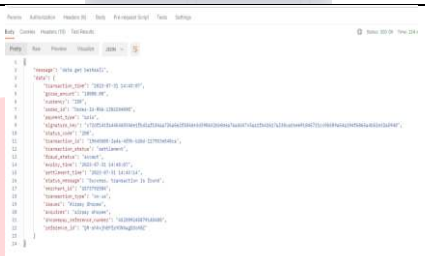
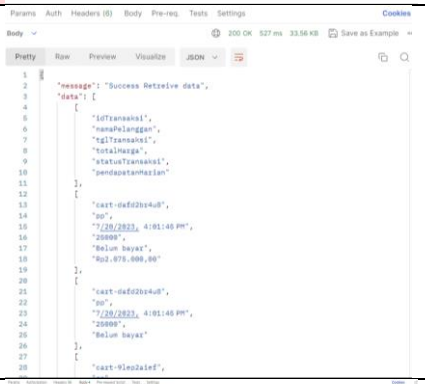
Dikarenakan *test case* dibuat berdasarkan *endpoint API* yang telah dibangun berdasar *product backlog* maka pengujian yang dilakukan adalah menguji fungsionalitas yang telah dikembangkan pada aplikasi Cafeasy. Namun,

dalam pengujian API terdapat beberapa *endpoint API* yang tidak dapat dilakukan pengujian, seperti *endpoint* API /login/success, dan /login/failed. Hal ini dikarenakan, *endpoint* API tersebut mengirim *response* berupa *redirect* halaman yang sesuai dengan kondisi API.

F. Hasil API Testing

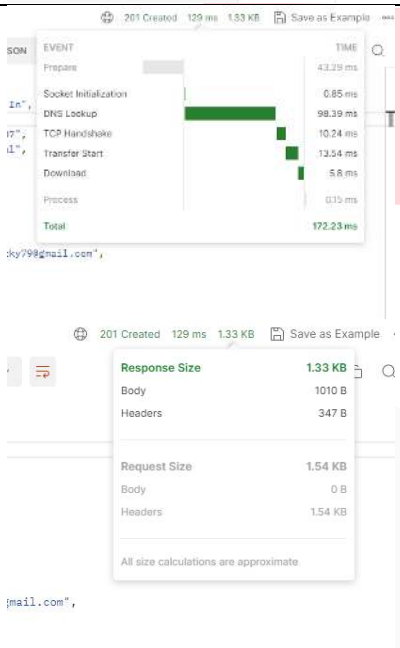
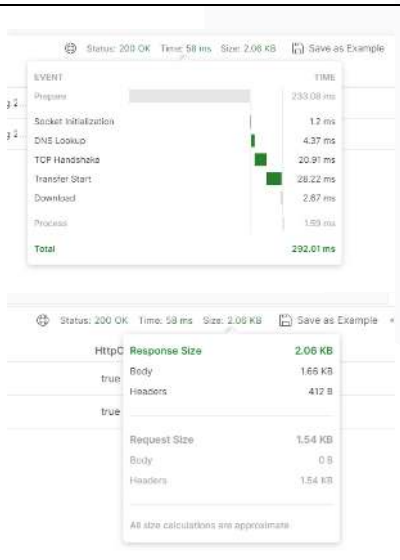
Hasil pengujian pada Postman melampirkan hasil dalam mengakses *endpoint* API untuk setiap daftar Test Case. Adapun hasil pengujian daftar *test case* pada Tabel 5 (Daftar Test Case) disajikan pada Tabel 6 (Test Case Result) dan hasil *response time* disajikan pada Tabel 7 (Test Case Result Response Time dan Size).

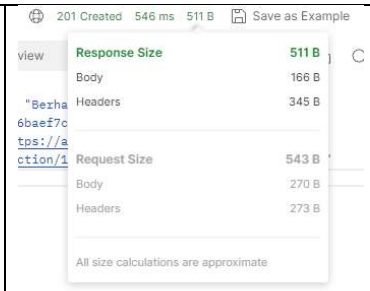
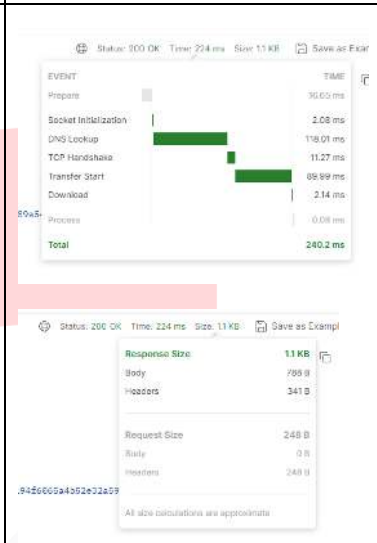
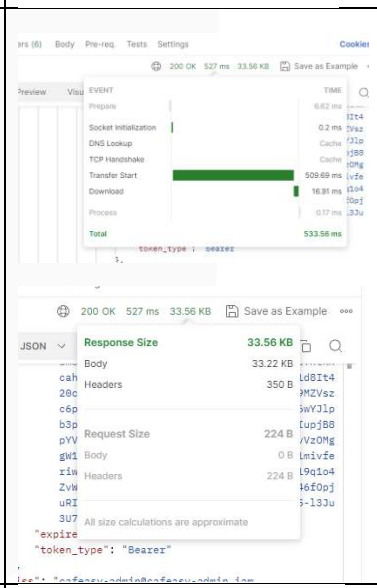
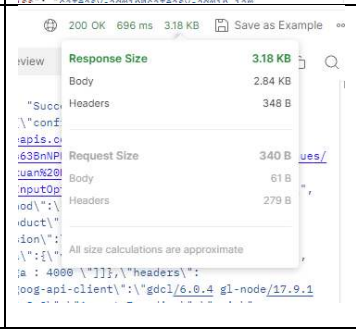
TABEL 6
(Test Case Result)

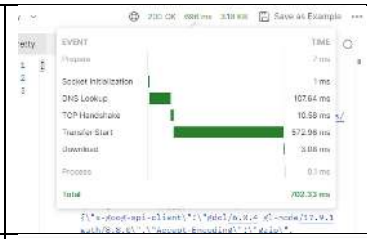
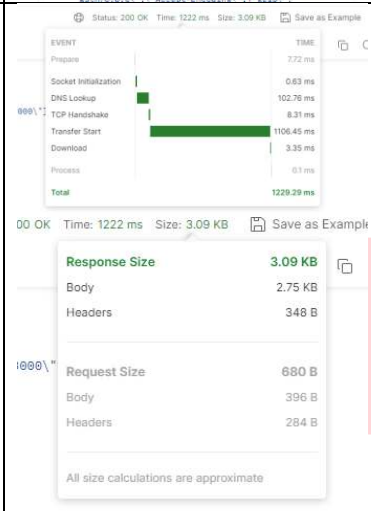
Test Case ID	Actual Output	Expected Output
TCs-1		Menampilkan halaman Google untuk login akun
TCs-2		Menampilkan data user Google Account setelah login
TCs-3		Dapat melakukan logout
TCs-4	 Tampilan url payment :	Mengirim url untuk melakukan payment dan detail payment sesuai inputan
		
TCs-5		Mendapatkan status pembayaran sesuai parameter *endpoint API*
TCs-6		Mendapatkan data dari Spreadsheet
TCs-7		Melakukan input data dengan Google Spreadsheet sesuai inputan *endpoint* API dan menampilkan data inputan tersebut.
TCs-8		Melakukan *create* sheets baru pada Google Spreadsheet dan input data pada sheets baru sesuai inputan *endpoint* API.

TABEL 7
(Test Case Result Response Time dan Size)

Test Case ID	Test Case Response Time dan Size
TCs-1	

	
TCs-2	
TCs-3	
TCs-4	

	
TCs-5	
TCs-6	
TCs-7	

	
TCs-8	

G. Analisis Hasil

TABEL 8
(Hasil Analisis Test)

No	Test Case ID	Analisis	Test Case Result
1	TCs-1	Pada **Error! Reference source not found.** baris TCs-1 menggunakan *endpoint* API /auth/google/callback untuk mengirim halaman *login* akun Google. Dalam implementasi API *testing* menggunakan *platform* Postman, *output* yang didapat berupa halaman *login* akun Google. Dengan begitu, *actual output* pada API ini telah sesuai dengan *expected output* pada TCs-1 untuk menampilkan halaman *login* akun Google. Meskipun ukuran data *response* mencapai 103 kilobytes, API ini memiliki kinerja yang sangat cepat pada angka 175 milidetik. Berdasarkan analisis tersebut, *endpoint* API /auth/google/callback dapat dinilai sangat baik dalam sisi performa waktu dan hasil pengujian sesuai dengan *expected result* TCs-1.	Berdasarkan hasil analisis, TCs-1 berstatus pass.
2	TCs-2	Berdasarkan **Error! Reference source not found.** baris TCs-2 menggunakan *endpoint* API /auth/login untuk melakukan otentikasi *login* Google Account. Pada pengujian API ini melakukan *login* pada web Cafeasy dengan akun yang tersedia pada *browser tester*. Ketika pengujian *endpoint* API /auth/login, *actual output* yang dihasilkan telah sesuai dengan akun Google yang digunakan saat melakukan *login*. Dengan demikian, *actual output* pada API ini telah sesuai dengan *expected output* pada TCs-2 untuk menampilkan data akun Google yang telah *login*. Pada sisi performa waktu, API ini memiliki *response time* 129 milidetik dengan ukuran data 1,33 kilobytes. Hal tersebut menandakan bahwa API ini memiliki performa waktu yang sangat cepat dan ukuran data menyesuaikan data pada akun Google. Berdasarkan analisis tersebut, *endpoint* API /auth/login telah memenuhi *expected result* TCs-2 dengan performa waktu yang sangat cepat saat mengakses API /auth/login.	Berdasarkan hasil analisis, TCs-2 berstatus pass.
3	TCs-3	Berdasarkan **Error! Reference source not found.** baris TCs-3 menggunakan *endpoint* API /auth/logout untuk melakukan *logout* Google Account. Pada pengujian API ini melakukan *logout* akun pada akun TCs-2. *Expected result* yang telah terpenuhi adalah *output* API dapat melakukan logout akun yang telah *login* sebelumnya pada TCs-2. Pada sisi performa waktu, API /auth/logout dapat menyelesaikan prosesnya dengan waktu 58 milidetik untuk ukuran data *response* 2,06 kilobytes.	Berdasarkan hasil analisis, TCs-3 berstatus pass.
4	TCs-4	Berdasarkan **Error! Reference source not found.** baris TCs-4 menunjukkan bahwa *actual result* memenuhi *expected result* sebagai *variable* keberhasilan TCs-4. Hal ini dikarenakan API yang dibuat menggunakan ExpressJS berhasil terhubung dengan Midtrans Payment Gateway. *Response time* yang dibutuhkan pada TCs-4 dibawah 1 milidetik 546 milidetik dengan ukuran data 511 *bytes*. Hal tersebut menunjukan servis *endpoint* memiliki *response* yang cepat dalam melakukan fungsinya. Kinerja ExpressJS untuk terhubung dengan Midtrans Payment Gateway dan server yang dibangun membuat API memiliki performa yang baik. Namun, koneksi internet yang tidak stabil dapat mempengaruhi kinerja dari API. Dengan demikian, *endpoint* API /midtransPayment memiliki kinerja yang cepat dan bekerja sesuai fungsi yang diharapkan.	Berdasarkan hasil analisis, TCs-4 berstatus pass.
5	TCs-5	Pada **Error! Reference source not found.** baris TCs-5 dalam kolom input menggunakan data “order_id” pada TCs-4 sebagai parameter untuk mengakses *endpoint* API status transaksi. Berdasarkan hasil pengujian tersebut, *output* API tidak hanya memiliki *response* data untuk status transaksi tetapi detail transaksi pada TCs-4. Data keluaran API sangat tepat dan dengan *response time* 224 milidetik untuk ukuran data 1,1 kilobytes. Hal tersebut menandakan, bahwa API bekerja dengan sangat baik yang mampu mendapatkan data dengan tepat dan performa yang sangat cepat. Dengan demikian, hasil pengujian sesuai dengan *expected result* TCs-5 dan memiliki performa yang sangat cepat.	Berdasarkan hasil analisis, TCs-5 berstatus pass
6	TCs-6	Berdasarkan pada **Error! Reference source not found.** baris TCs-6 menampilkan detail data dari Google Spreadsheet Cafeasy. *Endpoint* API /getSpreadsheet data menampilkan kinerja waktu 527 milidetik dan ukuran data 33 kilobytes dalam prosesnya. Proses kinerja API *get* spreadsheet memiliki performa yang cepat. Namun, kinerja waktu dan	Berdasarkan hasil analisis, TCs-6 berstatus pass.

		ukuran data dapat bertambah ketika data pada Google Spreadsheet bertambah. Dengan demikian, *endpoint* API pada TCs-6 bekerja sesuai dengan *expected result* yang menandakan hasil pengujian sesuai kriteria dengan performa yang cepat.	
7	TCs-7	Berdasarkan pada **Error! Reference source not found.** baris TCs-7 menggunakan *endpoint* API */writeSpreadsheet* untuk menambahkan data baru pada Sheet. Hasil pengujian mendapatkan *response time* 696 milidetik dengan ukuran data 3.18 kilobytes. Performa API cukup tinggi untuk ukuran data 3.18 kilobytes. Hal ini dikarenakan *response* data akan terkirim ke pengguna saat Google Sheet API telah selesai memasukkan data yang diinput. Meskipun demikian, hasil pengujian sesuai dengan *expected result* TCs-7 yang menandakan API telah sesuai kriteria.	Berdasarkan hasil analisis, TCs-7 berstatus pass.
8	TCs-8	Berdasarkan pada **Error! Reference source not found.** baris TCs-8 menggunakan *endpoint* API */createNewSheet* untuk membuat Sheet baru dan *insert* data baru pada Sheet tersebut. Hasil pengujian menunjukan *reponse time* lebih dari 1 detik (1222ms) untuk ukuran data 3.09 kilobytes. Penyebab *response time* tinggi pada proses Google Spreadsheet membuat Sheet baru secara otomatis. Meskipun demikian, hasil pengujian telah sesuai dengan *expeted result* TCs-8.	Berdasarkan hasil analisis, TCs-8 berstatus pass.

V. KESIMPULAN

A. Kesimpulan

Berdasarkan hasil proses pengembangan dan pengujian backend aplikasi Cafeasy dapat disimpulkan bahwasannya, terdapat dua komponen utama dalam pembentukan backend aplikasi yaitu ExpressJS sebagai framework API dan NodeJS sebagai bahasa pemrograman. Penggunaan dua komponen tersebut sangat fleksibel dalam membangun API untuk aplikasi Cafeasy dengan konsep MVC architecture pattern, terutama sebagai controller yang dapat menghubungkan third party APIs. API yang dibangun pada aplikasi terhubung dengan Midtrans untuk melakukan pemesanan secara online dan Google Sheet API untuk pembukuan secara otomatis. Hal tersebut dibuktikan pada pengujian bahwa semua test case berjalan sesuai dengan harapan fungsionalitas aplikasi Cafeasy.

Response time endpoint API memiliki hasil yang berbeda dikarenakan faktor kecepatan koneksi internet, ukuran data, third party APIs yang terhubung dan metode API. Terdapat satu API yang memiliki response time diatas satu detik dikarenakan faktor metode POST pada Google Sheet API membutuhkan waktu dalam proses input data. Pengujian keseluruhan API masih memiliki celah dan kekurangan yang dapat mengurangi fungsionalitas aplikasi, membuat error dalam penggunaan API dan membuat crash aplikasi Cafeasy.

B. Saran

Berdasarkan kesimpulan dari hasil proses pengembangan dan pengujian backend aplikasi Cafeasy, disarankan untuk menggunakan metode pengujian lain dan membuat error handler pada API. Hal ini dikarenakan pengujian yang dilakukan tidak memuat kondisi API dalam keadaan bad request dan masih terdapat API yang tidak dapat diuji menggunakan API test. Dengan demikian, dapat membuat aplikasi Cafeasy menjadi lebih stabil dalam menangani error, dan crash pada setiap penggunaan endpoint API backend aplikasi Cafeasy.

REFERENSI

[1] R. Shriwas, N. Patel, A. Bherani, A. Khajone, and M. Raut, “Touchscreen based ordering system for restaurants,” in *2014 International Conference on Communication and Signal Processing*, IEEE, 2014, pp. 1021–1024.

[2] N. Muthia, H. Amalia, A. Puspita, and A. F. Lestari, “Rancang Bangun Sistem Informasi Akuntansi Penjualan Dengan Model Waterfall Berbasis Java Desktop,” *JITK (Jurnal Ilmu Pengetahuan Dan Teknologi Komputer)*, vol. 5, no. 1, pp. 15–22, 2019.

[3] S. Sachdeva, “Scrum Methodology,” *Internationa Journal Of Engineering and Computer Science, URL: https://www. academia. edu/26010951/Scrum_Methodology (2.9. 2019)*, 2016.

[4] N. D. Naidu *et al.*, “E-Commerce web Application by using MERN Technology,” *International Journal for Modern Trends in Science and Technology*, vol. 7, pp. 1–5, 2021.

[5] S. Hoque, *Full-Stack React Projects: Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node. js*. Packt Publishing Ltd, 2020.

[6] L. P. Chitra and R. Satapathy, “Performance comparison and evaluation of Node. js and traditional web server (IIS),” in *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, IEEE, 2017, pp. 1–4.

[7] “Google Sheets API Overview | Google Developers.” https://developers.google.com/sheets/api/guides/concepts (accessed Nov. 28, 2022).

[8] V. A. Kusuma, M. I. A. Putra, and S. S. Suprapto, “Sistem Monitoring Stok dan Penjualan Minuman pada Vending Machine berbasis Internet of Things (IoT) Menggunakan Google Sheets dan Kodular,” *Jurnal Sistim Informasi dan Teknologi*, pp. 94–98, Aug. 2022, doi: 10.37034/jsisfotek.v4i3.136.

[9] D. Hodges W and K. Schlottmann, “The Code4Lib Journal – Reporting from the Archives: Better Archival Migration Outcomes with Python and the Google Sheets API,” *code{4}lib Journal*, Nov. 05, 2019. https://journal.code4lib.org/articles/14871?ref=https://githubhelp.com (accessed Nov. 28, 2022).

[10] P. S. Alfian and L. Magdalena, “Penerapan Payment Gateway pada Aplikasi Marketplace Waroeng Mahasiswa Menggunakan Midtrans,” *J. Inform. Univ. Pamulang*, vol. 5, no. 3, pp. 387–393, 2020.

[11] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, “The promises and perils of mining github,” in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 92–101.

[12] B. B. Rad, H. J. Bhatti, and M. Ahmadi, “An introduction to docker and analysis of its performance,” *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 3, p. 228, 2017.

[13] J. Ranta, “Testing AWS hosted Restful APIs with Postman,” 2023.

[14] M. Shershneu and A. Oskin, “Postman Platform for API Development in the Mobile Application" Musicians of Russia",” 2020.

[15] B. De and B. De, “API testing strategy,” *API Management: An Architect’s Guide to Developing and Managing APIs for Your Organization*, pp. 153–164, 2017.