LAPORAN TUGAS BESAR PEMODELAN DATA

Aplikasi Layanan *On Demand*

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah Pemodelan Data Praktikum.

Disusun oleh:

Rizka Auliarahmi	191524057
Dzakira Fabillah	191524040
Putri Syalwa Abdullah	191524054

PROGRAM STUDI D4 TEKNIK INFORMATIKA

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2021

DAFTAR ISI

LAPORAN TUGAS BESAR PEMODELAN DATA **0**

DAFTAR ISI **1**

DAFTAR GAMBAR **4**

DAFTAR TABEL **5**

1. PENDAHULUAN **7**

1.1 Rumusan Masalah 7

1.2 Ruang Lingkup 7

2. ANALISIS PROBLEM DOMAIN **11**

2.1 Deskripsi Masalah 11

2.2 Business Rules 12

2.3 Workload 15

2.4 Analisis Query 17

2.5 Analisis Laporan (Agregasi) 28

3. PERANCANGAN **33**

3.1 Collection Relationship Diagram (CRD) 33

3.2 Kamus Data 33

4. IMPLEMENTASI DAN EKSPERIMEN **43**

4.1 Implementasi 43

4.1.1 Implementasi Fitur Admin 48

4.1.2 Implementasi Fitur Customer 66

4.1.3 Implementasi Fitur Driver 75

4.2 Eksperimen 81

4.2.1 Rancangan eksperimen 82

4.2.2 Hasil Eksperimen 82

5. KESIMPULAN DAN LESSON-LEARNED **89**

5.1 Kesimpulan 89

5.2 Lesson-Learned 90

5.3 Kendala dan Solusi 90

6. REFERENSI **92**

DAFTAR GAMBAR

Gambar 1 BPMN Pemenuhan Pesanan Layanan 12

Gambar 2 Collection Diagram Relationship 33

DAFTAR TABEL

Tabel 1 Ruang Lingkup 6

Tabel 2 Aturan Bisnis 11

Tabel 3 List Operasi 14

Tabel 4 Hasil Analisis Kuantitatif Operasi 15

Tabel 5 Hasil Analisis Query 16

Tabel 6 Hasil Analisis Aggregasi: Total Order Per Type of Services 27

Tabel 7 Hasil Analisis Aggregasi: Total Order this month 28

Tabel 8 Hasil Analisis Aggregasi: Total Order this week 28

Tabel 9 Hasil Analisis Aggregasi: Total Order today 29

Tabel 10 Hasil Analisis Aggregasi: Total New Customer this month 30

Tabel 11 Hasil Analisis Aggregasi: Total order per month in one year 30

Tabel 12 Collection DataCustomer 32

Tabel 13 Collection DataDriver 34

Tabel 14 Collection AcitivityHistory 37

Tabel 15 Collection DataAdmin 40

Tabel 16 Collection Feedback 41

Tabel 17 Fitur Log In 42

Tabel 18 Fitur Sign Up 44

Tabel 19 Fitur Reports 47

Tabel 20 Fitur Profile Admin 50

Tabel 21 Fitur Read & Search Activity 52

Tabel 22 FiturRead Activity tertentu 54

Tabel 23 Fitur Read Feedback 55

Tabel 24 Fitur Kelola Data Admin 56

Tabel 25 Fitur Add New Admin 57

Tabel 26 Fitur Daftar DaraDriver 58

Tabel 27 Fitur Kelola Data Driver 60

Tabel 28 Fitur Daftar DataCustomer 62

Tabel 29 Fitur Daftar DaraDriver 64

Tabel 30 Fitur Read & Update customer 65

Tabel 31 Fitur Create Oderr 67
Tabel 32 Fitur Read Oder 71
Tabel 33 Fitur Review 72
Tabel 34 Fitur Customer Get All Order History + Per Category 73
Tabel 35 Fitur Read & Update New Order Status 74
Tabel 36 Fitur Read Review After Order Finished 76
Tabel 37 Fitur List New Order 77
Tabel 38 Fitur Driver Get All Order History 78
Tabel 39 Fitur Kelola Data Diri Driver 79
Tabel 40 Eksperimen Add Customer 81
Tabel 41 Load Driver 82
Tabel 42 Create Order 83
Tabel 43 Load Order Detail 84
Tabel 44 Update Activity Status 85
Tabel 45 Load Activity (Search by Category) 86
Tabel 46 Kendala dan Solusi 89

1. PENDAHULUAN

1.1 Rumusan Masalah

Kompleksitas atau masalah-masalah yang terdapat dalam membangun suatu model data pada NoSQL (*document-based*) terhadap studi kasus yang kami pilih yaitu Aplikasi Layanan *On Demand*, diantaranya:

- Bagaimana merancang dan membangun database yang dapat mengolah data yang banyak serta kompleks?
- Bagaimana merancang dan membangun document database yang dapat membantu pengolahan data menjadi lebih efektif dan mempercepat proses query?
- Bagaimana membangun database yang dapat menyimpan data dan memproses data sesuai dengan kebutuhan?
- Bagaimana membangun database yang dapat memenuhi kebutuhan sistem dan user?
- Bagaimana membangun Aplikasi Layanan *On Demand* sehingga memberikan banyak keuntungan dari sisi manajemen data?
- Bagaimana membangun sistem basis data dengan kebutuhan jumlah field yang berbeda antar *document* dalam satu *collection*?
- Bagaimana membangun User Interface Aplikasi Layanan *On Demand* yang interaktif?

1.2 Ruang Lingkup

Pada aplikasi ini, data yang ada pada database aplikasi digunakkan oleh 3 jenis pengguna, diantaranya yaitu admin, *driver,* dan *customer.* Masing-masing pengguna memiliki akses dan batasannya terhadap data. *Business Process* yang terjadi pada setiap *user*/Ruang lingkup pada aplikasi ini diantaranya adalah :

Tabel 1 Ruang Lingkup

No.	Nama Proses Bisnis	Nama Fitur Aplikasi	Stakeholder (User Aplikasi)	Kode Fitur
1.	Pengelolaan Data Driver	Input Data Driver Baru **(Create)**	Driver	FD1
		Melihat Seluruh Data Driver **(Read)**	Admin	FD2

		Melakukan operasi pencarian (*search*) pada data driver berdasarkan NIK. **(Read)**	Admin	FD3
		Delete driver. **(Delete)**	Admin	FD4
		Update Profile Driver **(Update)**	Admin, Driver	FD5
		Verifikasi driver **(Update)**	Admin	FD8
		Read Data Driver Tertentu **(Read)**	Admin, Driver, Customer (terbatas)	FD9
		Update active status **(update)**	Driver	FD10
		Login sebagai Driver	Driver	FD11
		Melihat jumlah seluruh driver, car driver, motorcycle driver, dan active driver (**Read)**	Admin	FD12
		Admin bokir driver (**update)**	Admin	FD13
2.	Pengelolaan Data Customer	**Create** Account (Input Data Customer).	Customer	FC1
		Update Data Diri Customer	Customer	FC2
		Melihat seluruh data customer. **(Read)**	Admin	FC3
		Melakukan operasi pencarian terhadap data customer berdasarkan Nama. **(Read)**	Admin	FC4
		Melihat data customer yang melakukan pemesanan layanan pada *driver.* **(Read)**	Driver	FC5
		Melihat data customer tertentu **(Read)**	Admin	FC6
		Admin dapat mengubah status customer (blokir) **(update)**	Admin	FC7
		Melihat Profil akun. **(Read)**	Customer	FC8
		Login sebagai Customer	Customer	FC9
		Menghapus Data Customer **(Delete)**	Admin	FC10

3.	Pengelolaan Data Riwayat Aktivitas	Melakukan Pemesanan/ Input Activity baru. **(Create)**	Customer, Driver	FR1
		Mengetahui biaya yang harus dibayar untuk suatu layanan.	Customer, Driver	FR2
		Pemberian feedback layanan driver oleh seorang customer. **(Create)**	Customer	FR3
		Melihat riwayat aktivitas *customer* dalam menggunakan layanan aplikasi. (**Create)**	Customer	FR4
		Melihat seluruh riwayat kegiatan (baik layanan transportasi, maupun layanan antar barang). **(Read)**	Admin	FR5
		Melihat riwayat aktivitas dalam jangka waktu tertentu (bisa dalam satu hari tertentu). **(Read)**	Admin	FR6
		Melihat riwayat pelayanan yang dilakukan oleh driver tertentu. **(Read)**	Driver	FR7
		Melihat report **(Read)**	Admin	FR8
		Melihat feedback yang diberikan customer setelah order selesai **(Read)**	Driver	FR9
		Melihat data aktivitas tertentu.	Driver, Customer, Admin	FR10
		Melihat activity berdasarkan category tertentu	Admin	FR11
		Melihat feedback tertentu	Admin	FR12
		Update status order/aktivitas	Driver	FR13

4.	Pengelolaan Data Admin	Membuat akun admin baru. **(Create)**	Admin	FA1
		Melihat seluruh data admin **(Read)**	Admin	FA2
		Melihat akun dirinya **(Read)**	Admin	FA3
		Mengganti password sendiri **(Update)**	Admin	FA4
		Menghapus akun admin **(Delete)**	Admin	FA5
		Login sebagai Admin	Admin	FA6

2. ANALISIS *PROBLEM DOMAIN*

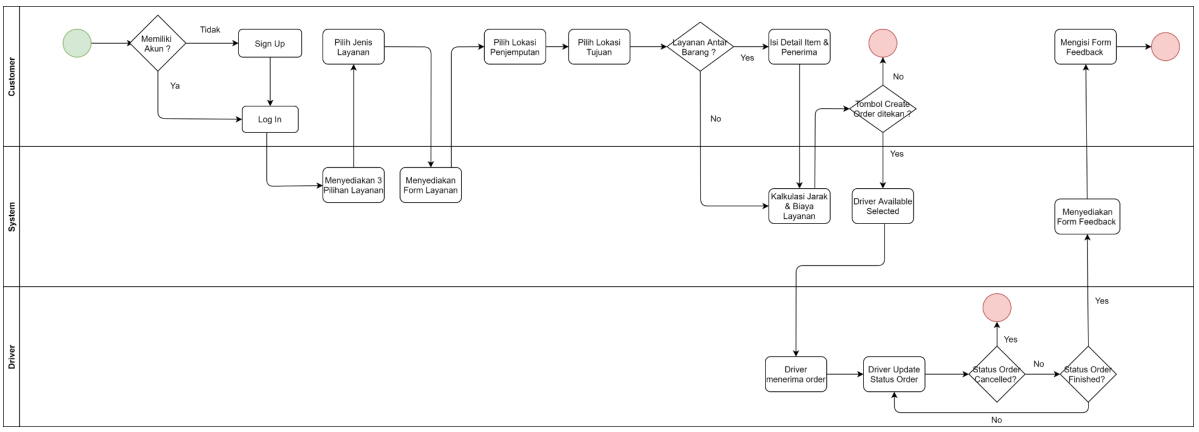
2.1 Deskripsi Masalah

- Aplikasi Layanan *On Demand* ini menyediakan 3 layanan, diantaranya layanan transportasi motor, transportasi mobil, dan layanan antar barang. Setiap layanan yang dipesan oleh *customer* akan dipenuhi oleh seorang *driver* yang terdaftar ke aplikasi dan sudah di verifikasi oleh admin.
- Seorang *driver* perlu mendaftarkan dirinya melalui aplikasi dengan mengisi data diri dan melampirkan berkas berkas yang diperlukan, baru kemudian akan diperiksa dan diverifikasi oleh admin. Driver harus memiliki minimal salah satu kendaraan di antara motor dan mobil, yang akan digunakan untuk memenuhi pesanan *customer.* Setiap document pada Data Driver dapat memiliki beberapa field yang berbeda (schemaless), misalnya adalah data kendaraan motor tidak memiliki field *capacity* karena kapasitasnya sudah pasti satu orang. Ketika driver sudah resign, maka status nya akan berubah sehingga tidak akan menerima pesanan lagi.
- Seorang customer harus login menggunakan akunnya sebelum melakukan pesanan layanan. Jika belum memiliki account, maka *customer* harus sign up terlebih dahulu dengan mendaftarkan email dan password akun.
- Ketika melakukan order layanan, *customer* akan memilih layanan yang akan dipesan terlebih dahulu. Kemudian *customer* harus menentukan titik penjemputan dan titik tujuan, jika layanan yang dipilih adalah layanan antar barang, maka data terkait barang (seperti jenis barang & berat barang) dan penerima (seperti nama & nomor telepon penerima) juga perlu diisi, jika tidak, maka field tersebut tidak akan ada pada document (schemaless). Setiap layanan memiliki parameter yang berbeda-beda dalam menentukan biaya layanan. Aturan biaya di spesifikasikan di bagian *Business Rules.*
- Aplikasi akan menentukan *driver* yang akan memenuhi pesanan *customer*. Driver yang dapat memenuhi pesanan merupakan driver yang memiliki tipe kendaraan yang sesuai dan statusnya *active* (dapat memenuhi pesanan. Driver harus melakukan konfirmasi terlebih dahulu terhadap pesanan yang masuk. Kemudian customer akan menerima data terkait identitas driver beserta kendaraannya.
- Setelah pesanan terpenuhi, *customer* dapat memberikan penilaian berupa rating dengan rentang 1 - 5, dan juga *review* bagi pelayanan yang diberikan *driver*. Jika rata-rata rating *driver* kurang dari 2.5, maka *driver* akan terblokir oleh aplikasi. Jika ingin diaktifkan kembali, *driver* perlu mengajukan/melakukan konfirmasi ke perusahaan, baru kemudian bisa diaktifkan kembali oleh admin.

- *Customer* dapat melihat riwayat aktivitas/data terkait layanan yang pernah dipesan melalui aplikasi. *Driver* juga dapat melihat riwayat aktivitas/pesanan-pesanan yang dipenuhi oleh *driver* tersebut.
- Admin memiliki tingkat akses read and write terhadap semua data. Admin dapat membuat akun admin lainnya. Admin juga dapat melihat data-data terkait laporan, seperti jumlah order dan jumlah akun baru dalam jangka waktu tertentu.

Berikut merupakan BPMN (*Business Process Model*) dari fungsi utama aplikasi yaitu pemenuhan pesanan layanan.

Gambar 1 BPMN Pemenuhan Pesanan Layanan



2.2 *Business Rules*

Fitur-fitur yang tidak memiliki aturan tertentu terkait rumus/algoritma/metode tertentu, maka fiturnya tidak akan dimunculkan pada tabel berikut.

Tabel 2 Aturan Bisnis

No.	Nama Fitur Aplikasi	Aturan
1.	Input Data Driver Baru (Create)	Ketika driver mendaftar, data driver yang diinputkan harus lengkap (semua field yang diperlukan/wajib ada terisi).
		Driver yang didaftarkan harus dipastikan memiliki kendaraan (motor atau mobil) untuk melakukan pelayanan.

		Driver dapat terblokir secara otomatis apabila total rating yang diterima ojek online dibawah 2/5.
		Setelah daftar, status driver otomatis belum terverifikasi dan belum dapat melakukan/menerima order. Untuk melakukan layanan admin harus memverifikasi driver terlebih dahulu.
	Update Data Driver	Data driver hanya dapat di submit satu kali. Setelah itu driver hanya dapat mengubah status aktifnya.
	Verifikasi driver **(Update)**	Admin dapat memverifikasi driver setelah melihat data dan berkas.
	Read Data Driver Tertentu **(Read)**	Admin dapat melihat seluruh data driver, dan driver dapat melihat data miliknya sendiri.
	Update active status **(update)**	Driver dapat mengubah status active nya. Jika bernilai *true,* maka driver dapat melakukan pelayanan, dan jika *false* artinya driver sedang tidak dapat menerima layanan.
2.	Create Account (Input Data Customer).	Customer mendaftarkan dirinya sebagai pengguna aplikasi (sign up) dengan mendaftarkan email dan password account, dan selanjutnya hanya perlu log-in.
		No HP yang sama tidak bisa dimiliki oleh lebih dari satu account.
	Admin dapat mengubah status customer (blokir) **(update)**	Admin dapat memblokir seorang *customer* jika misalkan mendapatkan laporan dari pihak lain yang menyebabkan seorang *customer* harus di blokir.
3.	Melakukan Pemesanan/ Input Activity baru. **(Create)**	Untuk setiap layanan, jarak maksimumnya adalah 30 km.
		Layanan yang tersedia : - Antar Barang - Transportasi Motor

		- Transportasi Mobil
		Pada pengiriman barang, berat barang maksimal yang bisa diantar adalah 15 kg.
		Layanan antar-barang hanya dilakukan oleh driver yang menggunakan pengendara motor.
		Activity status merupakan status dari order, misalnya : “new order” “cancelled” “finished” Khusus Transportasi : “on the way to sender” “picked up” “on the way to destination” Khusus Barang : “on the way to sender” “item(s) picked up” “on the way to recipient”
		Customer yang diblokir tidak dapat membuat pesanan.
		Driver yang sudah diblokir tidak dapat menerima pesanan sampai admin meng buka blokir tersebut.
	Mengetahui biaya yang harus dibayar untuk suatu layanan.	● Biaya pengantaran menggunakan motor Rp.2500,00 per kilometer. ● Biaya pengantaran menggunakan mobil Rp.3500,00 per kilometer. ● Biaya pengiriman Rp.2800,00 per kilometer.
	Pemberian rating/ penilaian layanan driver oleh seorang customer. (**Create)**	Dapat dilakukan satu kali oleh customer setiap setelah menerima suatu layanan.

	Melihat riwayat pelayanan yang telah dilakukan oleh driver tertentu. **(Read)**	Masing-masing driver hanya dapat melihat riwayat aktivitas dirinya sendiri.
	Melihat report **(Read)**	Admin dapat melihat order report dalam satu hari, satu bulan, maupun satu tahun. Yang dapat admin lihat diantaranya adalah jumlah order masing-masing layanan, dan juga jumlah user baru dalam jangka waktu tersebut.
4.	Membuat akun admin baru. **(Create)**	Admin akan didaftarkan/dibuatkan akunnya oleh admin lain berdasarkan pengetahuan perusahaan.

2.3 *Workload*

Dalam analisis workload ini, kami mengambil data dari aplikasi-aplikasi yang juga menyediakan layanan On-Demand seperti Gojek, Grab, dll. Jumlah driver 1.7 juta driver

- Jumlah pengguna transportasi online berjumlah 21.7 juta orang
- Aplikasi melayani kurang lebih 35 pesanan per detik. 3 juta Order per hari.

Karena aplikasi kami menyediakan layanan yang lebih sedikit (3 layanan) dari aplikasi-aplikasi serupa tersebut, maka perhitungan workload dibuat setengah dari data referensi.

- Jumlah pesanan per detik pada aplikasi ini ada 18 pesanan per detik dan 1.5 juta order perhari.
- Jumlah driver 1.7 driver didapat dari aplikasi gojek yang sudah aktif beroperasi sekitar 5 tahun, sehingga diperoleh data driver baru per hari ada 1.7 juta/(365 * 5) = 931 driver. Kemudian dibagi 2 menjadi 460 driver baru per hari dan total 850 ribu driver pada aplikasi.
- Jumlah pengguna transportasi online 21.7 juta orang didapat dari aplikasi grab yang sudah aktif beroperasi sekitar 6 tahun, sehingga diperoleh data customer baru per hari ada 21.7/(365*6) = ±10000. Kemudian, dibagi dua menjadi 5000 orang

2.3.1 List Operasi

Tabel 3 List Operasi

No	Query	Operation	Description
1	Add Customer	Write	Customer Sign Up.
2	Load Customer	Read	Customer Log In aplikasi.
3	Add Driver	Write	Driver Sign Up.

4	Load Driver	Read	Driver Log In aplikasi dan customer melihat profil driver yang menangani pesanannya.
5	Create Order	Write	Customer melakukan order salah satu layanan yang tersedia. Pemilihan driver dilakukan pada query ini, secara random mengambil driver yang memenuhi syarat active_status true, dan kendaraannya sesuai dengan jenis pesanan.
6	Load Order Detail	Read	Customer maupun Driver membuka informasi terkait order. Menampilkan data customer, driver, lokasi awal, lokasi akhir, dan detail barang apabila merupakan layanan antar barang.
7	Update Activity Status	Write	Mengupdate status order dari mulai driver dalam perjalanan, hingga order selesai. Dalam satu order hingga selesai, terjadi 3-4 kali perubahan status order (berbeda tergantung jenis layanan).
8	Add feedback	Write	Customer memberikan feedback setiap setelah order selesai (bersifat opsional bagi customer).
9	Update Active Status Driver	Write	Driver mengubah status aktifnya setiap ada perubahan state dari dapat memenuhi pesanan ke tidak dapat memenuhi pesanan, dan sebaliknya.
10	Load Activity (Search by Category)	Read	Melakukan pencarian terhadap aktivitas berdasarkan category.

2.3.2 Analisis Kuantitatif Operasi

Berdasarkan survey yang dilakukan oleh Littledata terhadap 3.973 site, *response time* rata rata adalah **530 ms**. Berdasarkan penelitian tersebut, *response time* yang kurang dari 317 ms termasuk kedalam 20% situs dengan *response time* yang baik.

Berikut merupakan beberapa query yang paling sering terjadi dalam aplikasi dan hasil analisa kuantifikasi yang kami lakukan.

Tabel 4 Hasil Analisis Kuantitatif Operasi

No	Query	Quantification
1	Add Customer	5000 write/day < 500 ms
2	Load Customer	20 read/second < 300 ms

3	Add Driver	460 write/day < 500 ms
4	Load Driver	25 read/second < 300 ms
5	Create Order	18 write/second < 300 ms
6	Load Order Detail	36 read/second < 300 ms
7	Update Activity Status	54 write/second < 300 ms
8	Add feedback	15 write/second < 300 ms
9	Update Active Status Driver	5 write/second < 500 ms
10	Load Activity (Search by Category)	<2000 ms

2.4 Analisis *Query*

Tabel 5 Hasil Analisis Query

No	Operasi	Kode Fitur	Collection	Query

1	Create data driver baru	FD1	DataDriver	`if(req.body.driver_password == req.body.password_verification){`
				`let new_profile = {`
				`nik: "",`
				`sim_no: "",`
				`name: req.body.profile.name,`
				`profpict: "",`
				`phone_number: "",`
				`gender: "",`
				`birth_of_date: "",`
				`};`
				
				`let new_vehicle_details = {`
				`transportation_type: "",`
				`plat_number: "",`
				`capacity: 0,`
				`merk_and_type: "",`
				`stnk_no_registrasion: "",`

| | | | | ```
        };

        let newAddress = {
          province: "",
          city: "",
          sub_district: "",
          zip_code: "",
          street: ""

        };

        let newDocuments = {
          skck: "",
          ktp: "",
          sim: "",
          stnk:"",

        };

        let newDriver = {
          driver_email:
req.body.driver_email,
          driver_password:
req.body.driver_password,
          profile: new_profile,
          vehicle_details:
new_vehicle_details,
          address: newAddress,
          documents:newDocuments,
          rating: null,
          submitted: false,
          active_status: false,
          blocked: false
        };

db_connect.collection("DataDriver").insertOne
(newDriver);
          res.status(201).json({
          message: "Succesfully inserted",
          newDriver
```  |
| 2 | Read data semua driver | FD2, FD12 | DataDriver | ```
  let mySort = {name: 1};
  db_connect
    .collection("DataDriver")
    .find({})
    .sort(mySort)
    .toArray(function (err, result) {
      if (err) throw err;
```  |

				`res.json(result);`

3	Search data driver berdasarkan NIK	FD3	DataDriver	`var myquery = { "profile.nik": (req.params.nik).toString() };`
				`db_connect`
				`.collection("DataDriver")`
				`.find(myquery)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

4	Delete Driver	FD4	DataDriver	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				
				`db_connect.collection("DataDriver").deleteOne (myquery, function (err, obj) {`
				`if (err) throw err;`
				`res.status(201).json({obj, message : "Deleted Succesfully"});`

5	Update Profile Driver	FD5	DataDriver	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`let newvalues = {`
				`$set: req.body`
				`};`
				`db_connect`
				`.collection("DataDriver")`
				`.updateOne(myquery, newvalues, function (err, result) {`
				`if (err) throw err;`
				`res.status(201).json({result, message : "Updated Succesfully"});`

6	Verifikasi driver	FD8	DataDriver	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`let newvalues = {`
				`$set: {`
				`verification_status: true`
				`},`
				`};`
				`db_connect`
				`.collection("DataDriver")`
				`.updateOne(myquery, newvalues, function (err, res) {`
				`if (err) throw err;`
				`console.log("1 document updated");`

7	Read Data Driver Tertentu	FD9	DataDriver	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`db_connect`
				`.collection("DataDriver")`
				`.find(myquery)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

8	Update active status	FD10	DataDriver	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`let _active_status = req.body.active_status;`
				`let newvalues = {`
				`$set: {`
				`active_status: _active_status`
				`},`
				`};`
				`db_connect`
				`.collection("DataDriver")`
				`.updateOne(myquery, newvalues, function (err, result) {`
				`if (err) throw err;`
				`res.status(201).json({result, message : "Updated Succesfully"})`

9	Login sebagai Driver	FD11	DataDriver	`db_connect.collection("DataDriver")`
				`.find({driver_email: req.body.driver_email})`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`if(result[0].driver_password == req.body.driver_password ){`
				`res.status(201).json({`
				`message: "Succesfully login",`
				`result`

10	Admin bokir driver	FD13	DataDriver	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`let newvalues = {`
				`$set: {`
				`blocked: req.body.blocked`
				`},`
				`};`
				`db_connect`

				.collection("DataDriver")
				.updateOne(myquery, newvalues, function (err, result) {
				if (err) throw err;
				res.status(201).json({result, message : "Updated Succesfully"});

11	Create Data Customer	FC1	DataCustomer	if(req.body.customer_password == req.body.password_verification){
				console.log("verifikasi password berhasil");
				let new_account = {
				customer_email: req.body.customer_email,
				customer_password: req.body.customer_password,
				profile:{
				name: req.body.profile.name,
				prof_pic: req.body.profile.prof_pic,
				phone_number: req.body.profile.phone_number,
				gender: req.body.profile.gender,
				birth_of_date: req.body.profile.birth_of_date
				},
				signUp_date: new Date(),
				blocked: req.body.blocked
				};
				db_connect.collection("DataCustomer").insertOne(new_account);
				res.status(201).json({
				message: "Succesfully inserted",
				new_account

12	Update Data Diri Customer	FC2	DataCustomer	let id_account = { _id : new mongodb.ObjectID(req.params.id)}
				let new_values = {
				$set: {
				customer_email: req.body.customer_email,
				customer_password: req.body.customer_password,
				profile:{
				name: req.body.profile.name,
				prof_pic: req.body.profile.prof_pic,

				phone_number: req.body.profile.phone_number,
				gender: req.body.profile.gender,
				birth_of_date: new Date(req.body.profile.birth_of_date)
				}
				}
				};
				
				db_connect.collection("DataCustomer").updateOne(id_account, new_values, function (err, result){
				if (err) throw err;
				res.status(201).json({
				message: "Succesfully updated",
				result

13	Melihat seluruh data customer.	FC3	DataCustomer	let mySort = {name: 1};
				db_connect
				.collection("DataCustomer")
				.find({})
				.sort(mySort)
				.toArray(function (err, result) {
				if (err) throw err;
				res.json(result);

14	Melakukan operasi pencarian terhadap data customer berdasarkan Nama.	FC4	DataCustomer	var myquery = { "profile.name": (req.params.name).toString() };
				db_connect
				.collection("DataCustomer")
				.find(myquery)
				.toArray(function (err, result) {
				if (err) throw err;
				res.json(result);

15	Melihat data customer tertentu	FC5
		FC6
		FC8
			DataCustomer	var myquery = { _id: new mongodb.ObjectID(req.params.id) };
				db_connect
				.collection("DataCustomer")
				.find(myquery)
				.toArray(function (err, result) {
				if (err) throw err;
				res.json(result);

16	Admin dapat mengubah status customer (blokir)	FC7	DataCustomer	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`let newvalues = {`
				`$set: {`
				`blocked: req.body.blocked`
				`},`
				`};`
				`db_connect`
				`.collection("DataCustomer")`
				`.updateOne(myquery, newvalues, function (err, result) {`
				`if (err) throw err;`
				`res.status(201).json({result, message : "Updated Succesfully"});`

17	Login sebagai Customer	FC9	DataCustomer	`db_connect.collection("DataCustomer")`
				`.find({customer_email: req.body.customer_email})`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				
				`if(result[0].customer_password == req.body.customer_password ){`
				
				`res.status(201).json({`
				`message: "Succesfully login",`
				`result`

18	Menghapus Data Customer	FC10	DataCustomer	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				
				`db_connect.collection("DataCustomer").deleteOne(myquery, function (err, obj) {`
				`if (err) throw err;`
				`res.status(201).json({obj, message : "Deleted Succesfully"});`

19	Melakukan Pemesanan/ Input Activity baru.	FR1	ActivityHistory	`var id_driver = "";`
				`if(req.body.type_of_service == "transportasi motor"){`
				`var myquery = { active_status: true , verification_status: true, "vehicle_detail.transportation_type": "Motorcycle"};`
				`}else{`
				`var myquery = { active_status: true ,`

				verification_status: true,
				"vehicle_detail.transportation_type": "Car"};
				}
				
				db_connect
				.collection("DataDriver")
				.find(myquery)
				.toArray(function (err, result) {
				if (err) throw err;
				let selected = Math.floor(Math.random()
				* result.length);
				id_driver = result[selected]._id;
				
				let new_order = {
				id_driver: id_driver.toHexString(),
				id_customer: req.body.id_customer,
				date: new Date(),
				type_of_service:
				req.body.type_of_service,
				start_loc: {
				longitude:
				req.body.start_loc.longitude,
				latitude:
				req.body.start_loc.latitude
				},
				end_loc: {
				longitude:
				req.body.end_loc.longitude,
				latitude:
				req.body.end_loc.latitude
				},
				activity_status: "new order",
				price: req.body.price
				};
				
				db_connect.collection("ActivityHistory").inse
				rtOne(new_order);
				res.status(201).json({
				message: "Succesfully inserted",
				new_order

20	Pemberian feedback layanan driver oleh	FR3	Feedback	let new_review = {
				rating: req.body.rating,
				review: req.body.review
				};
				
				db_connect.collection("Feedback").insertOne(n
				ew_review);

	seorang customer			`res.status(201).json({`
				`message: "Succesfully inserted",`
				`new_review`

21	Melihat riwayat aktivitas *customer* dalam aplikasi.	FR4	ActivityHistory	`let mysort = {date: -1}`
				`let myquery = {id_customer: req.params.id, type_of_service: req.params.category };`
				`db_connect`
				`.collection("ActivityHistory")`
				`.find(myquery)`
				`.sort(mysort)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

22	Melihat seluruh riwayat kegiatan (baik layanan transportasi, maupun layanan antar barang).	FR5	ActivityHistory	`let mysort = {date: -1}`
				`db_connect`
				`.collection("ActivityHistory")`
				`.find({})`
				`.sort(mysort)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

23	Melihat riwayat aktivitas dalam jangka waktu tertentu (bisa dalam satu hari tertentu).	FR6	ActivityHistory	`let mysort = {date: -1}`
				`let myquery = { date: {$gte: new Date(req.params.date), $lt: new Date(new Date(req.params.date).setDate(new Date(req.params.date).getDate()+1))} };`
				
				`db_connect`
				`.collection("ActivityHistory")`
				`.find(myquery)`
				`.sort(mysort)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

24	Melihat riwayat pelayanan yang dilakukan oleh driver tertentu.	FR7 FR10	ActivityHistory	`let mysort = {date: -1}`
				`let myquery = { id_driver: req.params.id, activity_status: {$in: ['finished', 'cancelled']} };`
				`db_connect`
				`.collection("ActivityHistory")`
				`.find(myquery)`
				`.sort(mysort)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

25	Melihat feedback yang diberikan customer setelah order selesai	FR9 FR12	Feedback	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`db_connect`
				`.collection("Feedback")`
				`.find(myquery)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

26	Melihat activity berdasarkan category tertentu	FR11	ActivityHistory	`let myquery = { type_of_service: req.params.category };`
				`let mysort = {date: -1}`
				`db_connect`
				`.collection("ActivityHistory")`
				`.find(myquery)`
				`.sort(mysort)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

27	Update status order/aktivitas	FR13	ActivityHistory	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`let newvalues = {`
				`$set: {`
				`activity_status: req.body.activity_status`
				`},`
				`};`
				`db_connect`
				`.collection("ActivityHistory")`
				`.updateOne(myquery, newvalues, function (err, result) {`
				`if (err) throw err;`

				`res.status(201).json({result, message : "Updated Succesfully"})`

28	Membuat akun admin baru.	FA1	DataAdmin	`if(req.body.admin_password == req.body.password_verification){`
				`console.log("verif pass berhasil");`
				`let newAdm = {`
				`admin_name: req.body.admin_name,`
				`admin_email: req.body.admin_email,`
				`admin_password: req.body.admin_password,`
				`created_at: new Date()`
				`};`
				`try{`
				`//console.log(req.data);`
				
				`db_connect.collection("DataAdmin").insertOne(newAdm);`
				`res.status(201).json({`
				`message: "Succesfully inserted",`
				`newAdm`

29	Melihat seluruh data admin	FA2	DataAdmin	`let mySort = {name: 1};`
				`db_connect`
				`.collection("DataAdmin")`
				`.find({})`
				`.sort(mySort)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

30	Melihat akun dirinya	FA3	DataAdmin	`var myquery = { _id: new mongodb.ObjectID(req.params.id) };`
				`db_connect`
				`.collection("DataAdmin")`
				`.find(myquery)`
				`.toArray(function (err, result) {`
				`if (err) throw err;`
				`res.json(result);`

31	Mengganti password sendiri	FA4	DataAdmin	`let myquery = { _id: new mongodb.ObjectID(req.params.id), admin_password: req.body.admin_curr_password};`
				
				`let newvalues = {`
				`$set: {`
				`admin_name: req.body.admin_name,`

				admin_email: req.body.admin_email,
				admin_password:
				req.body.admin_new_password
				}
				};
				
				db_connect
				.collection("DataAdmin")
				.updateOne(myquery, newvalues, function
				(err, result) {
				if (err) throw err;
				res.status(201).json({
				message: "Succesfully updated",
				result

32	Menghapus akun admin	FA5	DataAdmin	var myquery = { _id: new
				mongodb.ObjectID(req.params.id) };
				
				db_connect.collection("DataAdmin").deleteOne(
				myquery, function (err, obj) {
				if (err) throw err;
				res.status(201).json({obj, message :
				"Deleted Succesfully"});

33	Login sebagai Admin	FA6	DataAdmin	db_connect.collection("DataAdmin")
				.find({admin_email:
				req.body.admin_email})
				.toArray(function (err,
				result) {
				if (err) throw err;
				
				if(result[0].admin_password ==
				req.body.admin_password){
				
				res.status(201).json({
				message:
				"Succesfully login",
				result

2.5 Analisis Laporan (Agregasi)

Tabel 6 Hasil Analisis Aggregasi: *Total Order Per Type of Services*

Nama Fungsi Agregasi	Total Order Per Type of Services
Deskripsi	Menghasilkan array, yang setiap elemennya

	berisi object nama tipe layanan, dan jumla layanan tersebut.
Script	

```
adminRoutes.route("/admin/read/report/pertype").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      { $group : { _id : '$type_of_service', totalorder : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});
```

Tabel 7 Hasil Analisis Aggregasi: *Total Order this month*

Nama Fungsi Agregasi	Total Order this month
Deskripsi	Menghasilkan jumlah order ke aplikasi dalam 1 bulan terakhir.
Script	

```
adminRoutes.route("/admin/get/totalorder/thismonth").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let order_month = { date: {$gt: new Date(new Date().setDate(new
Date().getDate() - 30))} };
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      {$match : order_month},
      { $group : { _id : 0, totalorder : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });

});
```

Tabel 8 Hasil Analisis Aggregasi: *Total Order this week*

Nama Fungsi Agregasi	Total Order this week

Deskripsi	Menghasilkan jumlah order ke aplikasi dalam 1 minggu terakhir.
Script	

```
adminRoutes.route("/admin/get/totalorder/thisweek").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let order_week = { date: {$gt: new Date(new Date().setDate(new Date().getDate()
- 7))} };
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      {$match : order_week},
      { $group : { _id : 0, totalorder : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });

});
```

Tabel 9 Hasil Analisis Aggregasi: *Total Order today*

Nama Fungsi Agregasi	Total Order today
Deskripsi	Menghasilkan jumlah order ke aplikasi dalam 1 hari (hari ini).
Script	

```
adminRoutes.route("/admin/get/totalorder/today").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let order_day = { date: {$gt: new Date(new Date().setDate(new Date().getDate()
- 1))} };
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      {$match : order_day},
      { $group : { _id : 0, totalorder : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });

});
```

Tabel 10 Hasil Analisis Aggregasi: *Total New Customer this month*

Nama Fungsi Agregasi	Total New Customer this month
Deskripsi	Menghasilkan jumlah *customer* yang sign up ke aplikasi dalam 1 bulan terakhir.
Script	

```
adminRoutes.route("/admin/get/totalnewcust/thismonth").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let this_month = { signUp_date: {$gt: new Date(new Date().setDate(new
Date().getDate() - 30))} };
  db_connect
    .collection("DataCustomer")
    .aggregate([
      {$match : this_month},
      { $group : { _id : 0, totaluser : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });

});
```

Tabel 11 Hasil Analisis Aggregasi: *Total order per month in one year*

Nama Fungsi Agregasi	Total order per month in one year
Deskripsi	Menghasilkan jumlah order per bulannya untuk satu tahun (array of object).
Script	

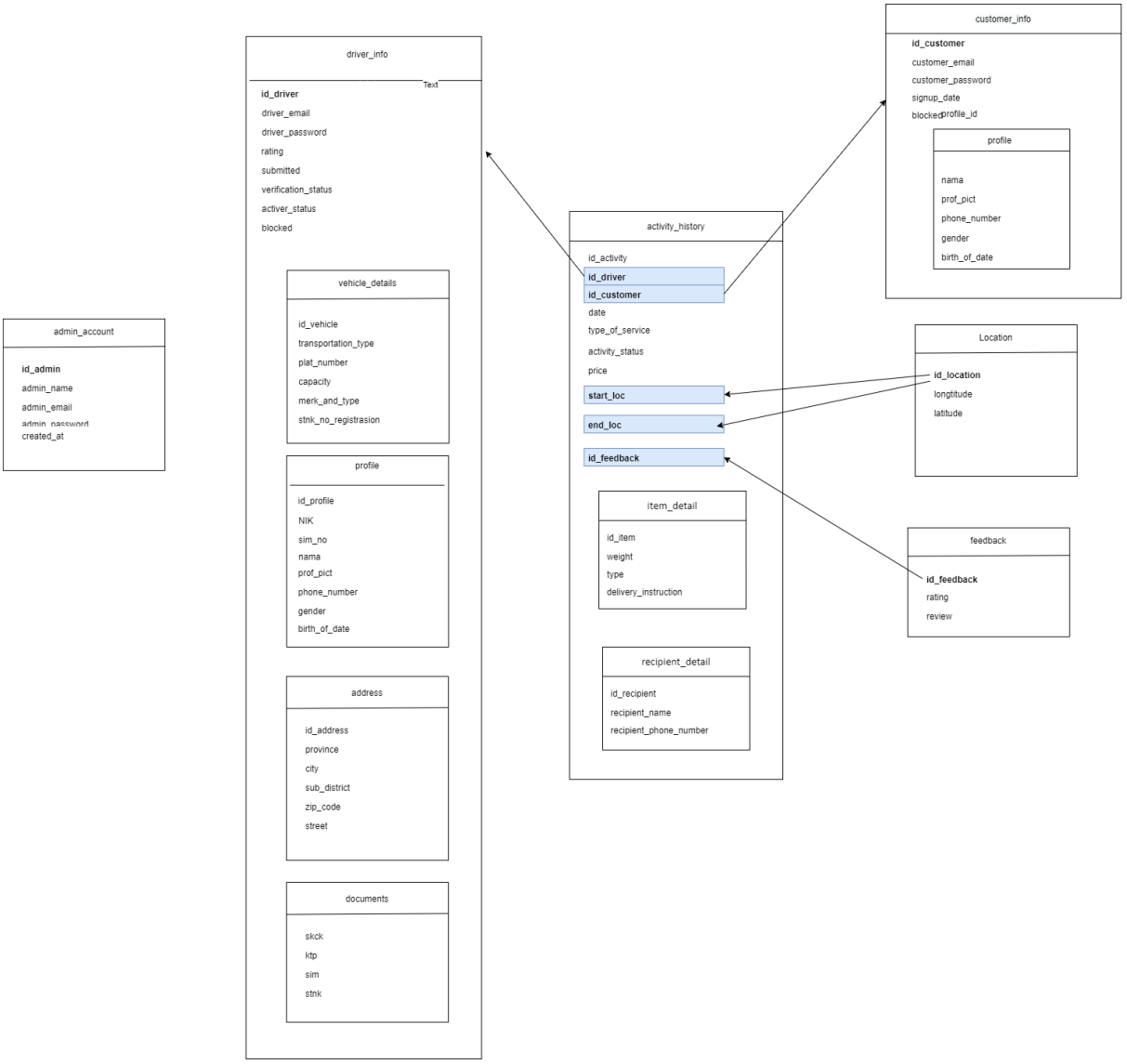
```
adminRoutes.route("/admin/get/totalorder/:year").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let year = req.params.year;
  let nextyear = (parseInt(year) + 1).toString();
  let myquery = { date: {$gte: new Date(year), $lt: new Date(new
Date(nextyear).setDate(new Date(nextyear).getDate() - 1))} };
  // console.log(new Date();
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      {$match: myquery },
      { $group : {  _id: {$substr: ['$date', 5, 2]} , totalorder : { $sum : 1 } }
}
    ])
```

```
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});
```

3. PERANCANGAN

3.1 Collection Relationship Diagram (CRD)

Gambar 2 Collection Diagram Relationship



3.2 Kamus Data

1. Collection DataCustomer

Tabel 12 Collection DataCustomer

Nama	DataCustomer
Deskripsi	Berisi document-document terkait data customer, yaitu document customer_info. Document profile embedded ke customer_info.

No.	Nama Dokumen	Nama Field	Tipe Data (BSON, object [embedded, linked])	Deskripsi
1.	customer_info	_id	ObjectID	ID document, di generate secara otomatis oleh mongodb
2.		customer_email	String	Email untuk log-in aplikasi
3.		customer_password	String	Password untuk log-in aplikasi
4.		signUp_date	Date	Tanggal customer sign-up, format yy/mm/dd
5.		blocked	boolean	Jika true maka *customer* tidak dapat melakukan pesanan
6.	profile	id_profile	ObjectID	ID dari profile
7.		name	String	Nama customer
8.		prof_pict	String	Photo profil customer
9.		phone_number	String	Nomor telepon customer
10.		gender	String	Gender customer
11.		birth_of_date	Date	Tanggal lahir customer format yy/mm/dd

Contoh Collection

```
{
"_id": ObjectID("60d01162959f90f88b6d3571"),
 customer_email: ”putri@gmail.com”,
 customer_password: ”abc123”,
 profile:{
      “name” : “Putri Syalwa”,
      “prof_pict” :
“https://www.google.com/search?q=foto+profil&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjK97ONkarwAhXDcn0KHVO0DtkQ_AUoAXoECAEQAw&biw=1366&bih=657#imgrc=zABhbT_dXLjPCM”,
      “phone_number” : “08577784468”,
      “gender”: “Laki-laki”,
      “birth_of_date”: “1998-11-15”
   },
 “signUp_date”: ”17-Jan-2021”,
 “blocked”: false
}
```

Indexing

Nama: _id_ (Default)

Deskripsi: Merupakan default index dari mongodb yang bersifat unique sehingga mencegah adanya document dengan id yang sama.

- _id

Nama: signup_date (Single Field Indexes)

Deskripsi: Untuk keperluan fitur report/laporan pelanggan bagi admin dimana admin dapat melihat jumlah customer baru dalam jangka waktu tertentu.

- signUp_date

Nama: email_customer (Unique Indexes)

Deskripsi: Email akun harus unique sehingga satu akun email hanya dapat dipakai untuk 1 akun.

- customer_email

2. Collection DataDriver

Tabel 13 Collection DataDriver

Nama		DataDriver		
Deskripsi		Berisi document document terkait data driver, yaitu document driver info. Document profile dan vehicle_details embedded kedriver_info. Address embedded ke profile.		
No.	**Nama Dokumen**	**Nama Field**	**Tipe Data (BSON, object [embedded, linked])**	**Deskripsi**
1.	`driver_info`	`_id`	ObjectID	ID document, di generate secara otomatis oleh mongodb
2.		`driver_email`	String	Email driver untuk log-in aplikasi
3.		`driver_passwo rd`	String	Password driver untuk log-in aplikasi
4.		`rating`	Float	Rating driver (maksimal 5)
5.		`submitted`	Boolean	Jika submitted bernilai true maka driver tidak dapat mengubah datanya lagi.
6.		`verification_ status`	Boolean	Jika true maka driver sudah di verifikasi oleh admin

7.		active_status	Boolean	Jika true, maka *driver* sedang berada dalam jam kerja nya
		available_status	Boolean	Jika true maka driver sedang tersedia dan dapat menerima pesanan
8.		blocked	Boolean	Jika true, maka *driver* terblokir
9.	profile	id_profile	ObjectID	ID dari profile
10.		nik	String	NIK driver
11.		sim_no	String	Nomor SIM driver
12.		name	String	Nama driver
13.		prof_pict	String	Photo profil driver
14.		phone_number	String	Nomor telepon driver
15.		gender	String	Gender *driver*
16.		birth_of_date	Date	Tanggal lahir *driver* format yy/mm/dd
17.	vehicle_details	id_vehicle	ObjectID	ID dari vehicle
18.		trasportation_type	String	Jenis kendaraan, apakah motor atau mobil
19.		plat_number	String	Plat kendaraan
20.		capacity	Integer	Kapasitas maksimal mobil
21.		merk_and_type	String	Merk mobil
22.		stnk_no_registration	String	Nomor registrasi STNK
23.	address	id_address	ObjectID	ID dari address
24.		province	String	Provinsi
25.		city	String	Kota
26.		sub_district	String	Kecamatan
27.		zip_code	String	Kode pos
28.		street	String	Nama jalan dan nomor tempat tinggal
29.	documents	skck	String	File pdf hasil scan skck driver
30.		ktp	String	File pdf hasil scan ktp driver
31.		sim	String	File pdf hasil scan sim driver
32.		stnk	String	File pdf hasil scan stnk driver

Contoh Collection


```
{
 "_id": ObjectID("60d01162959f90f88b6d3571"),
  "driver_email": "rizka@gmail.com",
  "driver_password": "asd567",
  "profile": {
      "nik" : "0989283744",
      "sim_no": "0875356787987500",
      "name" : "Rizka Aulia",
      "prof_pict" :
"https://www.google.com/search?q=foto+profil&source=lnms&tbm=isch&sa=X&ved=2a
hUKEwjK97ONkarwAhXDcn0KHVO0DtkQ_AUoAXoECAEQAw&biw=1366&bih=657#imgrc=zABhbT_d
XLjPCM",
      "phone_number" : "08987763568",
      "gender": "Perempuan",
      "birth_of_date": "1998-11-15"
   },
  "vehicle_details": {
       "trasportation_type": "Car",
       "plat_number": "B 1256 P",
       "capacity": 6,
       "merk_and_type": "Toyota",
       "stnk_no_registration": "08534245890"
  },
 "address" : {
       "province": "Jawa Barat",
       "city": "Kota Bandung",
       "sub_district": "Panyileukan",
       "zip_code": "40614",
       "street": "Jl. Panyawangan Raya No.40"
 },
  "documents":{
       "skck":"0875356787987500",
       "ktp":"3958699298585",
       "sim":"41824004838833",
       "stnk": "9595858595955"
  },
  "rating": 3,
  "submitted" : true,
  "verification_status": false,
```

```
  "active_status": false,
  "available_status": false,
  "blocked": false
}
}
```

Indexing

Nama: Single Field Indexes

Deskripsi: Untuk keperluan search dan juga update data driver, agar memudahkan pencarian dilakukan berdasarkan NIK.

- NIK

Nama: Single Field Indexes

Deskripsi: Untuk keperluan menampilkan driver yang belum di verifikasi, dan juga dalam mencari driver untuk memenuhi pesanan, maka dicari driver yang verification status nya adalah true.

- verification_status

Nama: Compound Indexes

Deskripsi: Query terhadap driver yaitu pencarian berdasarkan transportation_type dan activity_status akan dipanggil setiap kali ada order (untuk mencari driver yang dapat memenuhi pesanan *customer)*.

- active_status
- transportation_type

Nama: email_driver (Unique Indexes)

Deskripsi: Email akun harus unique sehingga satu akun email hanya dapat dipakai untuk 1 akun.

- driver_email

3. Collection ActivityHistory

Tabel 14 Collection AcitivityHistory

Nama		ActivityHistory		
Deskripsi		Berisi document-document terkait history layanan yang dipesan.		
No.	**Nama Dokumen**	**Nama Field**	**Tipe Data (BSON, object**	**Deskripsi**

			[embedded, linked])	
1.	activity_info	id_	ObjectID	ID document, di generate secara otomatis oleh mongodb
2.		id_driver	ObjectID	ID driver, di generate secara otomatis oleh mongodb
3.		id_customer	ObjectID	ID customer, di generate secara otomatis oleh mongodb
4.		id_feedback	ObjectID	ID feedback, di generate secara otomatis oleh mongodb
5.		date	Date	Tanggal dari aktivitas format yy/mm/dd
6.		type_of_service	String	Jenis layanan yang dipesan (transportasi motor/mobil, antar barang)
11.		activity_status	String	Status driver mengenai pelayanan
12.		price	Float	Harga satu kali pelayanan
	start_loc	Latitude	Float	
		Longitude	Float	
	end_loc	Latitude	Float	
		Longitude	Float	
13.	item_detail	id_item	ObjectID	Item detail untuk tipe service antar barang
14.		weight	Integer	Berat barang yang akan diantarkan (dalam satuan gram)
15.		type	String	Jenis barang yang akan diantarkan
16.		delivery_instruction	String	Instruksi dari customer kepada driver mengenai pengiriman
	recipient_detail	id_recipient	ObjectID	ID dari penerima barang
17.		recipient_name	String	Nama penerima
18.		recipient_phone_number	String	No HP penerima

Contoh Collection

```
{
 "_id": ObjectID("60d01162959f90f88b6d3571"),
 “id_driver”: {"96ke1162959f90f88b6d3571"},
 “id_customer”: {"167701162959f90f88b6d3571"},
 “id_feedback”: {"kf5fjy62959f90f88b6d3571"},
 “item_detail”: {
       “id_item”: ”I0000001”,
       “weight”: 400,
       “type”: ”Makanan”,
       “delivery_instruction”: ”Hati-hati tumpah”
  },
 “recipient_detail”: {
       “id_recipient”: ”R0000001”,
       “recipient_name”: ”Dzaki”,
       “recipient_phone_number”: ”089753567553”
  },
 “date”: “2020-12-09”,
 “type_of_service”: ”Antar Barang”,
 “start_loc”: ”1”,
 “end_loc”: ”2”,
 “activity_status”: ”Item picked up”,
 “price”: 23000.00
}
```

Indexing

Nama: _id_ (Default)

Deskripsi: Merupakan default index dari mongodb yang bersifat unique sehingga mencegah adanya document dengan id yang sama.

- _id

Nama: date_idx (Single Field Indexes)

Deskripsi: Untuk keperluan read riwayat aktivitas/order ke aplikasi dalam kurun waktu tertentu.

- date

Nama: type_of_service_1 (Single Field Indexes)

Deskripsi: Admin dapat melihat riwayat order dengan kategori tertentu

- type_of_service

Nama: id_cust_tos (Compound Field Indexes)

Deskripsi: Setiap *customer* pada aplikasi dapat melihat riwayat perjalanan berdasarkan jenis pelayanannya. *customer/*pengguna aplikasi tidak sedikit, maka query akan sering dilakukan.

- id_customer

- type_of_service

Nama: id_driver_tos (Compound Field Indexes)

Deskripsi: Untuk meningkatkan efektifitas fitur 'driver dapat melihat riwayat aktivitas yang dilakukan'. Karena driver berjumlah banyak dan akan terus berkembang, maka query juga sangat sering dilakukan.

- id_driver
- type_of_service

4. Collection DataAdmin

Tabel 15 Collection DataAdmin

Nama		DataAdmin		
Deskripsi		Menyimpan dokumen terkait data		
No.	**Nama Dokumen**	**Nama Field**	**Tipe Data (BSON, object [embedded, linked])**	**Deskripsi**
1.	admin_info	_id	ObjectID	ID document, di generate secara otomatis oleh mongodb
2.		admin_email	String	Email admin
3.		admin_password	String	Password admin
4.		created_at	Date	Tanggal
5.		admin_name	String	Nama admin

Contoh Collection

```
{
 "_id": ObjectID("60d01162959f90f88b6d3571"),
 "admin_name": "Admin Dzakira",
 "admin_pasword": "12345",
 "created_at": 2021-06-26T17:00:00.000Z
}
```

Indexing

Nama: _id_ (Default)

Deskripsi: Merupakan default index dari mongodb yang bersifat unique sehingga mencegah adanya document dengan id yang sama.

- _id

Nama: email_admin (Unique Indexes)

Deskripsi: Email akun harus unique sehingga satu akun email hanya dapat dipakai untuk 1 akun.

- admin_email

5. Collection Feedback

Tabel 16 Collection Feedback

Nama		DataAdmin		
Deskripsi		Menyimpan data review *customer* setelah order selesai.		
No.	**Nama Dokumen**	**Nama Field**	**Tipe Data (BSON, object [embedded, linked])**	**Deskripsi**
1.	`feedback`	`id_feedback`	ObjectID	ID document, di generate secara otomatis oleh mongodb
2.		`rating`	Float	Nilai yang dipilih customer untuk driver. Rentang 0 - 5
3.		`review`	String	Review customer mengenai driver

Contoh Collection

```
{
 "_id":ObjectID("60d01162959f90f88b6d3571"),
 "rating": 5.0,
 "review": "Baik banget abang ojolnyaa"
}
```

Indexing

Nama: _id_ (Default)

Deskripsi: Merupakan default index dari mongodb yang bersifat unique sehingga mencegah adanya document dengan id yang sama.

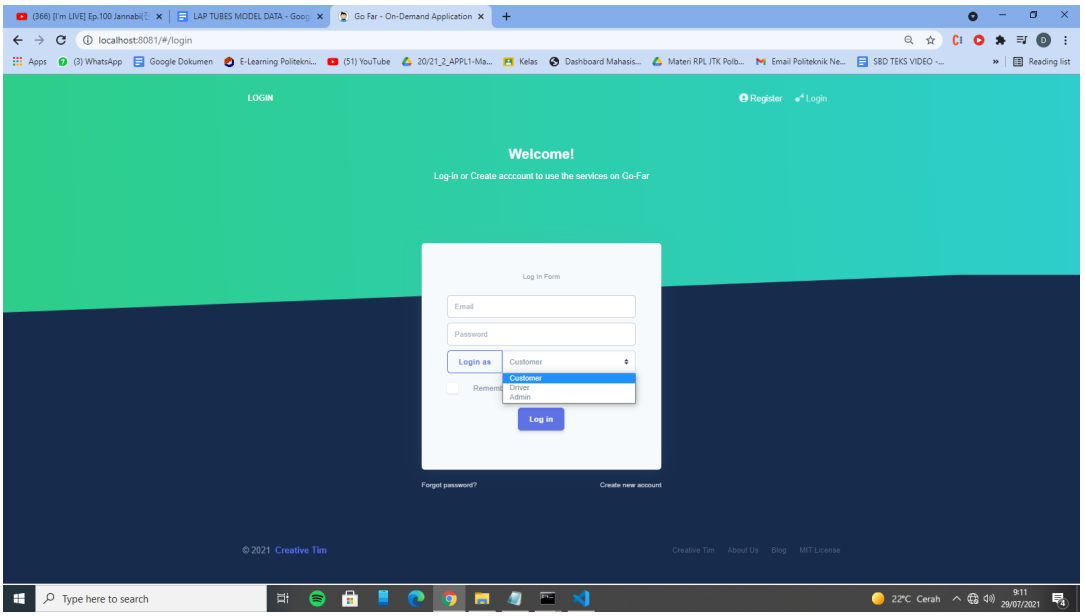
- _id

4. IMPLEMENTASI DAN EKSPERIMEN

4.1 Implementasi

- Login 3 User

Tabel 17 Fitur Log In



Fitur	Log In (FA6, FC9, dan FD11)
User	Admin, Customer, dan Driver
Deskripsi	Menyeleksi akses user masuk sebagai Admin, Customer, atau Driver.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	3 (1 untuk tiap user)
Script	Termasuk Important Queries

```
// ADMIN LOGIN
authRoutes.route("/auth/admin").post(function (req, res) {
    let db_connect = dbo.getDb("employees");
      try{
        db_connect.collection("DataAdmin")
                .find({admin_email: req.body.admin_email})
                .toArray(function (err, result) {
                    if (err) throw err;
                    if(result[0].admin_password == req.body.admin_password
){
```

```
                        res.status(201).json({
                            message: "Succesfully login",
                            result
                        });
                    }else{
                        res.status(400).json({
                            message: "Login unsuccessful"
                        });
                    }
                });

      }catch(err){
        console.log(err);
      }
  });

// CUST LOGIN
authRoutes.route("/auth/cust").post(function (req, res) {
    let db_connect = dbo.getDb("employees");
      try{
        db_connect.collection("DataCustomer")
                .find({customer_email: req.body.customer_email})
                .toArray(function (err, result) {
                    if (err) throw err;
                    if(result[0].customer_password ==
req.body.customer_password ){
                        res.status(201).json({
                            message: "Succesfully login",
                            result
                        });
                    }else{
                        res.status(400).json({
                            message: "Login unsuccessful"
                        });
                    }
                });

      }catch(err){
        console.log(err);
      }
  });

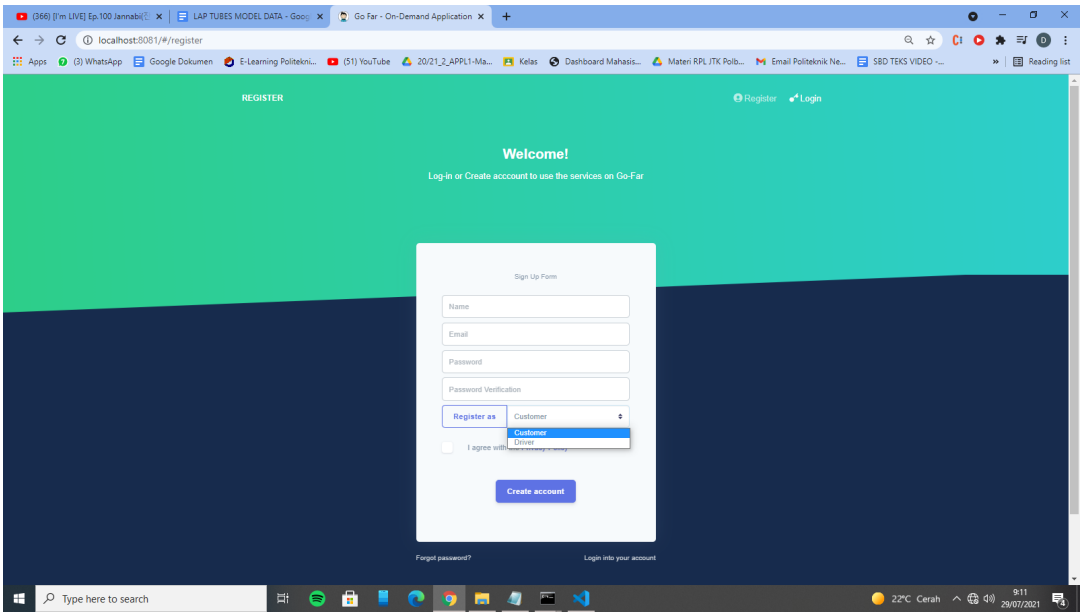
// DRIVER LOGIN
authRoutes.route("/auth/driver").post(function (req, res) {
    let db_connect = dbo.getDb("employees");
      try{
        db_connect.collection("DataDriver")
                .find({driver_email: req.body.driver_email})
                .toArray(function (err, result) {
```

```
                    if (err) throw err;
                    if(result[0].driver_password ==
req.body.driver_password ){
                        res.status(201).json({
                            message: "Succesfully login",
                            result
                        });
                    }else{
                        res.status(400).json({
                            message: "Login unsuccessful"
                        });
                    }
                });

        }catch(err){
          console.log(err);
        }
    });
```

- Sign Up 2 User

Tabel 18 Fitur Sign Up



Fitur	Sign Up (FC1 dan FD1)
User	Customer dan Driver
Deskripsi	Membuat account baru untuk user yang belum terdaftar baik sebagai Customer atau Driver.
Jumlah Koleksi yang Dibaca	1

Jumlah Proses Query	1
Script	Termasuk Important Queries

```
customerRoutes.route("/cust/create/newaccount").post(function (req, res) {
    let db_connect = dbo.getDb("employees");
    console.log(req)
    if(req.body.customer_password == req.body.password_verification){
        console.log("verifikasi password berhasil");
        let new_account = {
            customer_email: req.body.customer_email,
            customer_password: req.body.customer_password,
            profile:{
                name: req.body.profile.name,
                prof_pic: req.body.profile.prof_pic,
                phone_number: req.body.profile.phone_number,
                gender: req.body.profile.gender,
                birth_of_date: req.body.profile.birth_of_date
            },
            signUp_date: req.body.signUp_date,
            blocked: req.body.blocked
        };

        try{
            db_connect.collection("DataCustomer").insertOne(new_account);
            res.status(201).json({
            message: "Succesfully inserted",
            new_account
        });
        }catch(err){
            console.log(err);
        }
    }else{
        res.status(401).json({
            message: "Unsuccesful",
        });
    }
});

// DRIVER DAFTAR (email pass) (FD1)
driverRoutes.route("/driver/signup").post(function (req, res) {
    let db_connect = dbo.getDb("employees");
    if(req.body.driver_password == req.body.password_verification){
        // console.log("verif pass berhasil");
        let new_profile = {
          nik: "",
          sim_no: "",
          name: req.body.profile.name,
          profpict: "",
```

```
        phone_number: "",
        gender: "",
        birth_of_date: "",
      };

      let new_vehicle_details = {
        transportation_type: "",
        plat_number: "",
        capacity: 0,
        merk_and_type: "",
        stnk_no_registrasion: "",

      };

      let newAddress = {
        province: "",
        city: "",
        sub_district: "",
        zip_code: "",
        street: ""

      };

      let newDocuments = {
        skck: "",
        ktp: "",
        sim: "",
        stnk:"",

      };

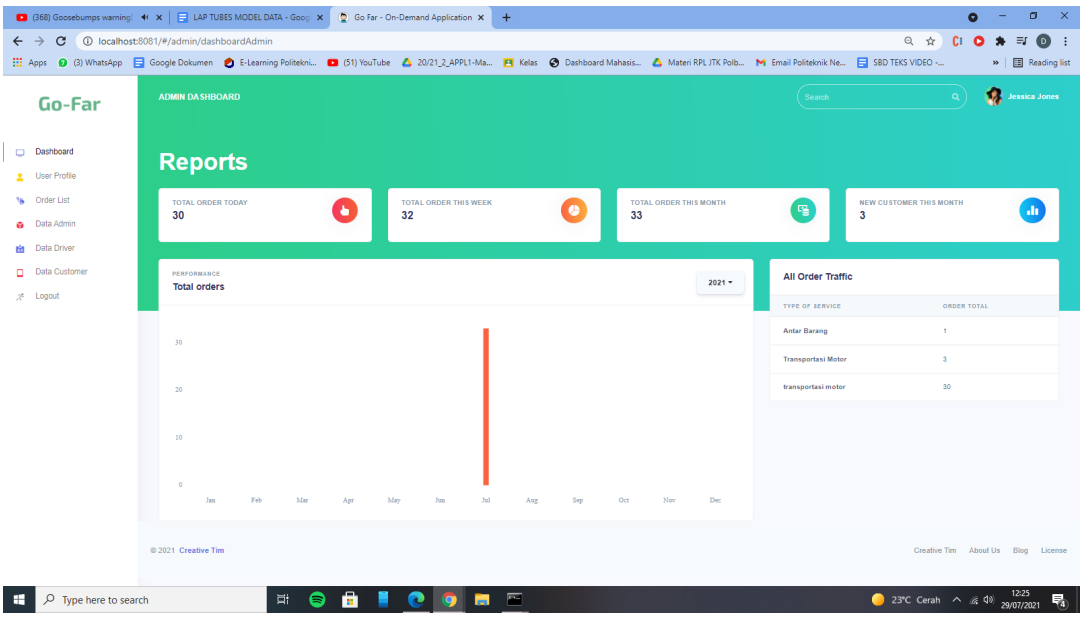
      let newDriver = {
        driver_email: req.body.driver_email,
        driver_password: req.body.driver_password,
        profile: new_profile,
        vehicle_details: new_vehicle_details,
        address: newAddress,
        documents:newDocuments,
        rating: null,
        submitted: false,
        active_status: false,
        blocked: false
      };
      try{
        db_connect.collection("DataDriver").insertOne(newDriver);
        res.status(201).json({
        message: "Succesfully inserted",
        newDriver
      });
```

```
        }catch(err){
          console.log(err);
        }
      }else{
        res.status(401).json({
          message: "Unsuccesful",
        });
      }
  });
```

4.1.1 Implementasi Fitur Admin

- Reports

Tabel 19 Fitur Reports



Fitur	Reports
User	Admin
Deskripsi	Laporan bulanan berisikan : - Total Order Per Layanan - Total Order dalam 1 bulan - Total Order dalam 1 minggu - Total Order dalam 1 hari - Jumlah akun *customer* baru dalam 1 bulan - Laporan dalam tahun tertentu (jumlah order dalam setiap bulannya untuk dijadikan chart)
Jumlah Koleksi yang Dibaca	2
Jumlah Proses Query	6

Script	Termasuk query penting

```
/* Total order per layanan*/

adminRoutes.route("/admin/read/report/pertype").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      { $group : { _id : '$type_of_service', totalorder : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});

/*REPORT THIS MONTH */
adminRoutes.route("/admin/get/totalorder/thismonth").get(function (req,
res) {
  let db_connect = dbo.getDb("employees");
  let order_month = { date: {$gt: new Date(new Date().setDate(new
Date().getDate() - 30))} };
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      {$match : order_month},
      { $group : { _id : 0, totalorder : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });

});

//TOTAL ORDER THIS WEEK
adminRoutes.route("/admin/get/totalorder/thisweek").get(function (req,
res) {
  let db_connect = dbo.getDb("employees");
  let order_week = { date: {$gt: new Date(new Date().setDate(new
Date().getDate() - 7))} };
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      {$match : order_week},
      { $group : { _id : 0, totalorder : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
```

```
      if (err) throw err;
      res.json(result);
    });

});

// TOTAL ORDER TODAY
adminRoutes.route("/admin/get/totalorder/today").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let order_day = { date: {$gt: new Date(new Date().setDate(new
Date().getDate() - 1))} };
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      {$match : order_day},
      { $group : { _id : 0, totalorder : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });

});

//NEW ACCOUNT THIS MONTH
adminRoutes.route("/admin/get/totalnewcust/thismonth").get(function (req,
res) {
  let db_connect = dbo.getDb("employees");
  let this_month = { signUp_date: {$gt: new Date(new Date().setDate(new
Date().getDate() - 30))} };
  db_connect
    .collection("DataCustomer")
    .aggregate([
      {$match : this_month},
      { $group : { _id : 0, totaluser : { $sum : 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });

});

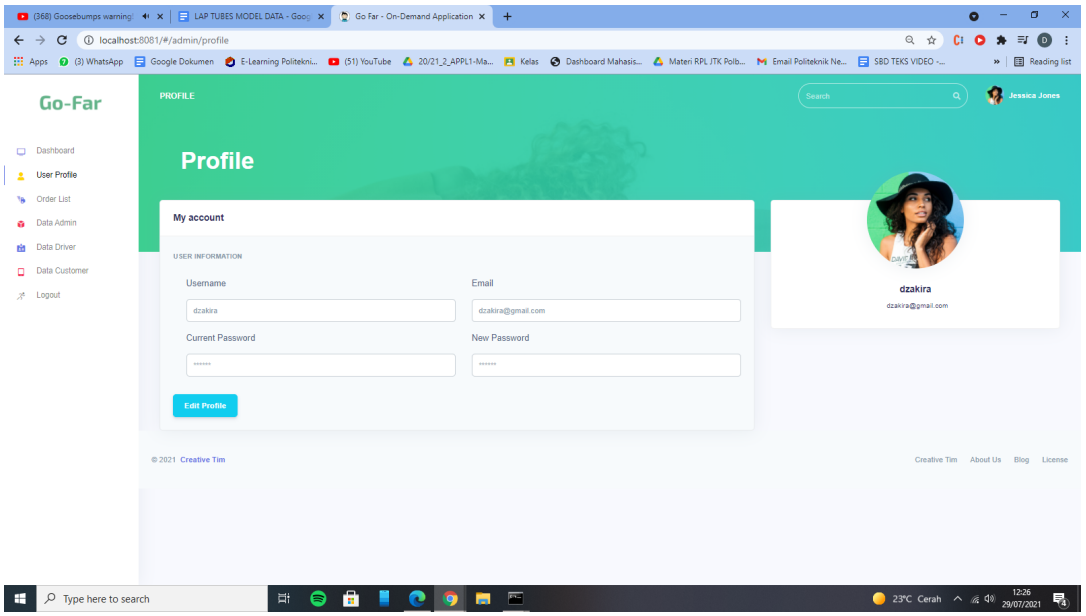
//REPORT IN 1 YEAR
adminRoutes.route("/admin/get/totalorder/:year").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let year = req.params.year;
  let nextyear = (parseInt(year) + 1).toString();
  let myquery = { date: {$gte: new Date(year), $lt: new Date(new
```

```
Date(nextyear).setDate(new Date(nextyear).getDate() - 1))} };
  // console.log(new Date();
  db_connect
    .collection("ActivityHistory")
    .aggregate([
      {$match: myquery },
      { $group : {  _id: {$substr: ['$date', 5, 2]} , totalorder : { $sum
: 1 } } }
    ])
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });

});
```

- Kelola Profile Admin

Tabel 20 Fitur Profile Admin



Fitur	• Read Profile (FA2) • Update admin account (FA3)
User	Admin
Deskripsi	Melihat dan mengubah data akun admin
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	2

Script	Bukan termasuk *query* penting

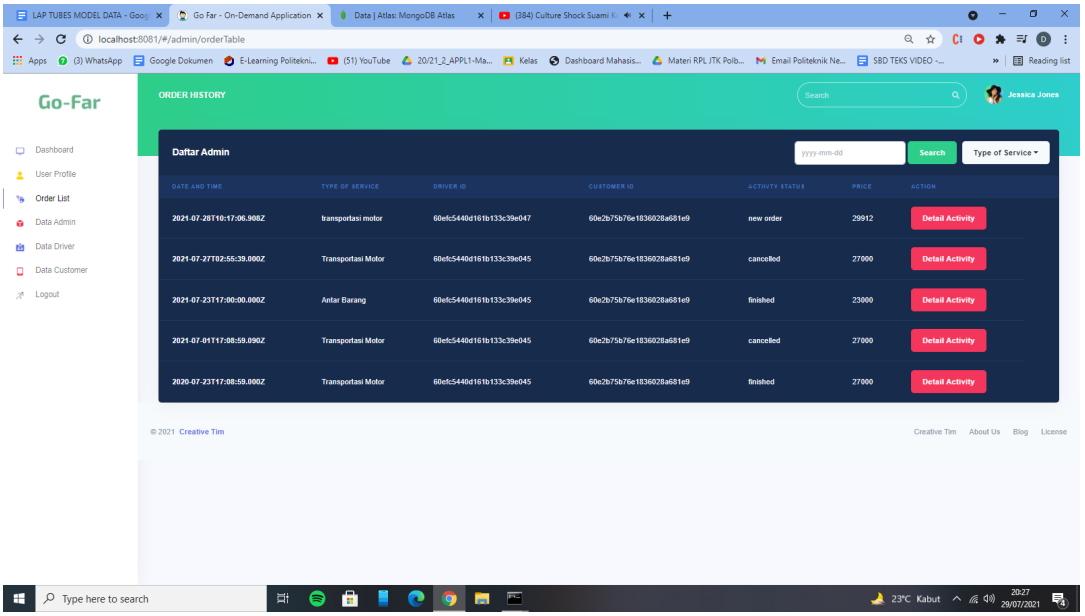
```
adminRoutes.route("/admin/read/admin/:id").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect
    .collection("DataAdmin")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});

// ADMIN UPDATE ADMIN ACCOUNT (FA4)
adminRoutes.route("/admin/update/admin/:id").post(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let myquery = { _id: new mongodb.ObjectID(req.params.id),
admin_password: req.body.admin_curr_password};

  let newvalues = {
    $set: {
      admin_name: req.body.admin_name,
      admin_email: req.body.admin_email,
      admin_password: req.body.admin_new_password
    }
  };
```

- Read Activity + Search

Tabel 21 Fitur Read & Search Activity



Fitur	Read All Activity (FR5) Read activity per day (FR6) Read Activity Per Category (FR11)
User	Admin
Deskripsi	• Melihat seluruh riwayat kegiatan (baik layanan transportasi, maupun layanan antar barang). • Melihat riwayat aktivitas dalam jangka waktu tertentu (bisa dalam satu hari tertentu). • Melihat activity berdasarkan category tertentu.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	3 (masing-masing 1 untuk setiap fitur)
Script	Bukan termasuk query penting

```
//Read All Activity

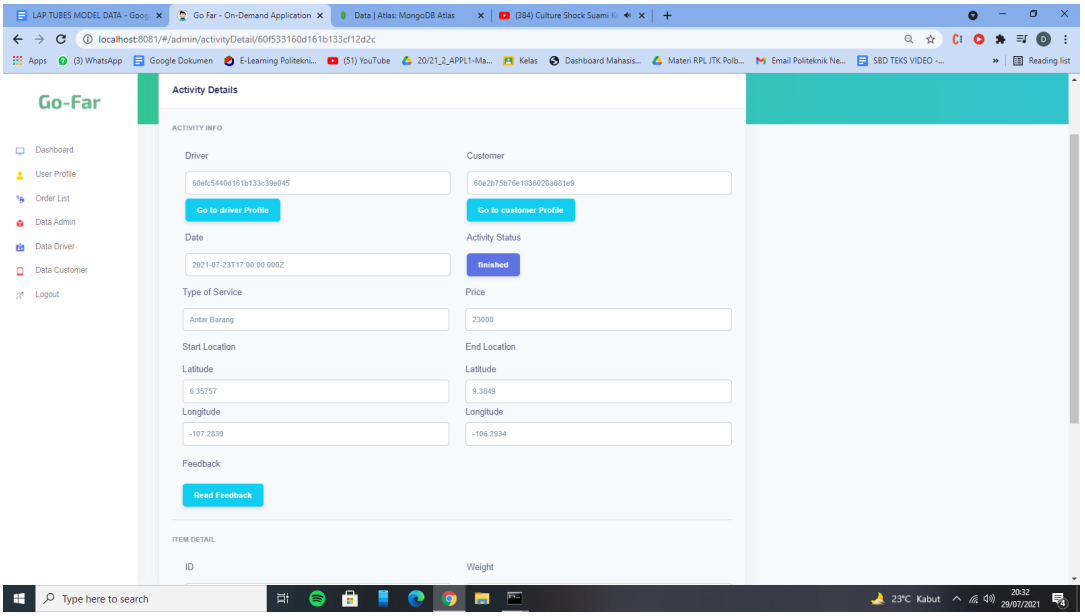
adminRoutes.route("/admin/read/allactivity").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  db_connect
    .collection("ActivityHistory")
    .find({})
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});
```

```
//Read Activity Search by Date
adminRoutes.route("/admin/get/history/perday/:date").get(function (req,
res) {
  let db_connect = dbo.getDb("employees");
  let myquery = { date: {$gte: new Date(req.params.date), $lt: new
Date(new Date(req.params.date).setDate(new
Date(req.params.date).getDate()+1))} };
  // console.log(new Date(new Date(req.params.date).setDate(new
Date(req.params.date).getDate()+1)));
  db_connect
    .collection("ActivityHistory")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});

//Read Per Category
adminRoutes.route("/admin/get/history/:category").get(function (req, res)
{
  let db_connect = dbo.getDb("employees");
  let myquery = { type_of_service: req.params.category };
  db_connect
    .collection("ActivityHistory")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});
```

- Read Activity Tertentu

Tabel 22 FiturRead Activity tertentu

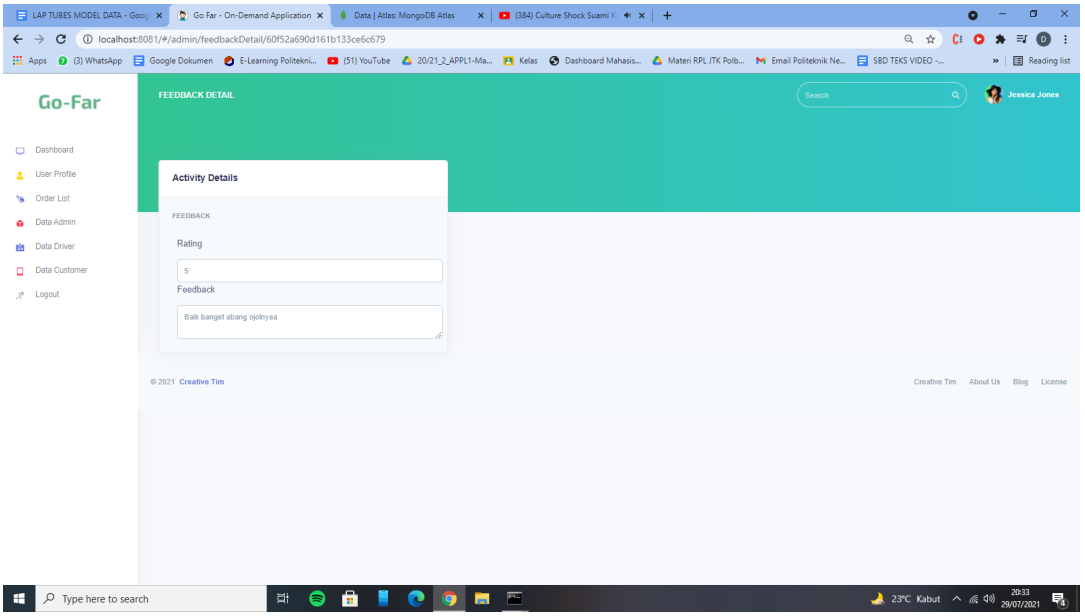


Fitur	Read activity (FR10)
User	Admin
Deskripsi	Membaca data aktivitas tertentu yang kemudian dapat melihat data driver, customer, dan juga feedback dari halaman tersebut.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	1
Script	Bukan termasuk *query* penting

```
adminRoutes.route("/admin/read/activity/:id").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect
    .collection("ActivityHistory")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});
```

- Read Feedback Tertentu

Tabel 23 Fitur Read Feedback

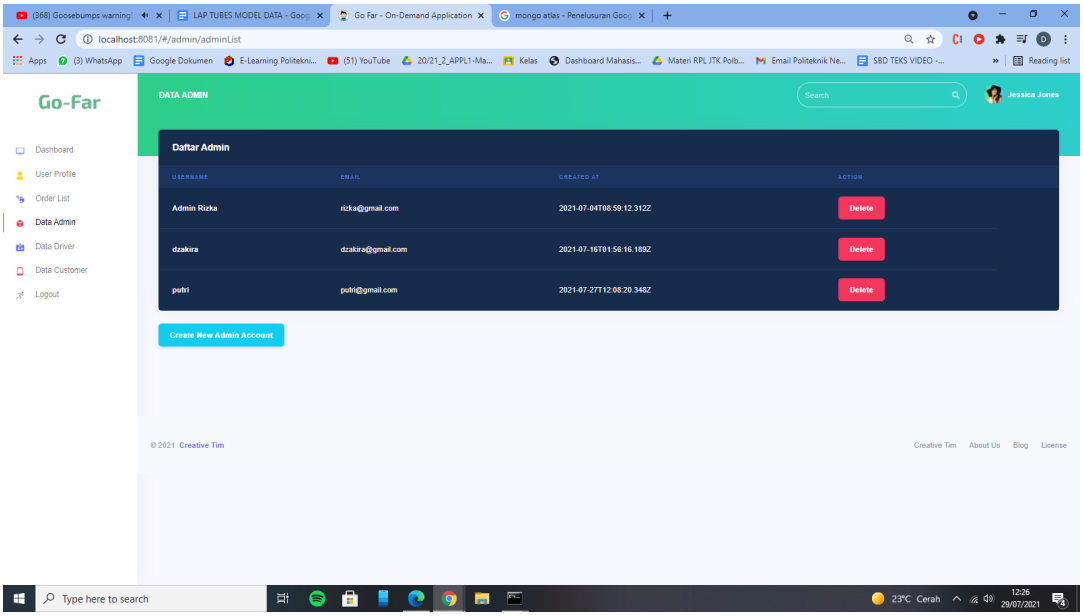


Fitur	Read Feedback tertentu (FR12)
User	Admin
Deskripsi	Melihat feedback tertentu.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	1
Script	Bukan termasuk *query* penting

```
adminRoutes.route("/admin/read/feedback/:id").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect
    .collection("Feedback")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});
```

- Kelola data admin (Tabel + Delete)

Tabel 24 Fitur Kelola Data Admin



Fitur	Kelola data admin (Tabel + Delete) // (FA3 dan FA5)
User	Admin
Deskripsi	Admin dapat melihat (read) akun dirinya sendiri dan dapat menghapus akun admin.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	2
Script	Termasuk *query* penting

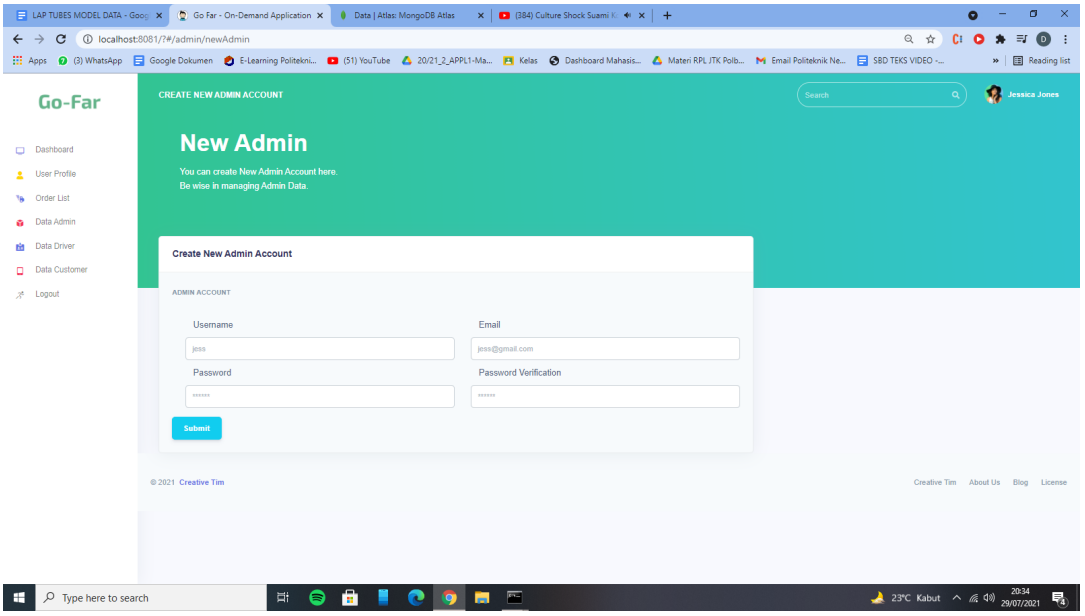
```
// ADMIN READ AKUN ADMIN TERENTU (FA3)
adminRoutes.route("/admin/read/admin/:id").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect
    .collection("DataAdmin")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});


// ADMIN DELETE ADMIN ACCOUNT (FA5)
adminRoutes.route("/admin/delete/admin/:id").delete((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect.collection("DataAdmin").deleteOne(myquery, function (err,
```

```
obj) {
    if (err) throw err;
    res.status(201).json({obj, message : "Deleted Succesfully"});
  });
});
```

- Form Add New Admin

Tabel 25 Fitur *Add New Admin*



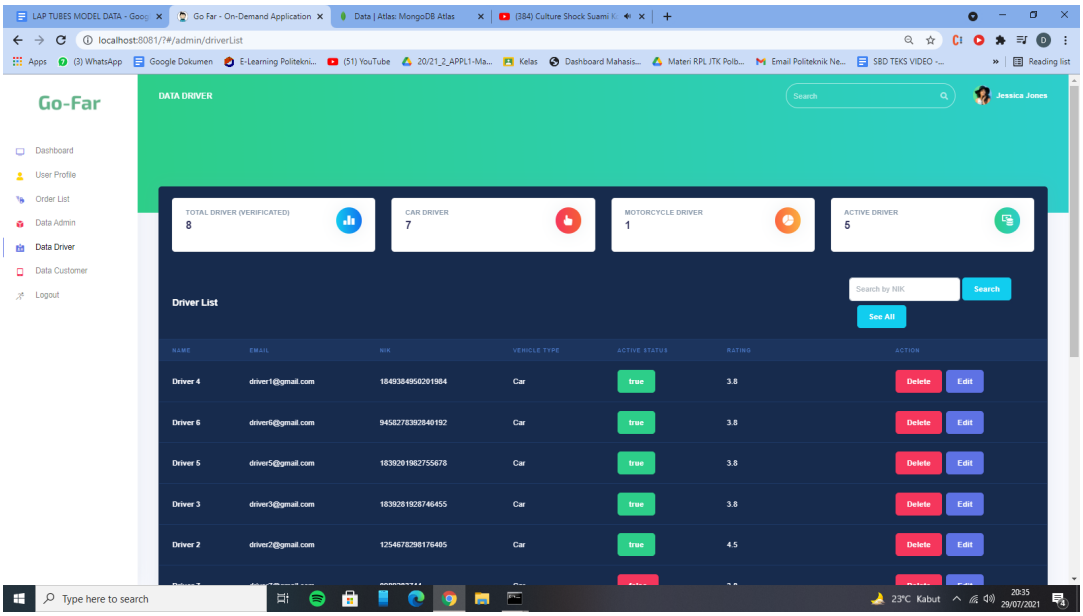
Fitur	Create Admin Account (FA1)
User	Admin
Deskripsi	Membuat akun admin baru.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	1
Script	Termasuk *query* penting

```
adminRoutes.route("/admin/add/newadmin").post(function (req, res) {
  let db_connect = dbo.getDb("employees");
  // console.log(req)
  if(req.body.admin_password == req.body.password_verification){
    console.log("verif pass berhasil");
    let newAdm = {
      admin_name: req.body.admin_name,
      admin_email: req.body.admin_email,
      admin_password: req.body.admin_password,
```

```
      created_at: new Date()
    };
    try{
      //console.log(req.data);
      db_connect.collection("DataAdmin").insertOne(newAdm);
      res.status(201).json({
      message: "Succesfully inserted",
      newAdm
    });
    }catch(err){
      console.log(err);
    }
  }else{
    res.status(401).json({
      message: "Unsuccesful",
    });
  }
});
```

- Daftar Daftar Driver

Tabel 26 Fitur Daftar DaraDriver



Fitur	• Read All driver (FD2) • Total Driver (FD12) • Total Car Driver (FD12) • Total Motor Driver (FD12) • Total Active Driver (FD12) • Search Driver by NIK (FD3)

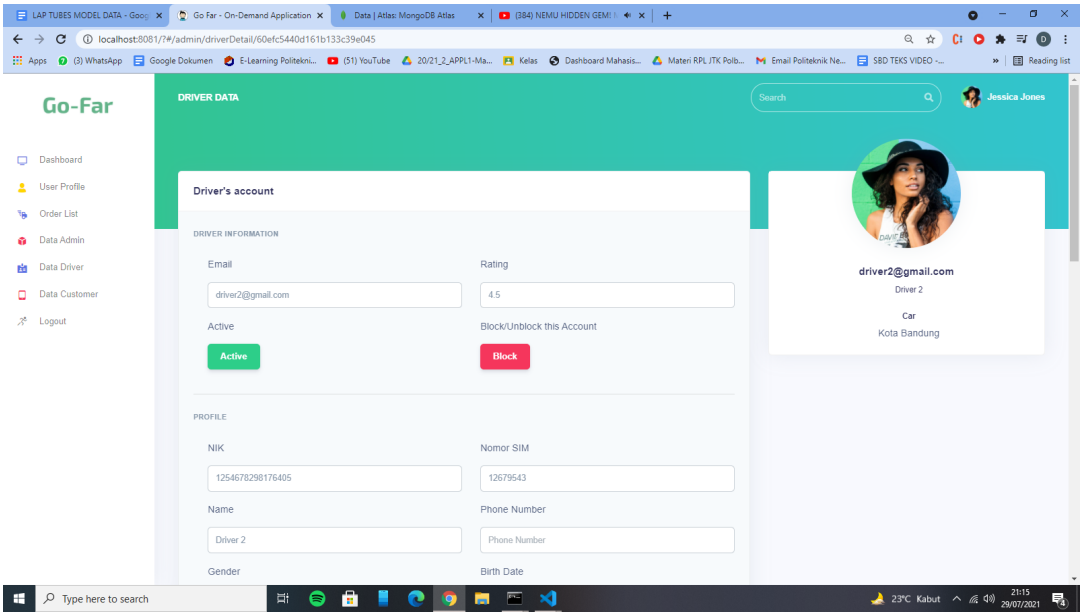
User	Admin
Deskripsi	Berisi tabel daftar driver, dan pada tampilan aplikasi akan ditampilkan data Total Driver, Total Car Driver, Total Motor Driver, Total Active Driver yang dihitung di bagian front end.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	2
Script	Bukan termasuk *query* penting

```
adminRoutes.route("/admin/read/alldriver").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let mySort = {name: 1};
  db_connect
    .collection("DataDriver")
    .find({})
    .sort(mySort)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
      // console.log(result);
    });
});

adminRoutes.route("/admin/search/driver/:nik").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { "profile.NIK": (req.params.nik).toString() };
  // console.log((req.params.nik).toString());
  db_connect
    .collection("DataDriver")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
      // console.log(result);
    });
});
```

- Kelola Data Seorang Driver

Tabel 27 Fitur Kelola Data Driver



Fitur	• Read 1 Driver (FD9) • Verifikasi Driver (FD8) • Update data driver (FD5) • Blokir Driver (FD13)
User	Admin
Deskripsi	Admin dapat mengelola Data Driver, seperti Read Data Driver Tertentu, Verifikasi driver, Update Profile Driver, dan admin pun dapat mengubah status (blokir) driver
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	4
Script	Termasuk query penting

```
// ADMIN READ 1 DRIVER (FD9)
adminRoutes.route("/admin/read/driver/:id").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect
    .collection("DataDriver")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
      // console.log(result);
    });
});
```

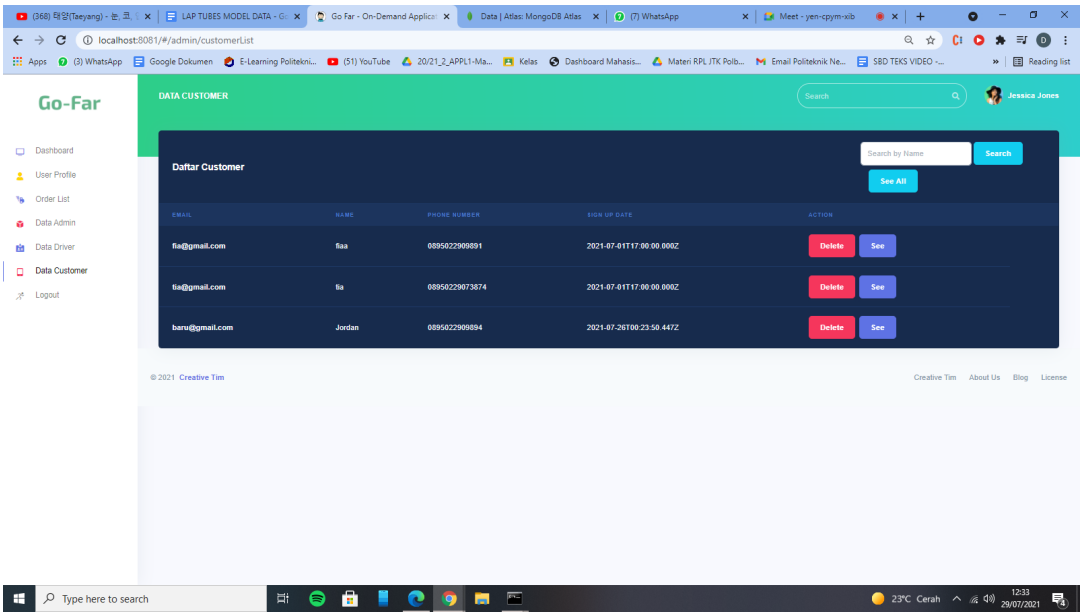
```
// ADMIN VERIFIKASI DRIVER (FD8)
adminRoutes.route("/admin/update/verif/:id").post(function (req, res) {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  let newvalues = {
    $set: {
      verification_status: true
    },
  };
  db_connect
    .collection("DataDriver")
    .updateOne(myquery, newvalues, function (err, res) {
      if (err) throw err;
      console.log("1 document updated");
    });
});

// ADMIN UPDATE DRIVER (FD5)
adminRoutes.route("/admin/update/driver/:id").post(function (req, res) {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  let newvalues = {
    $set: req.body
  };
  db_connect
    .collection("DataDriver")
    .updateOne(myquery, newvalues, function (err, result) {
      if (err) throw err;
      res.status(201).json({result, message : "Updated Succesfully"});
    });
});

// ADMIN UPDATE DRIVER STATUS BLOKIR (FC7)
adminRoutes.route("/admin/update/blokirdriver/:id").post(function (req,
res) {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  let newvalues = {
    $set: {
      blocked: req.body.blocked
    },
  };
  db_connect
    .collection("DataDriver")
    .updateOne(myquery, newvalues, function (err, result) {
      if (err) throw err;
      res.status(201).json({result, message : "Updated Succesfully"});
    });
});
```

- Kelola Daftar Customer (Read, Search & Delete)

Tabel 28 Fitur Daftar DataCustomer



Fitur	• Daftar Customer (FC3) • Search by name (FC4) • Delete (FC10)
User	Admin
Deskripsi	Admin melihat seluruh data customer, melakukan operasi pencarian terhadap data customer berdasarkan Nama, serta menghapus Data Customer.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	3
Script	Bukan termasuk *query* penting

```
// ADMIN READ CUSTOMER (FC3)
adminRoutes.route("/admin/read/allcust").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let mySort = {name: 1};
  db_connect
    .collection("DataCustomer")
    .find({})
    .sort(mySort)
    .toArray(function (err, result) {
      if (err) throw err;
```

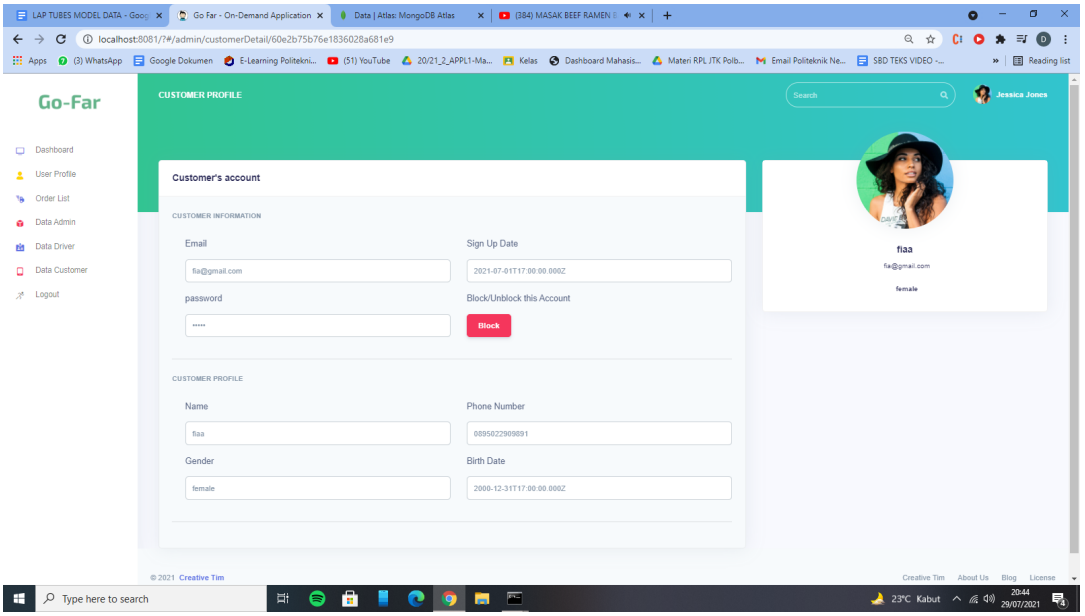
```
      res.json(result);
      // console.log(result);
    });
});

// ADMIN SEARCH CUSTOMER (FC4)
adminRoutes.route("/admin/search/cust/:name").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { "profile.name": (req.params.name).toString() };
  db_connect
    .collection("DataCustomer")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
      // console.log(result);
    });
});

//ADMIN DELETE CUSTOMER
adminRoutes.route("/admin/delete/customer/:id").delete((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect.collection("DataCustomer").deleteOne(myquery, function (err,
obj) {
    if (err) throw err;
    res.status(201).json({obj, message : "Deleted Succesfully"});
  });
});
```

- Read 1 Customer

Tabel 29 Fitur Daftar DaraDriver



Fitur	• Read 1 *customer* • Blokir *customer*
User	Admin
Deskripsi	Melihat data *customer* tertentu dan admin dapat blokir/unblokir suatu akun.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	2
Script	Bukan termasuk *query* penting

```
// ADMIN READ 1 DRIVER (FD9)
adminRoutes.route("/admin/read/driver/:id").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect
    .collection("DataDriver")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
      // console.log(result);
    });
});

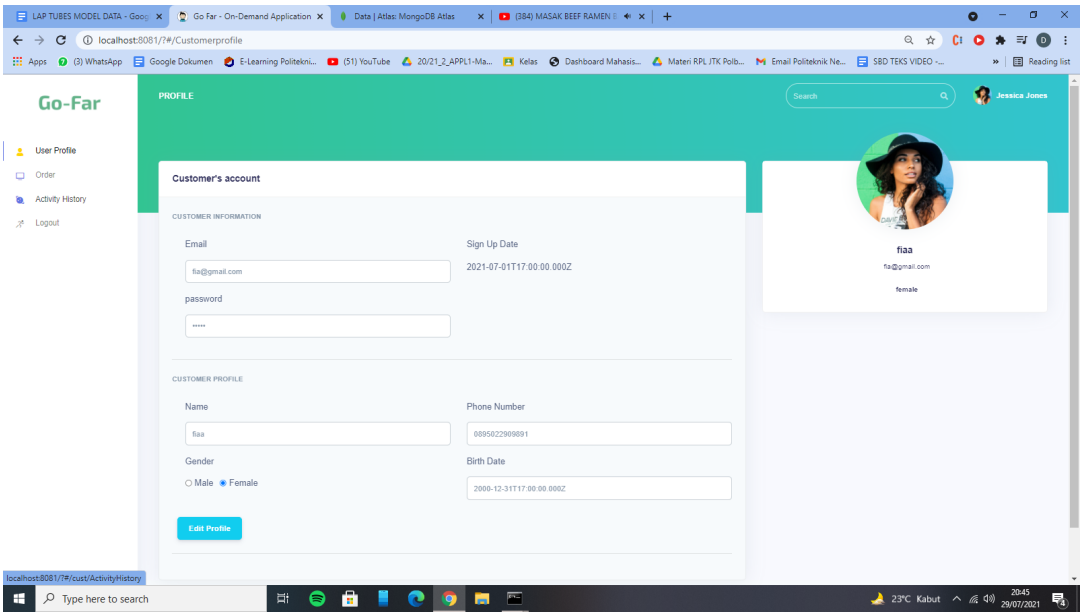
// ADMIN UPDATE DRIVER STATUS BLOKIR (FC7)
adminRoutes.route("/admin/update/blokirdriver/:id").post(function (req,
res) {
```

```
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  let newvalues = {
    $set: {
      blocked: req.body.blocked
    },
  };
  db_connect
    .collection("DataDriver")
    .updateOne(myquery, newvalues, function (err, result) {
      if (err) throw err;
      res.status(201).json({result, message : "Updated Succesfully"});
    });
});
```

4.1.2 Implementasi Fitur Customer

- Read dan Update Data Sendiri

Tabel 30 Fitur Read & Update *customer*



Fitur	• Read Data Sendiri (FC6) • Update Data Sendiri (FC7)
User	Customer
Deskripsi	Customer membaca data sendiri yang dari email, password, nama, gender, tempat tanggal lahir, foto profil, nomor telepon, tanggal sign up, dan status akun
Jumlah Koleksi yang Dibaca	1

Jumlah Proses Query	2
Script	Bukan termasuk important queries

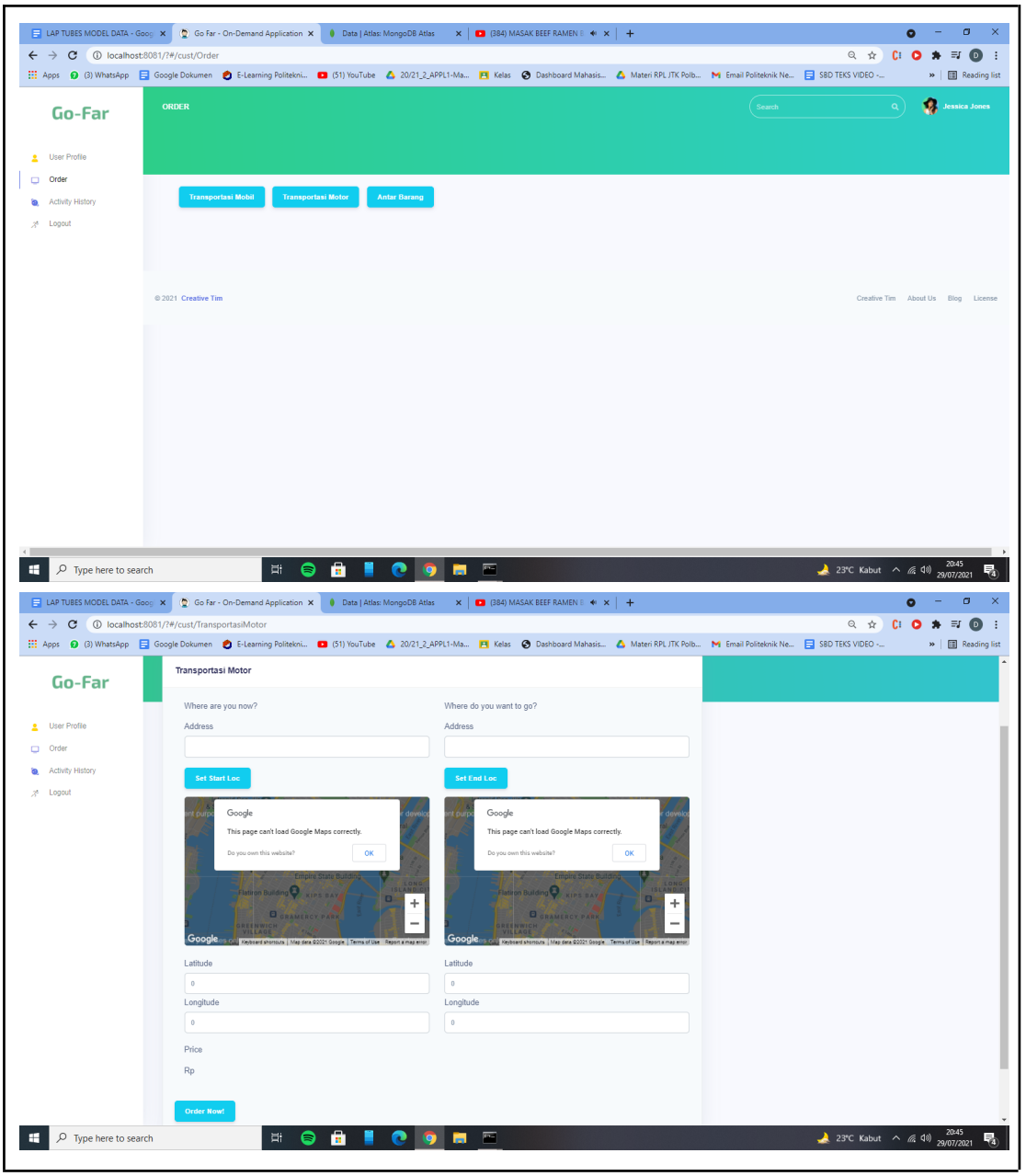
```
//CUSTOMER READ DATA SENDIRI (FC6)
customerRoutes.route("/cust/read/account/:id").get(function(req, res){
    let db_connect = dbo.getDb("employees");
    let id_account = { _id : new mongodb.ObjectID(req.params.id)}

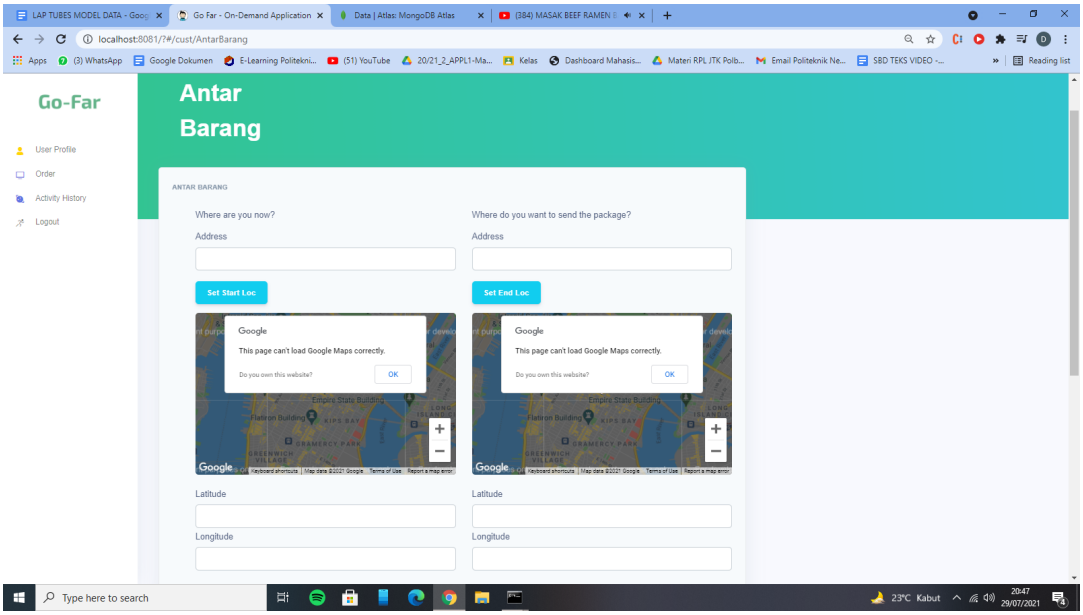
db_connect.collection("DataCustomer").find(id_account).toArray(function
(err, result) {
      if (err) throw err;
      res.json(result);
       console.log(result);
    });
});

// CUSTOMER UPDATE ACCOUNT // (FC2)
customerRoutes.route("/cust/update/account/:id").post(function (req, res)
{
    let db_connect = dbo.getDb("employees");
    let myquery = { _id : new mongodb.ObjectID(req.params.id)}
    let newvalues = {
        $set: {
        customer_email: req.body.customer_email,
        customer_password: req.body.customer_password,
        profile:{
            name: req.body.profile.name,
            prof_pic: req.body.profile.prof_pic,
            phone_number: req.body.profile.phone_number,
            gender: req.body.profile.gender,
            birth_of_date: req.body.profile.birth_of_date
        },
        signUp_date: req.body.signUp_date,
        blocked: req.body.blocked
        }
    };
```

- Create Order

Tabel 31 Fitur Create Oderr





Fitur	Create Order (FR1)
User	Customer
Deskripsi	Customer membuat order. Order terdiri dari 3 jenis yaitu transportasi motor, transportasi mobil, dan antar barang.
Jumlah Koleksi yang Dibaca	2
Jumlah Proses Query	2 (Untuk 3 Jenis Layanan)
Script	Termasuk important queries

```
// CUSTOMER CREATE ORDER // (FR1)
customerRoutes.route("/cust/create/order/transportasi").post(function
(req, res) {
    let db_connect = dbo.getDb("On-Demand");

    var id_driver = "";
    if(req.body.type_of_service == "transportasi motor"){
        var myquery = { active_status: true , verification_status: true,
"vehicle_detail.transportation_type": "Motorcycle"};
    }else{
        var myquery = { active_status: true , verification_status: true,
"vehicle_detail.transportation_type": "Car"};
    }

    db_connect
    .collection("DataDriver")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
```

```
      let selected = Math.floor(Math.random() * result.length);
      id_driver = result[selected]._id;

      let new_order = {
        id_driver: id_driver.toHexString(),
        id_customer: req.body.id_customer,
        date: new Date(),
        type_of_service: req.body.type_of_service,
        start_loc: {
            longitude: req.body.start_loc.longitude,
            latitude: req.body.start_loc.latitude
        },
        end_loc: {
            longitude: req.body.end_loc.longitude,
            latitude: req.body.end_loc.latitude
        },
        activity_status: "new order",
        price: req.body.price
    };

    try{
        db_connect.collection("ActivityHistory").insertOne(new_order);
        res.status(201).json({
        message: "Succesfully inserted",
        new_order
    });
    }catch(err){
        console.log(err);
    }

    });
});

customerRoutes.route("/cust/create/order/antarbarang").post(function (req,
res) {
    let db_connect = dbo.getDb("On-Demand");

    let id_driver = "";
    let length = 0;
    var myquery = {verification_status: true, active_status: true,
"vehicle_detail.transportation_type": "Motorcycle" };

    db_connect
    .collection("DataDriver")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      length = result.length;
      let selected = Math.floor(Math.random() * length);
```

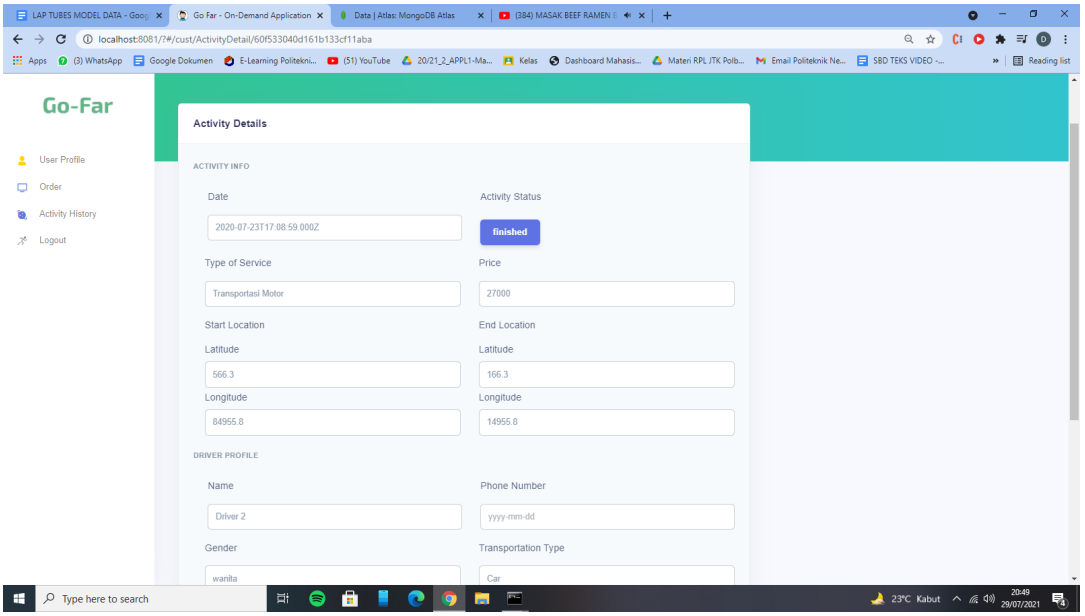
```
      id_driver = result[selected]._id;
    });

    let new_order = {
        id_driver: id_driver,
        id_customer:req.body.id_customer,
        item_detail: {
            id_item:new mongodb.ObjectID(),
            weight: req.body.item_detail.weight,
            type: req.body.item_detail.type,
            delivery_instruction:
req.body.item_detail.delivery_instruction
        },
        recipient_detail: {
            id_recepient: new mongodb.ObjectID(),
            recipient_name: req.body.recipient_detail.recipient_name,
            recipient_phone_number:
req.body.recipient_detail.recipient_phone_number
        },
        date: new Date(),
        type_of_service: 'antar barang',
        start_loc: {
            longitude: req.body.start_loc.longitude,
            latitude: req.body.start_loc.latitude
        },
        end_loc: {
            longitude: req.body.end_loc.longitude,
            latitude: req.body.end_loc.latitude
        },
        activity_status: "new order",
        price: req.body.price
    };

    try{
        db_connect.collection("ActivityHistory").insertOne(new_order);
        res.status(201).json({
        message: "Succesfully inserted",
        new_order
    });
    }catch(err){
        console.log(err);
    }
});
```

- Read 1 Order

Tabel 32 Fitur Read Oder



Fitur	Read 1 Order // (FR4) Get Driver Profile (FD9)
User	Customer
Deskripsi	Customer membaca detail 1 order dan profil driver yang akan menangani pesanan.
Jumlah Koleksi yang Dibaca	2
Jumlah Proses Query	2
Script	Tidak termasuk important queries

```
customerRoutes.route("/cust/read/order/:id").get(function (req, res) {
    let db_connect = dbo.getDb("employees");
    var myquery = { _id: new mongodb.ObjectID(req.params.id) };

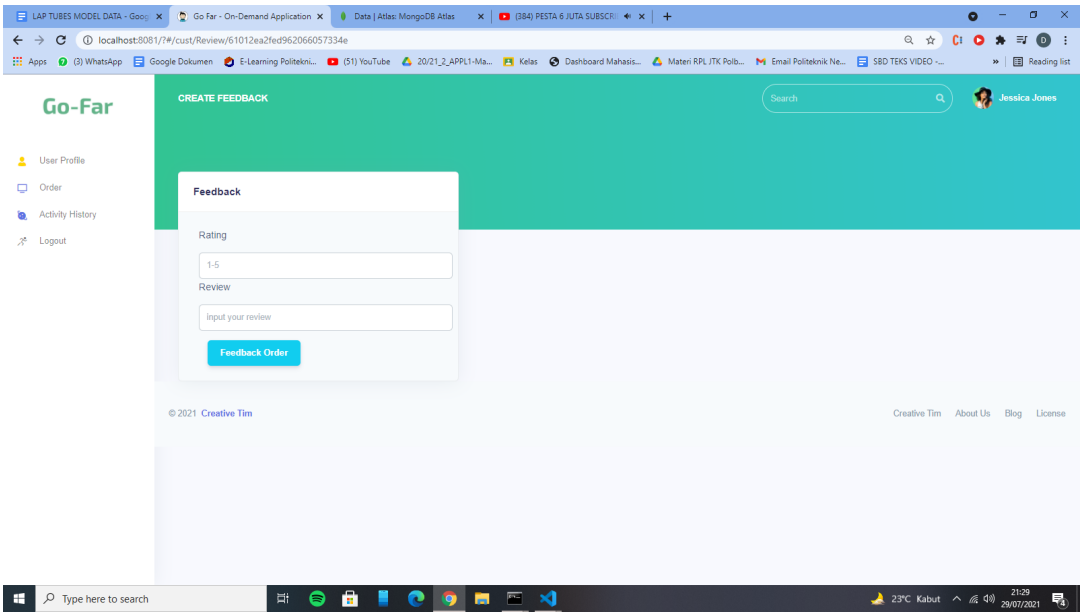
db_connect.collection("ActivityHistory").find(myquery).toArray(function
(err, result) {
            if (err) throw err;
            res.json(result);
          });
});

customerRoutes.route("/cust/get/driver_profile").get(function (req, res) {
    try{
        db_connect.collection("DataDriver").find();
        res.status(201).json({
        message: "Succesfully inserted",
```

```
        new_account
    });
    }catch(err){
        console.log(err);
    }
});
```

- Create Review

Tabel 33 Fitur Review



Fitur	Create Review // (FR3)
User	Customer
Deskripsi	Customer membuat review terkait
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	1
Script	tidak termasuk important queries

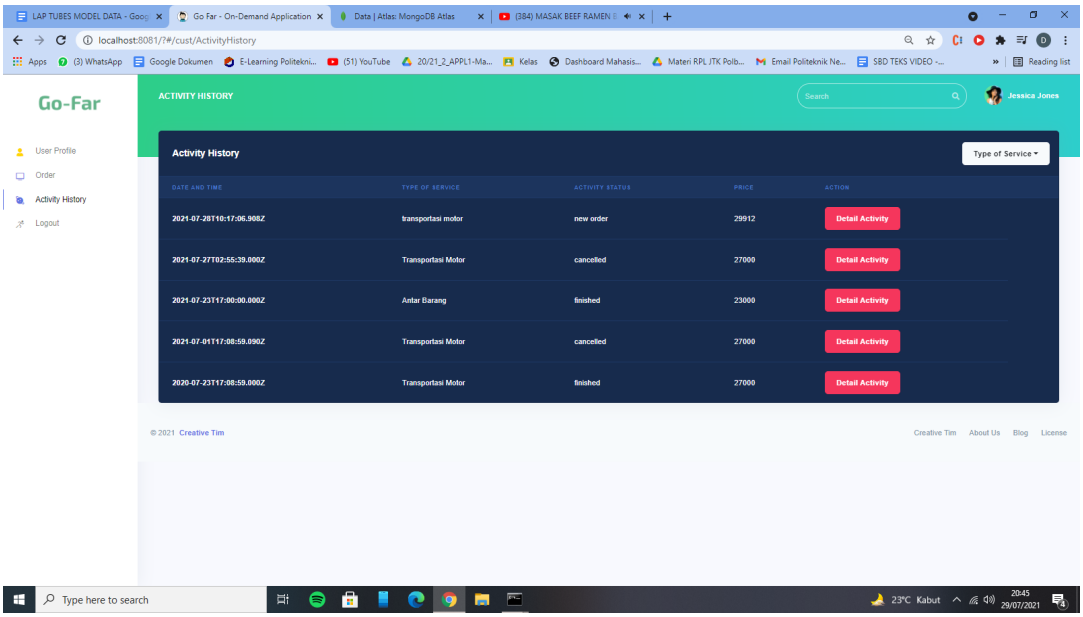
```
customerRoutes.route("/cust/create/review").post(function (req, res) {
    let db_connect = dbo.getDb("On-Demand");
    let new_review = {
        rating: req.body.rating,
        review: req.body.review
    };

    try{
```

```
        db_connect.collection("Feedback").insertOne(new_review);
        res.status(201).json({
        message: "Succesfully inserted",
        new_review
    });
    }catch(err){
        console.log(err);
    }
});
```

- Get All Order History + Per Category

Tabel 34 Fitur Customer Get All Order History + Per Category



Fitur	• Get All Order History // (FR4) • Get Order History (Motorcycle, Car, dan Item) // (FR4)
User	Customer
Deskripsi	Customer melihat daftar pesanan yang pernah dilakukannya. baik yang sudah selesai atau yang masih dilakukannya dan dari semua tipe pelayanan.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	1
Script	Tidak termasuk important queries

```
customerRoutes.route("/cust/get/allorder_history/:id").get(function (req,
```

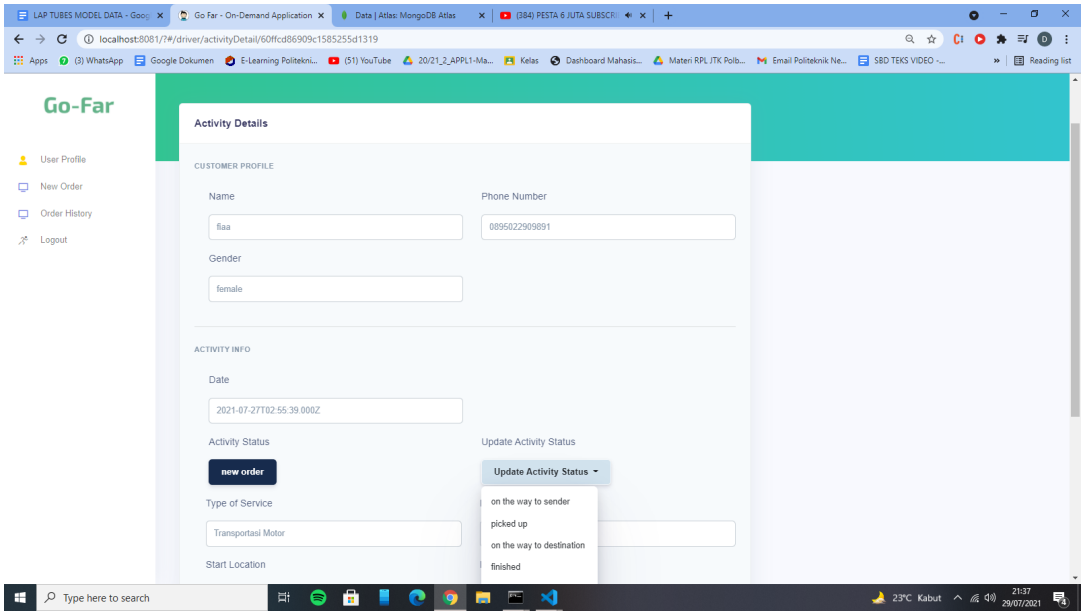
```
res) {
    let db_connect = dbo.getDb("employees");
    var myquery = { id_customer :req.params.id };
    db_connect
      .collection("ActivityHistory")
      .find(myquery)
      .toArray(function (err, result) {
        if (err) throw err;
        res.json(result);
      });
  });

customerRoutes.route("/cust/get/history/:category").get(function (req,
res) {
    let db_connect = dbo.getDb("employees");
    let myquery = { type_of_service: req.params.category };
    db_connect
      .collection("ActivityHistory")
      .find(myquery)
      .toArray(function (err, result) {
        if (err) throw err;
        res.json(result);
      });
  });
```

4.1.3 Implementasi Fitur Driver

- Read & Update New Order Status

Tabel 35 Fitur Read & Update New Order Status



Fitur	• Read Data Order tertentu (FR10)

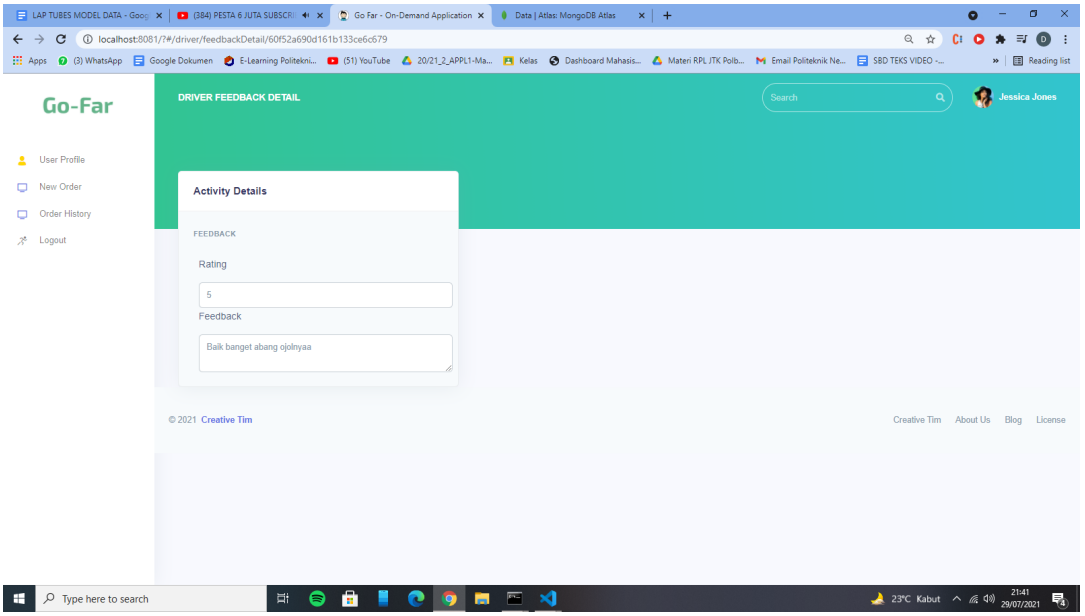
	• Update order status (FR13) • Lihat profil *customer*
User	Driver
Deskripsi	Mengubah status order sesuai dengan aturan pada *business rule* pada data order yang status nya belum *'finished'*
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	2
Script	Termasuk Query Penting

```
adminRoutes.route("/admin/read/activity/:id").get((req, res) => {
  let db_connect = dbo.getDb("employees");
  var myquery = { _id: new mongodb.ObjectID(req.params.id) };
  db_connect
    .collection("ActivityHistory")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});

driverRoutes.route("/driver/update/profile/:id").post(function (req, res)
{
    let db_connect = dbo.getDb("employees");
    var myquery = { _id: new mongodb.ObjectID(req.params.id), submitted:
false };
    let newvalues = {
      $set: req.body
    };
    db_connect
      .collection("DataDriver")
      .updateOne(myquery, newvalues, function (err, result) {
        if (err) throw err;
        console.log("1 document updated");
        res.status(201).json({result, message : "Updated Succesfully"})
      });
  });
```

- Read Review After Order Finished

Tabel 36 Fitur Read Review After Order Finished

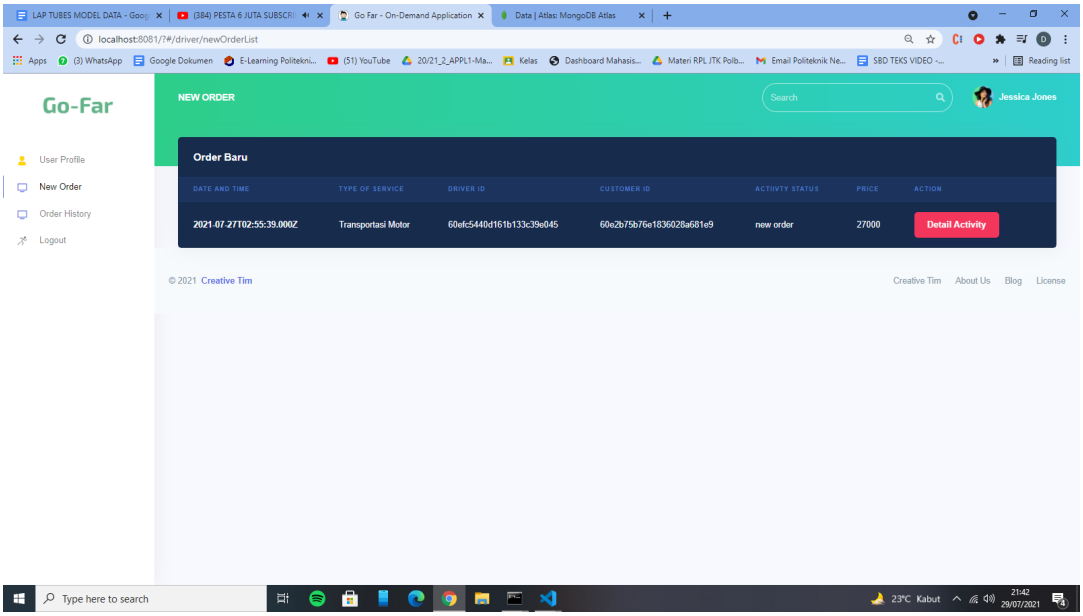


Fitur	Read Feedback (FR12)
User	Driver
Deskripsi	Melihat feedback dari *customer* setelah order selesai.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	1
Script	Bukan termasuk *query* penting

```
driverRoutes.route("/driver/read/feedback/:id").get((req, res) => {
    let db_connect = dbo.getDb("employees");
    var myquery = { _id: new mongodb.ObjectID(req.params.id) };
    db_connect
      .collection("Feedback")
      .find(myquery)
      .toArray(function (err, result) {
        if (err) throw err;
        res.json(result);
      });
  });
```

- List New Order

Tabel 37 Fitur List New Order

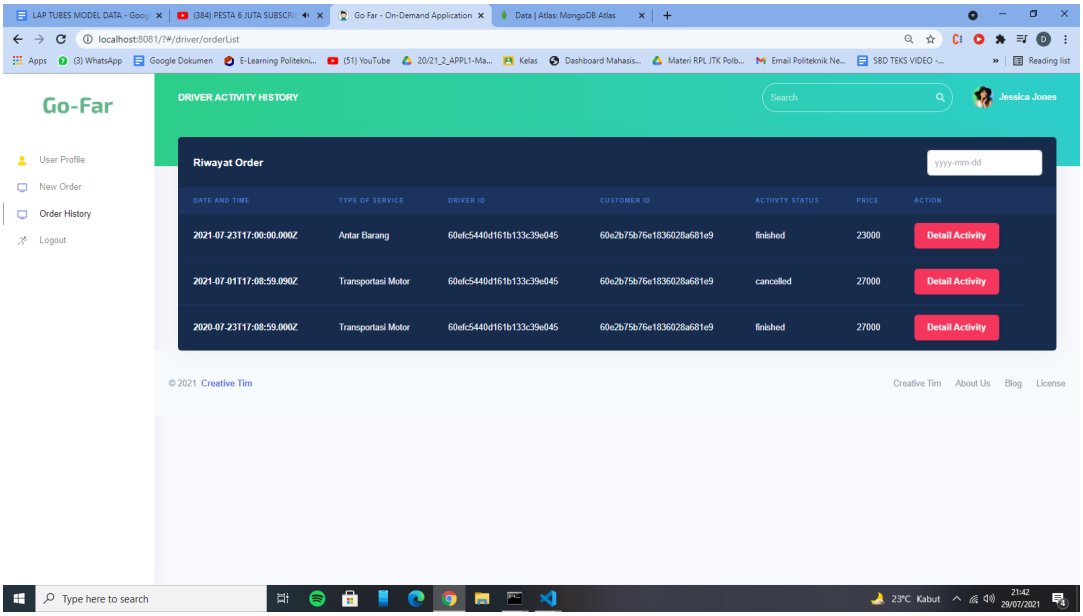


Fitur	Get New Order (FR4)
User	Driver
Deskripsi	Menampilkan data order baru/yang statusnya *'new order'*
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	1
Script	Termasuk *query* penting

```
driverRoutes.route("/driver/get/neworder/:id").get(function (req, res) {
  let db_connect = dbo.getDb("employees");
  let myquery = { id_driver: req.params.id, activity_status: 'new order'
};
  db_connect
    .collection("ActivityHistory")
    .find(myquery)
    .toArray(function (err, result) {
      if (err) throw err;
      res.json(result);
    });
});
```

- Get All Order History

Tabel 38 Fitur Driver Get All Order History

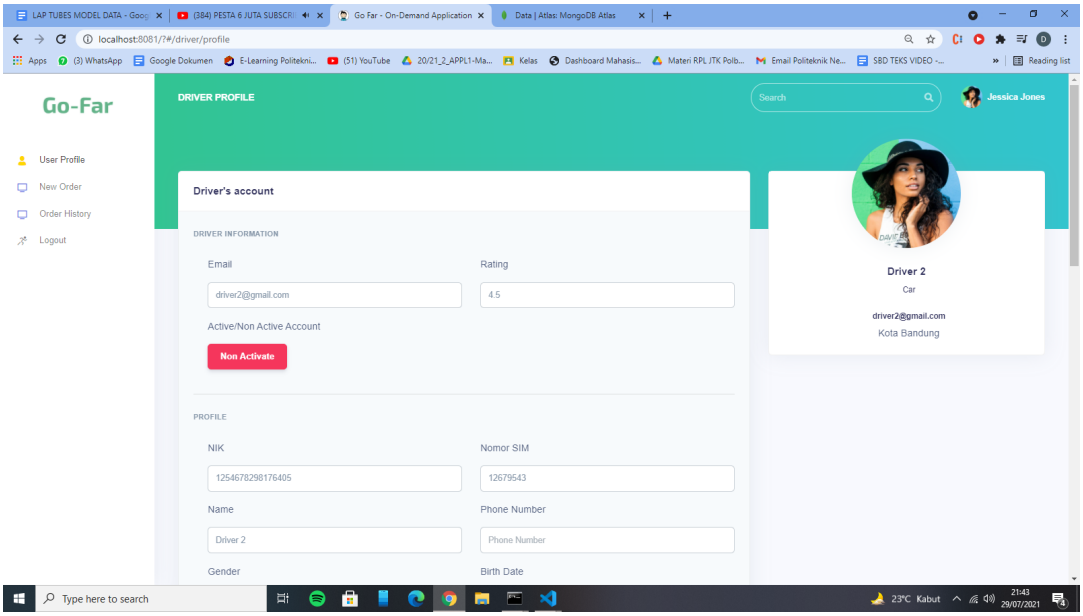


Fitur	Get All Order History (FR7)
User	Driver
Deskripsi	Menampilkan seluruh data riwayat history *driver* yang statusnya *finished* maupun *cancelled.*
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	1
Script	Bukan termasuk *query* penting

```
driverRoutes.route("/driver/get/history/:id").get(function (req, res) {
    let db_connect = dbo.getDb("employees");
    let myquery = { id_driver: req.params.id, activity_status: 'finished'
};
    db_connect
      .collection("ActivityHistory")
      .find(myquery)
      .toArray(function (err, result) {
        if (err) throw err;
        res.json(result);
      });
  });
```

- Kelola Data Diri Driver

Tabel 39 Fitur Kelola Data Diri Driver



Fitur	• Read Profile (FD9) • Update Profil (FD5) • Update Active Status (FD10)
User	Driver
Deskripsi	Melihat data diri driver dan update data diri yang hanya dapat di submit satu kali sebelum di verifikasi oleh admin. Dan update *active status* driver.
Jumlah Koleksi yang Dibaca	1
Jumlah Proses Query	3
Script	Termasuk *query* penting

```
driverRoutes.route("/driver/read/profile/:id").get((req, res) =>
{
    let db_connect = dbo.getDb("employees");
    var myquery = { _id: new mongodb.ObjectID(req.params.id) };
    db_connect
      .collection("DataDriver")
      .find(myquery)
      .toArray(function (err, result) {
        if (err) throw err;
        res.json(result);
      });
  });
```

```
driverRoutes.route("/driver/update/profile/:id").post(function
(req, res) {
    let db_connect = dbo.getDb("employees");
    var myquery = { _id: new mongodb.ObjectID(req.params.id),
submitted: false };
    let newvalues = {
      $set: req.body
    };
    db_connect
      .collection("DataDriver")
      .updateOne(myquery, newvalues, function (err, result) {
        if (err) throw err;
        console.log("1 document updated");
        res.status(201).json({result, message : "Updated
Succesfully"})
      });
  });


driverRoutes.route("/driver/update/active/:id").post(function
(req, res) {
    let db_connect = dbo.getDb("employees");
    var myquery = { _id: new mongodb.ObjectID(req.params.id) };
    let _active_status = req.body.active_status;
    let newvalues = {
      $set: {
        active_status: _active_status
      },
    };
    db_connect
      .collection("DataDriver")
      .updateOne(myquery, newvalues, function (err, result) {
        if (err) throw err;
        res.status(201).json({result, message : "Updated
Succesfully"})
      });
  });
```

4.2 Eksperimen

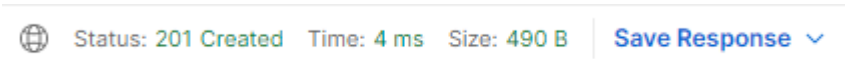
4.2.1 Rancangan eksperimen

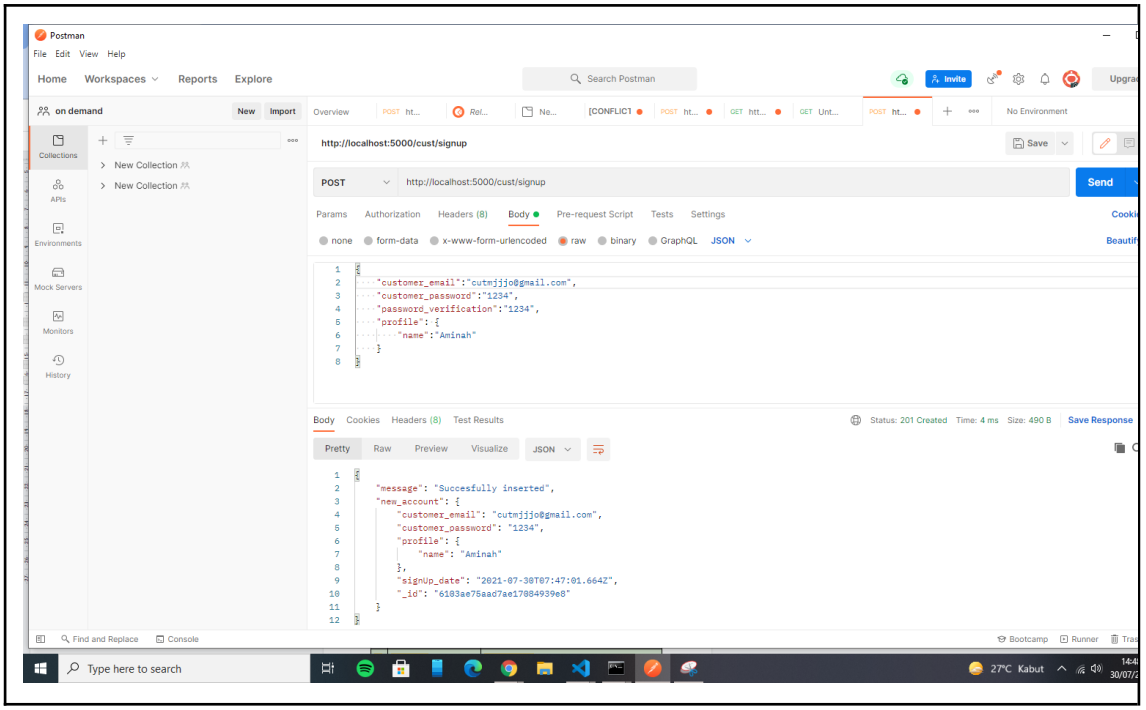
Kami melakukan percobaan terhadap beberapa query yang ada analisa workload pada bagian sebelumnya, untuk melihat apakah *response time* sistem memenuhi/kurang dari hasil analisis kuantifikasi pada analisis workload. Query yang akan dilakukan percobaan diantaranya adalah *Add Customer, Load Driver, Create Order, Load Order Detail, Update Order Status,* dan *Load Activity (Search by Category).*

Kami menggunakan *dummy data* dengan jumlah tertentu untuk menyesuaikan dengan analisa quantification. *Response Time* dilihat dari *response time* yang ada pada Postman ketika melakukan request. Eksperimen dilakukan setelah indexing pada database. Berikut merupakan hasil eksperimen yang kami lakukan.

4.2.2 Hasil Eksperimen

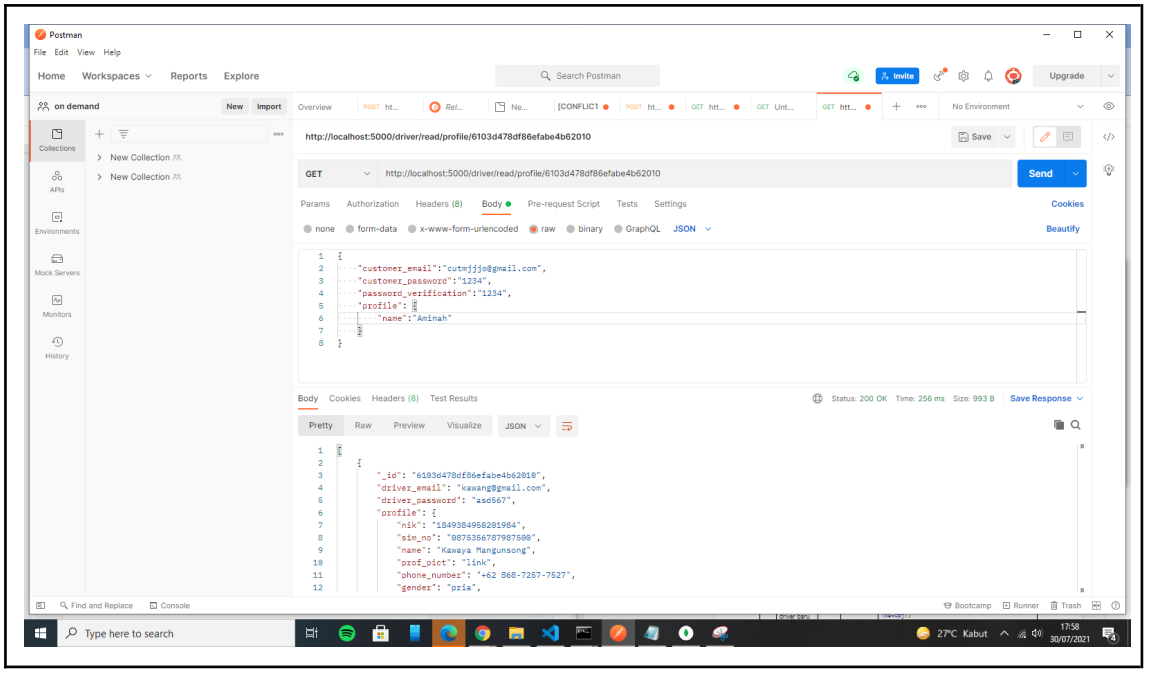
Tabel 40 Eksperimen Add Customer

Query	Add Customer
Description	Customer Sign Up.
Analysis Quantification	5000 write/day < 500 ms
Jumlah Percobaan	10 kali
Rata-rata Response Time Aplikasi	7.1 ms **(sesuai workload)**
Rentang Response Time	2 ms - 14 ms
Screenshot hasil (salah satu percobaan)	
	



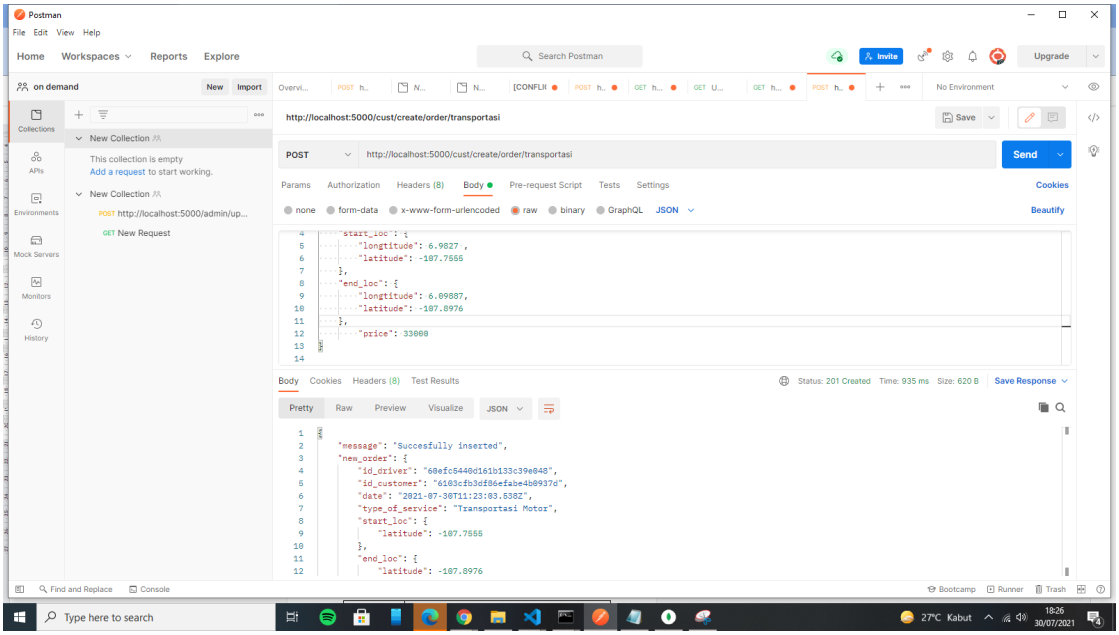
Tabel 41 Load Driver

Query	Load Driver
Description	Driver Log In aplikasi dan customer melihat profil driver yang menangani pesanannya.
Analysis Quantification	25 read/second < 300 ms
Jumlah Existing Document Driver	757 documents
Jumlah Percobaan	10
Rata-rata Response Time Aplikasi	275.4 ms **(sesuai workload)**
Rentang response time	275.4 ms 275.4 ms
Screenshot hasil (salah satu percobaan)	
Status: 200 OK Time: 256 ms Size: 993 B Save Response	



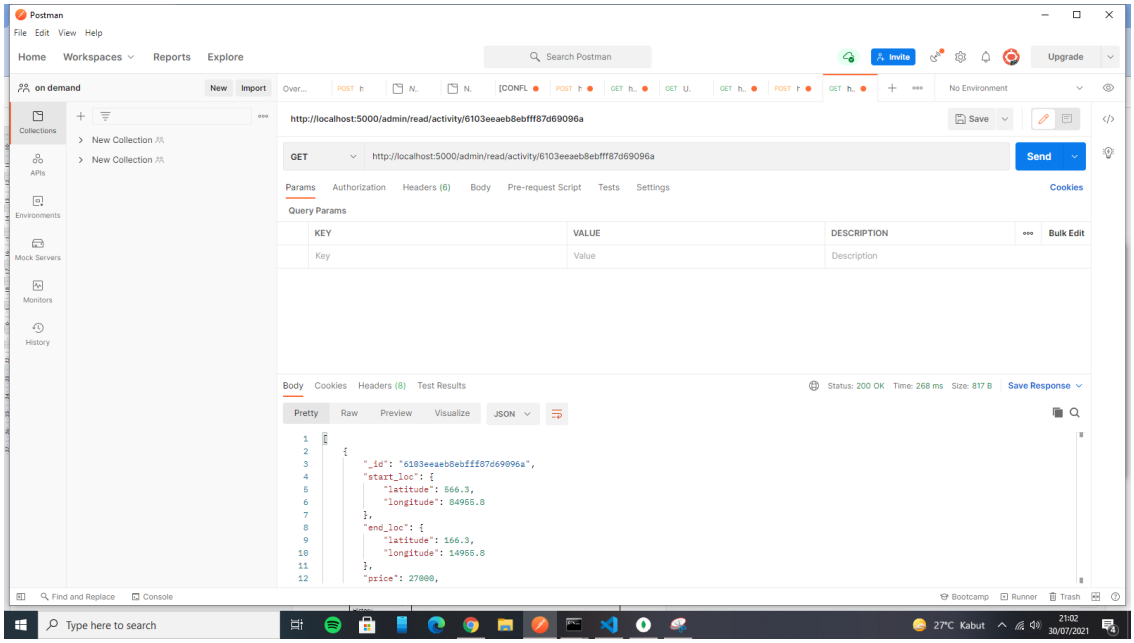
Tabel 42 Create Order

Query	Create Order
Description	Customer melakukan order salah satu layanan yang tersedia. Pemilihan driver dilakukan pada query ini, secara random mengambil driver yang memenuhi syarat active_status true, dan kendaraannya sesuai dengan jenis pesanan.
Analysis Quantification	18 write/second < 300 ms
Jumlah Percobaan	10
Jumlah Existing Document DataDriver	757 documents
Rata-rata Response Time Aplikasi	445 ms **(melebihi analisis kuantifikasi workload)**
Rentang response time	259 ms - 1.231 ms
Screenshot hasil (salah satu percobaan)	
Status: 201 Created Time: 935 ms Size: 620 B Save Response	



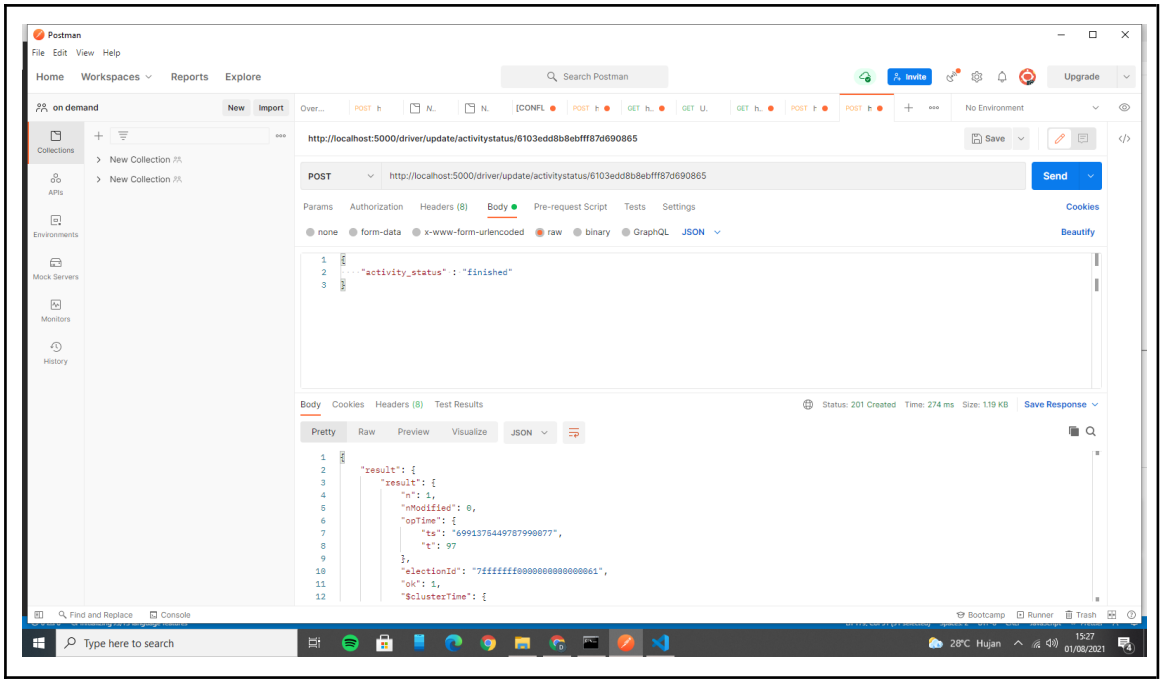
Tabel 43 Load Order Detail

Query	Load Order Detail
Description	Customer maupun Driver membuka informasi terkait order. Menampilkan data customer, driver, lokasi awal, lokasi akhir, dan detail barang apabila merupakan layanan antar barang.
Analysis Quantification	36 read/second < 300 ms
Jumlah Existing Document Activity History	442 + 516 + 566 + 291 + 616 +582 + 659 + 337 + 756 + 299
Jumlah Percobaan	10
Rata-rata Response Time Aplikasi	506.4 ms **(melebihi dari hasil analisa kuantifikasi workload)**
Rentang response time	291 ms - 756 ms
Screenshot hasil (salah satu percobaan)	
Status: 200 OK Time: 268 ms Size: 817 B Save Response	



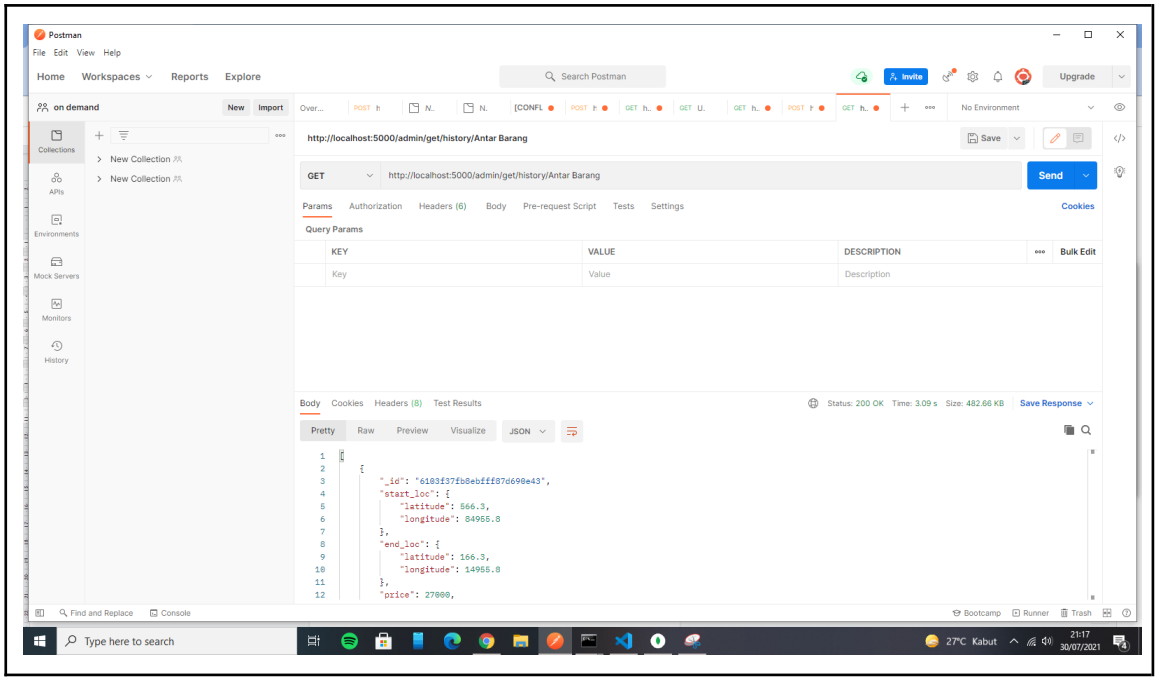
Tabel 44 Update Activity Status

Query	Update Activity Status
Description	Mengupdate status order dari mulai driver dalam perjalanan, hingga order selesai. Dalam satu order hingga selesai, terjadi 3-4 kali perubahan status order (berbeda tergantung jenis layanan).
Analysis Quantification	54 write/second < 300 ms
Jumlah Existing Document Activity History	2725
Jumlah Percobaan	10
Rata-rata Response Time Aplikasi	282.1 ms **(sesuai dengan analisis kuantifikasi workload)**
Rentang response time	272 ms - 308 ms
Screenshot hasil (salah satu percobaan)	
Status: 201 Created Time: 274 ms Size: 1.19 KB Save Response	



Tabel 45 Load Activity (Search by Category)

Query	Load Activity (Search by Category)
Description	Melakukan pencarian terhadap aktivitas berdasarkan category.
Analysis Quantification	<2000 ms
Jumlah Existing Document Activity History	2725
Jumlah Percobaan	10
Rata-rata Response Time Aplikasi	1.82 s (**sesuai dengan analisis kuantifikasi workload)** Keterangan : Sebelum indexing response time kisaran 2 - 11 seconds
Rentang response time	1 s - 2.9 s
Screenshot hasil (salah satu percobaan)	
Status: 200 OK Time: 3.09 s Size: 482.66 KB Save Response	


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Invite
Upgrade
on demand
New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
New Collection
New Collection
No Environment
http://localhost:5000/admin/get/history/Antar Barang
Save
GET
http://localhost:5000/admin/get/history/Antar Barang
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 3.09 s Size: 482.66 KB
Save Response
Pretty Raw Preview Visualize JSON
[
{
"_id": "6103f37fb8ebfff87d698e43",
"start_loc": {
"latitude": 566.3,
"longitude": 84955.8
},
"end_loc": {
"latitude": 166.3,
"longitude": 14955.8
},
"price": 27000,
Find and Replace
Console
Bootcamp
Runner
Trash
Type here to search
27°C Kabut
21:17
30/07/2021

5. KESIMPULAN DAN LESSON-LEARNED

5.1 Kesimpulan

Aplikasi layanan *On Demand* ini dirancang dengan menggunakan database NoSQL yaitu MongoDB, yang merupakan database berorientasi dokumen (*document based).* Aplikasi ini terinspirasi dari aplikasi layanan *on demand* lainnya seperti Gojek dan Grab yang semakin banyak digunakan karena mempermudah masyarakat dalam menjangkau suatu tempat dan meningkatkan produktivitas. Aplikasi ini sendiri menyediakan 3 fitur layanan, diantaranya Transportasi Motor, Transportasi Mobil, dan Layanan Antar Barang.

Kami menggunakan MEVN Stack untuk membangun aplikasi, MEVN sendiri merupakan singkatan dari MongoDB, Express.js, Vue.js, dan Node.js. Untuk *User Interface*, kami menggunakan template Vue.js & Bootstrap 4 yang sudah tersedia secara gratis dari https://www.creative-tim.com untuk mengefektifkan waktu dalam perancangan tampilan.

Dalam pembuatan aplikasi, masih terdapat beberapa kekurangan diantaranya adalah:

- Keamanan (*Security*)

  Tidak diperlukannya/ tidak adanya *authentication* dan *authorization* untuk mengakses Web API. *Authorization* memiliki tujuan untuk membatasi perilaku yang hanya dapat dilakukan oleh user tertentu. Dengan tidak adanya hal tersebut, maka hal tersebut (modul yang dibatasi) dapat digunakkan oleh siapa saja tanpa autorisasi, sehingga tingkat keamanan aplikasi masih rendah.

- Upload Data File belum berhasil

  Untuk memenuhi kebutuhan aplikasi, terdapat beberapa file yang memiliki value dengan tipe data file (dokumen/foto). Namun dalam implementasi yang kami lakukan, Web API, tidak dapat menerima/membaca object dengan tipe form-data. Sedangkan file sendiri dapat dikirim dengan *content type* multipart/form-data. Permasalah ini belum dapat terselesaikan.

- Kalkulasi biaya layanan belum akurat

  Dalam eksperimen yang dilakukan, kami menyadari bahwa hasil koordinat geometri yang dihasilkan ketika melakukan pencarian alamat tidaklah akurat, bahkan satu alamat dapat menghasilkan koordinat geometri yang berbeda setelah beberapa saat. Sehingga biaya yang dikeluarkan juga memiliki angka yang tidak masuk akal/tidak akurat.

- Redundant function pada sisi *back end*

  Selain itu kami menyadari kekurangan dalam membangun *back end* aplikasi dimana banyaknya function dengan fungsi yang sama karena router dibuat per *user* sedangkan setiap

user memiliki beberapa fungsi yang sama. Menurut kami akan lebih efektif jika router dikelompokan berdasarkan *collection.*

5.2 Lesson-Learned

Dengan tugas besar mata kuliah Pemodelan Data ini, kelompok kami mendapatkan lebih banyak pelajaran mengenai bagaimana kerjasama tim yang baik terutama ketika masing-masing dari anggota kami terpisahkan oleh jarak karena situasi pandemi yang belum usai. Kami mempelajari bahwa kerja sama tim yang baik dipengaruhi oleh komunikasi yang baik, sehingga tim kami menyelaraskan pemahaman terhadap ide tugas besar agar tidak adanya salah tangkap atau perbedaan pemahaman antar anggota kelompok ketika pengerjaan. Selain itu, tentu saja dengan adanya tugas besar ini membuat masing-masing anggota dari kami menjadi lebih memahami tentang hal-hal yang berkaitan dengan pemodelan data. Pengerjaan tugas ini menitikberatkan pada perancangan data yang baik, oleh karena itu kami banyak mempelajari hal-hal yang perlu di pertimbangkan dalam merancang database menggunakan NoSQL, seperti menentukan dan menganalisis setiap collection, merancang pengolahan data yaitu membuat/menentukan *Collection Relationship Diagram* (CRD) serta menentukan kamus data, indexing, dan masih banyak lagi.

Dalam proses pengimplementasian pada website, kami juga banyak mempelajari *tools tools* baru dalam membangun aplikasi, bagi itu dari segi *backend* (MongoDB, Node Js, Express.js) , *frontend* (Vue.js), dan lain lain.

Selain itu pula, karena studi kasus yang kami angkat adalah Aplikasi Layanan *On Demand*, kami juga mendapatkan pengetahuan mengenai bagaimana gambaran sistem aplikasi layanan *on demand* serta kebutuhan-kebutuhannya. Kami juga mempelajari pentingnya membuat rancangan yang baik yaitu dengan dibuatnya laporan ini, sehingga akan mempermudah kami saat melakukan implementasi.

5.3 Kendala dan Solusi

Tabel 46 Kendala dan Solusi

Kendala	Solusi
Post data melalui Front End (data json *undefined*)	Pendefinisian 'Content-Type' yang salah ketika melakukan post. 'Content-type' diubah menjadi 'application/json' sehingga post berhasil.
Pendefinisian perancangan collection terkait data alamat	Menggunakan bantuan maps api untuk alamat sehingga bisa mendapatkan koordinat geometri untuk kalkulasi biaya layanan.

Menyambungkan website dengan API Google Maps	Menggunakan https://opencagedata.com/api.
Tidak bisa membaca format *form-data*	Belum Solve sehingga tidak dapat mengunggah file (dokumen/gambar) karena harus menggunakan format multipart/form-data untuk mengunggah file
Menggunakan Maps API Javascript	Ada pada dokumentasi Maps API Javascript, namun belum bisa menggunakan maps untuk mencari titik penjemputan/tujuan secara manual.

6. REFERENSI

Dokumentasi Mongodb. Data Modelling Introduction. Diakses 17 Mei 2021 melalui https://docs.mongodb.com/manual/core/data-modeling-introduction/.

Dokumentasi Mongodb.Advanced Schema Design Patterns. Diakses 17 Mei 2021 melalui https://www.mongodb.com/presentations/advanced-schema-design-patterns.

Kumparan. 2020. Organisasi Ojol: Ada 4 bJuta Driver Ojol di Indonesia. Diakses 30 Juni 2021 melalui https://kumparan.com/kumparantech/organisasi-ojol-ada-4-juta-driver-ojol-di-indonesia-1tBrZLEXOEI/full.

Astutik, Y. 2020. 21,7 Juta Masyarakat Indonesia Pakai Transportasi Online. Diakses 30 Juni 2021 https://www.cnbcindonesia.com/tech/20200317150135-37-145529/217-juta-masyarakat-indonesia-pakai-transportasi-online.

Franedya, R. 2019. Fakta-fakta yang Orang Tidak Tahu Soal Gojek, 2 Juta Driver?. Diakses 30 Juni 2021 https://www.cnbcindonesia.com/tech/20190405134149-37-64882/fakta-fakta-yang-orang-tidak-tahu-soal-gojek-2-juta-driver.

Mongodb. How to Use MERN Stack: A Complete Guide. Diakses 12 Juni 2020 melalui https://www.mongodb.com/languages/mern-stack-tutorial.

Nielsen, J. 1993. *Usability Engineering.* San Francisco. Elsevier Science & Technology.

https://www.littledata.io/average/server-response-time