SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

RANCANG BANGUN APLIKASI WEB UNTUK PELACAKAN KENDARAAN MENGGUNAKAN NODEJS DAN *FRAMEWORK* LARAVEL

TUGAS AKHIR

RAFIF MULIA RESWARA
0110220260

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI
AGUSTUS 2024

STT TERPADU
NURUL FIKRI

SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

RANCANG BANGUN APLIKASI WEB UNTUK PELACAKAN KENDARAAN MENGGUNAKAN NODEJS DAN *FRAMEWORK* LARAVEL

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer

RAFIF MULIA RESWARA
0110220260

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI
AGUSTUS 2024

HALAMAN PERNYATAAN ORISINALITAS

Skripsi/Tugas Akhir ini adalah hasil karya penulis,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama : Rafif Mulia Reswara

NIM : 0110220260

Depok, 27 Juli 2024

Tanda Tangan

Rafif Mulia Reswara

iii

HALAMAN PENGESAHAN

Skripsi/Tugas Akhir ini diajukan oleh :

Nama	:	Rafif Mulia Reswara
NIM	:	0110220260
Program Studi	:	Teknik Informatika
Judul Skripsi	:	Rancang Bangun Aplikasi Web Untuk Pelacakan Kendaraan Menggunakan NodeJS dan *Framework* Laravel

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika Sekolah Tinggi Teknologi Terpadu Nurul Fikri

DEWAN PENGUJI

Pembimbing	Penguji
Dr. Lukman Rosyidi, ST., MM.	Dr. Sirojul Munir S.Si., M.Kom

Ditetapkan di : Depok
Tanggal : 27 Juli 2024.

iv

KATA PENGANTAR

Dengan penuh rasa syukur, penulis mengawali ungkapan terima kasih kepada Allah SWT yang telah memberikan berkah dan rahmat-Nya sehingga penulis berhasil menyelesaikan Tugas Akhir ini. Penulisan Tugas Akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Komputer Program Studi Teknik Informatika pada Sekolah Tinggi Teknologi Terpadu Nurul Fikri. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan tugas akhir ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Allah SWT.
2. Orang tua dan semua anggota keluarga yang telah memberikan dorongan baik secara moril maupun materiil dalam penyelesaian tugas ini.
3. Bapak Dr. Lukman Rosyidi, M.T., M.M. selaku Ketua Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
4. Ibu Tifani Nabarian, S.Kom., M.T.I. selaku Ketua Program Studi Teknik Informatika Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
5. Bapak Henry Saptono, S.Si., M.Kom. selaku Dosen Pembimbing Akademik yang telah membimbing penulis selama berkuliah di Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
6. Bapak Dr. Lukman Rosyidi, M.T., M.M. selaku Dosen Pembimbing Tugas Akhir penulis dalam menyelesaikan penulisan ilmiah ini.
7. Para Dosen di lingkungan Sekolah Tinggi Teknologi Terpadu Nurul Fikri yang telah membimbing penulis dalam menuntut ilmu yang telah diberikan.
8. Bapak Agus Prihanto selaku *Vice President* dari PT. Swa Nusa Multimedia yang telah memberikan izin untuk dijadikannya laporan karya tulis ilmiah.
9. PT. Swa Nusa Multimedia, Manajer beserta karyawan yang telah meluangkan waktunya untuk memberikan data yang diperlukan bagi penulisan ilmiah ini.

Dalam penulisan ilmiah ini tentu saja masih banyak terdapat kekurangan-kekurangan yang mungkin disebabkan oleh keterbatasan kemampuan dan pengetahuan yang penulis miliki. Walaupun demikian, penulis telah berusaha menyelesaikan penulisan ilmiah ini sebaik mungkin. Oleh karena itu apabila terdapat kekurangan di dalam penulisan ilmiah ini, dengan rendah hati penulis menerima kritik dan saran dari pembaca.

Akhir kata, penulis berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 27 Juli 2024

Rafif Mulia Reswara

vi

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Sekolah Tinggi Teknologi Terpadu Nurul Fikri, saya yang bertanda tangan di bawah ini:

Nama : Rafif Mulia Reswara
NIM : 0110220260
Program Studi : Teknik Informatika
Jenis Karya : Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada STT-NF **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty - Free Right*)** atas karya ilmiah saya yang berjudul :

"Rancang Bangun Aplikasi Web Untuk Pelacakan Kendaraan Menggunakan NodeJS dan *Framework* Laravel".

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini STT-NF berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.
Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 27 Juli 2024
Yang Menyatakan

Rafif Mulia Reswara

vii

ABSTRAK

Nama : Rafif Mulia Reswara

NIM : 0110220260

Program Studi : Teknik Informatika

Judul : Rancang Bangun Aplikasi Web Untuk Pelacakan Kendaraan Menggunakan NodeJS dan *Framework* Laravel.

Pengantaran barang merupakan salah satu aspek penting dalam kehidupan saat ini. Di era revolusi industri 4.0, pelacakan erat kaitannya dengan pengantaran barang. Dengan adanya *GPS*, proses pengantaran dapat menjadi lebih transparan. Cara yang cukup sering ditemukan adalah dengan memanfaatkan *smartphone*, namun cara ini menggunakan daya pada *smartphone* lebih banyak untuk pengiriman jarak jauh. Tujuan dari penelitian ini adalah melacak kendaraan dengan memanfaatkan perangkat *GPS Tracker*, sehingga sopir dapat memanfaatkan *smartphone* miliknya untuk produktivitas lainnya tanpa khawatir konsumsi daya yang boros. Metode pengembangan perangkat lunak yang digunakan adalah *scrum* dan *Unified Modelling Language* (*UML*) untuk perancangannya. Hasil dari penelitian ini berupa aplikasi web yang terbagi menjadi dua bagian, yaitu *backend service* menggunakan NodeJS dan *frontend* menggunakan *framework* Laravel. Hasil dari *backend service* NodeJS adalah bisa menerima data dari perangkat *GPS Tracker* secara *online* sesuai dengan spesifikasi protokol GT06 serta mekanisme sesi setiap koneksi yang datang, mendapatkan data alamat lengkap dengan memanfaatkan *API* dari OpenStreetMap dengan melakukan *reverse geocoding* dari data *latitude* dan *longitude*, dan menyimpannya ke basis data MariaDB dengan hasil pengujian *black box* didapatkan sebesar 76%. Dilanjutkan dengan hasil dari Laravel yang menampilkan peta secara interaktif dengan memanfaatkan *MapsAPI*, memanfaatkan metode *client-polling* untuk mendapatkan lokasi terkini secara *real-time*, serta membuat fitur manajemen kendaraan, manajemen pengemudi, dan manajemen perangkat untuk memudahkan admin menggunakan sistem dan mengelolanya dengan hasil pengujian *black box* didapatkan sebesar 100%.

Kata kunci : *GPS*, kendaraan, pelacakan, PT Swa Nusa Multimedia, web.

ABSTRACT

Name	:	*Rafif Mulia Reswara*
NIM	:	0110220260
Study Program	:	*Informatics Engineering*
Title	:	*Web Application Building for Vehicle Tracking Using NodeJS and Laravel Framework*

Goods delivery is an important aspect in life nowadays. In industry revolution 4.0, tracking is closely related to goods delivery. With GPS, delivery process will be more transparent. The method that is often is by using a smartphone. However, this method consumes more energy on smartphone during a long trip. The purpose of this research to track vehicle using GPS Tracker device, so that driver can use his smartphone for other productivity without worrying about the energy is drained. The development method uses scrum and Unified Modelling Language (UML) for architecting it. The results of this research a web application that is divided into two parts, first is backend service that uses NodeJS and the second is frontend that uses Laravel framework. The results of NodeJS as backend service are receiving data from GPS Tracker online based on GT06 specification protocol and session mechanism for every incoming connection, gets the complete address data from OpenStreetMap API that uses reverse geocoding from latitude and longitude, and stores all of them to MariaDB database and had been testing using black box at 76%. Then the results of Laravel that shows the map interactively that uses MapsAPI, use client-polling method for getting real-time location data, and create features like vehicle management, driver management, and device management that makes it easier for admin to use the system and manage it and had been testing using black box at 100%.

Key words : GPS, PT Swa Nusa Multimedia, tracking, vehicle, web.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR RUMUS

BAB I
PENDAHULUAN

1.1 Latar Belakang

Era revolusi industri 4.0 membawa perubahan di segala sektor di kehidupan manusia, salah satunya mengenai transportasi. Transportasi merupakan sebuah proses memindahkan manusia dan barang dari satu lokasi ke lokasi lainnya [1]. Dengan adanya transportasi dapat meningkatkan mobilitas setiap orang akibat aktivitas ekonomi, sosial, dan sebagainya yang memerlukan kecepatan waktu. Seperti jasa ojek atau pengiriman barang. Sebelumnya, jika ingin menggunakan jasa transportasi mengharuskan melakukan kontak langsung secara fisik atau melalui panggilan telepon [1]. Namun saat ini, hanya dengan memanfaatkan *smartphone* Android dapat membantu manusia menggunakan jasa transportasi yang dibutuhkannya.

Teknologi *Global Positioning System* (*GPS*) juga berkontribusi dalam hal revolusi industri 4.0 pada sektor industri transportasi. Dengan adanya *GPS*, memungkinkan sebuah objek dapat diketahui keberadaannya dengan memanfaatkan sinyal satelit. Hal ini dimanfaatkan para pelaku di industri transportasi seperti melacak proses pengiriman dan pengantaran barang, atau memberikan estimasi waktu untuk sampai ke tempat tujuan. Teknologi *GPS* dapat diterapkan pada *smartphone* Android. Salah satunya adalah Gojek, yang memanfaatkan *Mobile Location-Based Service* (*m-LBS*) yang diterapkan dalam layanan *Go-Food* [2]. Namun metode ini mengonsumsi daya yang cukup besar, sehingga tidak cocok untuk pengiriman logistik yang membutuhkan perjalanan jauh [3]. Sehingga dibutuhkan solusi lain untuk menghemat konsumsi daya, yaitu dengan memanfaatkan perangkat *GPS Tracker* yang dapat dipasang pada mesin kendaraan.

PT. Swa Nusa Multimedia merupakan perusahaan yang bergerak di sektor industri transportasi sebagai pelayanannya untuk melayani kebutuhan logistik dan

pelanggan tanpa membedakan jumlah besar kecilnya pengiriman barang yang dilayani dengan memanfaatkan armada yang tersedia. PT. Swa Nusa Multimedia membutuhkan sistem pelacakan kendaraan guna bersaing di dunia industri transportasi dan bertahan di era revolusi industri 4.0. Dengan memanfaatkan perangkat *GPS Tracker* untuk melacak keberadaan kendaraan, dan aplikasi web untuk perolehan data lokasi dari perangkat *GPS Tracker*, serta memvisualisasikannya ke dalam bentuk peta secara *real-time* dan interaktif. Sehingga memudahkan PT. Swa Nusa Multimedia dalam proses pelacakan dan pemantauan kendaraannya.

Berdasarkan permasalahan di atas penulis mengambil penelitian yang berjudul "Rancang Bangun Aplikasi Web Untuk Pelacakan Kendaraan Menggunakan NodeJS dan *Framework* Laravel" untuk membantu PT Swa Nusa Multimedia bersaing di era revolusi industri 4.0.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dijelaskan maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

a. Bagaimana merancang aplikasi web untuk mengumpulkan data posisi kendaraan secara *online* dengan memanfaatkan *GPS*?
b. Bagaimana merancang aplikasi web untuk melakukan manajemen dan pemantauan kendaraan secara *real-time*?

1.3 Tujuan dan Manfaat Penelitian

Adapun tujuan daripada penelitian ini adalah sebagai berikut:

a. Membuat *backend service* untuk memperoleh data dari perangkat *GPS Tracker* serta mendesain basis data untuk mengolah datanya.
b. Membuat aplikasi web dengan *framework* Laravel untuk memantau kendaraan dengan menerapkan metode *client polling* dan untuk manajemen terkait: data kendaraan, data pengemudi, dan data perangkat (*GPS Tracker*).

Adapun manfaat daripada penelitian ini adalah sebagai berikut:

a. Memudahkan PT Swa Nusa Multimedia dalam hal melacak dan memantau kendaraan.
b. Memudahkan PT Swa Nusa Multimedia dalam hal manajemen kendaraan, manajemen perangkat (*GPS Tracker*), dan manajemen pengemudi.
c. Membantu PT Swa Nusa Multimedia bersaing di era revolusi industri 4.0.
d. Membantu akademisi dalam hal penelitian persoalan pelacakan kendaraan dengan memanfaatkan perangkat *GPS Tracker*.

1.4 Batasan Masalah

Adapun batasan masalah daripada penelitian ini adalah sebagai berikut:

a. Luas dan lingkup masalah hanya mencakup pada PT Swa Nusa Multimedia.
b. Tidak membahas proses transaksi atau sampai melakukan *order* atau pemesanan.
c. Tidak membahas mengenai manajemen logistik dan pengantaran logistik. Dikarenakan yang dijadikan penelitian adalah pelacakan kendaraan.
d. Perangkat *GPS Tracker* hanya terbatas pada model GT06.
e. Tidak membahas komponen-komponen perangkat elektronik *GPS Tracker*. Dikarenakan cakupan daripada penelitian ini adalah memperoleh, menyimpan, memperbaiki, memanipulasi, atau menampilkan informasi yang bersifat geografis.
f. Fitur yang terdapat pada aplikasi web berupa pelacakan kendaraan, pemantauan kendaraan secara *real-time*, riwayat perjalanan, perhitungan jarak tempuh kendaraan, manajemen kendaraan, manajemen perangkat (*GPS Tracker*), manajemen pengemudi, dan *multi-role user*.
g. Tidak semua jenis kendaraan diuji, hanya menggunakan 1 jenis kendaraan ketika pengujian, yaitu dengan kendaraan bermotor roda dua. Karena yang diuji adalah pelacakannya, bukan tentang kompatibel atau tidaknya perangkat *GPS Tracker* pada semua jenis kendaraan.

h. Tidak membahas instalasi perangkat *GPS Tracker* pada kendaraan. Dikarenakan jenis kendaraan tidak hanya satu dan instalasi perangkat *GPS Tracker* berhubungan dengan bagian kelistrikan kendaraan yang di luar cakupan keilmuan dan penelitian penulis.
i. Data yang didapatkan terbatas yang disediakan oleh *GPS Tracker* yaitu *latitude*, dan *longitude*. Sehingga peta yang digunakan adalah peta 2d, bukan 3d (dikarenakan tidak memiliki data *altitude*).
j. Tidak membahas mengenai perancangan *User Interface dan User Experience* (*UI/UX*), dikarenakan lebih memfokuskan kepada fungsionalitas aplikasi.

1.5 Sistematika Penulisan

Adapun sistematika penulisan pada penelitian ini adalah sebagai berikut.

Bab I Pendahuluan

Pada bab ini membahas mengenai latar belakang yaitu menjelaskan kenapa penulis mengambil judul penelitian, rumusan masalah merupakan pokok permasalahan yang akan dibahas pada penelitian ini, tujuan dan manfaat penelitian merupakan hasil akhir yang diharapkan dari penelitian ini, dan batasan masalah merupakan hal-hal yang berada di luar cakupan penelitian.

Bab II Kajian Literatur

Pada bab ini penulis menjelaskan tentang literatur yang diperlukan seperti Sistem Informasi Geografis (SIG), *Global Positioning System* (*GPS*), *backend service*, bahasa pemrograman JavaScript, basis data MariaDB, bahasa pemrograman PHP, *Framework* Laravel, *Maps API*, *real-time*, formula haversine, metode pengembangan *agile*, *Unified Modelling Language* (*UML*), dan metode pengujian *black box* yang akan menjadi landasan berpikir dan kerangka penelitian dalam pengerjaannya. Serta penelitian terkait yang pernah dilakukan sebelumnya untuk dijadikan referensi sekaligus menghindari duplikasi.

Bab III Metodologi Penelitian

Pada bab ini menjelaskan tentang tahapan penelitian yang dilakukan. Rancangan penelitian berisi jenis penelitian pengembangan (RnD), metode analisis data pengujian adalah kuantitatif, metode pengumpulan data adalah observasi dan wawancara, metode pengujian menggunakan *black box*, metode implementasi menggunakan *scrum* dan metode evaluasi pengembangan dengan meninjau *velocity* dari *product backlog* yang dikerjakan, dan lingkungan pengembangan.

Bab IV Implementasi Dan Evaluasi

Pada bab ini menjelaskan tentang implementasi terkait tahapan dan metode yang sudah dijelaskan pada BAB III. Di antaranya implementasi perancangan sistem dan aplikasi dengan menggunakan *Unified Modelling Language* (*UML*), persiapan lingkungan pengembangan, implementasi metode pengembangan menerapkan prinsip *agile* dan penerapannya dengan *scrum*, dan implementasi metode pengujian menggunakan *black box*.

Bab V Kesimpulan Dan Saran

Pada bab ini penulis melakukan penarikan kesimpulan terkait hasil pengujian yang telah dilakukan dan evaluasi sebagai landasan dan saran untuk penelitian berikutnya.

BAB II
KAJIAN LITERATUR

2.1. Sistem Informasi Geografis

Sistem Informasi Geografis (SIG) terbentuk dari 3 definisi, yaitu:

1. Sistem yang definisinya merupakan sekumpulan komponen yang saling berkaitan dan bekerja sama untuk mencapai tujuan tertentu [4].
2. Informasi yang definisinya merupakan hasil dari pemrosesan data yang dapat digunakan untuk membantu pengambilan keputusan [4].
3. Geografis yang definisinya adalah segala hal yang berhubungan dengan lokasi di permukaan bumi [4].

Dari 3 definisi terpisah di atas, jika ditarik kesimpulannya maka Sistem Informasi Geografis (SIG) merupakan komponen-komponen yang terstruktur, saling terhubung, dan membentuk suatu kesatuan dan keterikatan untuk mengolah (memperoleh, menyimpan, memperbaiki, atau memanipulasi) data yang bersifat geografis dan menjadikannya informasi yang bermanfaat [4], [5].

Pada dasarnya SIG merupakan pemodelan konsep dunia nyata (*real world*) ke dalam suatu bentuk digital yang dapat diolah secara matematis. Tahapan pemodelan pada SIG berdasarkan buku [6] adalah sebagai berikut:

1. Realitas fisik merupakan tahapan pada saat analisis dunia nyata.
2. Model dunia nyata merupakan tahapan pada saat mengubah objek-objek dunia nyata menjadi model.
3. Model data merupakan tahapan pada saat mengubah model objek dunia nyata menjadi tipe data.
4. Basis data merupakan tahapan pada saat menyimpan data model ke dalam basis data.
5. Peta atau laporan merupakan tahapan pada saat penyajian model akhir dalam sebuah sistem.

Sistem Informasi Geografis (SIG) terdiri dari 2 data, yaitu data spasial dan data non-spasial [7]. Data spasial terdiri dari vektor dan raster data [4]. Vektor data merupakan format data spasial yang merepresentasikan bentuk bumi yang terdiri dari titik, garis, dan *polygon*. Sedangkan raster merupakan format data spasial yang merepresentasikan bentuk bumi menjadi sebuah *pixel* atau *grid cells*. Dan data non-spasial merupakan data atribut yang melekat pada objek spasial untuk menjelaskan dan menginformasikan objek spasial tersebut.

2.2. Global Positioning System (GPS)

Global Positioning System (*GPS*) pada awalnya dikembangkan oleh Departemen Pertahanan Amerika yang diperuntukkan bagi kepentingan militer. Secara definisi, *GPS* merupakan sistem yang bertujuan untuk menentukan posisi dan navigasi secara global dengan memanfaatkan satelit dan metode Triangulasi [8].

Sedangkan pengertian *Global Positioning System* (*GPS*) menurut penelitian yang dilakukan oleh [9] merupakan, "layanan berbasis lokasi yang mempertemukan 3 teknologi yaitu Sistem Informasi Geografis (SIG), layanan internet, dan perangkat seluler". Dari pengertian tersebut dapat dijelaskan bahwa *GPS* merupakan bagian dari teknologi SIG dikarenakan data yang dihasilkan bersifat geografis. Mengenai layanan internet dan perangkat seluler dikarenakan untuk menerima dan mengirim data geografis melalui internet dengan memanfaatkan sinyal seluler.

Dari kedua pengertian di atas dapat disimpulkan bahwa *GPS* merupakan bagian daripada SIG untuk menentukan posisi dan navigasi secara global dengan memanfaatkan satelit dan metode triangulasi serta layanan internet dan sinyal seluler untuk menerima dan mengirimkan data geografis.

GPS terbagi menjadi 3 bagian, yaitu bagian pengguna (*User Segment*) yang bertugas menerima data dari satelit dan memprosesnya untuk menentukan posisi, arah, jarak, dan waktu. Bagian satelit (*Space Segment*) yang bertugas untuk menerima dan menyimpan data yang ditransmisikan oleh stasiun-stasiun pengendali, menyimpan dan menjaga informasi waktu dengan ketelitian tinggi (jam atom di satelit), dan memancarkan sinyal dan informasi secara terus menerus ke

perangkat penerima. Saat ini terdapat 24 buah satelit yang mengorbit bumi, terdiri dari 21 buah yang aktif bekerja dan 3 buah sisanya sebagai cadangan. Terakhir bagian pengendali (*Control Segment*) yang bertugas untuk mengendalikan satelit dari bumi untuk melihat keadaan satelit, penentuan serta prediksi orbit, sinkronisasi waktu antar satelit, dan mengirimkan data ke satelit [8].

Pada bagian pengguna (*User Segment*) untuk mendapatkan posisi 2d atau titik koordinat (*latitude and longitude*) dibutuhkan minimal mengunci 3 satelit. Jika pada bagian pengguna (*User Segment*) dapat menerima 4 atau lebih satelit, maka dapat menghitung posisi 3d (*latitude, longitude, and altitude*) [9].

2.3. Real-Time

Real-time merupakan istilah dan konsep yang masih belum ada definisi secara pasti. Namun menurut [10], *real-time* dapat didefinisikan menjadi 3 definisi berdasarkan konteksnya, yaitu:

1. *Digital Real-time Signal Processing*

   Merupakan perangkat keras dan perangkat lunak yang didesain untuk menyelesaikan sebuah tugas dalam periode waktu yang telah dispesifikasikan.
2. *Real-Time System*

   Merupakan sistem yang harus menyajikan hasil logis yang tepat, namun sangat penting untuk menyajikannya dalam kurun waktu yang telah ditentukan.
3. *Real-Time Data Stream*

   Merupakan sistem yang harus menyediakan pemrosesan dalam jumlah volume data yang besar, memiliki latensi yang rendah, dan *response* yang cepat.

Dalam pengembangan aplikasi web, perolehan data secara *real-time* dapat dicapai dengan melakukan *request* secara *asynchronous*. Penerapannya dengan metode *client polling*, yaitu melakukan *request* secara berkala dari sisi *client* untuk selalu mendapatkan data terbaru. Metode *client polling* dapat dilakukan dengan

menggunakan *API* atau komunikasi *HTTP* ke sisi server, serta mendapatkan umpan balik berupa data *JavaScript Object Nation* (*JSON*).

2.4. Backend Service

Backend service merupakan sebuah layanan yang menerima permintaan dari *client* dan memprosesnya di sisi server [11]. *Backend service* merupakan sebuah *Application Programming Interface* (*API*) yang menghubungkan aplikasi satu dengan aplikasi yang lainnya untuk melakukan pertukaran data [12]. Dengan adanya *backend service* dapat menghubungkan aplikasi *client* seperti web *front-end*, *mobile application*, atau perangkat *Internet of Things* (*IoT*) [11]. *Backend service* berfungsi untuk memperoleh data, menyimpan data, mengambil data, memanipulasi data, mengirimkan data, atau mengembalikan data untuk menanggapi permintaan dari sisi *client*.

Backend service dapat berkomunikasi dan melakukan pertukaran data dengan *client* dengan menerapkan protokol komunikasi. Protokol komunikasi yang digunakan dapat bermacam-macam seperti *Hypertext Transfer Protocol* (*HTTP*), *WebSockets*, *Message Queueing Telemetry Transport* (*MQTT*), Advanced Message Queueing Protocol (*AMQP*), *GraphQL*, *Transmission Control Protocol/Internet Protocol* (*TCP/IP*), atau *User Datagram Protocol* (*UDP*) tergantung jenis kebutuhannya.

Pada penelitian ini, *backend service* berfungsi untuk memperoleh data dari perangkat *GPS Tracker*. Penerapan *backend service* menggunakan protokol komunikasi *TCP/IP* dikenal dengan teknik *TCP socket programming*. *TCP socket programming* merupakan jenis komunikasi yang berbasis *connection-oriented* [13], yaitu memastikan konektivitas yang sedang berlangsung.

2.5. Bahasa Pemrograman JavaScript dan NodeJS

JavaScript merupakan bahasa pemrograman yang dikenal dengan ECMAScript (ES) [14]. ECMAScript merupakan Bahasa pemrograman yang fleksibel. ECMAScript bisa menerapkan paradigma *Object-Oriented Programming* (*OOP*) dan paradigma *Functional Programming* (*FP*) [15], [16]. Sintaksis

JavaScript mengikuti spesifikasi dari ECMAScript. Spesifikasi ECMAScript ditentukan dari jenis *runtime* dan *engine* yang digunakan.

Runtime JavaScript merupakan lingkungan untuk menjalankan kode JavaScript. *Runtime* JavaScript menyediakan *API* sebagai antarmuka dengan sistem operasi [14]. Browser merupakan *runtime* JavaScript yang terdapat *Document Object Model* (*DOM*) untuk memanipulasi dan berinteraksi dengan tampilan yang ada di halaman web browser. Browser juga menyediakan *API* seperti *XMLHTTP* yang mengizinkan JavaScript melakukan permintaan *HTTP* ke server secara *asynchronous* [14].

NodeJS merupakan *runtime* JavaScript yang berjalan di atas sistem operasi secara langsung. NodeJS merupakan *open source* dan *cross-platform*, sehingga dapat berjalan di atas sistem operasi seperti Windows, Linux, Unix, dan MacOS [13]. NodeJS dapat menjalankan JavaScript dengan bantuan V8 *Engine* dari Google [14]. NodeJS menerapkan spesifikasi ECMAScript 2015 (ES6) dan seterusnya, sehingga selalu menerapkan dan mendukung versi terbaru [14].

NodeJS biasa dimanfaatkan untuk kebutuhan *backend service* atau menjalankan kode JavaScript di sisi server. Berikut kelebihan NodeJS sehingga digunakan sebagai *backend service* [17]:

1. *Fast Server Connections*

   NodeJS merupakan *runtime* yang mengeksekusi kode JavaScript hanya dengan memanfaatkan *single thread*, sehingga penggunaan memori relatif lebih kecil. NodeJS menerapkan konsep *non-blocking input output* dengan menggunakan *event-loop*, sehingga bisa menangani banyak permintaan secara sekaligus.

2. Satu Bahasa Pemrograman

   NodeJS mengatasi masalah *programmer* untuk menjadi *Full Stack Developer*. Melakukan pengkodean di JavaScript relatif lebih mudah untuk dipelajari, dikarenakan merupakan bahasa pemrograman yang *staticless typing*. Dengan JavaScript hanya dengan satu Bahasa, dapat dijalankan pada sisi *client-side* dan *server-side*.

3. Bahasa yang Banyak Digunakan

JavaScript banyak dipilih dan digunakan oleh banyak developer. Sejak JavaScript lebih disukai dan banyak direferensikan oleh developer dan belajar Bahasa JavaScript untuk *Front-end* dan *Back-end* lebih meringankan beban untuk banyak developer baru di NodeJS.

2.6. Basis Data MariaDB

Basis data terdapat 2 definisi [18], yaitu:

1. Basis Data

   Basis data merupakan data yang didesain untuk terintegrasi dan terorganisir untuk memenuhi kebutuhan pengguna atau organisasi.

2. Sistem Basis Data

   Sistem basis data merupakan sebuah sistem yang dirancang menerapkan metode yang tersedia dan didukung oleh komputer untuk menyimpan, Menyusun, atau mengelola *record-record* serta memelihara data operasional lengkap sebuah organisasi. Sehingga mampu menyediakan informasi yang optimal yang diperlukan pemakai dalam proses pengambilan keputusan.

Dengan adanya basis data dan sistem basis dapat memudahkan pemakai seperti yang dijelaskan pada bagian definisi. Namun Ketika mendesain basis data juga perlu diperhatikan sistem basis data jenis apa yang sesuai dengan kebutuhan. Berikut jenis basis data *SQL* dan *NoSQL* [19].

1. *SQL*

   SQL merupakan singkatan dari *Structured Query Language*. *SQL* dirancang untuk mendesain basis data yang terstruktur. SQL disebut juga sebagai *Relational Database Management System* (*RDBMS*). SQL sesuai untuk kebutuhan seperti manajemen, dan transaksi.

   Pada *SQL* terdapat prinsip *ACID* yaitu *atomicity, consistency, isolation*, *and durability*. *ACID* secara umum berguna untuk memastikan bahwa *query SQL* yang dieksekusi dapat diterima seutuhnya, tersimpan semuanya, dan eksekusi berjalan secara independen.

2. *NoSQL*

NoSQL merupakan basis data yang tidak terstruktur. *NoSQL* hadir karena adanya *big data*, yaitu untuk memproses banyak data dalam jumlah ukuran yang besar.

NoSQL dipilih karena kelebihannya untuk melakukan *skalabilitas*, ketersediaan data ketika ada permintaan, *fault tolerance* atau toleransi terhadap kesalahan atau jika terjadi gangguan pada salah satu komponen untuk terus melanjutkan pemrosesan, dan pemrosesan secara cepat pada data yang tidak terstruktur dalam jumlah yang besar.

NoSQL sendiri terdapat banyak variasi dengan memiliki kelebihan masing-masing. Untuk memilih variasi *NoSQL* tidak lagi relevan jika menggunakan *ACID*. Sehingga pada awal tahun 2000-an dikemukakan teori baru oleh Profesor Eric Bewer, yaitu prinsip *CAP* terdiri dari *consistency, availability, and partition tolerance*.

Pada penelitian ini digunakan sistem basis data MariaDB. MariaDB merupakan RDBMS, yang sesuai dengan kebutuhan penelitian untuk membangun aplikasi web di bagian *backend* sistem seperti melakukan manajemen, pencatatan lokasi dari perangkat *GPS Tracker*, dan pemrosesan data yang terstruktur.

2.7. Bahasa Pemrograman PHP

PHP singkatan dari *Hypertext Preprocessor*. PHP merupakan *scripting language* [20]. PHP merupakan Interpreter, sehingga tidak dapat berjalan di atas sistem operasi tanpa bantuan program lain. Tidak seperti Bahasa pemrograman lain seperti C dan Go yang memiliki *compiler* untuk melakukan kompilasi menjadi *executable file*. Dan seperti JavaScript yang memiliki mekanisme *Just-In-Time* (*JIT*) *compiler* untuk melakukan kompilasi pada saat *runtime* berlangsung [21].

PHP membutuhkan web server untuk menjalankan skrip PHP. PHP juga dikenal dengan pemrograman *server-side*, yaitu melakukan pemrograman di sisi server. PHP juga digunakan untuk pengembangan web dinamis, dikarenakan

kemampuan PHP untuk disisipkan kode *Hypertext Markup Language* (HTML), yang memungkinkan developer untuk membuat halaman web yang dinamis [20].

PHP pada awalnya dikembangkan oleh Rasmus Lerdorf pada tahun 1994 sebagai alat untuk melacak kunjungan ke halaman web. Saat ini PHP sudah digunakan untuk membuat halaman web yang dinamis, dan berganti nama dari PHP *Personal Home Page*, menjadi *Hypertext Preprocessor* [20].

2.8. Framework Laravel

Framework merupakan kerangka kerja perangkat lunak untuk mempermudah pengembangan aplikasi perangkat lunak. *Framework* memiliki aturan, konvensi, struktur, dan pola desain yang sudah teruji. Sehingga memungkinkan developer untuk membantu fokus pengembangan pada logika bisnis, dibandingkan harus memikirkan kembali infrastruktur dasar [22].

Framework memiliki arsitektur dan pola desain seperti *Model View Controller* (*MVC*), *Clean Architecture*, *Onion Architecture*, *Hexagonal Architecture*, atau *Model Driven Architecture* (*MDA*). Pertimbangan dalam memilih *framework* tidak hanya soal arsitektural, melainkan juga performa, fitur, utilitas, kemudahan, dan juga komunitas. Pada PHP terdapat variasi *framework* seperti Symfony, CodeIgniter, Phalcon, Yii, CakePHP, dan Laravel.

Pada penelitian ini penulis memilih *framework* Laravel dikarenakan sebagai berikut:

1. Perutean

   Salah satu keunggulan *framework Laravel* yaitu mengizinkan *developer* untuk melakukan kustomisasi perutean *HTTP URL*. Dengan adanya fitur ini, developer dapat dengan mudah memanajemen dan manipulasi *endpoint* sesuai yang dibutuhkan. Serta ke depannya Laravel mengharuskan *traffic* yang datang menggunakan *https* [23].

2. Performa

   Pada *framework* Laravel versi 4 dan 5, Laravel memiliki *response time* yang lebih kecil dibandingkan dengan Symfony 2.6. Perbandingannya Laravel dapat menampung jumlah koneksi sebanyak 90, namun memiliki

response time di antara 80 – 120 ms. Sedangkan pada Symfony memiliki *response time* yang jauh lebih tinggi di antara 100 – 200 ms [24].

3. Pengguna

   Dari sisi pengguna Laravel lebih banyak penggunanya dibandingkan *framework* lain. Sehingga memungkinkan untuk mempermudah proses *handover* ketika pergantian karyawan atau *programmer*.

4. Utilitas

   Dari sisi utilitas, *framework* Laravel memiliki banyak utilitas yang cukup membantu seperti melakukan *caching* terhadap *routing*, *caching views blade template*, utilitas *storage*, utilitas terkait *database* dan *migration*, utilitas terkait *queue* atau *scheduling* [22].

5. Manajemen dependensi

   Framework Laravel memiliki manajemen dependensi yang cukup baik. Laravel terintegrasi dengan alat Composer, sehingga memudahkan *developer* untuk melakukan instalasi, dan manajemen pustaka hanya dengan alat Composer tanpa perlu melakukannya secara manual.

6. Komunitas dan Keterbaruan

   Dari sisi komunitas *framework Laravel* yang paling progresif perkembangannya dibandingkan *framework* PHP lainnya. Sehingga *framework Laravel* akan terus dievaluasi dan diperbaiki, dan menerapkan teknologi terbaru serta keamanan terbaru.

2.9. *Maps API* OpenStreetMap dan LeafletJS

Dalam pengembangan Sistem Informasi Geografis (SIG), *Maps API* berperan penting dalam tahapan penyajian data dari sebuah sistem. Penyajian data pada dasarnya bisa dalam laporan, grafik, atau peta. Karena data geografis, maka dibutuhkan visualisasi data berupa peta. Peta dapat terbagi menjadi 2 [25], yaitu:

1. Peta dasar

   Peta dasar merupakan peta yang menunjukkan objek-objek di permukaan bumi pada posisi yang sebenarnya. Peta dasar mencakup pemetaan yang mengandung unsur *hypsografi*/relief (garis kontur, titik

tinggi, gunung, lembah, dan lainnya), unsur *hydrologi* (sungai, danau, laut), unsur vegetasi (hutan, belukar, kebun sawah), dan unsur buatan (jalan, pemukiman, pelabuhan).

2. Peta tematik

   Peta tematik merupakan peta yang menyajikan informasi berdasarkan tema yang ditentukan. Peta tematik sebenarnya merupakan peta dasar, hanya saja unsur topografi yang ditampilkan menyesuaikan dengan tema yang diambil. Peta tematik memperlihatkan data secara kualitatif atau kuantitatif, sesuai dengan tema yang diambil. Pada peta tematik tetap memerlukan detail topografi seperti batas administrasi, jalan, sungai, dan informasi penting lainnya yang dibutuhkan oleh tema tersebut.

Model data yang digunakan pada peta adalah model data vektor [4]. Model data vektor yang digunakan berupa titik, guna menampilkan lokasi yang tercatat pada sistem, dan model data vektor berupa garis, guna menampilkan perjalanan yang dilakukan oleh kendaraan. Lokasi yang ditampilkan menggunakan sistem koordinat geografis. Dengan memanfaatkan garis bujur dan garis lintang sebagai koordinat. Garis bujur merupakan *longitude*, merupakan garis yang membentang dari utara sampai selatan, dan mengukur dari barat ke timur [26]. Sedangkan garis lintang merupakan *latitude*, merupakan garis yang membentang dari barat hingga timur, dan mengukur dari utara ke selatan [26]. Pada penelitian ini sistem koordinat yang digunakan adalah WGS84, yang menjadi acuan di seluruh dunia.

Peta, data vektor, dan data koordinat akan ditampilkan dengan menggunakan *Maps* API OpenStreetMap, Mapbox, dan LeafjetJS. OpenStreetMap merupakan penyedia peta dan data peta yang dibangun oleh kontribusi komunitas dan untuk siapa saja. LeafletJS merupakan *open-source* JavaScript *library* untuk berinteraksi dengan komponen peta dan dengan penyedia data peta. Mapbox juga merupakan penyedia peta dan data peta, hanya saja pada penelitian ini terbatas digunakan untuk antarmuka peta

2.10. Formula *Haversine*

Pada penelitian ini, penulis menggunakan formula *haversine* secara praktis. Formula *haversine* untuk menghitung jarak terpendek antara 2 titik di permukaan bumi yang digambarkan dalam sebuah garis tanpa mengabaikan kelengkungan bumi [27]. Hal ini dibutuhkan untuk menghitung jarak tempuh sebuah kendaraan yang tercatat sebagai banyak titik koordinat, hingga membentuk sebuah garis. Berikut bentuk formula *haversine* (2.1).

$$d = 2r \arcsin\left(\sqrt{sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2) sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (2.1)$$

Dengan d = jarak, r = radius bumi sebesar 6371,1 Km, φ_1 dan φ_2 merupakan bujur (*latitude*) dalam radian, λ_1 dan λ_2 merupakan lintang (*longitude*) dalam radian, $\Delta\varphi$ adalah besaran perubahan bujur (*latitude*), dan $\Delta\lambda$ adalah besaran perubahan lintang (*longitude*) [27]. Secara sederhana untuk menghitung jarak antar 2 titik koordinat (φ_1, φ_2 dan λ_1, λ_2), maka masing-masing titik koordinat haruslah diubah dari bentuk *decimal degree* menjadi bentuk radian. Kemudian menghitung besaran perubahan pada bujur dan lintang (*latitude and longitude*) dengan melakukan selisih pada masing-masing bujur dan lintang. Kemudian menghitung sudut pusat dalam bentuk radian dengan memasukkan bujur dan lintang ke dalam fungsi trigonometri. Terakhir mengalikan sudut pusat dengan radius bumi, sehingga menghasilkan jarak dalam kilometer.

2.11. Metode Pengembangan *Agile*

Software Development Life Cycle (*SDLC*) merupakan tahapan pengembangan dalam proses pembuatan atau pengembangan perangkat lunak. Dalam pengembangan perangkat lunak, terdapat istilah f*ramework activity*, yaitu merupakan tahapan atau aktivitas pengembangan secara umum yang dapat dibagi menjadi tahapan [28]:

1. *Communication*

   Communication merupakan tahapan ketika proyek dimulai, dan melakukan identifikasi kebutuhan. Tahapan di mana tim *development* berkomunikasi dengan pelanggan (*customer*) atau pengguna (*user*) mengenai kebutuhan yang dihadapi, dan kebutuhan dari sisi bisnis.

2. *Planning*

   Planning merupakan tahapan ketika selesai mengidentifikasi kebutuhan, dan menerjemahkannya ke dalam kebutuhan teknis, risiko yang mungkin terjadi, fasilitas yang mungkin dibutuhkan, dan *timeline*.

3. *Modelling*

   Modelling merupakan tahapan ketika mendesain sistem yang memenuhi sesuai kebutuhan. Pada tahapan ini, dapat dikatakan perancangan aplikasi, atau *system design*. Dapat memanfaatkan *UML*, untuk proses pemodelannya.

4. *Construction*

   Construction merupakan tahapan implementasi atau *pengkodean*, dan juga pengujian perangkat lunak. Pada tahapan ini jika menerapkan prinsip *agile*, maka konstruksinya adalah *sprint*. *Sprint* adalah implementasi yang dilakukan dalam durasi waktu yang singkat, namun ruang lingkup yang stabil.

5. *Deployment*

   Deployment merupakan tahapan ketika perangkat lunak secara utuh selesai, atau hanya sebagian disampaikan ke pelanggan (*customer*) atau pengguna (*user*). Guna mengevaluasi dan mendapatkan *feedback*.

Tahapan di atas merupakan *framewok activity*. Pada penerapannya, tidak harus berurutan dari tahap pertama sampai terakhir (*linear process flow*). Namun bisa juga dilakukan secara iteratif (*iterative process flow*), atau bahkan secara paralel (*parallel process flow*). Semua itu tergantung dengan metode pengembangan yang dipilih [28].

Agile merupakan salah satu metode pengembangan *SDLC*. *Agile* biasa digunakan untuk siklus pengembangan yang durasinya singkat dan berfokus pada peningkatan berkelanjutan, atau dengan kata lain menitikberatkan pada kecepatan *delivery* dan memungkinkan perubahan setiap saat [29], [30].

Pengembangan *Agile* dimulai pada Februari 2001 di Utah, USA untuk mendiskusikan metodologi baru dalam pengembangan perangkat lunak [30]. *Agile* berisi konsep dasar untuk serangkaian berbagai macam metode pengembangan perangkat lunak. Implementasi dari *agile* terdapat 7 macam model di antaranya: *agile modelling*, *crystal*, *dynamic system development methodology*, *adaptive software development*, *feature driven development*, *extreme programming*, dan *scrum*. Di antara model yang telah disebutkan, *scrum* merupakan model yang paling efektif dikarenakan mengedepankan proses kerja yang cepat dalam pengembangannya [30].

Asal mula kata "*scrum*" berasal dari sebuah aktivitas yang terjadi dalam permainan *rugby* [28]. Dalam model *scrum*, terdapat istilah *iteration* atau pengulangan yang dibagi menjadi bagian-bagian kecil yang dinamakan sprint. Sehingga sprint yang dilakukan secara iteratif atau berulang, dinamakan *scrumban* [30]. Dalam *scrum*, terdapat pola proses yang mendefinisikan sekumpulan tindakan dalam pengembangan seperti [28]:

1. *Backlog*

   Backlog merupakan daftar yang diprioritaskan dari kebutuhan proyek atau fitur yang mempunyai nilai bisnis untuk pelanggan. Item yang ditambahkan ke *backlog*, dapat dilakukan kapan pun. Sehingga *backlog* perlu di prioritaskan ulang, dan hal ini biasa dilakukan oleh manajer produk.

2. *Sprints*

   Sprints terdiri dari sekumpulan unit kerja yang dibutuhkan untuk mengimplementasikan kebutuhan yang ada di *backlog* dalam durasi waktu tertentu. Perubahan yang terjadi pada *backlog*, tidak mempengaruhi sprint. Oleh karena itu sprint dilakukan dalam durasi waktu yang singkat, namun ruang lingkup yang stabil.

3. *Scrum meetings*

   Dalam *scrum* terdapat pertemuan yang biasa diadakan harian, biasanya 15 menit. Pertemuan di pimpin oleh *scrum master*. Kegunaan dari pertemuan, adalah untuk membantu tim menemukan potensi masalah sedini mungkin.

4. Demo

   Demo merupakan tahapan untuk menunjukkan ke pelanggan bahwa *software* dapat difungsikan, didemonstrasikan, dan dievaluasi oleh pelanggan. Dalam tahap demo, tidak selalu terdapat semua fungsionalitas yang telah diimplementasikan. Melainkan hanya fungsionalitas yang dapat diberikan ke pelanggan dalam kurun waktu yang telah ditetapkan.

2.12. *Unified Modelling Language* (*UML*)

Perancangan perangkat lunak sangat berperan penting dalam pembuatan atau pengembangan perangkat lunak. *Unified Modelling Language* (*UML*) merupakan pemodelan yang paling bermanfaat untuk memvisualisasikan sistem yang kompleks. Dengan adanya *UML* dapat memberikan dampak mengenai bagaimana perangkat lunak dibuat dan di dokumentasikan, seperti pada kasus *Vehicle Routing Problem* (*VRP*) [31].

UML dapat membantu *developer* mengimplementasikan perangkat lunak yang sesuai dengan kebutuhan. Dengan adanya *UML* juga dapat memudahkan proses komunikasi antara *developer* dengan *non-developer*. *UML* terdapat jenis pemodelan, yaitu [28], [32]:

1. *Context Model*

   Context model digunakan untuk mengidentifikasi dan menjelaskan batasan dari sebuah sistem (*system boundaries*), yaitu apa yang menjadi dan bukan bagian dari sistem, yang akan, atau sedang dikembangkan. Untuk mendefinisikan *system boundaries* dapat menggunakan model arsitektur, supaya dapat terlihat hubungan antara sistem dengan sistem yang lainnya. Untuk menggambarkan *context model* dari sisi aktivitas, dapat menggunakan *activity diagram*.

2. *Interaction Model*

   Interaction model digunakan untuk merancang interaksi dari beberapa jenis entitas dari semua sistem yang ada. Untuk menampilkan interaksi dari sisi penggunaan secara visual, dapat menggunakan *use case diagram*. Dan untuk menampilkan interaksi sistem secara lebih mendetail secara visual dapat menggunakan *sequence diagram*.

3. *Structural Modelling*

   Structural modelling digunakan untuk menampilkan komponen yang membentuk suatu sistem dan hubungannya dengan komponen lain. Untuk merancang arsitektur suatu sistem dapat memanfaatkan *class diagram*.

Pada penelitian ini ketiga jenis pemodelan *UML* yang telah disebutkan akan digunakan untuk perancangan dan memvisualisasikannya ke dalam bentuk diagram. Dengan adanya *UML* dapat membantu mempermudah pemahaman sistem yang akan dibangun secara keseluruhan.

2.13. Metode Pengujian *Black Box*

Dalam pengembangan perangkat lunak, untuk memastikan perangkat lunak layak digunakan dan sesuai dengan hasil yang diharapkan maka dibutuhkan tahapan pengujian perangkat lunak.

Pada pengembangan perangkat lunak, terdapat beragam metode pengujian seperti: *Black Box Testing*, *White Box Testing*, *User Acceptance Testing* (*UAT*), *Performance Testing*, atau *Security Testing*. Semua jenis metode memiliki pengukuran dan hasil akhir yang beragam. Pada implementasinya tidak semua jenis pengujian dilakukan, disesuaikan dengan produk yang ada, dan pertimbangan-pertimbangan lainnya yang tidak bisa penulis tuliskan di sini.

Pada penelitian ini, peneliti akan menggunakan metode pengujian *Black Box*. Metode pengujian *Black Box* merupakan pengujian yang menargetkan fungsional aplikasi. Black Box testing merupakan pelengkap dari hal-hal yang tidak tercakup pada pengujian *White Box* [33].

Manfaat metode pengujian jenis ini adalah hanya dengan membuat skenario pengujian atau *test case*, semua komponen sistem dapat teruji tanpa harus melakukannya secara detail dan spesifik pada komponen tertentu. Sehingga lebih efektif dan efisien, serta menggunakan energi dan waktu lebih sedikit, namun mendapatkan hasil pengujian yang cukup signifikan.

2.14. Penelitian Terkait

Tabel 2.1 merupakan penelitian-penelitian terkait mengenai topik "pelacakan kendaraan" ataupun judul penelitian "Rancang Bangun Aplikasi Web Untuk Pelacakan Kendaraan Menggunakan NodeJS dan *Framework* Laravel".

Tabel 2. 1 Penelitian Terkait

No	Nama dan Tahun	Judul	Topik	Subjek	Hasil
1	Muksan Junaidi (2020)	Sistem Keamanan Pelacakan Kendaraan Bermotor Menggunakan *Raspberry Pi 3* Dengan *Module Gps* Secara *Real-Time* Berbasis *Web*	Pelacakan Kendaraan	Motor Pribadi	Perangkat *IoT* dengan *module GPS* dibuat menggunakan *Raspberry Pi 3*, *fullstack web application* menggunakan python dengan flask ditambah socket.io untuk fitur *real-time*, dan *database* SQLite sebagai tempat pengolahan datanya
2	Kevin Winata Tanjung, Yulius Hari, Yoga Alif Kurnia Utama (2021)	Sistem Pelacak Sepeda Motor Berbasis Web Menggunakan Esp32	Pelacakan Kendaraan	Motor Pribadi	*Mikrokontroller* ESP32 dibuat dengan arduino sebagai *GPS*, *fullstack web application* menggunakan Laravel sebagai pemantauan

3	Farhan Syaibir (2023)	Sistem Pelacakan Dan Monitoring Perjalanan Bus Menggunakan Gps Berbasis Web	Pelacakan Kendaraan	Bus	Pemanfaatan *GPS* yang sudah ada serta menghubungkan nya ke Firebase, dan *front-end web application* memanfaatkan *Maps API* dari Google Maps API + LeafletJS dan bootstrap
4	Dian Citra Kesuma (2022)	Rancang Bangun Aplikasi Pelacakan Pengiriman Mobil Pada Pt.Sultan Perintis Perkasa Berbasis Web	Pelacakan Kendaraan	Mobil dan PT. Sultan Perintis Perkasa	*Fullstack web application* menggunakan PHP, MySQL, dan AppServ sebagai lingkungan untuk mempermudah proses pembuatan aplikasi
5	Muskan M. Pathan, Ujwala A. Bongale (2022)	*Design And Implementation Of Real-Time Web-Based Vehicle Tracking System Using Sim*	Pelacakan Kendaraan	Kendaraan secara umum	*Fullstack and real-time web application* menggunakan MariaDB, Express.js, Angular, dan Node.js serta pelacakan kendaraan menggunakan *Subscriber Identity Module* (*SIM*)
6	Idriss Moumen, Najat Rafalia, Jaafar Abouchabaka, Marouane Aoufi (2023)	*Real-Time GPS Tracking System For Iot-Enabled Connected Vehicles*	Pelacakan Kendaraan	Kendaraan secara umum	Pembuatan *IoT* dengan Ardunio Uno R3, SIM800L, NEO6M GPS, dan *fullstack real-time web application* dengan Next.js, socket.io, tailwind CSS dan Firebase

					sebagai *database*.
7	Rafif Mulia Reswara (2024)	Rancang Bangun Aplikasi Web Untuk Pelacakan Kendaraan Menggunakan Nodejs Dan *Framework* Laravel	Pelacakan Kendaraan	Kendaraan secara umum dan PT. Swa Nusa Multimedia	*Fullstack web application* menggunakan *framework* Laravel, *backend service* menggunakan NodeJS, dan pelacakan kendaraan memanfaatkan perangkat *GPS Tracker*

2.15. Posisi Penelitian

Tabel 2.2 menegaskan posisi penelitian yang diteliti penulis, di antara penelitian-penelitian yang memiliki keterkaitan dengan topik ataupun judul penelitian penulis. Sekaligus menjadi pembanding, pembeda, dan keunggulan dari masing-masing penelitian yang ada.

Tabel 2. 2 Posisi Penelitian

Nama dan Tahun	**Judul**	**Alat Pelacakan**	***Backend Service***	**Teknologi Web**	**Basis Data**	***Maps API***
Muksan Junaidi (2020)	Sistem Keamanan Pelacakan Kendaraan Bermotor Menggunakan *Raspberry Pi 3* Dengan *Module Gps* Secara *Real-Time* Berbasis *Web*	Raspberry Pi 3 dan Modul *GPS*	Flask	*Framework* flask dan Socket.io	SQLite	Tidak tertulis pada penelitian

Kevin Winata Tanjung, Yulius Hari, Yoga Alif Kurnia Utama (2021)	Sistem Pelacak Sepeda Motor Berbasis Web Menggunakan Esp32	*Mikrokontroller* ESP32	Laravel	*Framework* Laravel	Tidak tertulis pada penelitian	Tidak tertulis pada penelitian
Farhan Syaibir (2023)	Sistem Pelacakan Dan Monitoring Perjalanan Bus Menggunakan Gps Berbasis Web	Perangkat GPS dengan modul Neo-M8M dan L76X GPS HAT serta sudah jadi	*Firebase Realtime Database*	Tidak tertulis pada penelitian	*Firebase Realtime Database*	*Google Maps API* dan LeafletJS
Dian Citra Kesuma (2022)	Rancang Bangun Aplikasi Pelacakan Pengiriman Mobil Pada Pt.Sultan Perintis Perkasa Berbasis Web	Tidak tertulis pada penelitian	Tidak ada	AppServ	MySQL	Tidak tertulis pada penelitian
Muskan M. Pathan, Ujwala A. Bongale (2022)	*Design And Implementation Of Real-Time Web-Based Vehicle Tracking System Using Sim*	*Subscriber Identity Module* (*SIM*)	NodeJS dengan Express.js	AngularJS	MariaDB	*Google Maps API*
Idriss Moumen, Najat Rafalia, Jaafar Abouchabaka, Marouane Aoufi (2023)	*Real-Time GPS Tracking System For Iot-Enabled Connected Vehicles*	SIM800L dan NEO6M GPS	NodeJS dengan Express.js	*Framework* Next.js dengan Tailwind CSS dan Socket.io	*Firebase Realtime Database*	OpenStreet Map dan LeafletJS

Rafif Mulia Reswara (2024)	Rancang Bangun Aplikasi Web Untuk Pelacakan Kendaraan Menggunakan Nodejs Dan *Framework* Laravel	Perangkat GPS Tracker Model GT06 dan sudah jadi	NodeJS dengan modul net	*Framework* Laravel	MariaDB	OpenStreet Map, MapBox, dan LeafletJS

Tabel 2.2 terkait posisi penelitian, membandingkan dengan tolak ukur teknologi yang digunakan adalah bervariasi. Setiap penelitian memiliki kesamaan topik, yaitu "pelacakan kendaraan". Berikut perbedaan pada masing-masing penelitian yang akan dijelaskan secara deskriptif, adalah sebagai berikut.

1. Muksan Junaidi (2020)

   Pada penelitian yang dilakukan oleh Muksan Junaidi pada tahun 2020, adalah menjawab permasalahan terkait pencurian kendaraan bermotor. Studi kasus dan subjek penelitian adalah kendaraan bermotor pribadi. Perangkat *GPS* yang digunakan, dirakit dari awal. Fitur pada aplikasi web yang paling mencolok adalah *real-time tracking* yang disajikan dalam bentuk vektor titik, *log tracking*, dan *log actvitiy* yang hanya disajikan dalam bentuk daftar tabel.

2. Kevin Winata Tanjung, Yulius Hari, Yoga Alif Kurnia Utama (2021)

   Pada penelitian yang dilakukan oleh Kevin Winata Tanjung, Yulius Hari, dan Yoga Alif Kurnia Utama pada tahun 2020, adalah menjawab permasalahan terkait kehilangan kendaraan bermotor. Studi kasus dan subjek penelitian adalah kendaraan bermotor pribadi. Perangkat *GPS* yang digunakan, *mikrokontroller* ESP32. Fitur pada aplikasi web paling mencolok terdapat pelacakan kendaraan. Dan administrasi mengenai kendaraan, dan perangkat GPS *mikrokontroller* ESP32, menjadi satu pada bagian kendaraan.

3. Farhan Syaibir (2023)

   Pada penelitian yang dilakukan oleh Farhan Syaibir pada tahun 2023, adalah menjawab permasalahan terkait menemukan posisi bus

untuk dinaiki dan keterlambatan tiba di tempat tujuan. Studi kasus dan subjek penelitian adalah bus secara umum. Perangkat *GPS* yang digunakan, memanfaatkan yang sudah jadi. Fitur pada aplikasi web yang paling mencolok adalah *live tracking*, yang merupakan *latitude* dan *longitude* hasil metode *haversine formula*. Terdapat program rute bus dan diterapkan *inisialisasi routing machine* agar *maps* dapat memberikan rute tercepat dari satu titik ke titik lainnya, dan dapat menampilkan total estimasi waktu dari titik pemberhentian pertama hingga titik pemberhentian terakhir.

4. Dian Citra Kesuma (2022)

   Pada penelitian yang dilakukan oleh Dian Citra Kesuma pada tahun 2022, adalah menjawab permasalahan terkait pelacakan pengiriman mobil dan sistem informasi terkait pengiriman mobil. Studi kasus dan subjek penelitian adalah mobil pada PT. Sultan Perintis Perkasa. Perangkat *GPS* yang digunakan, tidak tertulis pada penelitian. Fitur pada aplikasi web yang paling mencolok adalah pendataan informasi pengiriman mobil, manajemen sopir, manajemen kota, manajemen data pemakai, *multi-role user* yaitu admin dan sopir, dan halaman pengecekan resi.

5. Muskan M. Pathan, Ujwala A. Bongale (2022)

   Pada penelitian yang dilakukan oleh Muskan M. Pathan dan Ujwala A. Bongale pada tahun 2022, adalah menjawab permasalahan terkait efisiensi pelacakan yang lebih murah dan nyaman hanya dengan memanfaatkan *SIM*, dan tanpa memerlukan koneksi internet dan konsumsi daya baterai yang besar. Studi kasus dan subjek penelitian adalah kendaraan secara umum yang dapat diadopsi oleh banyak perusahaan. Tidak menggunakan perangkat *GPS*, hanya memanfaatkan *SIM*. Fitur secara garis besar yang paling mencolok adalah manajemen rute, manajemen jadwal *transport*, *real-time visibility*, optimalisasi rute, pengiriman *SMS* dan email, analisa panggilan dan *text*.

6. Idriss Moumen, Najat Rafalia, Jaafar Abouchabaka, Marouane Aoufi (2023)

   Pada penelitian yang dilakukan oleh Idriss Moumen, Najat Rafalia, Jaafar Abouchabaka, Marouane Aoufi pada tahun 2023, adalah menjawab permasalahan terkait pemanfaatan dan integrasi perangkat *IoT* dan *GPS* untuk berbagai sektor industri. Studi kasus dan subjek penelitian adalah kendaraan secara umum untuk keperluan berbagai sektor industri. Perangkat *GPS* dirakit dari awal menggunakan S IM800L GSM modul dan Neo6M GPS modul. Fitur secara garis besar yang paling mencolok adalah pelacakan secara *real-time* dan akurat, menyediakan pemantauan yang komprehensif seperti *diagnostic menis*, konsumsi bahan bakar, dan kecepatan.

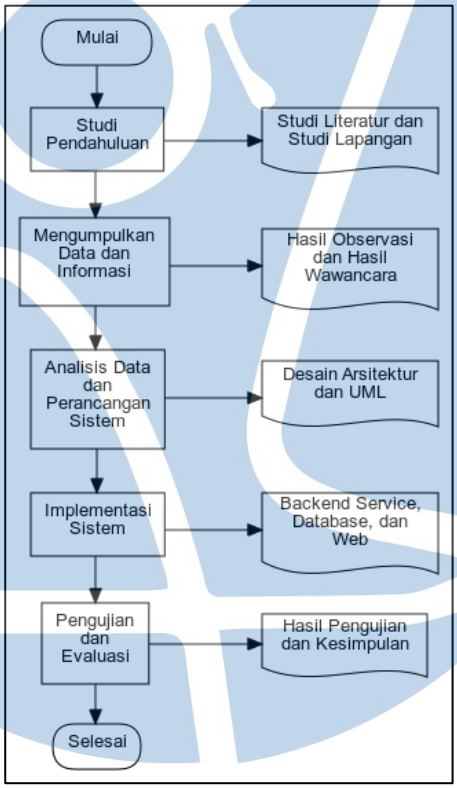
7. Rafif Mulia Reswara (2024)

   Sedangkan pada penelitian yang dilakukan Rafif Mulia Reswara pada tahun 2024, adalah menjawab permasalahan terkait pelacakan kendaraan yang akan dimanfaatkan untuk kebutuhan pengiriman logistik. Studi kasus pada PT. Swa Nusa Multimedia, dan kendaraan secara umum. Perangkat *GPS* menggunakan yang sudah tersedia dengan model GT06. Fitur pada aplikasi web tidak hanya pelacakan secara *real-time* dengan menerapkan *client polling*. Fitur pemantauan disajikan menggunakan *Maps API*, dalam bentuk vektor garis, atau dengan kata lain riwayat perjalanan bisa langsung dilihat dalam bentuk peta (interaktif). Dan terdapat fitur manajemen sopir, manajemen kendaraan, dan manajemen perangkat *GPS Tracker* yang masing-masing adalah komponen terpisah yang saling berhubungan. Dan terdapat perhitungan jarak tempuh dengan menggunakan metode *haversine formula* yang merupakan bagian dari fitur pemantauan kendaraan dan manajemen kendaraan. Serta fitur *multi-role user* untuk membagi mengotorisasi pengguna.

BAB III
METODOLOGI PENELITIAN

3.1. Tahapan Penelitian

Dalam melakukan penelitian ini, peneliti dalam pelaksanaannya membagi menjadi tahapan-tahapan sebagai berikut.



Gambar 3. 1 Tahapan Penelitian

Gambar 3.1 merupakan tahapan penelitian yang penulis lakukan dengan rincian sebagai berikut:

1. Studi Pendahuluan

   Studi pendahuluan dilakukan supaya peneliti mendapatkan gambaran besar terkait hal yang ingin diteliti. Peneliti melakukan studi literatur untuk mendapatkan pengetahuan dari penelitian-penelitian sebelumnya, dan referensi seperti jurnal, dan buku. Peneliti juga melakukan

studi lapangan dengan melakukan observasi ke tempat yang dijadikan penelitian.

2. Mengumpulkan Data dan Informasi

   Tahap selanjutnya adalah melakukan pengumpulan data, yang dilakukan menggunakan metode observasi dan wawancara secara langsung.

3. Analisis Data dan Perancangan Sistem

   Tahapan analisis data dan perancangan dilakukan secara bersamaan. Dikarenakan penelitian yang dilakukan adalah pengembangan, dan lingkup objek penelitiannya adalah studi kasus, maka jenis datanya kualitatif. Sehingga data yang didapat, akan diterjemahkan menjadi sebuah model atau perancangan yang akan diimplementasi.

4. Implementasi Sistem

   Tahapan selanjutnya adalah melakukan implementasi terhadap perancangan dan pemodelan yang sudah dibuat. Metode yang digunakan dengan menerapkan *scrum* [28].

5. Pengujian dan Evaluasi

   Tahapan terakhir adalah melakukan pengujian terhadap sistem yang telah dibuat. Hal ini dilakukan untuk menguji apakah sistem dapat digunakan sebagaimana mestinya. Dan melakukan evaluasi terhadap kesimpulan yang didapat.

3.2. Rancangan Penelitian

Pada bagian ini, penulis akan menjelaskan terkait jenis penelitian yang dilakukan, metode analisis data untuk menganalisis data pengujian, metode pengumpulan data, metode pengujian, metode implementasi dan evaluasi, dan lingkungan pengembangan.

3.2.1. Jenis Penelitian

Menurut [34], penelitian pengembangan dikenal dengan metode penelitian *Research and Development* (*RnD*). Metode *RnD* merupakan

penelitian untuk menemukan atau mengembangkan solusi dalam menyelesaikan atau meningkatkan suatu hal dari suatu kondisi atau permasalahan dengan suatu cara tertentu, hingga menghasilkan suatu layanan atau produk yang teruji [34], [35].

Jenis penelitian merupakan variasi dari penelitian yang dilakukan. Jenis penelitian dapat ditentukan berdasarkan masalah yang diangkat dan tujuan penelitian. Pada penelitian ini objektifnya adalah membuat aplikasi web dengan merancangnya, hingga hasilnya sesuai dengan kondisi, permasalahan, dan kebutuhan dari PT. Swa Nusa Multimedia. Sehingga berdasarkan premis di atas, jenis penelitian yang dilakukan adalah penelitian pengembangan [36].

Luaran dari jenis penelitian pengembangan dapat berupa perancangan atau pemodelan dan sebuah layanan atau produk. Pada penelitian ini perancangan atau pemodelannya berupa *activity diagram*, *sequence diagram*, dan *class diagram* yang merupakan bagian dari *Unified Modelling Language* [28], [32]. Kemudian mengenai hasilnya berupa layanan *backend service* untuk perolehan data *GPS* secara *online* dan aplikasi web untuk manajemen dan memantau kendaraan secara *real-time*.

3.2.2. Metode Analisis Data

Metode analisis data merupakan analisis data terkait hasil pengujian. Metode pengujian *black box* merupakan pengujian yang menganalisis berdasarkan antarmuka perangkat lunak [28]. Pengujian *black box* dilakukan dengan membuat skenario pengujian pada antarmuka perangkat lunak yang menguji aplikasi secara fungsional untuk menemukan *error*. Sehingga metode analisis data pada pengujian ini adalah kuantitatif, karena hasil dari pengujian merupakan angka yang berupa jumlah skenario pengujian yang berhasil dan yang gagal. Berikut adalah rumus untuk menganalisis pengujian *black box* (3.1).

$$Tingkat\ Keberhasilan\ Pengujian = \frac{Jumlah\ Pengujian\ Berhasil}{Jumlah\ Seluruh\ Pengujian} \times 100\% \quad (3.\ 1)$$

Pada rumus 3.1 berguna untuk mengetahui tingkat keberhasilan pengujian dengan mendapatkan presentasi keberhasilan pengujian dari jumlah pengujian yang berhasil.

3.2.3. Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan pada penelitian ini, berupa:

1. Observasi

   Observasi dilakukan secara langsung ke perusahaan PT. Swa Nusa Multimedia. Sehingga peneliti dapat terlibat secara aktif dengan karyawan yang ada dan pihak yang terlibat di dalamnya. Dan lebih memahami kebutuhan yang ada dengan mengalaminya secara langsung.

2. Wawancara

   Peneliti juga melakukan metode pengumpulan data dengan melakukan wawancara kepada narasumber yang berhubungan terkait judul penelitian penulis. Dengan melakukan wawancara, data yang didapatkan menjadi lebih objektif dibandingkan dengan observasi yang bisa menimbulkan asumsi pribadi penulis.

3. Eksperimen

   Peneliti juga melakukan jenis pengumpulan data dengan melakukan eksperimen terhadap perangkat *GPS Tracker* untuk perolehan datanya ke *backend service*, dan melakukan simulasi atau uji coba dengan kendaraan untuk mengintegrasikan data yang didapat dari *backend service* dan tampil secara *real-time* di aplikasi web.

3.2.4. Metode Pengujian

Pada penelitian ini terdapat 2 sistem yang diuji, yaitu sistem terkait aplikasi web untuk melakukan hal administratif dan pemantauan kendaraan,

dan sistem *backend service* terkait perolehan data pelacakan kendaraan dari *GPS Tracker*.

Dengan kondisi yang ada, penulis memilih metode pengujian *black box*. Dikarenakan memudahkan menemukan masalah dari 2 sistem yang terintegrasi. Sehingga pengujian menjadi lebih efektif dan efisien, dibandingkan jika harus menguji per-sistem.

Pada pengujian ini akan dibagi menjadi 2 fase, yaitu fase administratif terkait sopir, kendaraan, dan perangkat *GPS Tracker*. Dan fase pemantauan serta riwayat perjalanan, dengan melakukan perjalanan menggunakan kendaraan. Dan dibagi lagi menjadi 2 *role*, yaitu *role* sebagai administrator yang dapat melakukan hal administratif, dan pemantauan serta riwayat perjalanan. Dan *role special tracking*, yang hanya dapat melihat pelacakan dan memantau beberapa kendaraan saja.

Pada metode pengujian *black box*, yang menjadi penguji adalah penulis. Tujuan dari pada pengujian *black box* adalah menguji fungsionalitas aplikasi pada bagian terluar aplikasi. Format pengujian dapat terlihat pada tabel-tabel berikut, sesuai dengan yang telah dijelaskan pada paragraf sebelumnya.

Tabel 3. 1 Format Pengujian *Black Box*

SRS Code	***Test Code***	***Test Case***	***Expectation***	***Result***
R.01	T.01	-	-	-

Penjelasan dari pada tabel 3.1 format pengujian *black box* adalah sebagai berikut:

1. *SRS Code*

   System requirement specification code menandakan pada baris tersebut, menguji berdasarkan spesifikasi kebutuhan yang telah dibuat. *SRS Code* didapatkan dari hasil analisis metode pengumpulan data.

2. *Test Code*

Test code merupakan kode skenario pengujian yang akan dilakukan. Hal ini diterapkan supaya pengujian yang akan dilakukan mudah untuk dikenali.

3. *Test Case*

   Test case merupakan skenario pengujian yang akan dilakukan, dan dijelaskan secara deskriptif pada fungsi yang mau diuji. Hal ini diterapkan supaya pelaksanaan dan hasil pengujian lebih spesifik.

4. *Expectation*

   Expectation merupakan ekspektasi dari skenario pengujian yang dilakukan. Hal ini menjadi standar untuk memastikan bahwa fungsi tersebut berfungsi semestinya. *Expectation* dijelaskan secara deskriptif.

5. *Result*

   Result merupakan hasil pengujian yang didapat. Hasil pengujian dituliskan dalam bentuk 2 nilai, yaitu berhasil atau gagal.

Ketika pengujian dilaksanakan, peneliti mengurutkannya sesuai dengan alur yang akan dilakukan ketika penggunaan aplikasinya:

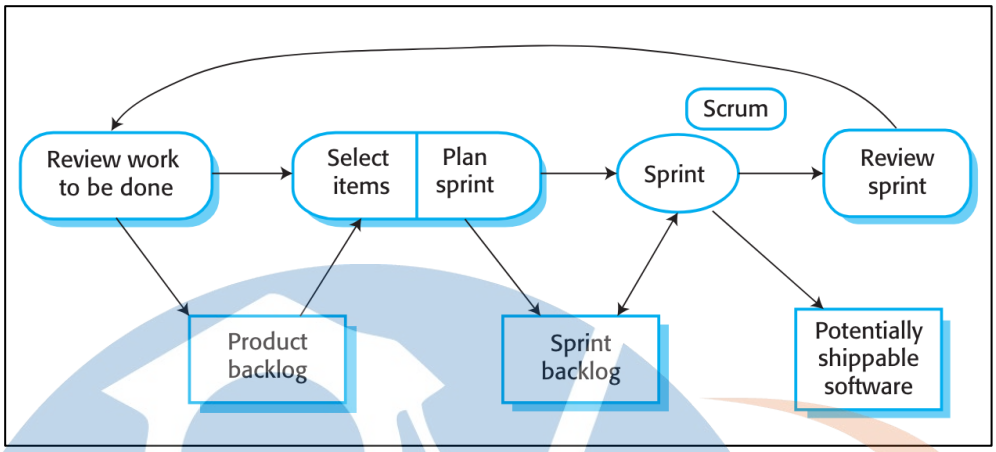
1. Penulis mengakses halaman web dengan *role* sebagai administrator. Kemudian melakukan hal administratif seperti manajemen sopir, manajemen kendaraan, dan manajemen perangkat *GPS Tracker*.
2. Penulis masih sebagai *role* administrator melakukan konfigurasi perangkat *GPS Tracker*, dan memastikan perangkat *GPS Tracker* dapat digunakan dan mengirimkan datanya ke *backend service*.
3. Penulis masih sebagai *role* administrator kemudian membawa perangkat *GPS Tracker* bersamaan dengan kendaraan. Tujuan dari pada fase ini adalah mendapatkan data pelacakan, dan riwayat perjalanan.
   a. Kendaraan yang akan digunakan adalah kendaraan bermotor roda dua.
   b. Kendaraan akan dibawa melewati gang perumahan dengan kecepatan kurang lebih 10 – 20 Km/h.

c. Kendaraan akan dibawa melewati jalan raya besar yang wajib menaati peraturan lalu lintas dengan kecepatan kurang lebih 40 – 80 Km/h.
d. Halaman web bagian pelacakan kendaraan akan dimonitor untuk memastikan proses pelacakan berjalan secara *real-time*.

4. Penulis masih sebagai *role* administrator melakukan pengujian pada halaman web bagian pelacakan, untuk melihat riwayat perjalanan secara interaktif.
5. Penulis masih sebagai *role* administrator melakukan manajemen *user* dengan menambahkan *user* baru dengan *role* sebagai *special tracking*.
6. Penulis berganti *role* sebagai *special tracking*, pada fase ini memastikan menu yang diakses hanya menu yang berfungsi sebagai pelacakan dan pemantauan kendaraan. Dan menguji apakah proses pelacakan tetap dapat dilakukan secara *real-time* dan melihat riwayat perjalanan secara interaktif.

3.2.5. Metode Implementasi dan Evaluasi

Metode implementasi dilakukan dengan menerapkan prinsip *agile*, dengan penerapannya yaitu *scrum*. Pada tahapannya akan dilakukan pengaturan skala prioritas yang didefinisikan menjadi *backlog*, dan durasi *sprint* yang akan dilakukan. Kemudian dilanjutkan dengan pemodelan dan implementasi pemodelan. Secara keseluruhan metode implementasi dengan *scrum* dapat mengacu pada gambar 3.2.



Gambar 3. 2 Metode Implementasi *Scrum*

Dari gambar 3.2 metode implementasi *scrum*, dapat dijelaskan sebagai berikut [28], [32]:

1. *Product Backlog*

   Pada tahapan ini merupakan daftar pekerjaan yang harus dikerjakan untuk proses pengerjaan (*sprint*). Daftar pekerjaan dapat berupa definisi fitur untuk sebuah perangkat lunak, kebutuhan perangkat lunak, *user stories*, definisi arsitektur, atau dokumentasi pengguna.

2. *Sprint Planning*

   Pada tahapan ini terdapat *product owner* yang mengatur skala prioritas pada *product backlog* dengan memberikan nilai *story point* untuk memberikan estimasi seberapa *effort* dan bernilainya pekerjaan tersebut.

3. *Sprint Backlog*

   Pada tahapan ini merupakan daftar pekerjaan yang akan dikerjakan, sedang dikerjakan, sudah selesai, dan gagal dikerjakan. Kegunaan dari *sprint backlog* adalah dapat diketahuinya pekerjaan ini akan dilakukan oleh siapa, kapan dilakukannya, tenggat waktunya, serta statusnya.

4. *Sprint Review*

   Pada tahapan ini sebenarnya merupakan pertemuan yang diadakan setelah *sprint* selesai untuk dilakukan evaluasi. Hasil dari *sprint review*

berupa *velocity* dari akumulasi *story point* dari pekerjaan yang berhasil dikerjakan sebagai indikator performa dan pertimbangan *sprint* selanjutnya.

Metode evaluasi merupakan evaluasi terhadap pengembangan. Berdasarkan implementasi *scrum*, evaluasi yang digunakan adalah mengakumulasikan *velocity* dari *story point* yang tercantum pada *product backlog*. *Velocity* adalah bobot berat yang memiliki dampak cukup besar dari sisi bisnis dan pengguna. Dengan adanya *velocity*, dapat meninjau keberhasilan dari pengembangan yang mengimplementasikan *scrum*.

3.2.6. Lingkungan Pengembangan

Penelitian ini dilakukan di PT. Swa Nusa Multimedia yang beralamat di Jalan RC Veteran 11A, Rempoa, Bintaro, Pesanggrahan, Kota Jakarta Selatan 12330. Adapun perangkat lain yang dibutuhkan dalam penelitian ini adalah:

a. *GPS Tracker* Model GT06

   Kartu *SIM M2M* (*Machine-to-Machine*) dari AUTOTRONIC. Hal ini dibutuhkan supaya perangkat *GPS Tracker* dapat mengirimkan datanya ke *backend service*.

b. Server Ubuntu 20.04 LTS
   - Minimum *RAM*: 2GB
   - Minimum *vCPU*: 1

c. PHP v7.2.34

d. *Framework* Laravel v7.30.6

e. Nginx v1.25.4

f. NodeJS v12.22.12

g. Google Chrome versi rilis tahun 2021 ke atas.

BAB IV
IMPLEMENTASI DAN EVALUASI

4.1. Pengumpulan Data dan Informasi

4.1.1. Identifikasi Kebutuhan

Identifikasi masalah diperlukan supaya sistem yang dikembangkan pada penelitian ini solutif dan tidak melebar. Berdasarkan proses observasi dan wawancara yang dilakukan oleh peneliti, bahwa PT. Swa Nusa Multimedia memerlukan sistem pelacakan kendaraan untuk mengantarkan logistik baik untuk jarak kecil seperti dalam kelurahan atau antar-provinsi dalam lingkup pulau Jawa.

Umumnya proses pelacakan kendaraan dapat dilakukan menggunakan *smartphone* yang memiliki fitur *GPS*. Namun berdasarkan informasi dari penelitian [3], bahwa pelacakan dengan memanfaatkan perangkat *smartphone* mengonsumsi daya baterai lebih banyak. Selain itu juga tidak efisien untuk pengiriman antar-provinsi dalam lingkup pulau Jawa, dikarenakan sopir membutuhkan *smartphone* untuk keperluan lainnya.

Berdasarkan pertimbangan tersebut peneliti menemukan solusi pelacakan kendaraan dengan memanfaatkan perangkat *GPS Tracker*. Sehingga lebih efisien serta tidak membebani sopir untuk memasang sebuah aplikasi yang melacak lokasi dan membuat *smartphone* miliknya menjadi boros baterai.

4.1.2. *Software Requirement Specification* (*SRS*)

Software Requirement Specification (*SRS*) merupakan sebuah dokumen yang berisi seluruh aspek dari sebuah perangkat lunak yang akan dibangun harus dispesifikasikan sebelum pengerjaan itu dimulai [28]. *SRS* dapat berisi kebutuhan pengguna atau spesifikasi sistem secara mendetail, dan terkadang itu di satukan menjadi satu deskripsi [32].

Tabel 4.1 adalah *SRS* yang penulis dapatkan dari hasil observasi terhadap PT Swa Nusa Multimedia, melakukan wawancara kepada informan yang berwenang, dan eksperimen terhadap perangkat *GPS Tracker*.

Tabel 4. 1 *Software Requirements Specification* (*SRS*)

Kode	Deskripsi	Aspek
R.01	*Backend service* dapat menerima data dari *GPS Tracker*.	Sistem
R.02	*Backend service* dapat membaca *login message* dan membalas *login message* dari dan untuk *GPS Tracker*.	Sistem
R.03	*Backend service* dapat membaca *location data* dari *GPS Tracker* dan menyimpannya ke basis data.	Sistem
R.04	*Backend service* dapat membaca *heartbeat data* dari *GPS Tracker* dan menyimpannya ke basis data.	Sistem
R.05	*Backend service* dapat membaca *alarm data* dari *GPS Tracker* dan menyimpannya ke basis data.	Sistem
R.06	*Backend service* dapat membedakan setiap perangkat yang terhubung berdasarkan koneksi untuk disimpan ke basis data setiap kali perangkat mengirimkan data (mengidentifikasi *id* perangkat berdasarkan koneksi).	Sistem
R.07	*Backend service* dapat melakukan *reverse geocoding* dengan memanfaatkan *API* dari OpenStreetMap untuk mendapatkan alamat lengkap, dikarenakan data dari perangkat *GPS Tracker* hanya berupa *latitude* dan *longitude*.	Sistem
R.08	*User* dapat melakukan *login*.	*User*
R.09	Terdapat *role* admin & *role* untuk pelacakan saja.	*User*

R.10	Semua *role* dapat mengetahui lokasi terkini kendaraan di *dashboard* dan dapat memantaunya secara *real-time*.	*User*
R.11	Semua *role* dapat mengetahui status pergerakan kendaraan di *dashboard*.	*User*
R.12	Semua *role* dapat mengetahui informasi jarak tempuh kendaraan di *dashboard*.	*User*
R.13	Semua *role* dapat mengetahui informasi kendaraan di *dashboard*.	*User*
R.14	Semua *role* dapat mengetahui informasi pengemudi di *dashboard*.	*User*
R.15	Semua *role* dapat melacak riwayat perjalanan kendaraan di *dashboard*.	*User*
R.16	*Role* admin dapat mengelola kendaraan dan melihat informasi jarak tempuh kendaraan.	*User*
R.17	*Role* admin dapat mengelola pengemudi.	*User*
R.18	*Role* admin dapat mengelola pengguna web.	*User*
R.19	*Role* admin dapat mengatur akses yang spesifik terhadap *role* pelacakan terkait kendaraan yang dilihatnya pada halaman *dashboard*.	*User*
R.20	*Role* admin dapat mengelola perangkat *GPS Tracker*.	*User*
R.21	*Role* admin dapat memantau kesuksesan integrasi perangkat *GPS Tracker* dengan sistem.	*User*

Tabel 4.1 terdapat 3 kolom, yaitu kolom kode, deskripsi, dan aspek.

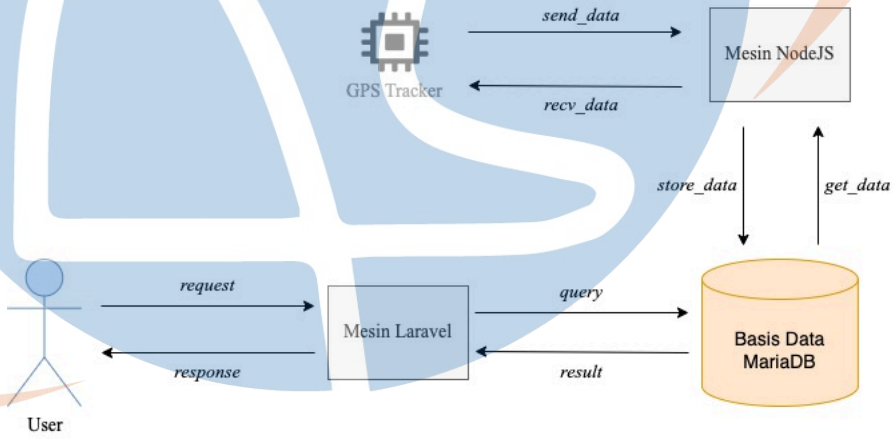
1. Kode: Kolom kode bertujuan supaya lebih mudah diidentifikasi dibandingkan jika harus menjelaskan deskripsinya secara berulang.
2. Deskripsi: Kolom deskripsi merupakan deskripsi spesifikasi perangkat lunak yang akan dibuat.

3. Aspek: Kolom aspek merupakan keterangan untuk kolom deskripsi yang melakukan spesifikasi dari aspek *user*, sistem, atau keduanya.

4.2. Perancangan Sistem

4.2.1. Arsitektur Sistem

Arsitektur sistem pada penelitian ini memiliki 2 buah mesin, yaitu mesin NodeJS untuk menerima, memproses, dan menyimpan data dari perangkat *GPS Tracker* ke basis data MariaDB. Kemudian mesin Laravel bertujuan untuk membuat aplikasi web seperti mengelola kendaraan, perangkat *GPS Tracker*, pengemudi, mengelola *users*, dan melihat *dashboard*. Berikut gambar 4.1 yang menggambarkan arsitektur sistem secara keseluruhan.



Gambar 4. 1 Arsitektur Sistem

Berikut penjelasan dan cara kerja sistem dari gambar 4.1 adalah sebagai berikut:

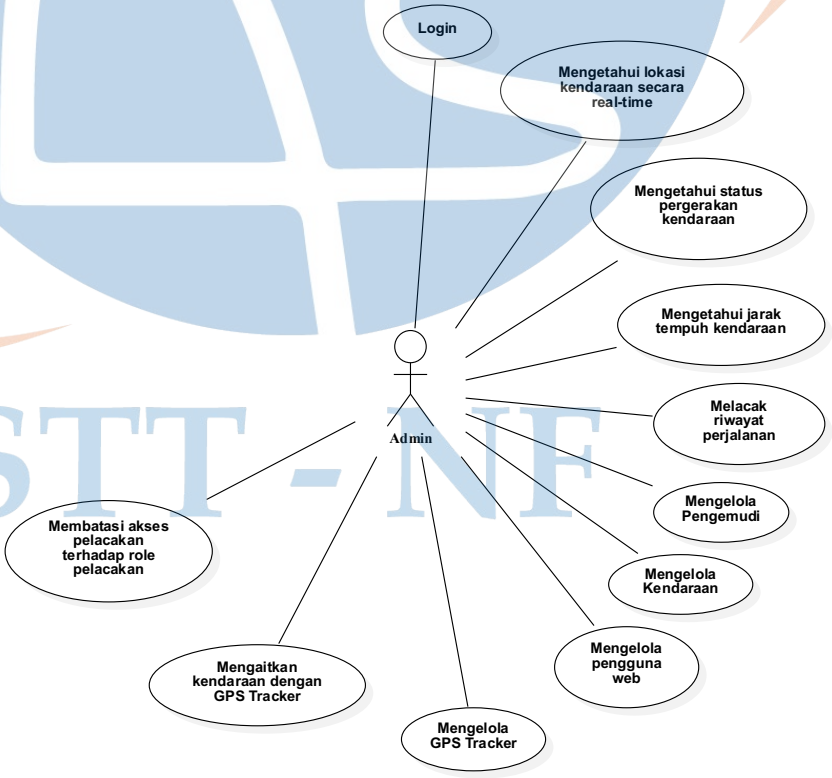
- *User* yang memiliki peran admin atau pelacakan akan mengakses *website* melalui Laravel.
- Laravel menerapkan konsep *Model View Controller* (*MVC*) untuk menerima *request* dari *user* dan memberikan *response* kepada *user*.

Laravel juga melakukan *query* ke basis data MariaDB dan memberikan *result* kepada *user* yang diterima sebagai *response*.

- MariDB sebagai basis data untuk menampung dan mengolah data dari Laravel yang dikelola oleh *user* dan dari perangkat *GPS Tracker* melalui NodeJS.
- NodeJS sebagai *backend service* untuk memperoleh dan mengolah data perangkat *GPS Tracker*.

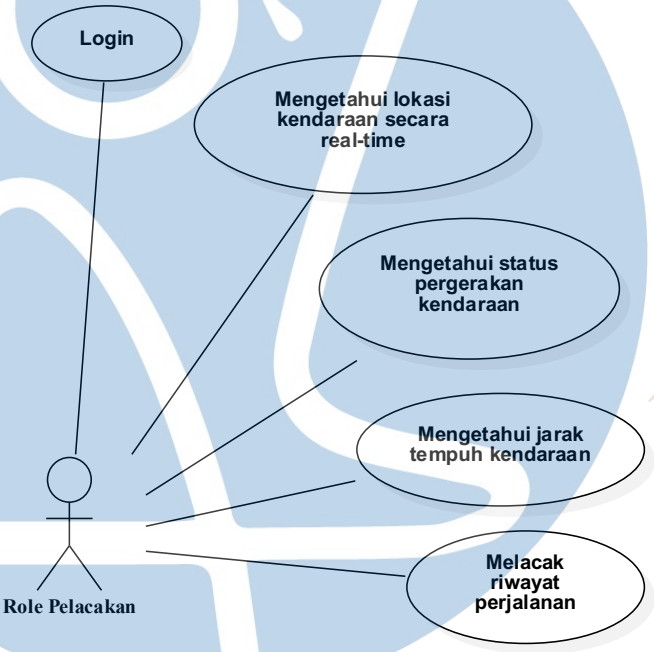
4.2.2. Use Case Diagram

Use case diagram merupakan salah satu pemodelan pada *UML* untuk mendesain interaksi antara aktor (*user*) dengan sistem secara umum dan menyeluruh. *Use case diagram* mendeskripsikan apa saja yang bisa dilakukan oleh *user* terhadap sistem. Berikut gambar 4.2 merupakan *Use case diagram* admin pada sistem pelacakan kendaraan.



Gambar 4. 2 *Use Case Diagram* Admin

Pada sistem pelacakan terdapat *role user* yang lainnya, yaitu *role* pelacakan. Berbeda dengan *role* admin yang dapat mengakses semua fitur, *role* pelacakan hanya dapat melihat *dashboard* terhadap kendaraan yang sudah di batasi aksesnya oleh *role* admin. Tujuan dari *role* pelacakan adalah memberikan akses kepada sopir untuk melacak kendaraannya. Berikut gambar 4.3 *use case diagram role* pelacakan.

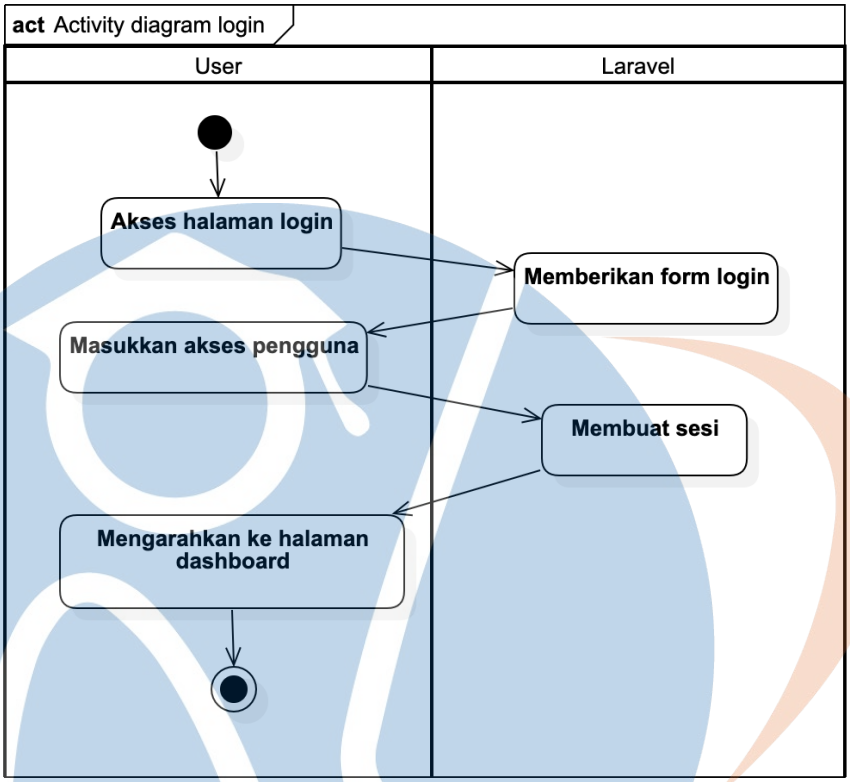


Gambar 4. 3 *Use Case Diagram Role* Pelacakan

4.2.3. Activity Diagram

Activity diagram merupakan salah satu pemodelan pada *UML* untuk menggambarkan proses bisnis baik saat suatu sistem digunakan atau rencana penggunaan sistem yang setiap aktivitasnya merepresentasikan sebuah proses. *Activity diagram* didesain berdasarkan *use case diagram* yang telah dibuat untuk memahaminya lebih detail, berikut *activity diagrams*:

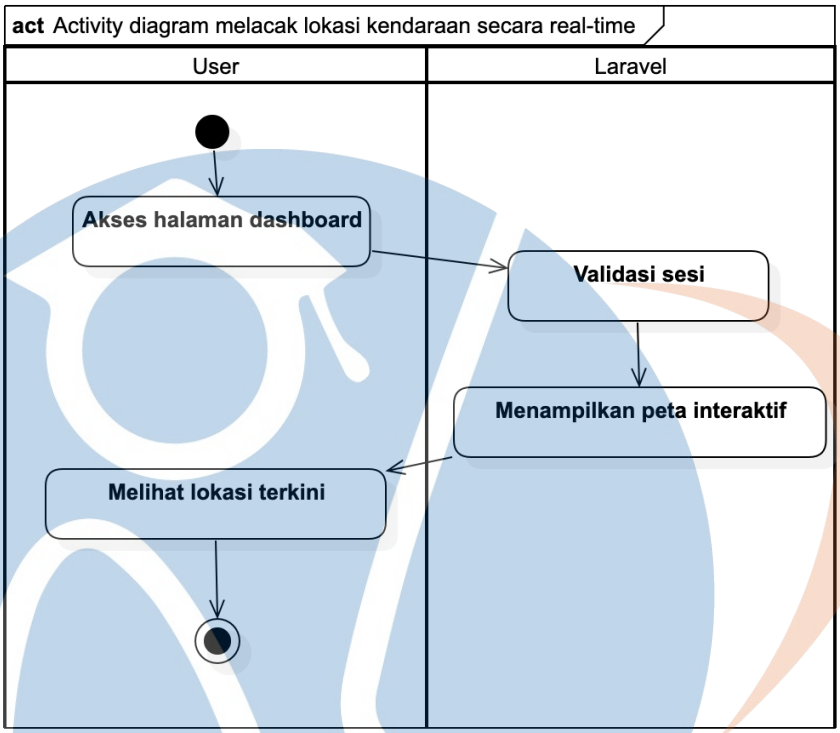
1. *Activity diagram login*



Gambar 4. 4 *Activity Diagram Login*

Keterangan untuk gambar 4.4 bahwa *role* admin dan *role* pelacakan dapat melakukan *login*.

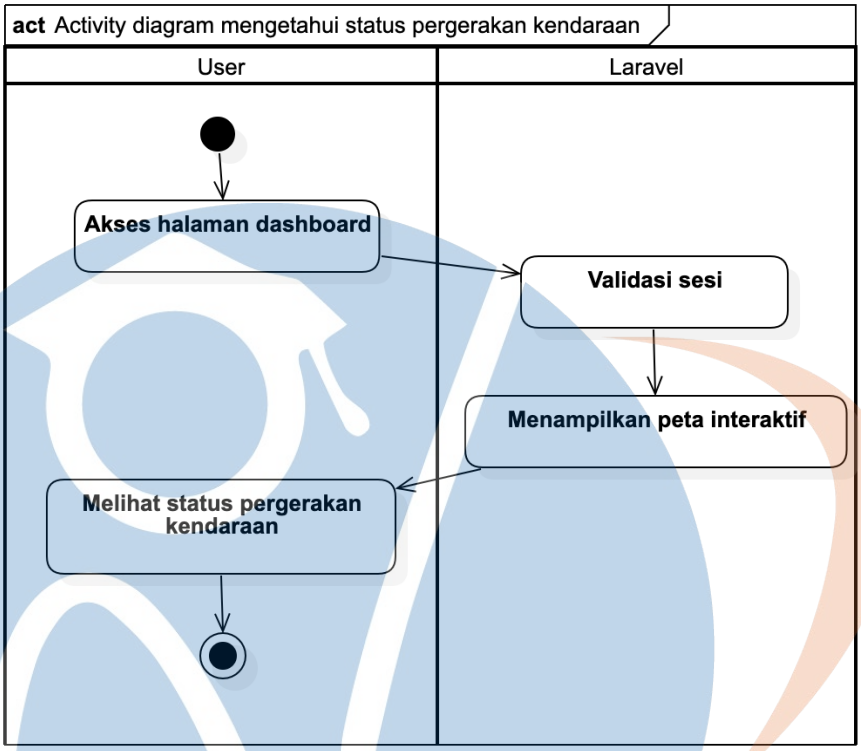
2. *Activity diagram* melacak lokasi kendaraan secara *real-time*



Gambar 4. 5 *Activity Diagram* Melacak Lokasi Kendaraan Secara *Real-Time*

Keterangan untuk gambar 4.5 bahwa *role* admin dan *role* pelacakan juga dapat melakukannya.

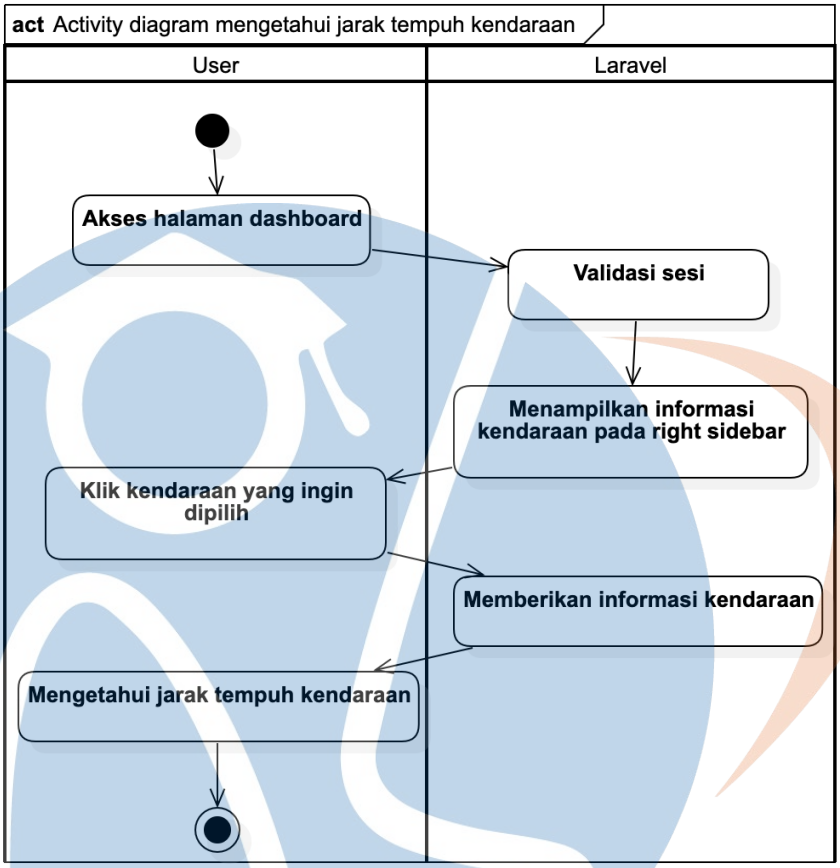
3. *Activity diagram* mengetahui status pergerakan kendaraan



Gambar 4. 6 *Activity Diagram* Mengetahui Status Pergerakan Kendaraan

Keterangan untuk gambar 4.6 bahwa *role* admin dan *role* pelacakan juga dapat melakukannya.

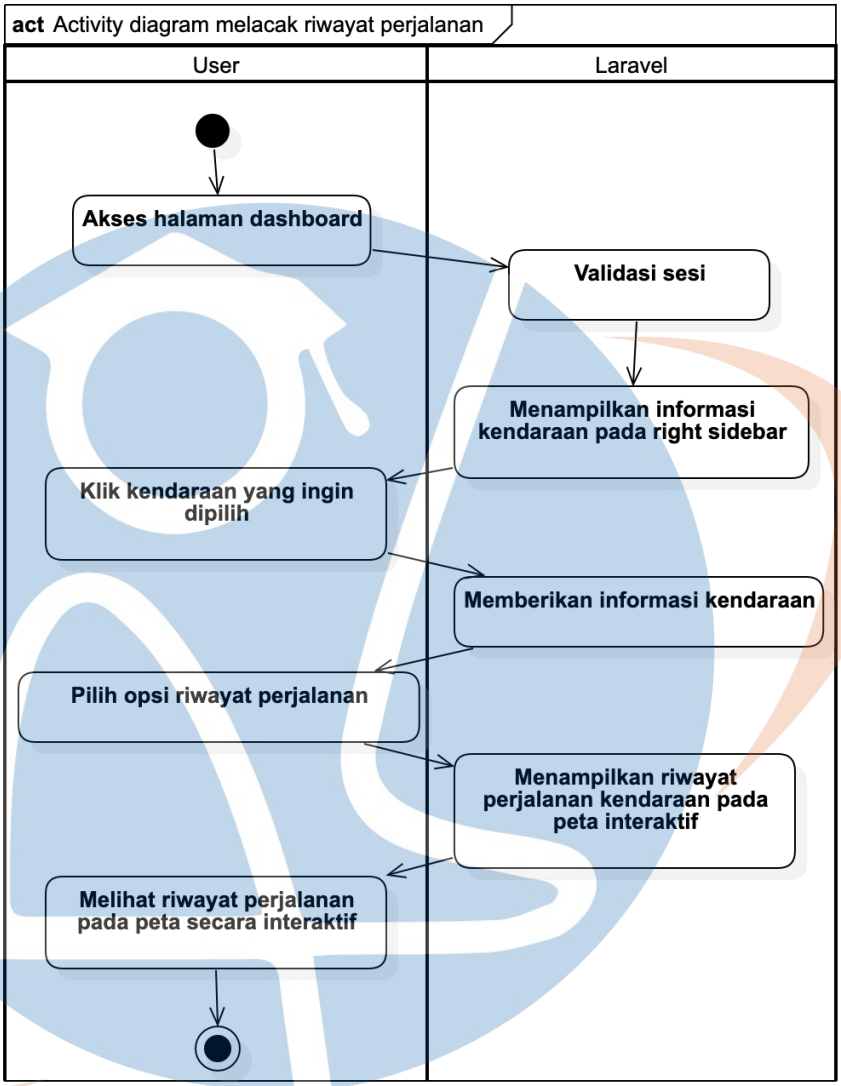
4. *Activity diagram* mengetahui jarak tempuh kendaraan



Gambar 4. 7 *Activity Diagram* Mengetahui Jarak Tempuh Kendaraan

Keterangan untuk gambar 4.7 bahwa *role* admin dan *role* pelacakan juga dapat melakukannya.

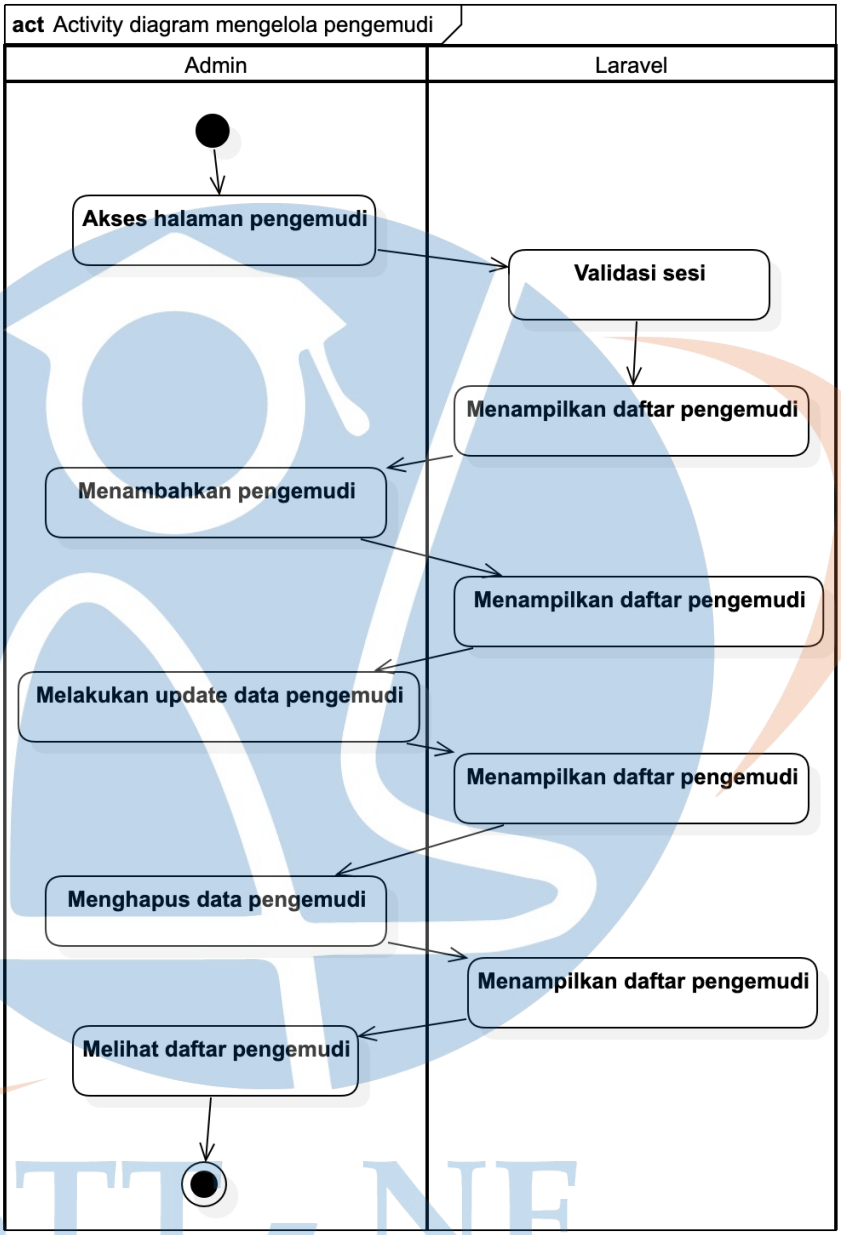
5. *Activity diagram* melacak riwayat perjalanan



Gambar 4. 8 *Activity Diagram* Melacak Riwayat Perjalanan

Keterangan untuk gambar 4.8 bahwa *role* admin dan *role* pelacakan juga dapat melakukannya.

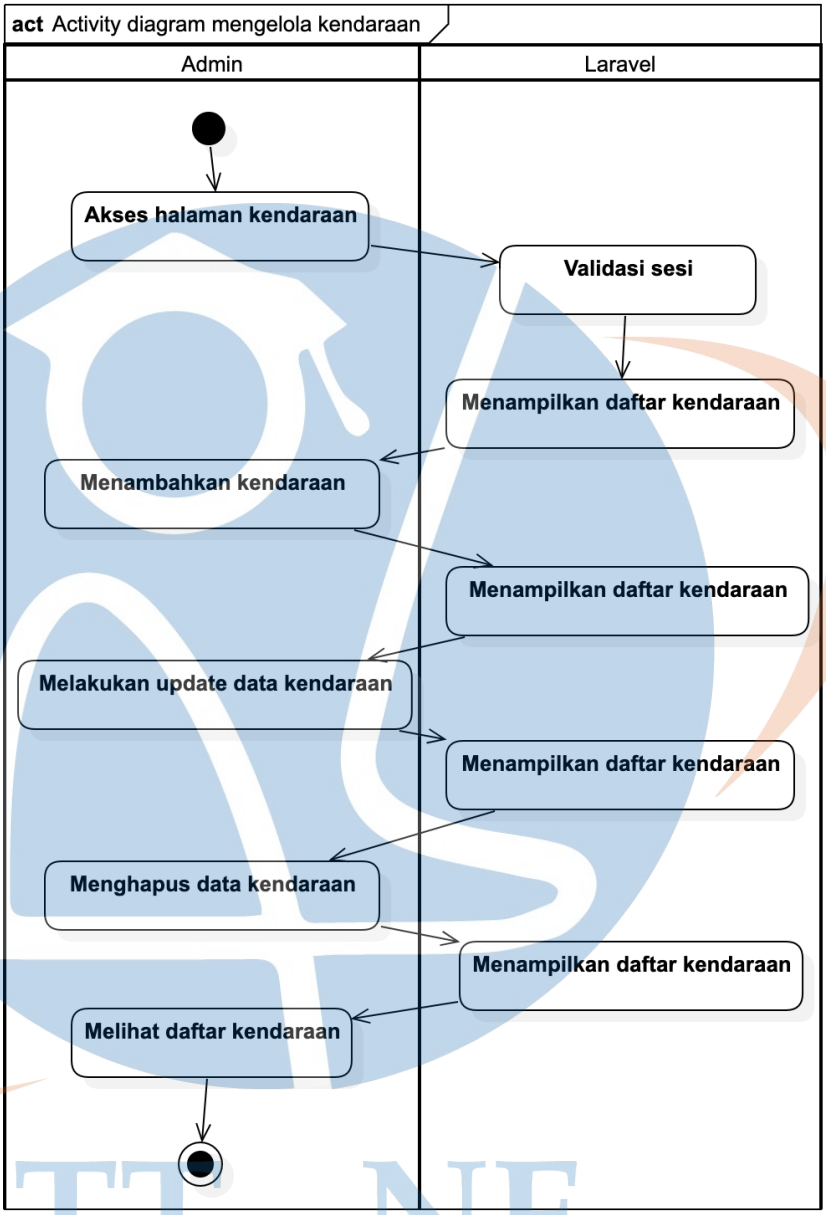
6. *Activity diagram* mengelola pengemudi



Gambar 4. 9 *Activity Diagram* Mengelola Pengemudi

Keterangan untuk gambar 4.9 bahwa hanya *role* admin yang dapat melakukannya.

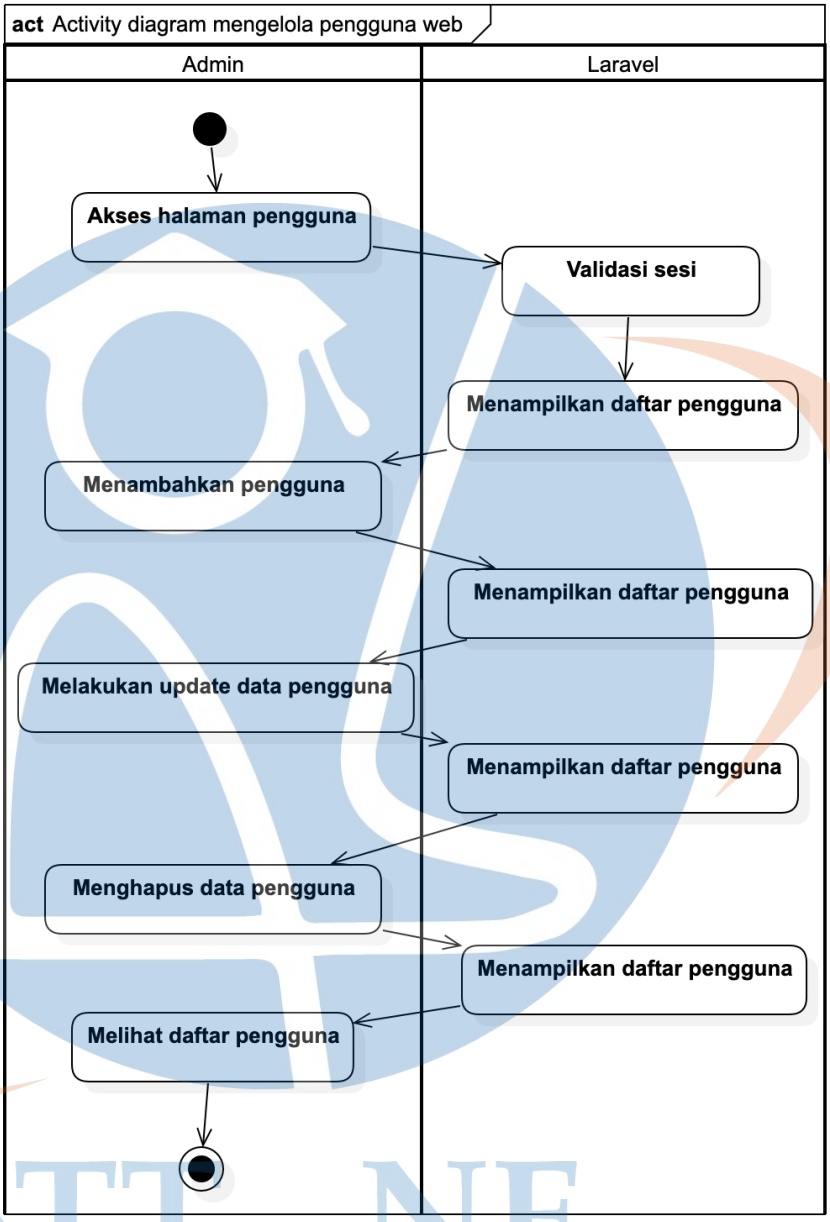
7. *Activity diagram* mengelola kendaraan



Gambar 4. 10 *Activity Diagram* Mengelola Kendaraan

Keterangan untuk gambar 4.10 bahwa hanya *role* admin yang dapat melakukannya.

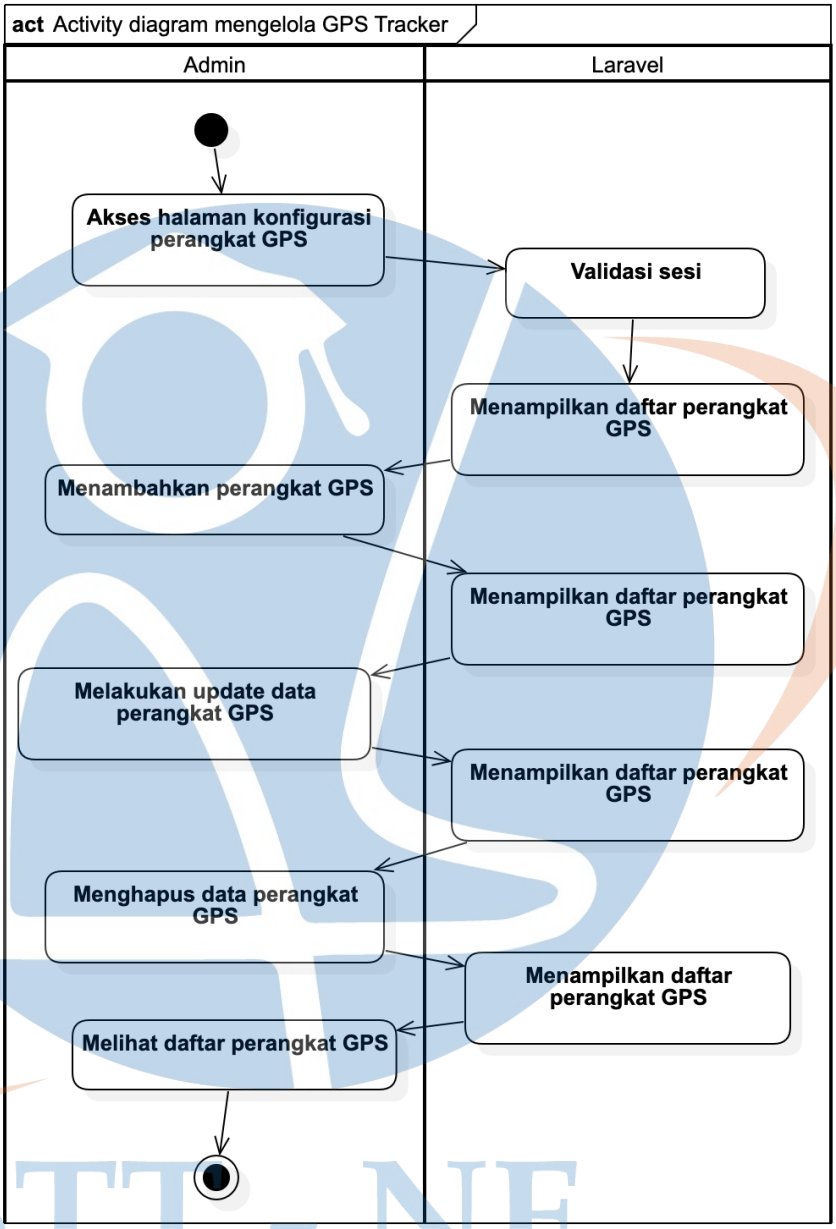
8. *Activity diagram* mengelola pengguna web



Gambar 4. 11 *Activity Diagram* Mengelola Pengguna Web

Keterangan untuk gambar 4.11 bahwa hanya *role* admin yang dapat melakukannya.

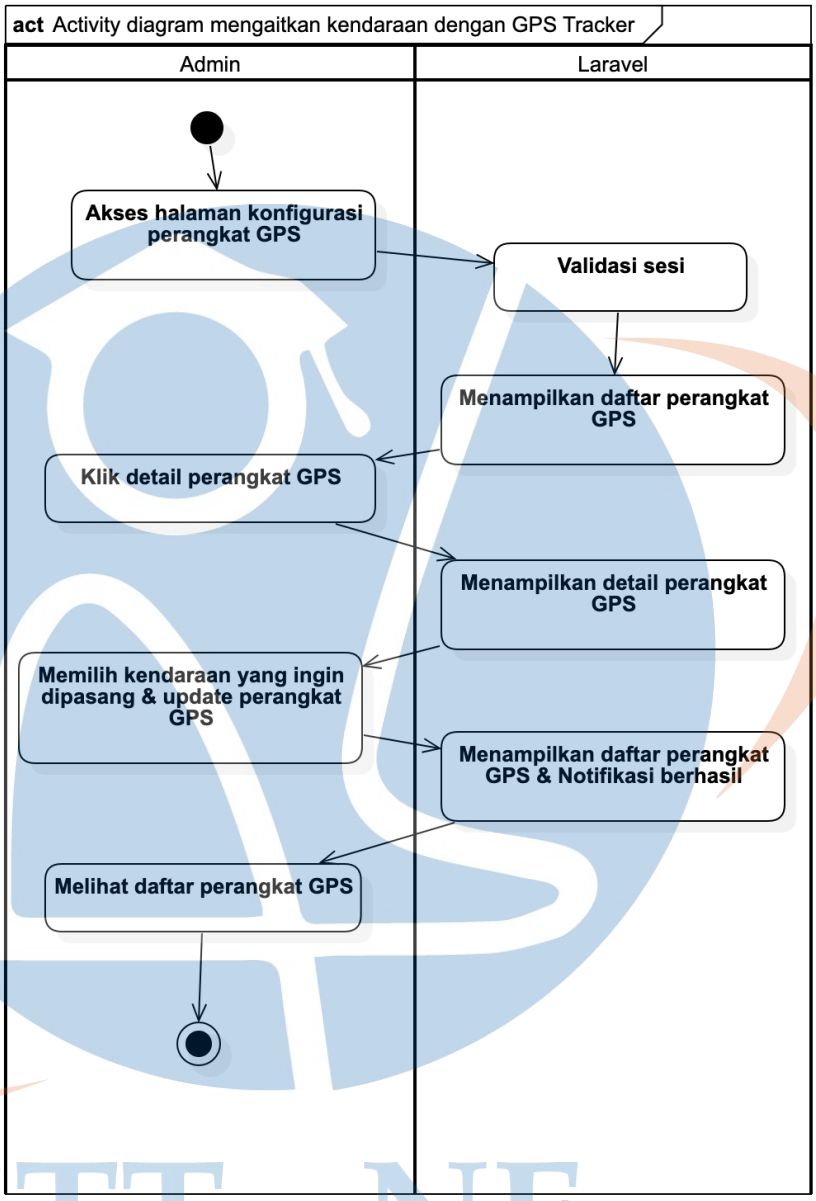
9. *Activity diagram* mengelola *GPS Tracker*



Gambar 4. 12 *Activity Diagram* Mengelola *GPS Tracker*

Keterangan untuk gambar 4.12 bahwa hanya *role* admin yang dapat melakukannya.

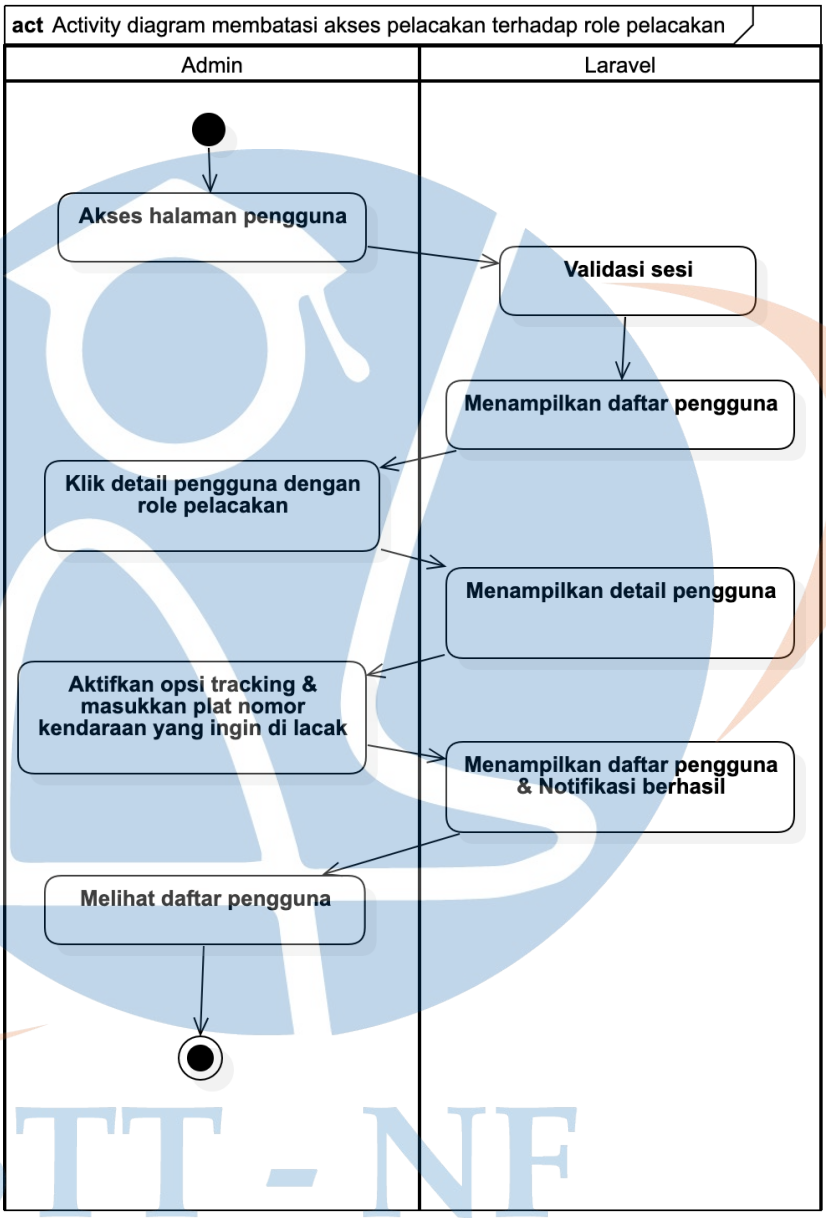
10. *Activity diagram* mengaitkan kendaraan dengan *GPS Tracker*



Gambar 4. 13 *Activity Diagram* Mengaitkan Kendaraan Dengan *GPS Tracker*

Keterangan untuk gambar 4.13 bahwa hanya *role* admin yang dapat melakukannya.

11. *Activity diagram* membatasi akses pelacakan terhadap *role* pelacakan



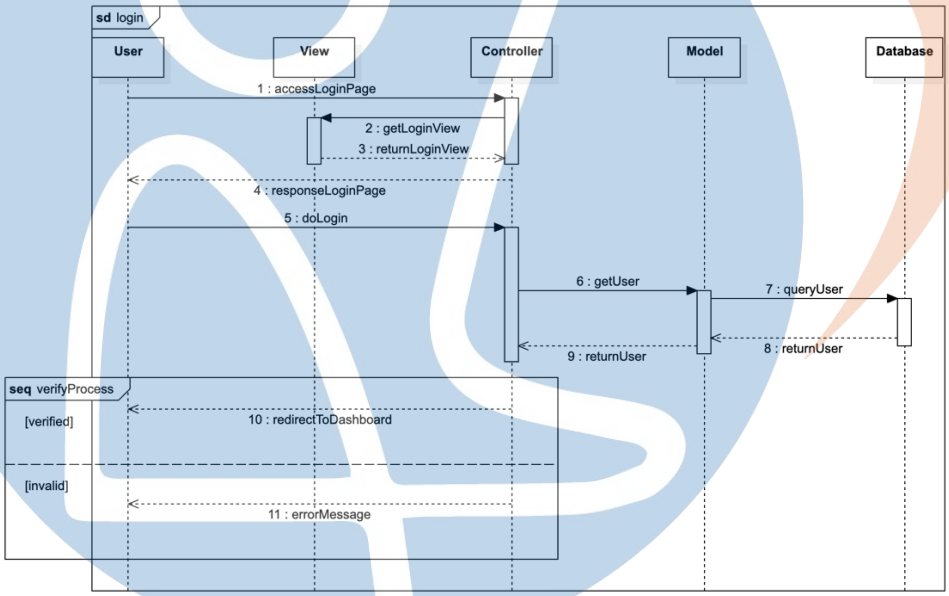
Gambar 4. 14 *Activity Diagram* Membatas Akses Pelacakan Terhadap Role Pelacakan

Keterangan untuk gambar 4.14 bahwa hanya *role* admin yang dapat melakukannya.

4.2.4. Sequence Diagram

Sequence diagram merupakan pemodelan yang digunakan untuk menggambarkan interaksi antara komponen-komponen sistem dengan aktor (*user*). *Sequence diagram* menggambarkan interaksi dengan sistem secara mendetail, dibandingkan *activity diagram* yang hanya menggambarkan proses bisnis dan cara kerja sistem secara umum. Berikut *sequence diagrams* pada sistem pelacakan.

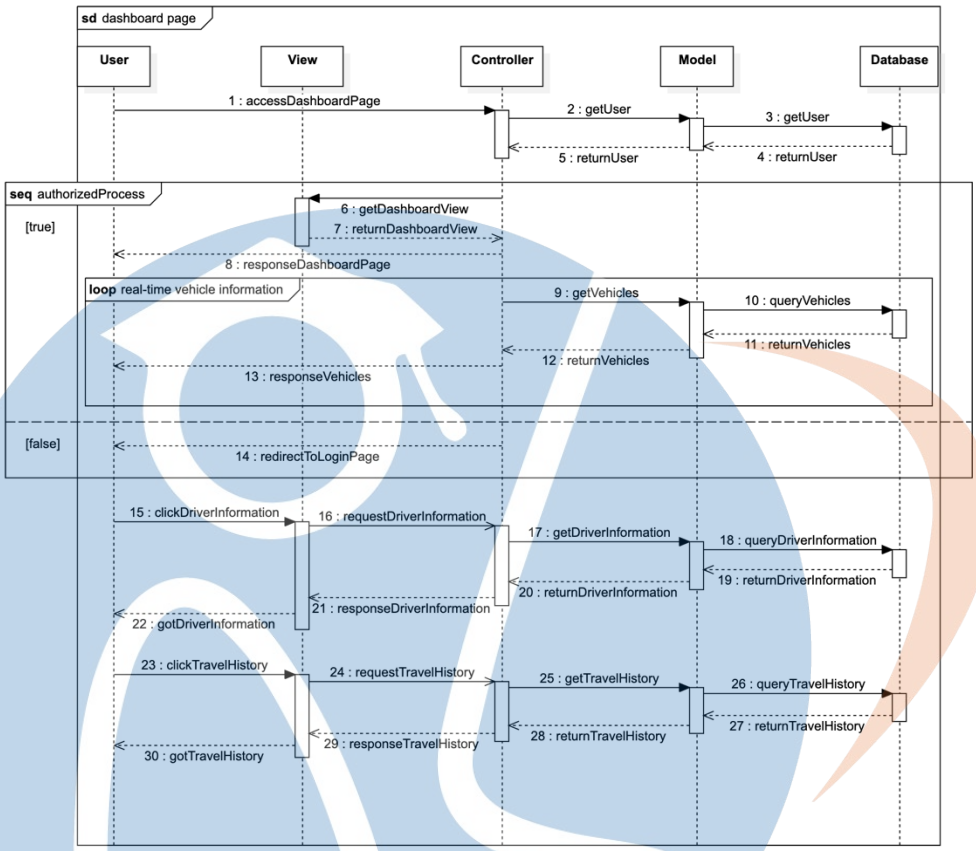
1. *Sequence diagram login*.



Gambar 4. 15 *Sequence Diagram Login*

Pada gambar 4.15 menjelaskan interaksi antar komponen sistem ketika *user* melakukan *login*. Proses melakukan *login* terbagi menjadi 2 tahapan. Tahapan pertama dari 1- 4 adalah mengakses halaman *login*. Tahapan kedua dari 5 – 11 yaitu mengecek kredensial yang dimasukkan oleh *user* ketika *login*.

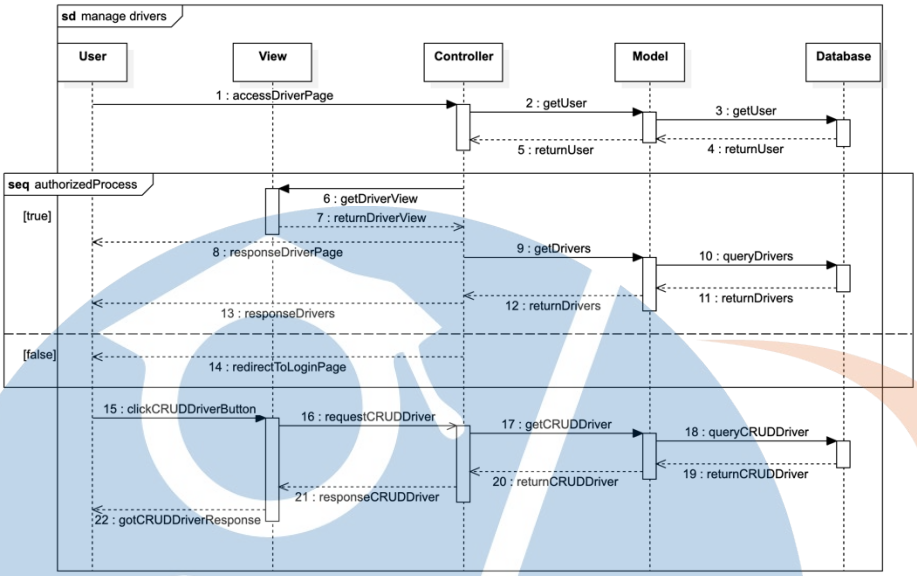
2. *Sequence diagram dashboard page.*



Gambar 4. 16 *Sequence Diagram Dashboard Page*

Pada gambar 4.16 menjelaskan interaksi antar komponen sistem ketika *user* mengakses atau berada di *dashboard page*. Tahapan pada *sequence diagram dashboard page* terbagi menjadi 4 tahapan. Tahapan pertama adalah proses otorisasi yang terjadi pada 1 – 14. Tahapan kedua adalah mendapatkan data kendaraan secara *real-time* pada 9 - 13, termasuk di dalamnya lokasi kendaraan, status pergerakan kendaraan, dan jarak tempuh kendaraan. Tahapan ketiga adalah proses mendapatkan detail pengemudi pada 15 - 22. Tahapan keempat adalah proses mendapatkan riwayat perjalanan pada 23 - 30.

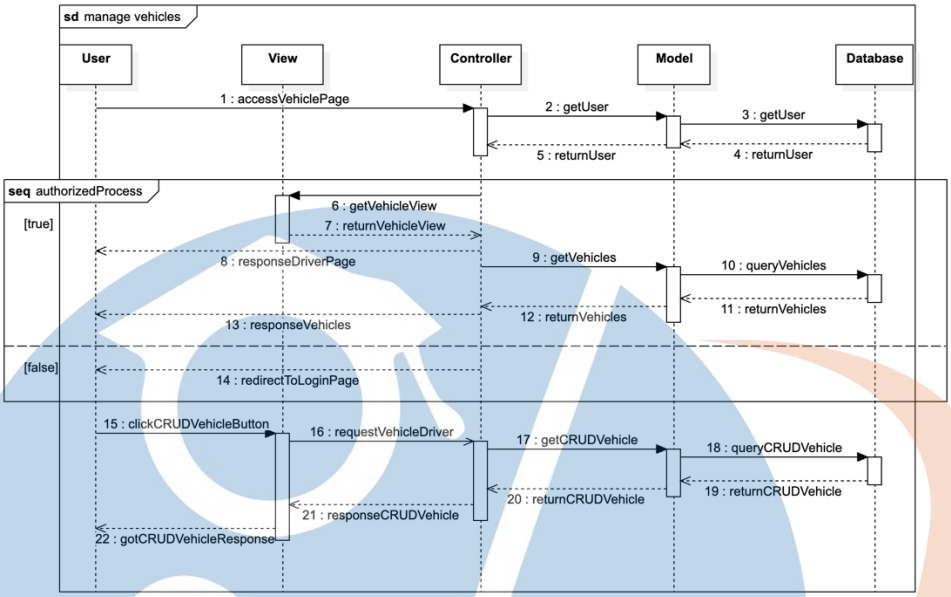
3. *Sequence diagram* mengelola pengemudi.



Gambar 4. 17 *Sequence Diagram* Mengelola Pengemudi

Pada gambar 4.17 menjelaskan interaksi antar komponen sistem ketika *user* mengakses atau berada di *driver page*. Pada *sequence diagram* mengelola pengemudi terbagi menjadi 3 tahapan. Tahapan pertama pada 1 – 14 merupakan proses otorisasi. Tahapan kedua pada 9 – 13 merupakan proses mendapatkan daftar tabel pengemudi. Tahapan ketiga pada 15 – 22 merupakan proses melakukan penambahan data, detail data, *update* data, dan hapus data terkait pengemudi.

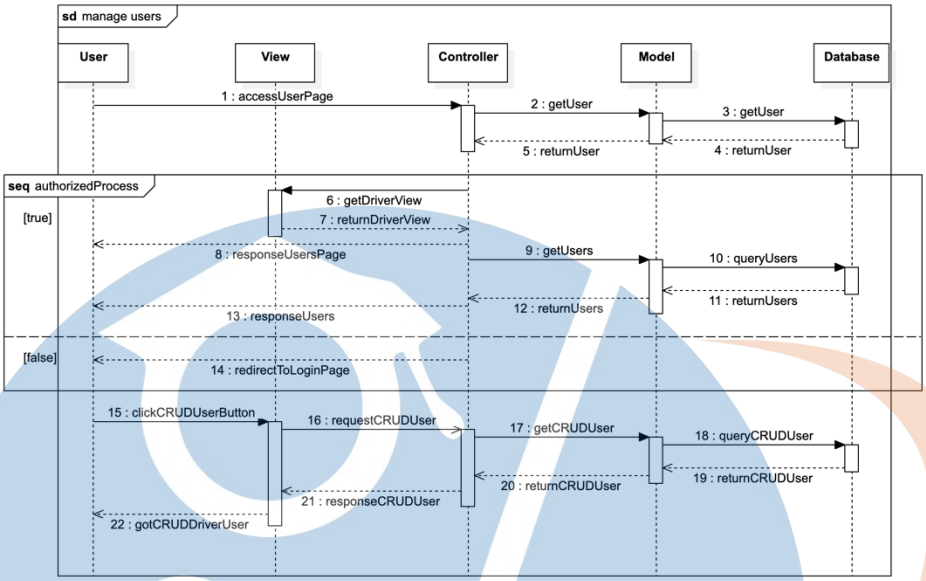
4. *Sequence diagram* mengelola kendaraan.



Gambar 4. 18 *Sequence Diagram* Mengelola Kendaraan

Pada gambar 4.18 menjelaskan interaksi antar komponen sistem ketika *user* mengakses atau berada di *vehicle page*. Pada *sequence diagram* mengelola kendaraan terbagi menjadi 3 tahapan. Tahapan pertama pada 1 – 14 merupakan proses otorisasi. Tahapan kedua pada 9 – 13 merupakan proses mendapatkan daftar tabel kendaraan. Tahapan ketiga pada 15 – 22 merupakan proses melakukan penambahan data, detail data, *update* data, dan hapus data terkait kendaraan.

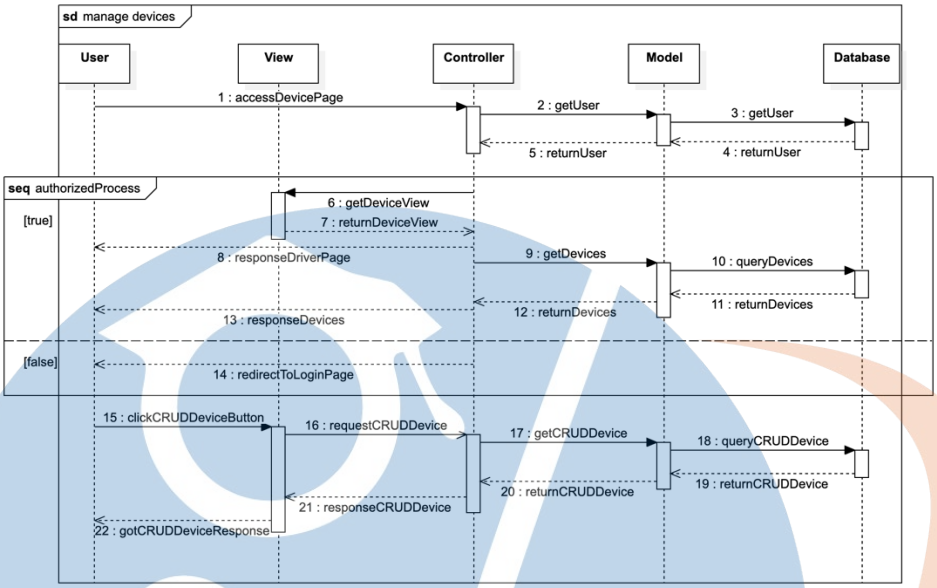
5. *Sequence diagram* mengelola pengguna web.



Gambar 4. 19 *Sequence Diagram* Mengelola Pengguna Web

Pada gambar 4.19 menjelaskan interaksi antar komponen sistem ketika *user* mengakses atau berada di *user page*. Pada *sequence diagram* mengelola pengguna web terbagi menjadi 3 tahapan. Tahapan pertama pada 1 – 14 merupakan proses otorisasi. Tahapan kedua pada 9 – 13 merupakan proses mendapatkan daftar tabel pengguna web. Tahapan ketiga pada 15 – 22 merupakan proses melakukan penambahan data, detail data, *update* data, dan hapus data terkait pengguna web.

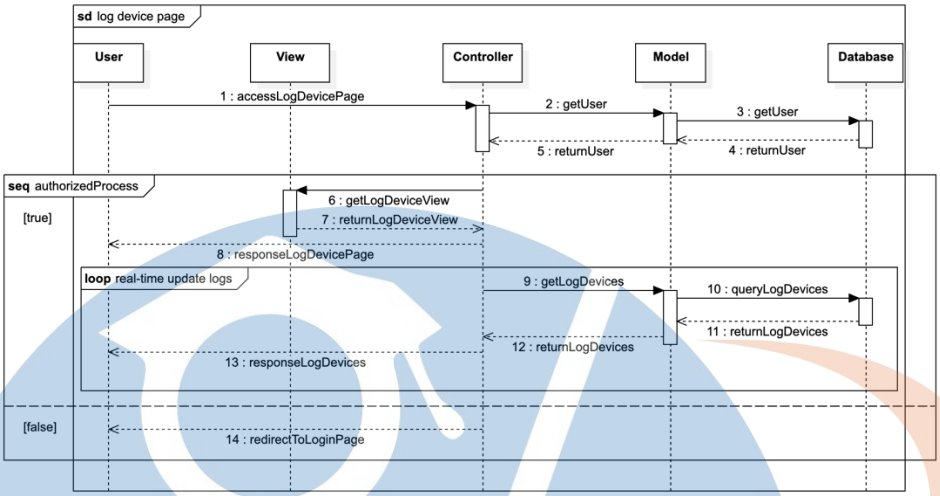
6. *Sequence diagram* mengelola *GPS Tracker*.



Gambar 4. 20 *Sequence Diagram* Mengelola *GPS Tracker*

Pada gambar 4.20 menjelaskan interaksi antar komponen sistem ketika *user* mengakses atau berada di *device page*. Pada *sequence diagram* mengelola *GPS Tracker* terbagi menjadi 3 tahapan. Tahapan pertama pada 1 – 14 merupakan proses otorisasi. Tahapan kedua pada 9 – 13 merupakan proses mendapatkan daftar tabel *GPS Tracker*. Tahapan ketiga pada 15 – 22 merupakan proses melakukan penambahan data, detail data, *update* data, dan hapus data terkait *GPS Tracker*. Pada tahapan ketiga proses mengaitkan perangkat *GPS Tracker* dengan kendaraan juga termasuk di dalamnya.

7. *Sequence diagram* memantau proses integrasi *GPS Tracker* ke sistem.

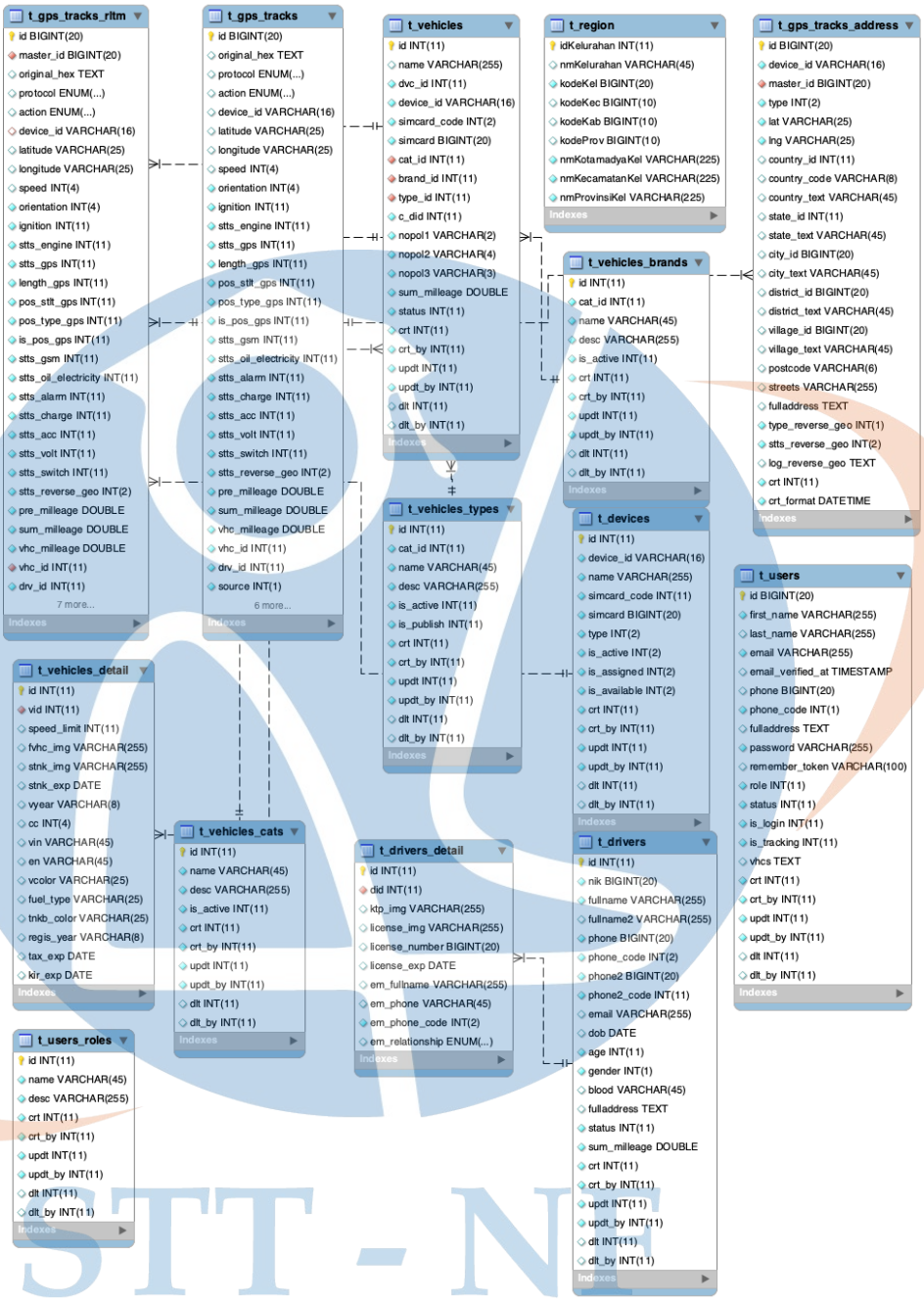


Gambar 4. 21 *Sequence Diagram* Memantau Proses Integrasi *GPS Tracker* ke Sistem

Pada gambar 4.21 menjelaskan interaksi antar komponen sistem ketika *user* mengakses atau berada di log *device page*. Pada *sequence diagram* memantau proses integrasi *GPS Tracker* ke sistem terbagi menjadi 2 tahapan. Tahapan pertama pada 1 – 14 merupakan proses otorisasi. Tahapan kedua pada 9 – 13 merupakan proses memantau integrasi *GPS Tracker* ke sistem.

4.2.5. *Entity Relationship Diagram* (*ERD*)

Entity relationship diagram (*ERD*) merupakan perancangan yang digunakan untuk mendesain sebuah basis data. *ERD* merepresentasikan hubungan antar tabel pada sebuah model secara visual. Dengan adanya *ERD* lebih efisien dalam hal perancangan, karena *ERD* bisa langsung dikonversi menjadi sebuah basis data yang nyata. Berikut gambar 4.22 yang merupakan *ERD* untuk sistem pelacakan kendaraan.



Gambar 4. 22 *ERD* Sistem Pelacakan Kendaraan

Gambar 4.22 terdapat banyak tabel dan relasi, tidak cukup jelas hanya dengan gambar, berikut ringkasan penjelasan dari masing-masing tabel.

1. *t_users*: Tabel untuk menampung data pengguna web.
    - *id*: *id* kepanjangan dari *identifier* sebagai identitas dari sebuah data, merupakan *primary key* dan nilainya berupa *auto increment*.
    - *role*: Merupakan *foreign key* yang memiliki relasi ke *t_users_roles.id* dengan relasi *one to many*, karena satu *role* bisa digunakan oleh banyak *users*.
2. *t_users_roles*: Tabel untuk menampung *role* pengguna web.
3. *t_drivers*: Merupakan data pengemudi.
4. *t_drivers_detail*: Merupakan detail pengemudi.
    - *id*: *id* kepanjangan dari *identifier* sebagai identitas dari sebuah data, merupakan *primary key* dan nilainya berupa *auto increment*.
    - *did*: *did* merupakan singkatan dari *driver id*, yang memiliki relas ke *t_drivers.id*, dengan relasi *one to one*, karena hanya 1 pengemudi yang memiliki 1 detail.
5. *t_vehicles*: Merupakan data kendaraan.
    - *id*: *id* kepanjangan dari *identifier* sebagai identitas dari sebuah data, merupakan *primary key* dan nilainya berupa *auto increment*.
    - *device_id*: Merupakan *foreign key* yang memiliki relas ke *t_devices.device_id* dengan relasi *one to one*, karena hanya 1 kendaraan yang dapat terkaitkan dengan satu *device*.
    - *cat_id*: Merupakan *foreign key* yang memiliki relas ke *t_vehicles_cats.id* dengan relasi *one to many*, karena satu label jenis kendaraan dapat digunakan oleh banyak kendaraan.
    - *brand_id*: Merupakan *foreign key* yang memiliki relas ke *t_vehicles_brands.id* dengan relasi *one to many*, karena satu label *brand* kendaraan dapat digunakan oleh banyak kendaraan.

- *type_id*: Merupakan *foreign key* yang memiliki relas ke *t_vehicles_types.id* dengan relasi *one to many*, karena satu label tipe kendaraan dapat digunakan oleh banyak kendaraan.
- *c_did*: Kolom ini bermakna *current driver id*, untuk mengetahui kendaraan ini digunakan oleh pengemudi siapa. Kolom ini merupakan *foreign key* yang memiliki relas ke *t_drivers id* dengan relasi *one to one*, karena hanya satu pengemudi yang dapat mengemudikan kendaraan.

6. *t_vehicles_detail*: Merupakan detail kendaraan.
    - *id*: *id* kepanjangan dari *identifier* sebagai identitas dari sebuah data, merupakan *primary key* dan nilainya berupa *auto increment*.
    - *vid*: *vid* merupakan singkatan dari *vehicle id*, yang memiliki relas ke *t_vehicles.id*, dengan relasi *one to one*, karena hanya 1 kendaraan yang memiliki 1 detail.
7. *t_vehicles_cats*: Merupakan jenis kendaraan seperti motor, mobil, dan truk.
8. *t_vehicle_brands*: Merupakan *brand* kendaraan seperti Honda, Mitsubishi, Hino, dan lain-lain.
9. *t_vehicle_types*: Merupakan tipe-tipe kendaraan seperti mobil box, verza, dan lain-lain.
10. *t_devices*: Merupakan perangkat *GPS Tracker*.
11. *t_gps_tracks*: Merupakan data yang didapatkan dari perangkat *GPS Tracker*.
    - *id*: *id* kepanjangan dari *identifier* sebagai identitas dari sebuah data, merupakan *primary key* dan nilainya berupa *auto increment*.
    - *original_hex*: Merupakan karakter *hexadecimal* yang didapatkan dari *GPS Tracker*.

- *protocol*: Merupakan nama protocol yang digunakan untuk membaca karakter *hexadecimal* tersebut.
- *action*: Merupakan jenis data yang dikirimkan dari *GPS Tracker*, nilainya berupa *login*, *location*, *heartbeat*, *alarm*, dan *other*.
- *device_id*: Merupakan *International Mobile Equipment Identity* (*IMEI*) pada perangkat *GPS Tracker* dan memiliki relasi ke *t_devices.device_id*, jenis relasinya *one to many*, karena satu *device_id* dapat memiliki banyak catatan pelacakan.
- *latitude*: Merupakan titik koordinat lokasi dalam bentuk *decimal degree* yang mengukur utara dan selatan bumi.
- *longitude*: Merupakan titik koordinat lokasi dalam bentuk *decimal degree* yang mengukur barat dan timur bumi.
- *speed*: Merupakan kecepatan dalam satuan *km/h*.
- *orientation*: Merupakan arah perangkat *GPS Tracker* dalam bentuk 360 derajat.
- *ignition*: Merupakan kondisi perapian apakah tinggi atau rendah.
- *stts_engine*: Merupakan status pergerakan kendaraan terdapat nilai *idling*, *moving*, dan *stopping*.
- *stts_gps*: Merupakan status *gps* pada perangkat *GPS Tracker* apakah aktif atau tidak.
- *length_gps*: Berdasarkan format protokol GT06 merupakan informasi panjang *GPS*.
- *post_stlt_gps*: Berdasarkan format protokol GT06 merupakan jumlah satellit yang terhubung ketika memposisikan posisi.
- *pos_type_gps*: Berdasarkan format protokol GT06 merupakan *GPS real-time* / *differential positioning*.
- *is_pos_gps*: Berdasarkan format protokol GT06 merupakan *GPS* telah memposisikan posisi atau tidak.
- *stts_gsm*: Status sinyal *GSM*.

- *stts_oil_electricity*: Status minyak dan kelistrikan terhubung atau tidak.
- *stts_alarm*: Berdasarkan format protokol GT06 merupakan *SOS*, alarm baterai lemah, *power cut alarm*, *shock alarm*, normal.
- *stts_charge*: Status sedang di *charge* atau tidak.
- *stts_acc*: Berdasarkan format protokol GT06 merupakan status *ACC* sedang tinggi atau rendah.
- *stts_volt*: Berdasarkan format protokol GT06 merupakan status kelistrikan dimulai dari 0 yang merupakan paling rendah sampai paling tinggi bernilai 6.
- *stts_switch*: Berdasarkan format protokol GT06 merupakan informasi terminal apakah aktif atau tidak.
- *stts_reverse_geo*: Merupakan status yang dibuat oleh peneliti untuk menentukan apakah sudah dilakukan *reverse geocoding* untuk mendapatkan alamat lengkap.
- *pre_milleage*: Merupakan jarak antara titik saat ini ke titik sebelumnya dalam satuan kilometer.
- *sum_milleage*: merupakan jarak kumulatif sejak perangkat digunakan sampai saat ini.
- *vhc_milleage*: Merupakan jarak kumulatif sejak kendaraan digunakan sampai saat ini.
- *vhc_id*: Merupakan *id* dari *t_vehicles*, merupakan relasi *many to many*, karena banyak kendaraan bisa mempunyai banyak catatan pelacakan.
- *drv_id*: Merupakan *id* dari t_drivers, merupakan relasi *many to many*, karena banyak pengemudi bisa mempunyai banyak catatan pelacakan.

- *source*: Merupakan kolom yang ditambahkan oleh penulis untuk menentukan sumber data ini didapatkan dari perangkat *GPS Tracker*, bukan dari perangkat yang lain.
- *crt*: Merupakan kolom untuk menyimpan waktu yang diterima oleh *server* ketika menerima data dalam bentuk *unix timestamp*.
- *crt_format*: Merupakan kolom untuk menyimpan waktu yang diterima oleh *server* ketika menerima data dalam bentuk YYYY-MM-DD HH:MM:SS, contoh: 2024-06-07 10:41:30.
- *crt_d*: Merupakan kolom untuk menyimpan waktu yang dikirimkan dari *GPS Tracker* dalam bentuk *unix timestamp*.
- *crt_d_format*: Merupakan kolom untuk menyimpan waktu yang dikirimkan dari *GPS Tracker* dalam bentuk YYYY-MM-DD HH:MM:SS, contoh: 2024-06-07 10:41:30.
- *crt_s*: Merupakan kolom untuk menyimpan waktu yang diterima oleh *server* ketika menerima data dalam bentuk *unix timestamp*.
- *crt_s_format*: Merupakan kolom untuk menyimpan waktu yang diterima oleh *server* ketika menerima data dalam bentuk YYYY-MM-DD HH:MM:SS, contoh: 2024-06-07 10:41:30.

12. *t_gps_tracks_address*: Merupakan detail alamat dari setiap data yang masuk ke tabel *t_gps_tracks*.
    - *master_id*: Merupakan *id* dari *t_gps_tracks*, merupakan relasi *one to one*, karena hanya satu catatan pelacakan yang dapat memiliki satu alamat.
    - *state_id*: Merupakan *kodeProv* dari *t_gps_tracks_address*, merupakan relasi *one to many*, karena satu *kodeProv* dapat digunakan oleh banyak catatan alamat pelacakan. *kodeProv* merupa kode provinsi.
    - *city_id*: Merupakan *kodeKab* dari *t_gps_tracks_address*, merupakan relasi *one to many*, karena satu kodeKab dapat

digunakan oleh banyak catatan alamat pelacakan. *kodeKab* merupa kode kabupaten.

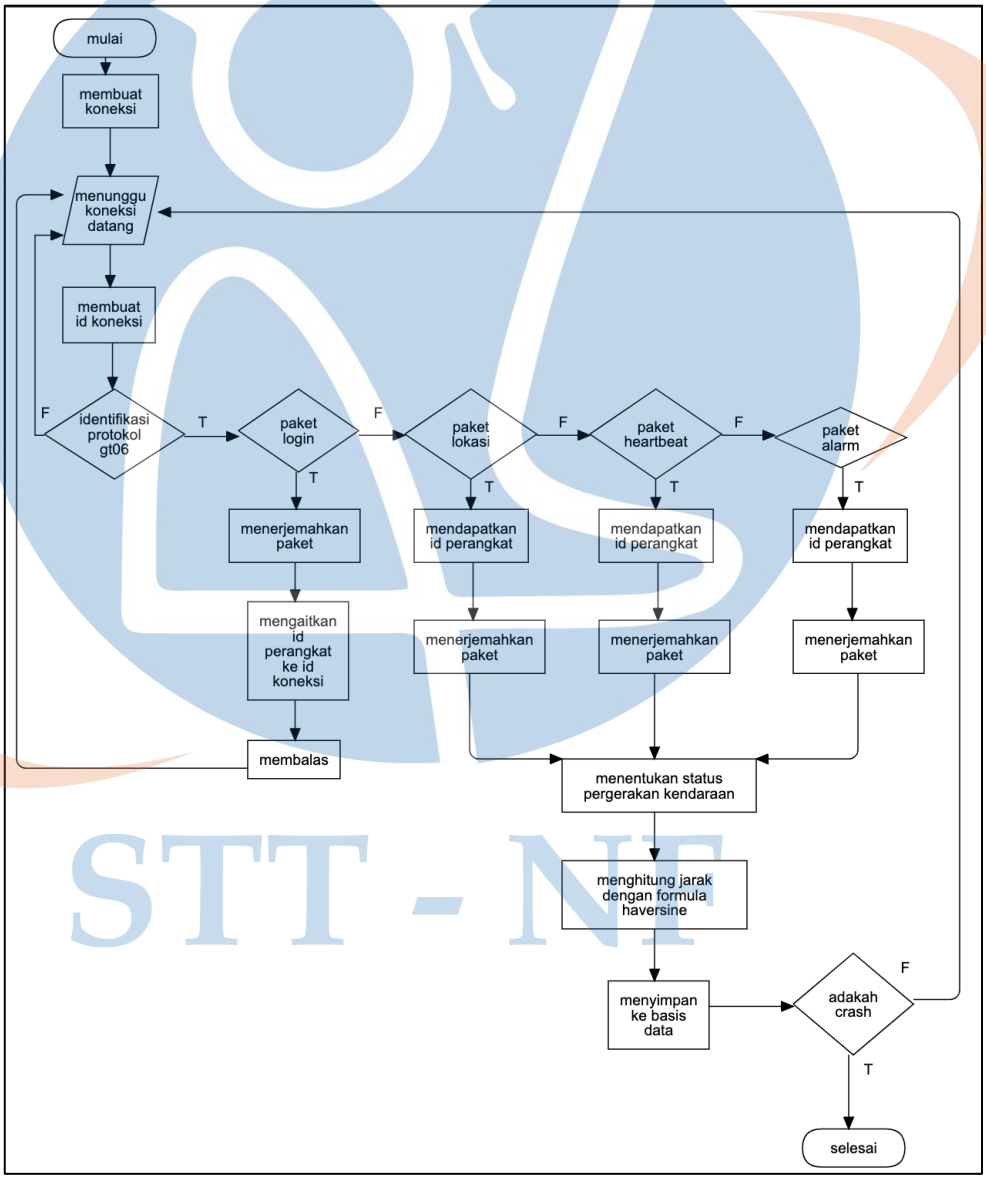
- *district_id*: Merupakan *kodeKec* dari *t_gps_tracks_address*, merupakan relasi *one to many*, karena satu *kodeKec* dapat digunakan oleh banyak catatan alamat pelacakan. *kodeKec* merupa kode kecamatan.
- *village_id*: Merupakan *kodeKel* dari *t_gps_tracks_address*, merupakan relasi *one to many*, karena satu *kodeKel* dapat digunakan oleh banyak catatan alamat pelacakan. *kodeKel* merupakan kode kelurahan.

13. *t_gps_tracks_rltm*: Merupakan tabel untuk *update* lokasi terkini.
    - *master_id*: Merupakan *id* dari *t_gps_tracks*, merupakan relasi *one to one*, karena hanya satu catatan pelacakan yang dapat memiliki satu catatan pelacakan.
    - *device_id*: Merupakan *International Mobile Equipment Identity* (*IMEI*) pada perangkat *GPS Tracker* dan memiliki relasi ke *t_devices.device_id*, jenis relasinya *one to many*, karena satu *device_id* dapat memiliki banyak catatan pelacakan.
    - *vhc_id*: Merupakan *id* dari *t_vehicles*, merupakan relasi *many to many*, karena banyak kendaraan bisa mempunyai banyak catatan pelacakan.
    - *drv_id*: Merupakan *id* dari t_drivers, merupakan relasi *many to many*, karena banyak pengemudi bisa mempunyai banyak catatan pelacakan.
14. *t_region*: Merupakan tabel untuk menyimpan data pembagian wilayah di Indonesia.

Tidak semua kolom pada tabel di ERD dijelaskan, hanya yang penting dan memiliki relasi saja yang dijelaskan, dikarenakan nama kolom pada tabel sudah mendeskripsikan fungsinya untuk apa.

4.2.6. Flowchart Program

Flowchart program menjelaskan cara kerja program yang dibuat menggunakan JavaScript dan dijalankan menggunakan NodeJS. Mulai dari program menerima data dari perangkat *GPS Tracker*, memprosesnya, dan hubungannya dengan basis data. Berikut gambar 4.23 yang menjelaskan *flowchart program* NodeJS.



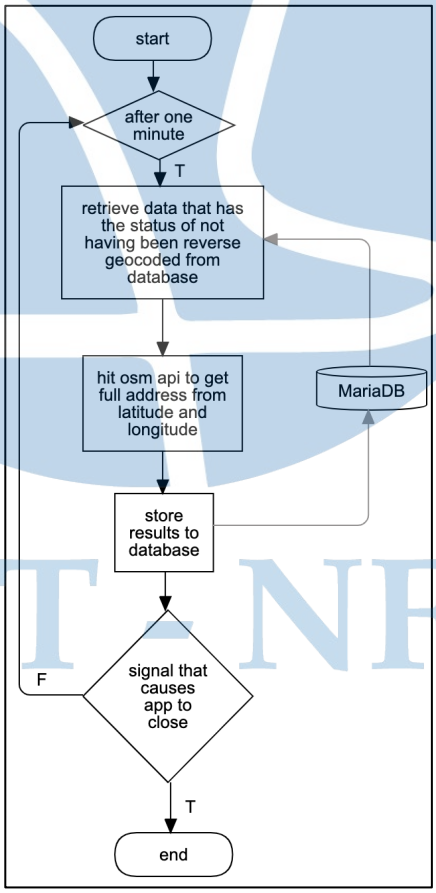
Gambar 4. 23 *Flowchart Program* NodeJS

Flowchart program NodeJS pada gambar 4.23 menggambarkan cara kerja *backend service* yang akan diimplementasikan. Berikut penjelasan lebih rinci dari gambar 4.23:

1. Ketika program dijalankan akan membuat sebuah koneksi untuk menerima data dari perangkat *GPS Tracker*.
2. Kemudian NodeJS akan menunggu sampai adanya koneksi yang masuk dan memprosesnya.
3. Setelah ada koneksi yang masuk NodeJS akan memberikan *id* pada koneksi tersebut supaya dikenali.
4. Kemudian NodeJS akan mengidentifikasi protokol dari paket data yang diterima.
5. Jika teridentifikasi sebagai protokol GT06 maka akan masuk ke bagian pengecekan jenis paket. Jika tidak maka tidak akan dilanjutkan dan kembali menunggu koneksi datang.
6. Jika teridentifikasi sebagai paket *login*, maka NodeJS akan membaca pesan paket tersebut sesuai dengan dokumentasi protokol GT06. Kemudian dilanjutkan dengan mengaitkan *id* perangkat terhadap *id* koneksi. Terakhir melakukan pesan balasan terhadap perangkat dan kembali pada tahapan menunggu koneksi datang.
7. Jika teridentifikasi sebagai paket lokasi, maka NodeJS akan mendapatkan *id* perangkat berdasarkan *id* koneksi dan membaca pesan paket tersebut sesuai dengan dokumentasi protokol GT06.
8. Jika teridentifikasi sebagai paket *heartbeat*, maka NodeJS akan mendapatkan *id* perangkat berdasarkan *id* koneksi dan membaca pesan paket tersebut sesuai dengan dokumentasi protokol GT06.
9. Jika teridentifikasi sebagai paket *alarm*, maka NodeJS akan mendapatkan *id* perangkat berdasarkan *id* koneksi dan membaca pesan paket tersebut sesuai dengan dokumentasi protokol GT06.
10. Selanjutnya baik itu paket lokasi, *heartbeat*, dan *alarm* akan masuk ke proses menentukan status pergerakan kendaraan.

11. Kemudian masuk ke proses untuk menghitung jarak lokasi antara titik saat ini dengan titik sebelumnya menggunakan formula *haversine*.
12. Setelah itu menyimpan data ke basis data.
13. Jika tidak ada sinyal yang membuat program selesai, maka proses akan Kembali ke menunggu koneksi datang.

Program pada NodeJS juga memiliki proses alternatif, yaitu proses *reverse geocoding*. Proses ini diperlukan untuk mendapatkan alamat lengkap dari sebuah data lokasi berdasarkan *latitude* dan *longitude* yang sudah didapatkan. Proses ini berjalan di belakang, sehingga tidak mengganggu proses utama untuk menerima paket dari *GPS Tracker*.



Gambar 4. 24 *Flowchart* NodeJS *Reverse Geocoding*

Pada gambar 4.24 merupakan cara kerja proses *reverse geocoding*. Proses ini berjalan setiap satu menit sekali. Setiap satu menit akan menarik semua data dari basis data yang memiliki status belum pernah dilakukan *reverse geocode*. Kemudian melakukan *hit* ke API OpenStreetMap untuk mendapatkan alamat lengkapnya berdasarkan *latitude* dan *longitude*.

4.3. Perencanaan Pengembangan

4.3.1. Sprint Backlog

Pada metode pengembangan *agile* yang mengimplementasikan *scrum*, semua pekerjaan didefinisikan menjadi *product backlog* yang kemudian diberikan nilai *story point* untuk menentukan seberapa banyak usaha yang perlu tim keluarkan untuk menyelesaikan pekerjaan tersebut. Setelah *story point* didapatkan pada setiap *task*, akan dilakukan estimasi pengerjaan dan pembagian tahapan *sprint*, dilanjutkan dengan mengakumulasikan *velocity* pada *task* yang berhasil dikerjakan. Berikut tabel 4.2 yang merupakan perencanaan *sprint* yang akan dilakukan.

Tabel 4. 2 Perencanaan *Sprint*

Task Code	***Sprint Phase***	***Date***	***Product Backlog***	***Task Name***	***PIC***	***Point***	***Velocity***
T.01	1	26 Februari 2024	R.01	Membuat *TCP Socket* di NodeJS.	P1	3	63
T.02				Konfigurasi perangkat *GPS Tracker*.	P1	3	
T.03		27 Februari 2024	R.02	Menerjemahkan paket *login* dalam bentuk *byte* dan mengirimkannya	P1	6	

				kembali dalam bentuk *byte* sesuai dengan format GT06.			
T.04		28 - 29 Februari 2024	R.03	Menerjemahkan paket data lokasi dalam bentuk *byte* sesuai dengan format GT06.	P1	7	
T.05				Mengubah data lokasi pada paket data lokasi ke dalam bentuk *decimal degree*.	P1	7	
T.06		01 & 04 Maret 2024		Menentukan status pergerakan kendaraan antara *move*, *idling*, dan *stop*.	P1	8	
T.07		05 Maret 2024		Menghitung jarak dari lokasi saat ini dengan lokasi sebelumnya menggunakan formula *haversine*.	P1	9	
T.08		06 Maret 2024		Menyimpan data lokasi ke basis data MariaDB.	P1	6	

T.10		07 – 08 Maret 2024	R.04	Menerjemahkan paket data *heartbeat* dalam bentuk *byte* sesuai dengan format GT06.	P1	7	
T.11				Mengubah data lokasi pada paket data *heartbeat* ke dalam bentuk *decimal degree*.	P1	7	
T.12	2	11 & 12 Maret 2024		Menentukan status pergerakan kendaraan antara *move*, *idling*, dan *stop*.	P1	8	64
T.13		13 Maret 2024		Menghitung jarak dari lokasi saat ini dengan lokasi sebelumnya menggunakan formula *haversine*.	P1	9	
T.14				Menyimpan data lokasi ke basis data MariaDB.	P1	6	
T.15		14 – 15 Maret 2024	R.05	Menerjemahkan paket data *alarm* dalam bentuk *byte*	P1	7	

				sesuai dengan format GT06.			
T.16				Mengubah data lokasi pada paket data *alarm* ke dalam bentuk *decimal degree*.	P1	7	
T.17		18 – 19 Maret 2024		Menentukan status pergerakan kendaraan antara *move*, *idling*, dan *stop*.	P1	8	
T.18		20 Maret 2024		Menghitung jarak dari lokasi saat ini dengan lokasi sebelumnya menggunakan formula *haversine*.	P1	9	
T.19				Menyimpan data lokasi ke basis data MariaDB.	P1	6	
T.20		21 – 22 Maret 2024	R.06	Menerapkan mekanisme *session* berdasarkan koneksi untuk menentukan koneksi ini berasal dari *id device* yang mana.	P1	7	

T.21	3	25 – 26 Maret 2024	R.07	Membuat *background process* untuk melakukan *reverse geocoding* setiap satu menit.	P1	9	68
T.22		27 Maret 2024	R.08	Membuat *login* di Laravel.	P1 & P2	6	
T.23		28 -29 Maret 2024	R.09	Membuat fitur *multi-role user*.	P1 & P2	7	
T.24		01 – 02 April 2024	R.10	Membuat tampilan peta pada halaman *dashboard*.	P1 & P2	8	
T.25				Menerapkan metode *client-polling* untuk mendapatkan data secara *real-time*.	P1 & P2	8	
T.26		03 April 2024	R.11	Membuat tampilan *badge stop*, *move*, dan *idling* dan mendapatkan data dari basis data.	P1 & P2	9	
T.27		04 April 2024	R.12	Membuat tampilan *tooltip* untuk menampilkan informasi jarak	P1 & P2	7	

				tempuh kendaraan dan mendapatkan data dari basis data.			
T.28		05 April 2024	R.13	Membuat tampilan *right-sidebar* untuk menampilkan informasi kendaraan dan mendapatkan data dari basis data.	P1 & P2	7	
T.29			R.14	Membuat tampilan *right-sidebar* untuk menampilkan informasi pengemudi dan mendapatkan data dari basis data.	P1 & P2	7	
T.30	4	08 –09 April 2024	R.15	Membuat fitur riwayat perjalanan dari waktu saat ini sampai sebelumnya dengan maksimal data 500.	P1 & P2	9	56
T.31				Membuat filter tanggal untuk	P1 & P2	9	

				riwayat perjalanan dengan maksimal data 5000.			
T.32		10 - 11 April 2024	R.16	Membuat *CRUD* pada data kendaraan.	P1 & P2	6	
T.33				Terdapat fitur untuk mengaitkan *GPS Tracker* dengan kendaraan pada halaman kelola kendaraan.	P1 & P2	7	
T.34		12 April 2024	R.17	Membuat *CRUD* pada data pengemudi.	P1 & P2	6	
T.35		15 April 2024	R.18	Membuat *CRUD* pada data *users*.	P1 & P2	6	
T.36		16 – 17 April 2024	R.19	Mengatur akses kendaraan yang dilihat pada *dashboard* untuk *role* pelacakan.	P1 & P2	7	
T.37		18 April 2024	R.20	Membuat *CRUD* pada data *GPS Tracker*.	P1 & P2	6	
T.38	5	19 & 22 April 2024		Terdapat fitur untuk mengaitkan *GPS Tracker*	P1 & P2	8	14

				dengan kendaraan yang tersedia.			
T.39		23 April 2024	R.21	Membuat daftar tabel yang menunjukkan data *GPS Tracker* saat ini.	P1 & P2	6	

Berikut penjelasan kolom-kolom terkait tabel 4.2 perencanaan *sprint* yang dilakukan.

1. *Task code*: merupakan kode pekerjaan untuk memudahkan identifikasi pekerjaan tersebut berdasarkan kode.
2. *Sprint*: Merupakan tahapan *sprint*.
3. *Date*: durasi pengerjaan *task*.
4. *Product backlog*: *Product backlog* dapat berupa definisi fitur, *software requirements*, *user stories*, atau deskripsi dari *supplementary task* seperti definisi arsitektur dan dokumentasi *user* [32].
5. *Task name*: nama pekerjaan, rincian dari *product backlog*.
6. *PIC*: Merupakan kepanjangan dari *Person in Charge* (*PIC*). P1 memiliki keterangan *programmer* 1, dan P2 memiliki keterangan *programmer* 2. Peneliti berperan sebagai *programmer* 1.
7. *Point*: pecahan *velocity* dari *product backlog*.
8. *Velocity*: Nilai dari usaha yang dikeluarkan oleh tim untuk menyelesaikan *product backlog* tersebut.

4.4. Implementasi Rancangan Sistem

4.4.1. Konfigurasi perangkat *GPS Tracker*

Fokus dari penelitian ini adalah merancang dan membangun aplikasi web serta *backend service* untuk menerima data dari perangkat *GPS Tracker*. Namun tetap perlu dijelaskan cara konfigurasi perangkat *GPS*

Tracker supaya dapat mengirimkan data ke *backend service*. Ini juga termasuk mengimplementasi R.01.

1. Pastikan kondisi kartu *M2M* dari AUTOTRONIC sudah aktif.
2. Pasang kartu *SIM* ke perangkat *GPS Tracker*.
3. Gunakan *handphone* atau *smartphone* yang dapat mengirim *SMS* dan pastikan memiliki pulsa (menggunakan kartu *SIM* biasa, bukan *M2M*).
4. Kirimkan *SMS* "APN,M2MAUTOTRONIC#" ke nomor *M2M*.

   Perintah ini untuk mengatur *Access Point Name* (*APN*) ke M2MAUTOTRONIC. Hilangkan tanda kutip ketika mengirim pesan. Pastikan pesan mendapatkan balasan *OK*.
5. Kirimkan *SMS* "RESET#" ke nomor *M2M*.

   Perintah ini akan membuat perangkat melakukan *reboot* setelah 20 detik menerima pesan. Hilangkan tanda kutip ketika mengirim pesan. Pastikan pesan mendapatkan balasan *OK*.
6. Kirimkan *SMS* "SERVER,0,103.74.5.95,6014#" ke nomor *M2M*.

   Perintah ini merupakan alamat *Internet Protocol* (*IP*) *backend service* beserta nomor *port*. Hilangkan tanda kutip ketika mengirim pesan. Pastikan pesan mendapatkan balasan *OK*.
7. Kirimkan *SMS* "GPRSON,1#" ke nomor *M2M*.

   Perintah ini untuk konfigurasi GPRSON menjadi hidup. Hilangkan tanda kutip ketika mengirim pesan. Pastikan pesan mendapatkan balasan *OK*.
8. Kirimkan *SMS* "TIMER,10,10#" ke nomor *M2M*.

   Perintah ini untuk konfigurasi interval mengirim data ke *backend service*. 10 pada bagian pertama memiliki arti mengirimkan data setiap 10 detik ketika kondisi aki hidup. 10 pada bagian kedua memiliki arti mengirimkan data setiap 10 detik ketika kondisi aki mati. Hilangkan tanda kutip ketika mengirim pesan. Pastikan pesan mendapatkan balasan *OK*.

9. Kirimkan *SMS* “GMT,E,7#” ke nomor *M2M*. Hilangkan tanda kutip ketika mengirim pesan.

   Perintah ini untuk mengatur zona waktu +7. Hilangkan tanda kutip ketika mengirim pesan. Pastikan pesan mendapatkan balasan *OK*.

10. Kirimkan *SMS* “PARAM#” ke nomor *M2M*. Hilangkan tanda kutip ketika mengirim pesan.

    Perintah ini mengecek konfigurasi yang telah dilakukan pada perangkat. Hilangkan tanda kutip ketika mengirim pesan. Pastikan pesan mendapatkan balasan *OK*.

4.4.2. Kode Program NodeJS

Implementasi program yang peneliti terapkan pada NodeJS tidak menerapkan *framework*. Untuk pembagian kode programnya, secara sederhana peneliti membagi 3, yaitu modul *net* untuk menerima koneksi dan data, kelas *LibDevice* untuk menerjemahkan data dari sebuah *buffer*, dan fungsi *commitMessage* yang terdapat logika bisnis dan menyimpannya ke basis data.

Dalam proses menerjemahkan data protokol GT06, peneliti mengacu pada dokumen yang terlampir pada lampiran 2. Berikut penjelasan kode program NodeJS yang mengimplementasikan R.01 sampai R.07.

1. Membuat *TCP socket* di NodeJS dan menerapkan mekanisme sesi

```
require('dotenv').config({ path: '.env' });
require('events').EventEmitter.prototype._maxListeners = 30;
// start for gps-tracking
const net = require('net');
const LibDevice = require('./library/LibDevice');
const nanoid = require('nanoid').nanoid;
// end for gps-tracking

net.createServer({
  allowHalfOpen: true,
}, function (c) { // c mean connection
  // save connections
  c.id = nanoid();
  c.pipe(c);

  c.on('data', async (buffer_req) => {
    const me = LibDevice.identifyProtocolFromBuffer(buffer_req);
  });
  c.on('end', () => {
  });
  c.on('close', () => {
  });

  // unused
  c.on('drain', (a) => {
    console.log('client drain', a);
  });
  c.on('error', (a) => {
    console.error('client error', a);
  });
  c.on('lookup', (a) => {
    console.log('client lookup', a);
  });
  c.on('ready', (a) => {
    console.log('client ready', a);
  });
  c.on('timeout', (a) => {
    console.log('client timeout', a);
  });

  c.pipe(c);
}).listen(process.env.PORT, () => {
  console.log('Server gps tracker running at port ' + process.env.PORT);
});
// end for gps-tracking TCP ONLY
```

Gambar 4. 25 Implementasi *TCP Socket* dan Sesi di NodeJS

Untuk membuat *TCP socket* di NodeJS membutuhkan modul yang sudah tersedia sejak NodeJS dipasang pertama kali, bernama *net*. Modul *net* tidak memiliki mekanisme sesi, karena ia ditujukan untuk layer 4 pada *OSI layer*, yaitu *transport layer*. Maka dibutuhkan

mekanisme sesi yang berada di *layer* 5. Caranya dengan memberikan nilai unik pada setiap koneksi yang terhubung. Peneliti menggunakan *nanoid* untuk membangkitkan *id* yang acak dan unik.

Modul *net* menerapkan paradigma *non-blocking*, sehingga dapat memproses data tanpa harus mengantre. Ini dapat terlihat pada baris kode 16 sampai 39 pada gambar 4.25.

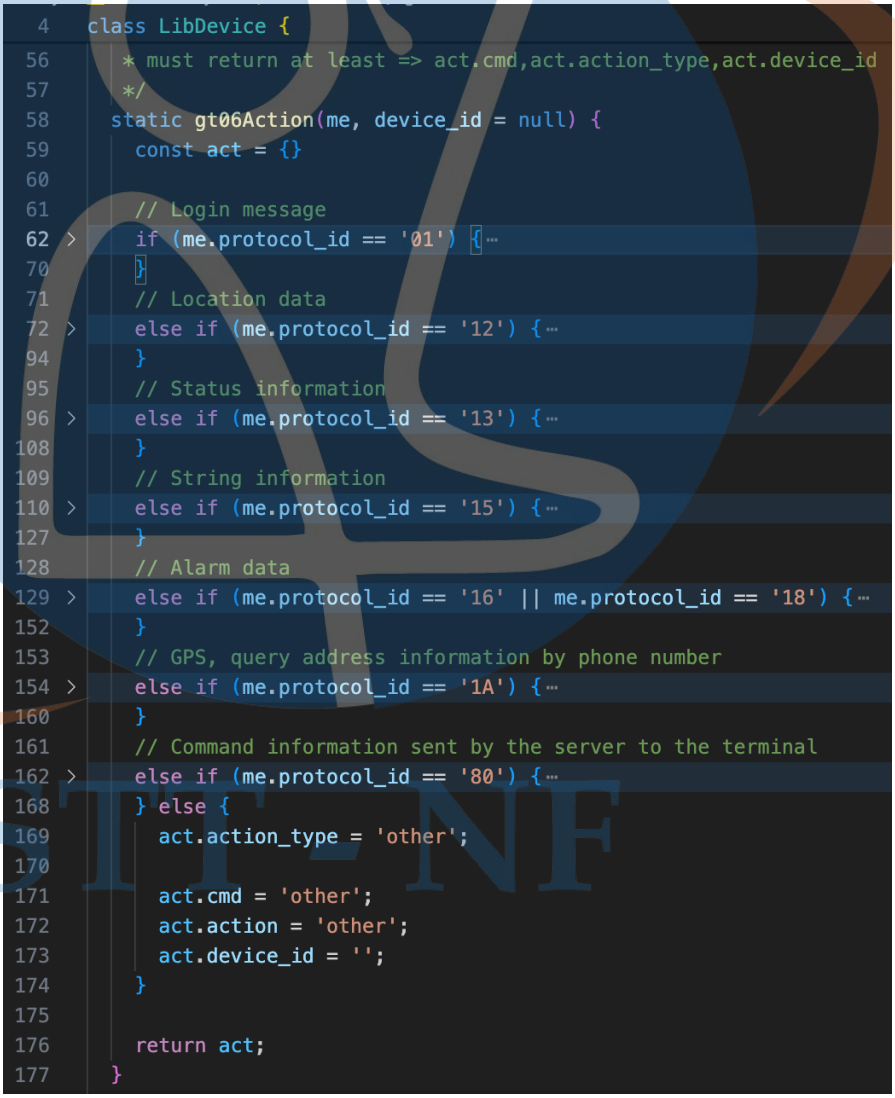
2. Mengidentifikasi protokol GT06

```
4   class LibDevice {
13    static identifyProtocolFromBuffer(buffer, opts = {}) {
14      let ori_hex_str = '';
15      if (typeof opts.skip_buffer != 'undefined' && opts.skip_buffer === true) {
16        ori_hex_str = buffer;
17      } else {
18        ori_hex_str = buffer.toString('hex'); // 16
19      }
20
21      const data = ori_hex_str;
22      const me = {
23        ori_string: data,
24      };
25
26      // gt06
27      if (data.slice(0, 4) == '7878') {
28        me.protocol_name = 'gt06';
29
30        me.start = data.slice(0, 4);
31        me.length = parseInt(data.slice(4, 6), 16);
32        me.protocol_id = data.slice(6, 8);
33        me.serial_number = data.slice(-12, -8)
34        me.error_check = data.slice(-8, -4)
35        me.finish = data.slice(-4); // data.substr(6 + me.length * 2, 4);
36
37        if (me.finish != '0d0a') {
38          throw 'finish code incorrect!';
39        }
40
41        me.ori_string = data;
42      }
43      // unknown
44      else {
45        me.protocol_name = 'unknown';
46      }
47
48      return me;
49    }
```

Gambar 4. 26 Identifikasi Protokol GT06 di NodeJS

Pada gambar 4.26 merupakan kelas bernama *LibDevice* dan metode untuk mengidentifikasi protokol dari sebuah *buffer*. Untuk membaca sebuah *buffer* harus dikonversi terlebih dahulu menjadi bentuk heksadesimal. Karakter pertama sampai keempat pada heksadesimal merupakan karakter untuk mengidentifikasi jenis protokol.

3. Mengidentifikasi paket data secara keseluruhan



Gambar 4. 27 Identifikasi Paket GT06 di NodeJS

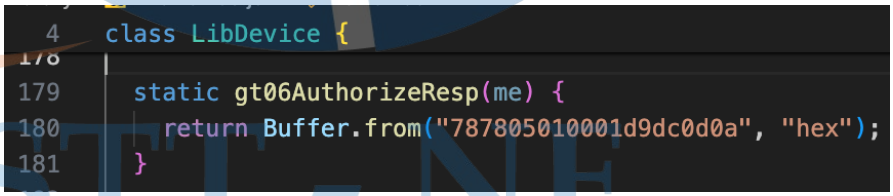
Pada gambar 4.27 merupakan kelas bernama *LibDevice* dan metode untuk mengidentifikasi paket-paket data yang tersedia pada GT06. Pada metode ini paket data sudah dalam bentuk heksadesimal. Identifikasinya hanya perlu disesuaikan dengan format pada dokumen yang tertera pada lampiran 2 pada bagian 4.3 *Protocol Number*.

4. Mengidentifikasi dan menerjemahkan paket *login*

Gambar 4. 28 Identifikasi dan Menerjemahkan Paket *login*

Pada gambar 4.28 merupakan implementasi menerjemahkan paket *login*. Paket *login* hanya terdapat *id* perangkat, yang nantinya akan dikaitkan dengan *id* koneksi. Implementasi ini masih berada di kelas *LibDevice*.

5. Balasan paket *login*



Gambar 4. 29 Balasan Paket *Login* di NodeJS

Pada gambar 4.29 merupakan balasan dari server setelah menerima paket *login*. Jika tidak memberikan balasan, perangkat *GPS Tracker* tidak akan mengirimkan data lokasi.

6. Mengaitkan *id* perangkat berdasarkan sesi

```
159  }, function (c) { // c mean connection
166    c.on('data', async (buffer_req) => {
207      if (me.protocol_name == 'gt06') {
208        // const act = LibDevice.gt06Action(me, device.device_id || null);
209        let dvc_id = devices[c.id];
210        const act = LibDevice.gt06Action(me, dvc_id || null);
211
212        if (act.action_type == 'login') {
213          logDevice.action = act.action_type;
214          logDevice.device_id = act.device_id;
215
216          devices[c.id] = act.device_id;
217          netConn[act.device_id] = c;
218          if (typeof act.buffer_resp != 'undefined') {
219            c.write(act.buffer_resp);
220          }
221        }
```

Gambar 4. 30 Mengaitkan *id* Perangkat dengan *id* Koneksi di NodeJS

Pada gambar 4.30 merupakan implementasi untuk mengaitkan *id* perangkat yang telah didapatkan dari proses menerjemahkan paket *login*. *Id* perangkat harus dikaitkan dengan koneksi, karena setelahnya tidak akan mengirimkan lagi *id* perangkat. Implementasi ini terdapat pada modul *net*.

7. Mengidentifikasi paket lokasi

```
4   class LibDevice {
58    static gt06Action(me, device_id = null) {
71      // Location data
72      else if (me.protocol_id == '12') {
73        act.action_type = 'location';
74
75        act.cmd = 'ping';
76        act.action = 'ping';
77        act.device_id = device_id; // because device_id only sent when login
78
79        // content data
80        // me.ori_string.slice(8).slice(0, 52); // me.ori_string.substr(8, me.length * 2);
81
82        // gps information
83        act.gps_string = me.ori_string.slice(8).slice(0, 36);
84        act.gps_data = LibDevice.gt06ParseLocation(me, act.gps_string);
85        // if (!act.gps_data) {
86        //   //Something bad happened
87        //   _this.do_log('GPS Data can\'t be parsed. Discarding packet...');
88        //   return false;
89        // }
```

Gambar 4. 31 Identifikasi Paket Lokasi di NodeJS

Pada gambar 4.31 merupakan implementasi untuk mengidentifikasi paket lokasi sesuai dengan format dokumen protokol GT06. Implementasi ini terdapat pada kelas *LibDevice*.

8. Menerjemahkan paket lokasi

```
class LibDevice {
  static gt06ParseLocation(me, gps_string = null) {
    let year = (parseInt(gps_string.slice(0, 2), 16) + '').padStart(2, 0);
    year = '20' + year;
    let month = (parseInt(gps_string.slice(2, 4), 16) + '').padStart(2, 0);
    let day = (parseInt(gps_string.slice(4, 6), 16) + '').padStart(2, 0);
    let hour = (parseInt(gps_string.slice(6, 8), 16) + '').padStart(2, 0);
    let minute = (parseInt(gps_string.slice(8, 10), 16) + '').padStart(2, 0);
    let second = (parseInt(gps_string.slice(10, 12), 16) + '').padStart(2, 0);

    let ob1 = LibHelper.hex2bin(gps_string.slice(32, 34)); // orientation_byte1
    let ob1_bit7 = ob1.slice(0, 1);
    let ob1_bit6 = ob1.slice(1, 2);
    let ob1_bit5 = ob1.slice(2, 3); // 0 (realtime  GPS) or differential positioning
    let ob1_bit4 = ob1.slice(3, 4); // 1 (GPS has been positioned)
    let ob1_bit3 = ob1.slice(4, 5); // 0 (east longitude) || 1 (west longitude)
    let ob1_bit2 = ob1.slice(5, 6); // 1 (north latitude) || 0 (south latitude)
    let ob1_bit1 = ob1.slice(6, 7);
    let ob1_bit0 = ob1.slice(7, 8);
    let ob2 = LibHelper.hex2bin(gps_string.slice(34, 36)); // orientation_byte2
    // let ob2_bit7 = ob2.slice(0, 1);
    // let ob2_bit6 = ob2.slice(1, 2);
    // let ob2_bit5 = ob2.slice(2, 3);
    // let ob2_bit4 = ob2.slice(3, 4);
    // let ob2_bit3 = ob2.slice(4, 5);
    // let ob2_bit2 = ob2.slice(5, 6);
    // let ob2_bit1 = ob2.slice(6, 7);
    // let ob2_bit0 = ob2.slice(7, 8);

    let lat_wind = ''; // wind direction N,S
    let lng_wind = ''; // wind direction W,E
    if (ob1_bit3 == 1) {
      lng_wind = 'W';
    }
    if (ob1_bit3 == 0) {
      lng_wind = 'E';
    }
    if (ob1_bit2 == 1) {
      lat_wind = 'N';
    }
    if (ob1_bit2 == 0) {
      lat_wind = 'S';
    }

    const data = {
      'date_raw': gps_string.slice(0, 12),
      'date': `${year}-${month}-${day} ${hour}:${minute}:${second}`,
      'quantity_pos_satellites_raw': gps_string.slice(12, 14),
      'quantity_pos_satellites_c': parseInt(gps_string.slice(12, 13), 16), // length of gps information
      'quantity_pos_satellites_b': parseInt(gps_string.slice(13, 14), 16), // number of positioning satellites
      realtime_dif_gps: ob1_bit5, // 0 (realtime  GPS) or differential positioning
      positioning_gps: ob1_bit6, // 1 (GPS has been positioned)
      'latitude_raw': gps_string.slice(14, 22),
      'longitude_raw': gps_string.slice(22, 30),
      lat_wind_direction: lat_wind,
      lng_wind_direction: lng_wind,
      'latitude': LibDevice.gt06Hex2DD(gps_string.slice(14, 22), lat_wind),
      'longitude': LibDevice.gt06Hex2DD(gps_string.slice(22, 30), lng_wind),
      'speed': parseInt(gps_string.slice(30, 32), 16), // km/h
      'orientation_raw': gps_string.slice(32, 36),
      'orientation': parseInt(`${ob1_bit1}${ob1_bit0}${ob2}`, 2), // -360 ~ 360 derajat
    };

    return data;
  }
```

Gambar 4. 32 Menerjemahkan Paket Lokasi di NodeJS

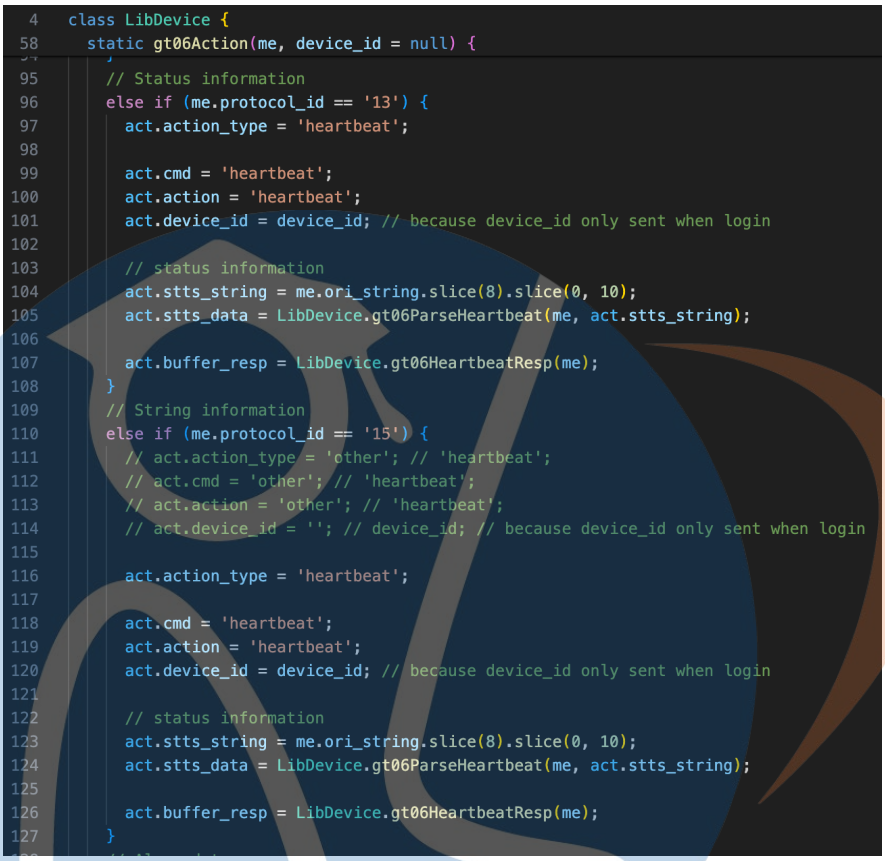
Pada gambar 4.32 merupakan implementasi untuk menerjemahkan paket lokasi sesuai dengan format dokumen protokol GT06. Implementasi ini terdapat pada kelas *LibDevice*.

9. Mengubah lokasi menjadi bentuk *decimal degree*.

Gambar 4. 33 Mengubah Lokasi Menjadi Bentuk *Decimal Degree*

Pada gambar 4.33 merupakan implementasi untuk mendapatkan data lokasi dalam bentuk heksadesimal menjadi bentuk *decimal degree*. Rumus konversinya dapat dilihat pada format dokumen protokol GT06 bagian 5.2.1.6 *Latitude*. Implementasi ini terdapat pada kelas *LibDevice*.

10. Mengidentifikasi paket *heartbeat*

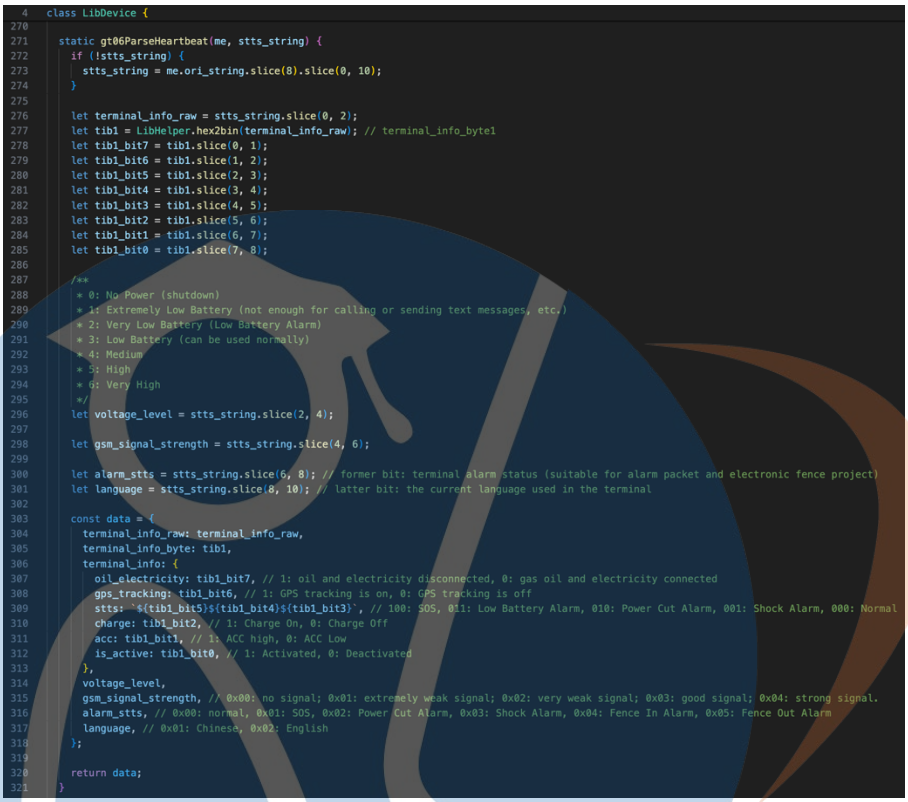


```
4   class LibDevice {
58    static gt06Action(me, device_id = null) {
95      // Status information
96      else if (me.protocol_id == '13') {
97        act.action_type = 'heartbeat';
98
99        act.cmd = 'heartbeat';
100       act.action = 'heartbeat';
101       act.device_id = device_id; // because device_id only sent when login
102
103       // status information
104       act.stts_string = me.ori_string.slice(8).slice(0, 10);
105       act.stts_data = LibDevice.gt06ParseHeartbeat(me, act.stts_string);
106
107       act.buffer_resp = LibDevice.gt06HeartbeatResp(me);
108     }
109     // String information
110     else if (me.protocol_id == '15') {
111       // act.action_type = 'other'; // 'heartbeat';
112       // act.cmd = 'other'; // 'heartbeat';
113       // act.action = 'other'; // 'heartbeat';
114       // act.device_id = ''; // device_id; // because device_id only sent when login
115
116       act.action_type = 'heartbeat';
117
118       act.cmd = 'heartbeat';
119       act.action = 'heartbeat';
120       act.device_id = device_id; // because device_id only sent when login
121
122       // status information
123       act.stts_string = me.ori_string.slice(8).slice(0, 10);
124       act.stts_data = LibDevice.gt06ParseHeartbeat(me, act.stts_string);
125
126       act.buffer_resp = LibDevice.gt06HeartbeatResp(me);
127     }
```

Gambar 4. 34 Identifikasi Paket *Heartbeat* di NodeJS

Pada gambar 4.34 merupakan implementasi untuk mengidentifikasi paket *heartbeat* sesuai dengan format dokumen protokol GT06. Implementasi ini terdapat pada kelas *LibDevice*.

11. Menerjemahkan paket *heartbeat*



```
class LibDevice {

  static gt06ParseHeartbeat(me, stts_string) {
    if (!stts_string) {
      stts_string = me.ori_string.slice(8).slice(0, 10);
    }

    let terminal_info_raw = stts_string.slice(0, 2);
    let tib1 = LibHelper.hex2bin(terminal_info_raw); // terminal_info_byte1
    let tib1_bit7 = tib1.slice(0, 1);
    let tib1_bit6 = tib1.slice(1, 2);
    let tib1_bit5 = tib1.slice(2, 3);
    let tib1_bit4 = tib1.slice(3, 4);
    let tib1_bit3 = tib1.slice(4, 5);
    let tib1_bit2 = tib1.slice(5, 6);
    let tib1_bit1 = tib1.slice(6, 7);
    let tib1_bit0 = tib1.slice(7, 8);

    /**
     * 0: No Power (shutdown)
     * 1: Extremely Low Battery (not enough for calling or sending text messages, etc.)
     * 2: Very Low Battery (Low Battery Alarm)
     * 3: Low Battery (can be used normally)
     * 4: Medium
     * 5: High
     * 6: Very High
     */
    let voltage_level = stts_string.slice(2, 4);

    let gsm_signal_strength = stts_string.slice(4, 6);

    let alarm_stts = stts_string.slice(6, 8); // former bit: terminal alarm status (suitable for alarm packet and electronic fence project)
    let language = stts_string.slice(8, 10); // latter bit: the current language used in the terminal

    const data = {
      terminal_info_raw: terminal_info_raw,
      terminal_info_byte: tib1,
      terminal_info: {
        oil_electricity: tib1_bit7, // 1: oil and electricity disconnected, 0: gas oil and electricity connected
        gps_tracking: tib1_bit6, // 1: GPS tracking is on, 0: GPS tracking is off
        stts: `${tib1_bit5}${tib1_bit4}${tib1_bit3}`, // 100: SOS, 011: Low Battery Alarm, 010: Power Cut Alarm, 001: Shock Alarm, 000: Normal
        charge: tib1_bit2, // 1: Charge On, 0: Charge Off
        acc: tib1_bit1, // 1: ACC high, 0: ACC Low
        is_active: tib1_bit0, // 1: Activated, 0: Deactivated
      },
      voltage_level,
      gsm_signal_strength, // 0x00: no signal; 0x01: extremely weak signal; 0x02: very weak signal; 0x03: good signal; 0x04: strong signal.
      alarm_stts, // 0x00: normal, 0x01: SOS, 0x02: Power Cut Alarm, 0x03: Shock Alarm, 0x04: Fence In Alarm, 0x05: Fence Out Alarm
      language, // 0x01: Chinese, 0x02: English
    };

    return data;
  }
```

Gambar 4. 35 Menerjemahkan Paket *Heartbeat* di NodeJS

Pada gambar 4.35 merupakan implementasi untuk menerjemahkan paket *heartbeat* sesuai dengan format dokumen protokol GT06. Implementasi ini terdapat pada kelas *LibDevice*.

12. Balasan paket *heartbeat*

```
class LibDevice {

  static gt06HeartbeatResp(me) {
    return Buffer.from("787805130001d9dc0d0a", "hex");
  }
```

Gambar 4. 36 Balasan Paket *Heartbeat* di NodeJS

Pada gambar 4.36 merupakan implementasi untuk balasan paket *heartbeat* sesuai dengan format dokumen protokol GT06. Implementasi ini terdapat pada kelas *LibDevice*.

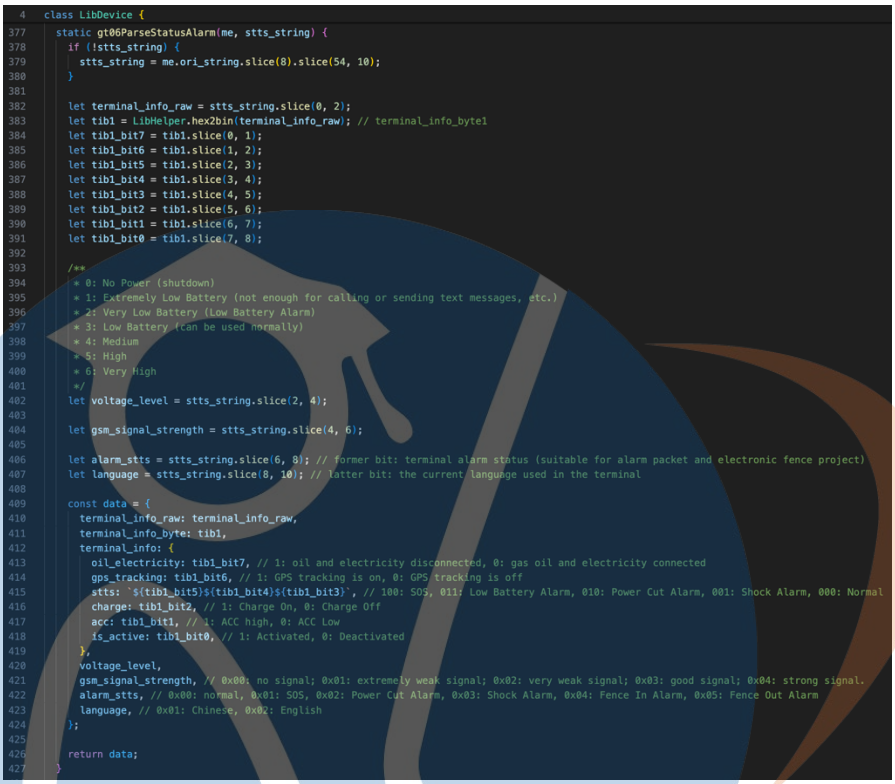
13. Mengidentifikasi paket *alarm*

```
4   class LibDevice {
58    static gt06Action(me, device_id = null) {
128     // Alarm data
129     else if (me.protocol_id == '16' || me.protocol_id == '18') {
130       act.action_type = 'alarm';
131
132       act.cmd = 'alarm';
133       act.action = 'alarm';
134       act.device_id = device_id; // because device_id only sent when login
135
136       // content data
137       // me.ori_string.slice(8).slice(0, 64); // me.ori_string.substr(8, me.length * 2);
138
139       // gps information
140       act.gps_string = me.ori_string.slice(8).slice(0, 36);
141       act.gps_data = LibDevice.gt06ParseLocation(me, act.gps_string);
142
143       // lbs information
144       act.lbs_string = me.ori_string.slice(8).slice(36, 54);
145       act.lbs_data = LibDevice.gt06ParseLbs(me, act.lbs_string);
146
147       // status information
148       act.stts_string = me.ori_string.slice(8).slice(54, 64);
149       act.stts_data = LibDevice.gt06ParseStatusAlarm(me, act.stts_string);
150
151       act.buffer_resp = LibDevice.gt06AlarmResp(me);
152     }
```

Gambar 4. 37 Identifikasi Paket *Alarm* di NodeJS

Pada gambar 4.37 merupakan implementasi untuk mengidentifikasi paket *alarm* sesuai dengan format dokumen protokol GT06. Implementasi ini terdapat pada kelas *LibDevice*.

14. Menerjemahkan paket *alarm*



```
class LibDevice {
  static gt06ParseStatusAlarm(me, stts_string) {
    if (!stts_string) {
      stts_string = me.ori_string.slice(8).slice(54, 10);
    }

    let terminal_info_raw = stts_string.slice(0, 2);
    let tib1 = LibHelper.hex2bin(terminal_info_raw); // terminal_info_byte1
    let tib1_bit7 = tib1.slice(0, 1);
    let tib1_bit6 = tib1.slice(1, 2);
    let tib1_bit5 = tib1.slice(2, 3);
    let tib1_bit4 = tib1.slice(3, 4);
    let tib1_bit3 = tib1.slice(4, 5);
    let tib1_bit2 = tib1.slice(5, 6);
    let tib1_bit1 = tib1.slice(6, 7);
    let tib1_bit0 = tib1.slice(7, 8);

    /**
     * 0: No Power (shutdown)
     * 1: Extremely Low Battery (not enough for calling or sending text messages, etc.)
     * 2: Very Low Battery (Low Battery Alarm)
     * 3: Low Battery (can be used normally)
     * 4: Medium
     * 5: High
     * 6: Very High
     */
    let voltage_level = stts_string.slice(2, 4);

    let gsm_signal_strength = stts_string.slice(4, 6);

    let alarm_stts = stts_string.slice(6, 8); // former bit: terminal alarm status (suitable for alarm packet and electronic fence project)
    let language = stts_string.slice(8, 10); // latter bit: the current language used in the terminal

    const data = {
      terminal_info_raw: terminal_info_raw,
      terminal_info_byte: tib1,
      terminal_info: {
        oil_electricity: tib1_bit7, // 1: oil and electricity disconnected, 0: gas oil and electricity connected
        gps_tracking: tib1_bit6, // 1: GPS tracking is on, 0: GPS tracking is off
        stts: `${tib1_bit5}${tib1_bit4}${tib1_bit3}`, // 100: SOS, 011: Low Battery Alarm, 010: Power Cut Alarm, 001: Shock Alarm, 000: Normal
        charge: tib1_bit2, // 1: Charge On, 0: Charge Off
        acc: tib1_bit1, // 1: ACC high, 0: ACC Low
        is_active: tib1_bit0, // 1: Activated, 0: Deactivated
      },
      voltage_level,
      gsm_signal_strength, // 0x00: no signal; 0x01: extremely weak signal; 0x02: very weak signal; 0x03: good signal; 0x04: strong signal.
      alarm_stts, // 0x00: normal, 0x01: SOS, 0x02: Power Cut Alarm, 0x03: Shock Alarm, 0x04: Fence In Alarm, 0x05: Fence Out Alarm
      language, // 0x01: Chinese, 0x02: English
    };

    return data;
  }
```

Gambar 4. 38 Menerjemahkan Paket *Alarm* di NodeJS

Pada gambar 4.38 merupakan implementasi untuk menerjemahkan paket *alarm* sesuai dengan format dokumen protokol GT06. Implementasi ini terdapat pada kelas *LibDevice*.

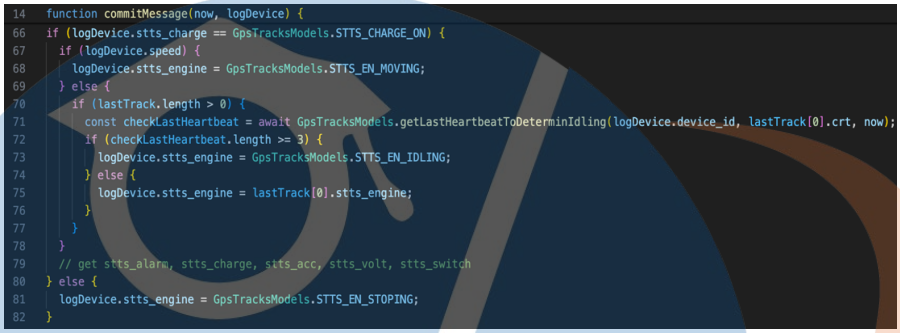
15. Balasan paket *alarm*

```
library > JS LibDevice.js > LibDevice > gt06AlarmResp
  4   class LibDevice {
428
429     static gt06AlarmResp(me) {
430       return Buffer.from("787805160001d9dc0d0a", "hex");
431     }
432
433   }
```

Gambar 4. 39 Balasan Paket *Alarm* di NodeJS

Pada gambar 4.39 merupakan implementasi untuk balasan paket *alarm* sesuai dengan format dokumen protokol GT06. Implementasi ini terdapat pada kelas *LibDevice*.

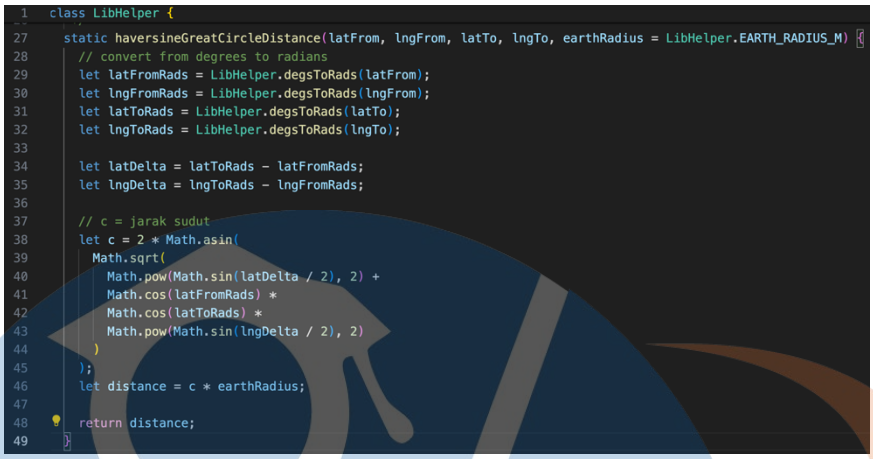
16. Menentukan status pergerakan kendaraan pada setiap paket



Gambar 4. 40 Menentukan Status Pergerakan Kendaraan Pada Setiap Paket

Pada gambar 4.40 merupakan implementasi untuk menentukan status kendaraan. Status kendaraan akan mati jika mesin dimatikan. Jika data memiliki kecepatan tidak nol, maka status kendaraan adalah bergerak. Jika kondisi mesin menyala, kecepatan nol, namun sebelumnya terdapat paket *heartbeat* sebanyak lebih sama dengan dari 3 kali, maka statusnya adalah *idle*. Jika tidak memenuhi persyaratan tersebut, maka status kendaraan tidak ada yang berubah. Implementasi ini terdapat pada fungsi *commitMessage*.

17. Menghitung jarak tempuh kendaraan menggunakan formula *haversine*



```
class LibHelper {
  static haversineGreatCircleDistance(latFrom, lngFrom, latTo, lngTo, earthRadius = LibHelper.EARTH_RADIUS_M) {
    // convert from degrees to radians
    let latFromRads = LibHelper.degsToRads(latFrom);
    let lngFromRads = LibHelper.degsToRads(lngFrom);
    let latToRads = LibHelper.degsToRads(latTo);
    let lngToRads = LibHelper.degsToRads(lngTo);

    let latDelta = latToRads - latFromRads;
    let lngDelta = lngToRads - lngFromRads;

    // c = jarak sudut
    let c = 2 * Math.asin(
      Math.sqrt(
        Math.pow(Math.sin(latDelta / 2), 2) +
        Math.cos(latFromRads) *
        Math.cos(latToRads) *
        Math.pow(Math.sin(lngDelta / 2), 2)
      )
    );
    let distance = c * earthRadius;

    return distance;
  }
```

Gambar 4. 41 Menghitung Jarak Tempuh Kendaraan Menggunakan Formula *Haversine*

Pada gambar 4.41 merupakan implementasi formula *haversine* untuk mendapatkan jarak tempuh kendaraan dari lokasi sebelumnya ke lokasi saat ini. Caranya dengan konversi semua *latitude* dan *longitude* dalam bentuk *decimal degree* menjadi bentuk radians. Kemudian menghitung besaran perubahan (*delta*) dengan mengurangi *latitude* (*radian*) saat ini dengan *latitude* sebelumnya (*radian*), begitu juga dengan *longitude*. Setelah mendapatkan besaran perubahan (*delta*), dimasukkan ke dalam fungsi trigonometri untuk didapatkan jarak sudutnya (*c*). Terakhir mengalikan jarak sudut (*c*) dengan radius bumi bernilai 6371,1 Km sehingga mendapatkan hasil dalam bentuk kilometer.

18. Menyimpan data dari perangkat *GPS Tracker* ke basis data

```
14  async function commitMessage(now, logDevice) {
137     if (logDevice.latitude != null && logDevice.longitude != null) { // && vhc.length > 0
138       // log tracking
139       await GpsTracksModels.bundleCreate2(logDevice, logDevice);
140     } else {
141       // log tracking
142       GpsTracksModels.bundleCreate2(logDevice, logDevice);
143     }
144   } catch (e) {
145     console.error(e);
146   }
147 }
```

Gambar 4. 42 Menyimpan Data dari Perangkat *GPS Tracker* ke Basis Data

Pada gambar 4.42 merupakan implementasi untuk menyimpan data yang telah didapat dari perangkat *GPS Tracker* ke basis data MariaDB. Implementasi ini terdapat pada fungsi *commitMessage*.

19. Membuat *background process* untuk melakukan *reverse geocoding*

```
371 const reverseGeoBackgrounds = require('./backgrounds/reverseGeoBackgrounds.js');
372 setInterval(() => {
373   reverseGeoBackgrounds.reverseGeo()
374 }, process.env.SCHEDULE_REVERSE_GEO_TIME);
```

Gambar 4. 43 *Background Process Reverse Geocoding* di NodeJS

Pada gambar 4.43 merupakan implementasi untuk melakukan *reverse geocoding* setiap beberapa waktu sesuai dengan yang telah dikonfigurasi, pada implementasi ini adalah setiap satu menit sekali.

4.4.3. Kode Program Laravel

Implementasi kode program Laravel cukup mudah untuk dipahami, karena Laravel merupakan *framework* sehingga pembagian kode menjadi lebih struktur. Sederhananya untuk mengakses halaman web (*view*) harus melewati rute yang kemudian diarahkan ke sebuah *controller* yang berisi logika aplikasi. Sebelum diarahkan ke *controller*, akan diarahkan ke

middleware terlebih dahulu untuk dilakukan otorisasi. Pada *controller* memerlukan akses ke basis data melalui sebuah model.

Selain konsep *MVC* juga terdapat implementasi *Maps API* yang memanfaatkan *library LeafletJS*, sehingga data yang telah didapatkan dari *GPS Tracker* dapat dilihat dan dianalisis lebih mudah, dan interaktif. Berikut kode program Laravel yang mengimplementasikan R.08 sampai R.21.

1. Peta Rute Laravel

Gambar 4. 44 Implementasi Peta Rute Laravel

Pada gambar 4.44 merupakan implementasi peta rute Laravel. Kegunaan dari rute adalah mengarahkan lalu lintas yang masuk sesuai dengan fungsinya.

2. *Laravel Middleware*

Gambar 4. 45 Implementasi *Laravel Middleware*

Pada gambar 4.45 merupakan implementasi *Laravel middleware*. Kegunaan dari *middleware* adalah memastikan lalu lintas yang masuk sesuai memang berhak untuk mengaksesnya (otorisasi). Pada gambar 4.45 merupakan otorisasi untuk *role* admin, *role* pelacak, dan tidak dikenali.

3. *Laravel Controller*

Gambar 4. 46 Implementasi *Laravel Controller*

Pada gambar 4.46 merupakan implementasi *Laravel controller*. Kegunaan dari *controller* adalah melakukan kontrol lalu lintas yang masuk akan diberikan data apa atau tampilan apa. Pada gambar 4.46 merupakan kontrol untuk menampilkan tampilan pengguna dan kontrol untuk menampilkan data pengguna.

4. *Laravel Model*

Gambar 4. 47 Implementasi *Laravel Model*

Pada gambar 4.47 merupakan implementasi *Laravel model*. Kegunaan dari *model* adalah sebagai perantara untuk menyimpan dan mengambil data dari basis data. Pada gambar 4.47 merupakan metode untuk mendapatkan daftar pengguna.

5. *Laravel View*

app.blade.php

resources > views > app > app.blade.php > html > body > script

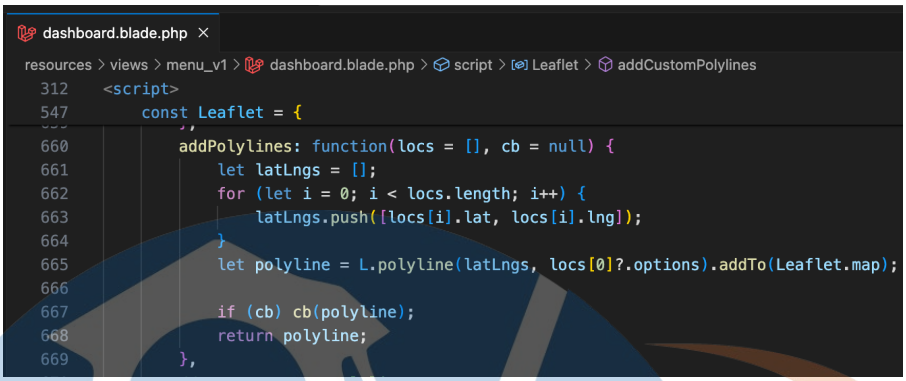
```
<html lang="\en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <link rel="icon" type="image/x-icon" href="{{ asset('images/logogram-bonceng.png') }}">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
        href="https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;
        rel="stylesheet">
    <link href="{{ asset('assets/vendor/bootstrap-5.0.2-dist/css/bootstrap.css') }}" rel="stylesheet">
    <link href="{{ asset('assets/vendor/leaflet-1.7.1/leaflet.css') }}" rel="stylesheet">
    <link href="{{ asset('assets/vendor/ionicons-v2.0-1/css/ionicons.css') }}" rel="stylesheet">
    <link href="{{ asset('assets/vendor/select2-4.1.0-rc.0/dist/css/select2.css') }}" rel="stylesheet">
    <link href="{{ asset('assets/vendor/DataTables/datatables.css') }}" rel="stylesheet">
    <link href="{{ asset('assets/css/bootstrap-datepicker.min.css') }}" rel="stylesheet">
    <link href="{{ asset('assets/css/meus.css') }}" rel="stylesheet">
    <style>…
    </style>
    @yield('customcss')
</head>

<body onload="startTime()">
    @include('template/navbar')
    @yield('content')
    {{-- Toast --}}
    <div class="position-fixed top-0 end-0 p-3" style="z-index: 99999">…
    </div>
    {{-- End Toast --}}
    <script src="{{ asset('assets/vendor/jquery-3.6.0/jquery-3.6.0.js') }}"></script>
    <script src="{{ asset('assets/vendor/bootstrap-5.0.2-dist/js/bootstrap.bundle.min.js') }}"></script>
    <script src="{{ asset('assets/vendor/select2-4.1.0-rc.0/dist/js/select2.full.min.js') }}"></script>
    <script src="{{ asset('assets/vendor/DataTables/datatables.js') }}"></script>
    <script src="{{ asset('https://cdn.datatables.net/fixedcolumns/4.0.1/js/dataTables.fixedColumns.min.js') }}">
    </script>
    <script src="{{ asset('assets/js/moment.min.js') }}"></script>
    <script src="{{ asset('assets/js/bootstrap-datepicker.min.js') }}"></script>
    <script src="{{ asset('assets/js/helper.js') }}"></script>
    <script>…
    </script>
    @yield('customjs')
</body>
```

Gambar 4. 48 Implementasi *Laravel View*

Pada gambar 4.48 merupakan implementasi *Laravel view*. Kegunaan dari *view* adalah sebagai tempat kode yang berisi tampilan. Pada gambar 4.48 merupakan kode *HTML* secara keseluruhan untuk menampilkan tampilan.

6. *Maps API*



Gambar 4. 49 Implementasi *Maps API* dengan LeafletJS

Pada gambar 4.49 merupakan implementasi *Maps API*. Dengan memanfaatkan *library LeafletJS*, peneliti dapat membuat garis yang membentuk sebuah jalan. Selain itu, *LeafletJS* juga dapat digunakan untuk membuat *marker*.

7. *Client-polling*

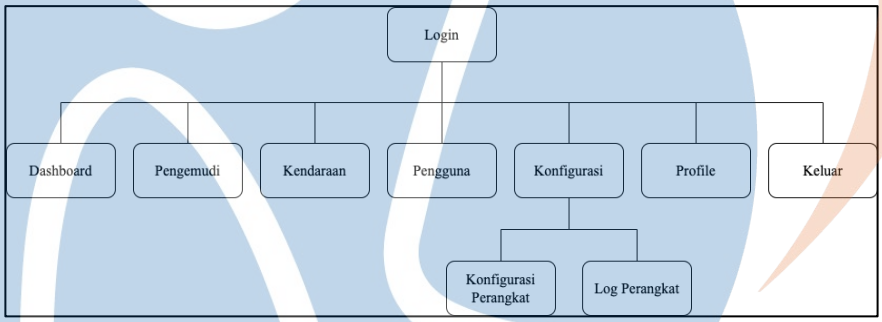
Gambar 4. 50 Implementasi Metode *Client-Polling* Untuk Mendapatkan Data Secara *Real-Time* di Laravel

Pada gambar 4.50 merupakan implementasi metode *client-polling* untuk memperbaharui data secara *real-time* dengan interval waktu yang telah ditentukan, pada implementasi ini adalah setiap satu menit sekali.

4.4.4. *User Interface* (*UI*)

User Interface merupakan antarmuka bagi pengguna untuk mengakses dan mengelola sistem pelacakan. Berikut struktur menu dan *UI* yang mengimplementasikan R.08 sampai R.21.

1. Struktur Menu



Gambar 4. 51 Struktur Menu

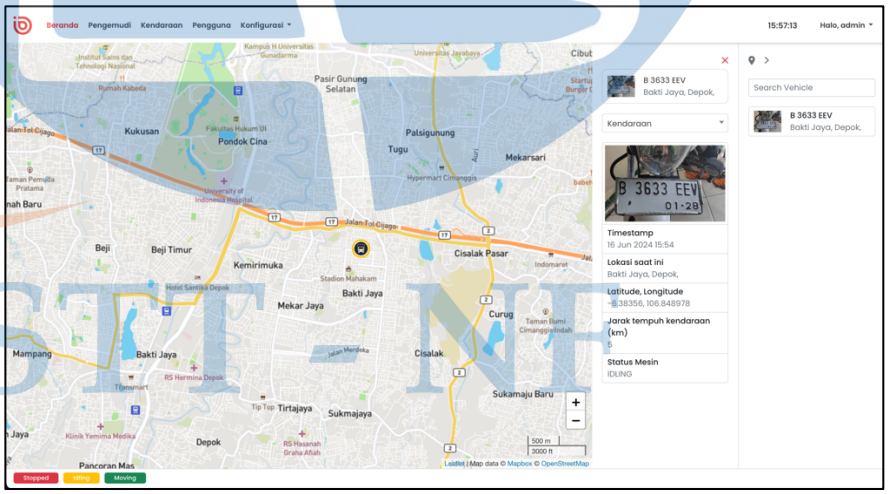
Pada gambar 4.51 merupakan struktur menu halaman web, struktur menu berguna untuk melihat secara keseluruhan pengguna dapat menjelajah halaman apa saja.

2. Halaman *Login*

Gambar 4. 52 Halaman *Login*

Pada gambar 4.52 merupakan halaman *login*. Halaman *login* berguna untuk melakukan autentikasi pengguna sebelum mengakses fitur-fitur yang ada di aplikasi *web* tersebut.

3. Halaman *Dashboard*

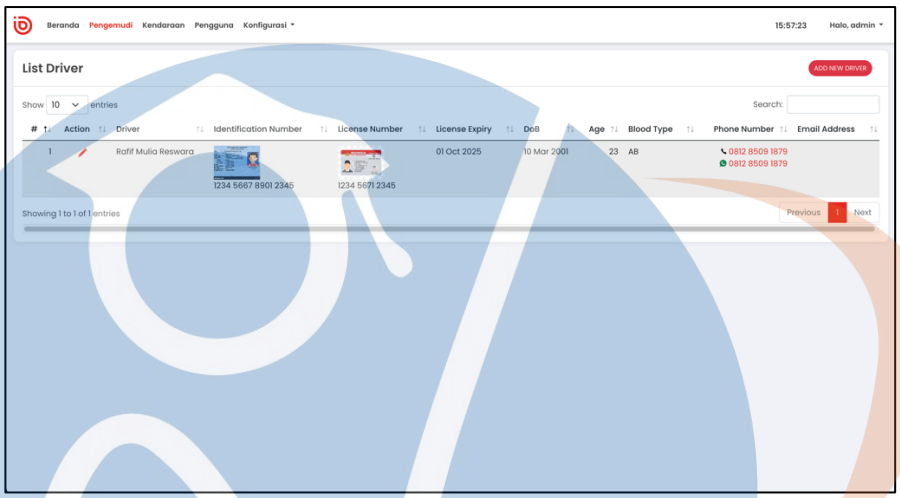


Gambar 4. 53 Halaman *Dashboard*

Pada gambar 4.53 merupakan halaman *dashboard*. Halaman *dashboard* dibuat untuk pengguna melihat kendaraan secara *real-time*,

melihat kecepatan dan jarak tempuh kendaraan, dan melihat riwayat perjalanan.

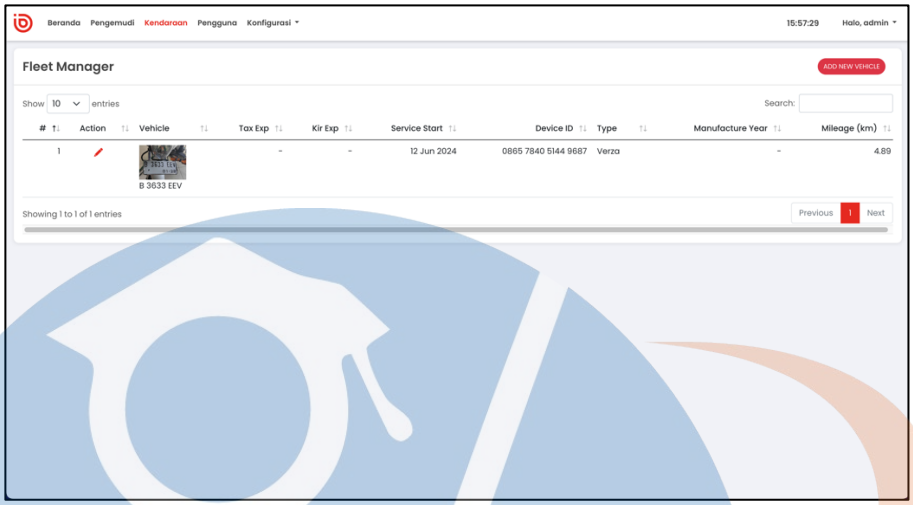
4. Halaman Pengemudi



Gambar 4. 54 Halaman Pengemudi

Pada gambar 4.54 merupakan halaman pengemudi. Halaman pengemudi dibuat untuk admin mengelola pengemudi-pengemudi yang mengendarai kendaraan.

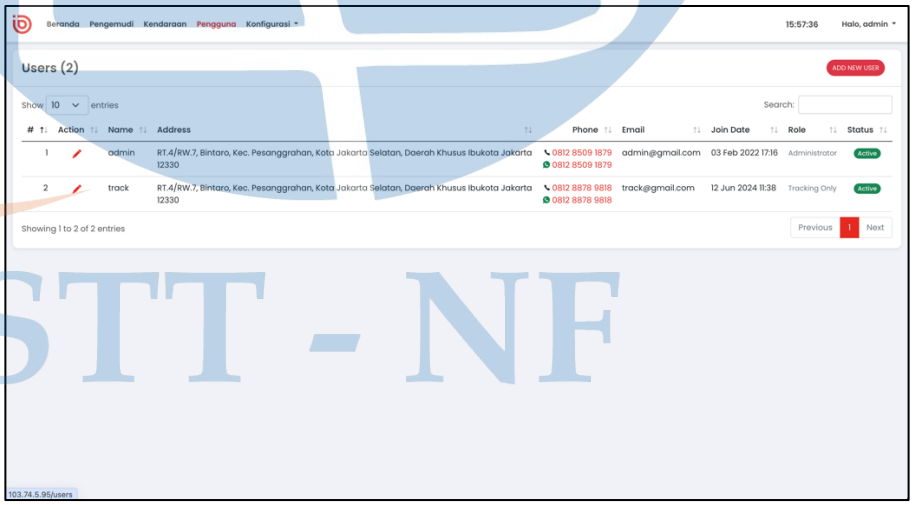
5. Halaman Kendaraan



Gambar 4. 55 Halaman Kendaraan

Pada gambar 4.55 merupakan halaman kendaraan. Halaman kendaraan dibuat untuk admin mengelola kendaraan-kendaraan yang tersedia.

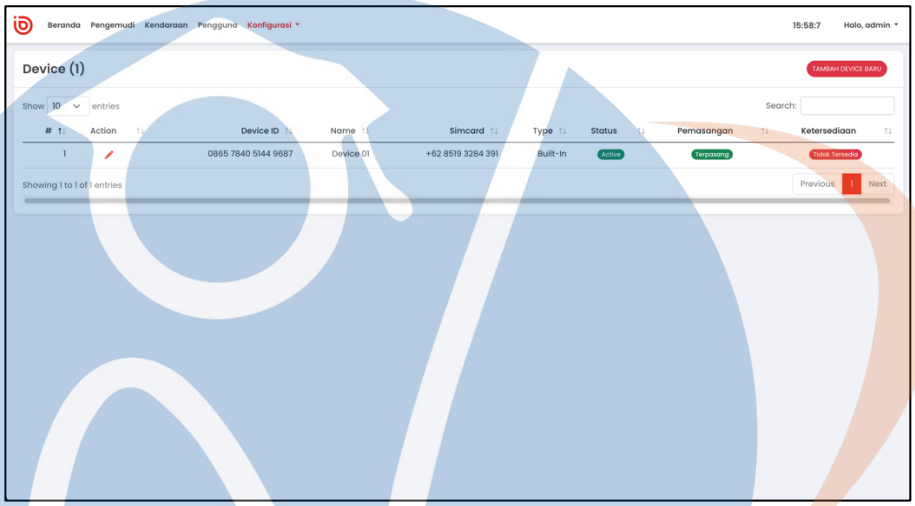
6. Halaman Pengguna



Gambar 4. 56 Halaman Pengguna

Pada gambar 4.56 merupakan halaman pengguna. Halaman pengguna dibuat untuk admin mengelola pengguna-pengguna yang mengakses aplikasi *web*.

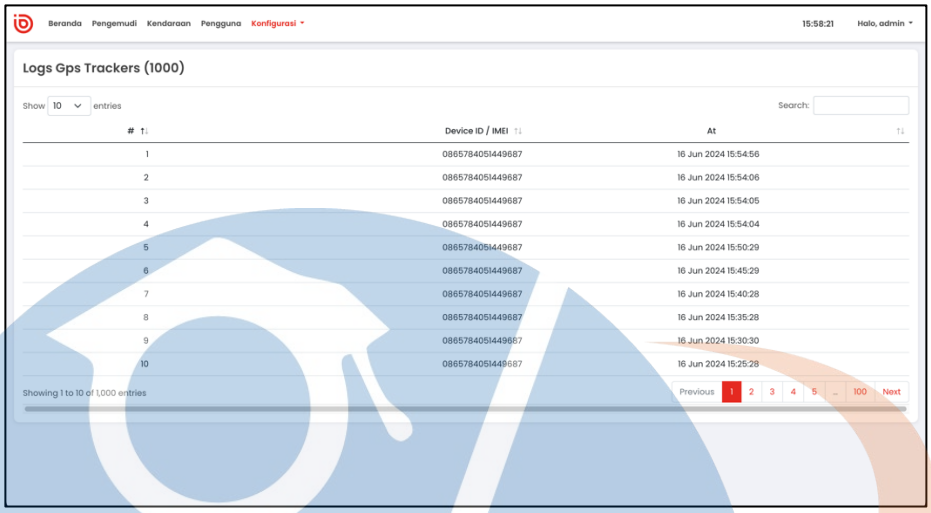
7. Halaman Konfigurasi Perangkat



Gambar 4. 57 Halaman Konfigurasi Perangkat

Pada gambar 4.57 merupakan halaman konfigurasi perangkat. Halaman konfigurasi perangkat dibuat untuk admin mengelola perangkat-perangkat yang akan atau sedang digunakan.

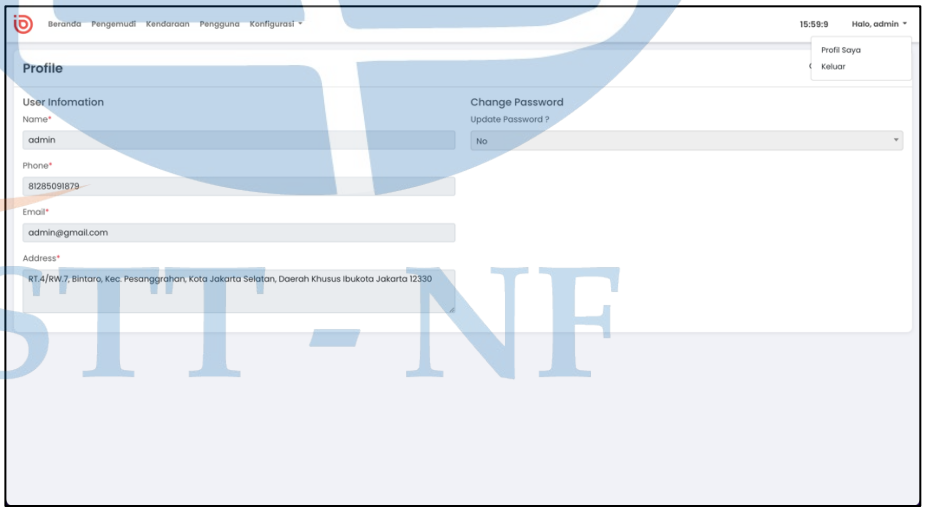
8. Halaman *Log* Perangkat



Gambar 4. 58 Halaman *Log* Perangkat

Pada gambar 4.58 merupakan halaman *log* perangkat. Halaman *log* perangkat dibuat untuk admin memastikan perangkat yang aktif dapat mengirim data ke *backend service*.

9. Halaman *Profile*



Gambar 4. 59 Halaman *Profile*

Pada gambar 4.59 merupakan halaman *profile*. Halaman *profile* dibuat untuk pengguna melakukan perubahan terkait akun yang sedang digunakannya.

4.5. Pengujian *Black Box*

Sesuai dengan perencanaan pengujian pada bab 3 tentang metodologi penelitian, pengujian dibagi menjadi 2 fase, yaitu fase administratif dan fase integrasi.

4.5.1. Fase Administratif

Fase administratif menguji terkait *login*, otorisasi, melakukan operasi *CRUD* terhadap pengemudi, kendaraan, pengguna web, perangkat *GPS Tracker* (*devices*), dan *logout*. Berikut tabel pengujian *black box* pada fase administratif pada tabel 4.3.

Tabel 4. 3 Tabel Pengujian *Black Box* Fase Administratif

SRS Code	***Test Code***	***Test Case***	***Expectation***	***Result***
R.08	T1.01	Melakukan *login*.	Otomatis diarahkan ke halaman *dashboard*.	Berhasil
	T1.02	Melakukan *logout*.	Otomatis diarahkan ke halaman *login*.	Berhasil
R.16	T1.03	Menambahkan pengemudi baru.	Menampilkan pengemudi baru pada daftar tabel.	Berhasil
	T1.04	Mengubah pengemudi.	Menampilkan hasil perubahan pada daftar tabel.	Berhasil

	T1.05	Menghapus pengemudi.	Menghilangkan pengemudi pada daftar tabel.	Berhasil
	T1.06	Mencari nama pengemudi berdasarkan daftar tabel.	Menampilkan hasil yang dicari.	Berhasil
	T1.07	Dapat melakukan pengurutan dari yang terkecil atau terbesar.	Menampilkan hasil yang diurutkan.	Berhasil
R.17	T1.08	Menambahkan kendaraan baru.	Menampilkan kendaraan baru pada daftar tabel.	Berhasil
	T1.09	Mengubah kendaraan.	Menampilkan hasil perubahan pada daftar tabel.	Berhasil
	T1.10	Menghapus kendaraan.	Menghilangkan kendaraan pada daftar tabel.	Berhasil
	T1.11	Mencari nomor kendaraan berdasarkan daftar tabel.	Menampilkan hasil yang dicari.	Berhasil
	T1.12	Dapat melakukan pengurutan dari yang terkecil atau terbesar.	Menampilkan hasil yang diurutkan.	Berhasil
R.18	T1.13	Menambahkan pengguna baru.	Menampilkan pengguna baru pada daftar tabel.	Berhasil

	T1.14	Mengubah pengguna.	Menampilkan hasil perubahan pada daftar tabel.	Berhasil
	T1.15	Menghapus pengguna.	Menghilangkan pengguna pada daftar tabel.	Berhasil
	T1.16	Mencari nama pengguna berdasarkan daftar tabel.	Menampilkan hasil yang dicari.	Berhasil
	T1.17	Dapat melakukan pengurutan dari yang terkecil atau terbesar.	Menampilkan hasil yang diurutkan.	Berhasil
R.20	T1.18	Menambahkan perangkat *GPS Tracker* baru.	Menampilkan perangkat *GPS Tracker* baru pada daftar tabel.	Berhasil
	T1.19	Mengubah perangkat *GPS Tracker*.	Menampilkan hasil perubahan pada daftar tabel.	Berhasil
	T1.20	Menghapus perangkat *GPS Tracker*.	Menghilangkan perangkat *GPS Tracker* pada daftar tabel.	Berhasil
	T1.21	Mencari kode perangkat *GPS Tracker* berdasarkan daftar tabel.	Menampilkan hasil yang dicari.	Berhasil

	T1.22	Dapat melakukan pengurutan dari yang terkecil atau terbesar.	Menampilkan hasil yang diurutkan.	Berhasil
R.18 & R.08 & R.09	T1.23	*Login* sebagai *role* pelacak dan mengakses menu pengemudi, kendaraan, dan perangkat *GPS Tracker*.	Menampilkan halaman aksi tidak diperbolehkan.	Berhasil
R.08 & R.09 & R.19	T1.24	*Role* admin membatasi akses kendaraan yang dapat dilihat oleh *user* dengan *role* pelacak pada *dashboard*, dengan mengaturnya melalui menu pengguna.	Informasi data berhasil terubah.	Berhasil
	T1.25	*Role* pelacak melakukan *login* dan hanya dapat melihat kendaraan yang telah diatur oleh admin.	Hanya dapat melihat kendaraan yang telah diatur oleh admin.	Berhasil
R.13	T1.26	Melihat *dashboard* dan melihat informasi kendaraan.	Tampil informasi kendaraan.	Berhasil
R.14	T1.27	Melihat *dashboard* dan melihat informasi pengemudi.	Tampil informasi pengemudi.	Berhasil

4.5.2. Fase Integrasi

Fase integrasi menguji terkait proses perangkat *GPS Tracker* sampai bisa mengirim data ke *backend service*. Juga mendapatkan data *latitude* dan *longitude* pada setiap paket lokasi, *heartbeat*, dan *alarm*. Juga proses *reverse geocoding* yang terjadi di belakang. Terakhir kesesuaian lokasi yang didapatkan dengan jalur yang ada di peta. Berikut tabel pengujian *black box* pada fase integrasi pada tabel 4.4.

Tabel 4. 4 Tabel Pengujian *Black Box* Fase Integrasi

SRS Code	***Test Code***	***Test Case***	***Expectation***	***Result***
R.01 - R.06 & R.21	T2.01	Menyalakan perangkat *GPS Tracker*.	Terlihat di daftar *log* perangkat *GPS Tracker* dengan *id device* tersebut.	Berhasil
R.01 - R.07 & R.10	T2.02	Penguji mengendarai kendaraan bermotor roda dua yang dipasang *GPS Tracker* dengan kecepatan 10 – 20 Km/h dari jam dan menit 12:45 sampai 13:00.	Status pergerakan kendaraan berwarna hijau / *moving* di *dashboard*.	Berhasil
- R.12 & R.15	T2.03		Pergerakan dapat dipantau secara *real-time*.	Berhasil
	T2.04		Lokasi yang ditampilkan pada peta sesuai dengan kondisi jalan.	Berhasil

	T2.05		Dapat melihat riwayat perjalanan sesuai dengan jalan yang telah dilewati.	Berhasil
	T2.06		Dapat melakukan filter riwayat perjalanan berdasarkan tanggal.	Berhasil
	T2.07		Tidak ada titik lokasi yang tampil pada peta keluar dari garis perjalanan.	Gagal
	T2.08		Dapat melihat jarak tempuh kendaraan.	Berhasil
	T2.09		Perangkat *GPS* dapat mengirimkan data sebanyak 90 kali.	Gagal
	T2.10		Dapat melihat alamat lengkap dari setiap titik lokasi yang tercatat setelah 2	Berhasil

			menit perjalanan berakhir.	
	T2.11	Penguji mengendarai kendaraan bermotor roda dua yang dipasang *GPS Tracker* dengan kecepatan 40 – 80 Km/h dari jam dan menit 13:05 sampai 13:25.	Status pergerakan kendaraan berwarna hijau / *moving* di *dashboard*.	Berhasil
	T2.12		Pergerakan dapat dipantau secara *real-time*.	Berhasil
	T2.13		Lokasi yang ditampilkan pada peta sesuai dengan kondisi jalan.	Berhasil
	T2.14		Dapat melihat riwayat perjalanan sesuai dengan jalan yang telah dilewati.	Berhasil
	T2.15		Dapat melakukan filter riwayat perjalanan berdasarkan tanggal.	Berhasil
	T2.16		Tidak ada titik lokasi yang tampil pada peta	Gagal

			keluar dari garis perjalanan.	
	T2.17		Dapat melihat jarak tempuh kendaraan.	Berhasil
	T2.18		Perangkat *GPS* dapat mengirimkan data lokasi sebanyak 120 kali.	Gagal
	T2.19		Dapat melihat alamat lengkap dari setiap titik lokasi yang tercatat setelah 2 menit perjalanan berakhir.	Berhasil
	T2.20	Diam di tempat selama 2 menit dengan kondisi mesin menyala.	Status pergerakan kendaraan berubah menjadi *idling* atau berwarna oren di *dashboard*.	Berhasil
	T2.21	Mesin dalam keadaan dimatikan.	Status pergerakan kendaraan berubah menjadi *stop* atau	Gagal

			berwarna merah di *dashboard*.	

4.6. Evaluasi Sistem

4.6.1. Evaluasi Pengujian Fase Administratif

Berdasarkan pengujian yang telah dilakukan pada fase administratif dengan total 27 skenario pengujian, telah didapatkan 27 pengujian berhasil. Jika menggunakan rumus tingkat keberhasilan *black box testing*, didapatkan persentase sebesar 100%. Sehingga pengguna dengan *role* admin dan pelacak dapat menggunakan aplikasi web tanpa adanya masalah fungsionalitas pada aplikasi.

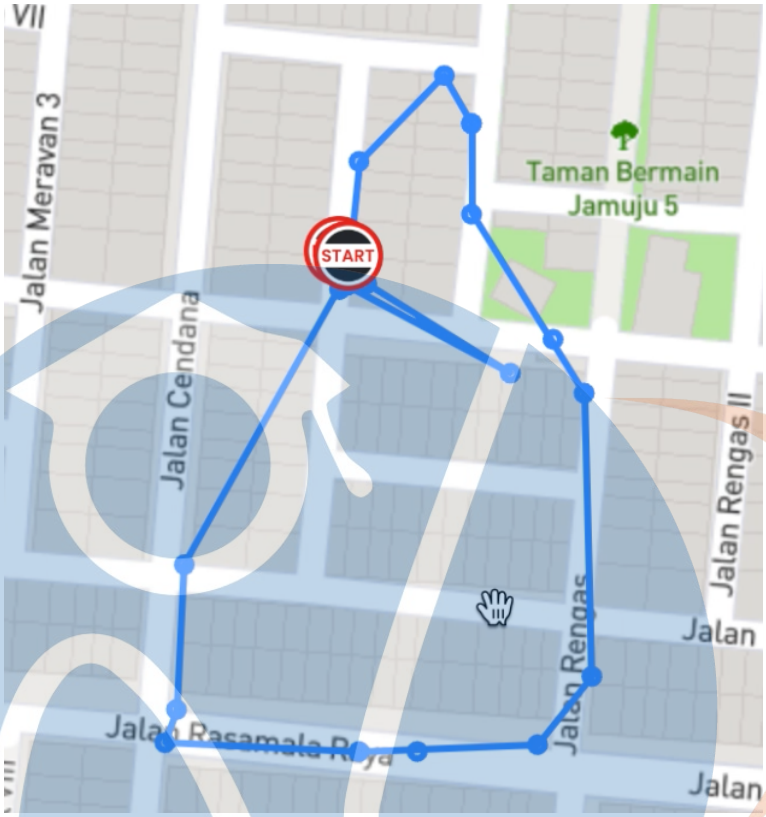
4.6.2. Evaluasi Pengujian Fase Integrasi

Evaluasi memiliki arti mengamati dari berbagai macam bukti untuk mengukur dampak dan efektivitas terhadap suatu proses yang berkaitan dengan spesifikasi sebelumnya.

Berdasarkan pengujian yang telah dilakukan pada fase integrasi dengan total 21 skenario pengujian, telah didapatkan 16 pengujian berhasil dan 5 pengujian gagal. Berikut penjelasan 5 pengujian yang gagal.

1. Pengujian T2.07

   Pada pengujian T2.07 yang melakukan perjalanan dengan kecepatan 10 – 20 Km/h dari jam dan menit 12:45 sampai 13:00 memiliki ekspektasi tidak ada titik yang keluar dari garis yang ada pada peta, namun terdapat titik yang keluar garis.



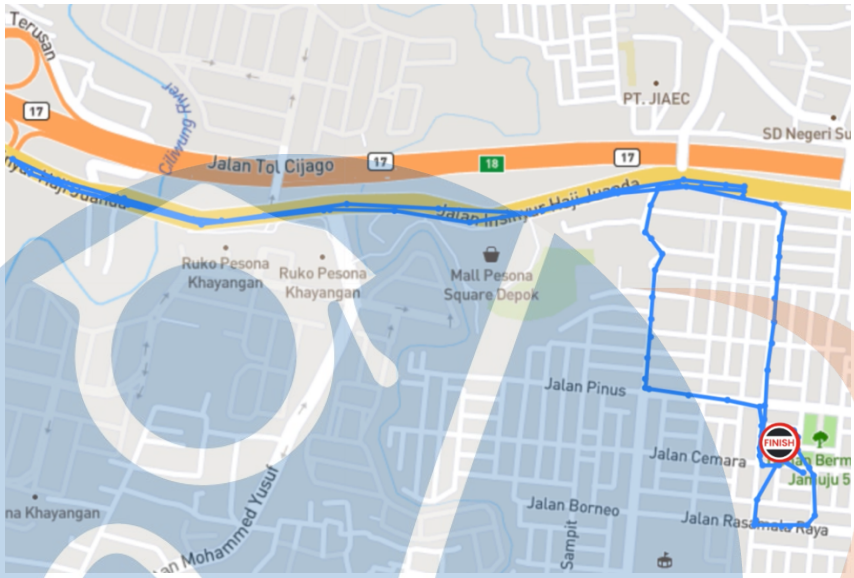
Gambar 4. 60 Riwayat Perjalanan di Gang Sekitar Perumahan

Dapat terlihat pada gambar 4.60 bahwa ada titik yang keluar dari garis. Hal ini disebabkan titik koordinat dari perangkat *GPS Tracker* tidak selalu dalam keadaan memiliki akurasi yang tinggi, terkadang akurasi *GPS* juga bisa rendah. Tetapi tingkat akurasi ini masih dapat ditoleransi.

2. Pengujian T2.09

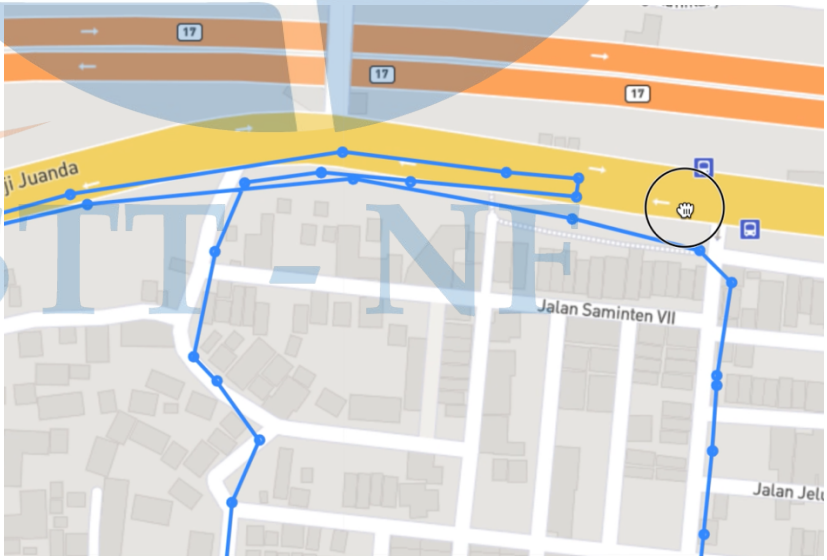
Pada pengujian T2.09 yang dilakukan dari 12:45 sampai 13:00, memiliki ekspektasi data dikirimkan sebanyak 90 kali. Hal ini dikarenakan perangkat *GPS Tracker* telah dikonfigurasi untuk mengirim data setiap 10 detik. Penyebab kegagalan pada skenario ini terletak pada konektivitas *GSM*, sehingga setiap kali *GSM* menghubungkan ulang, pengatur waktu akan mengulang dari awal dan data yang diterima akan berkurang, yaitu berjumlah 22.

3. Pengujian T2.16



Gambar 4. 61 Riwayat Perjalanan Secara Keseluruhan di Jalan Raya

Terlihat pada gambar 4.61 merupakan perjalanan secara keseluruhan pada pengujian T2.16 dengan kecepatan 40 – 80 Km/h dari jam dan menit 13:05 sampai 13:25.



Gambar 4. 62 Sebagian Riwayat Perjalanan di Jalan Raya

Pada gambar 4.62 jika dilihat lebih seksama masih memiliki titik yang keluar dari garis pada peta. Hal ini disebabkan titik koordinat dari perangkat *GPS Tracker* tidak selalu dalam keadaan memiliki akurasi yang tinggi, terkadang akurasi *GPS* juga bisa rendah. Tetapi tingkat akurasi ini masih dapat ditoleransi.

4. Pengujian T2.18

   Pada pengujian T2.18 yang dilakukan dari 13:05 sampai 13:25, memiliki ekspektasi data dikirimkan sebanyak 120 kali. Hal ini dikarenakan perangkat *GPS Tracker* telah dikonfigurasi untuk mengirim data setiap 10 detik. Penyebab kegagalan pada skenario ini terletak pada konektivitas *GSM*, sehingga setiap kali *GSM* menghubungkan ulang, pengatur waktu akan mengulang dari awal dan data yang diterima akan berkurang, yaitu berjumlah 70.

5. Pengujian T2.21

   Pada pengujian T2.21 ketika mesin kendaraan dalam keadaan dimatikan memiliki ekspektasi status pergerakan kendaraan berubah menjadi berhenti atau berwarna merah. Hasil yang didapatkan tidak sesuai dengan ekspektasi. Hal ini dikarenakan status pengapian kendaraan yang dikirimkan oleh *GPS Tracker* selalu dalam keadaan rendah, baik ketika kendaraan menyala atau dimatikan yang bisa disebabkan karena pemasangan kabel atau kondisi aki pada sebuah kendaraan.

Jika menggunakan rumus tingkat keberhasilan *black box testing* pada fase integrasi, didapatkan persentase sebesar 76% yang disebabkan karena tingkat akurasi *GPS* kurang presisi, konektivitas GSM, dan terkait instalasi pemasangan *GPS* pada kendaraan atau kondisi aki pada kendaraan.

4.6.3. Evaluasi Pengembangan Berbasis *Scrum*

Berdasarkan tabel 4.2 terkait perencanaan *sprint*, telah dilakukan *sprint* dengan 5 fase yang masing-masing fasenya adalah 2 pekan, dengan dikerjakan oleh 2 *programmer* yang terbagi dari *backend* dan *frontend*; dan *UI/UX* dan *frontend*, ditambah 1 orang sebagai *scrum master* yang bertindak sebagai penanggung jawab dari pengembangan ini dengan total durasi pengembangan selama 3 bulan, maka kecepatan pengembangan didapatkan hasil sebagai berikut.

1. Rata-rata kecepatan P1

   Total terlibat sprint: 5.

   Total *velocity* dari setiap poin pada *task* yang diselesaikan: 63 + 64 + 68 + 56 + 16 = 267.

   Kecepatan rata-rata P1: 267/5 * 100% = 53,4%
2. Rata-rata kecepatan P2

   Total terlibat sprint: 3.

   Total *velocity* dari setiap poin pada *task* yang diselesaikan: 68 + 56 + 16 = 140.

   Kecepatan rata-rata P2: 140/3 * 100% = 46,6%
3. Rata-rata kecepatan tim secara keseluruhan

   Total terlibat sprint: 5.

   Total *velocity* dari setiap poin pada *task* yang diselesaikan: 63 + 64 + 68 + 56 + 16 = 267.

   Kecepatan tim rata-rata: 267/5 * 100% = 53,4%

4.7. Evaluasi Saran

Berdasarkan pengujian yang telah dilakukan dan evaluasi terhadap pengujian, peneliti menyadari beberapa kekurangan dan memiliki beberapa masukan yang di antaranya sebagai berikut.

1. Pada fase pengujian integrasi yang pengujiannya gagal pada butir pengujian T2.07 dan T2.16 telah didapatkan hasil berupa masih terdapat titik lokasi

yang keluar garis jalan pada peta, baik perbedaannya tipis sehingga dapat ditoleransi atau cukup besar, sehingga peneliti memiliki saran untuk pengembangan selanjutnya terdapat mekanisme atau algoritma untuk membedakan titik yang keluar garis pada peta dan mana yang sesuai, sehingga data yang tidak sesuai dapat dibuang atau dikoreksi jika memang perbedaannya kecil dan data yang tampil pada peta lebih bagus.

2. Pada fase pengujian integrasi yang pengujiannya gagal pada ada butir pengujian T2.21 yang seharusnya status kendaraan berhenti namun masih tetap menyala dikarenakan data yang didapatkan dari *GPS Tracker* memang status pengapian kendaraan baik ketika kendaraan dinyalakan atau dimatikan selalu dalam keadaan pengapian rendah, sehingga saran peneliti untuk pengembangan berikutnya adalah memastikan instalasi kabel pada *GPS Tracker* serta pemasangan pada kendaraan sudah tepat atau menggunakan cara lain untuk menentukan status pergerakan kendaraan selain dari pengapian, misalnya berdasarkan waktu, cara ini sangat diperlukan jika instalasi kabel pada *GPS Tracker* serta pemasangannya pada kendaraan memang tidak ada yang keliru.
3. Pada fase pengujian integrasi yang pengujiannya berhasil pada butir pengujian T2.03 dan T2.12 terkait pemantauan secara *real-time* dengan menerapkan metode *client polling* yaitu melakukan *request* secara berkala setiap satu menit sekali terasa kurang *real-time* dan memiliki *delay* yang terasa cukup lama, dan apabila dikecilkan interval *request* secara berkala bisa menyebabkan performa menjadi lambat, yang disebabkan karena banyak *request* atau *query SQL* yang tidak optimal, sehingga peneliti menyarankan untuk pengembangan berikutnya untuk mencoba teknologi *websocket* atau optimasi *query* bahkan relokasi tabel supaya performa tetap cepat dan terasa lebih *real-time* tanpa *delay*.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan dengan judul "Rancang Bangun Aplikasi Web Untuk Pelacakan Kendaraan Menggunakan NodeJS dan *Framework* Laravel", dapat disimpulkan sebagai berikut:

1. Aplikasi web untuk mengumpulkan data posisi kendaraan secara *online* memanfaatkan *GPS* telah dirancang dan dikembangkan menggunakan teknologi NodeJS untuk menerima data dan menangani koneksi yang datang dari perangkat *GPS*, menerjemahkan data sesuai dengan spesifikasi protokol GT06, memanfaatkan *API* dari OpenStreetMap untuk melakukan *reverse geocoding* dari data *latitude* dan *longitude* menjadi data alamat lengkap, dan menyimpannya ke basis data MariaDB sehingga aplikasi web bagian *backend* dapat menerima data dari *GPS* secara *online*. Pengujian secara *black box* mendapatkan hasil tingkat keberhasilan sesuai dengan spesifikasi sebesar 76% disebabkan oleh akurasi perangkat *GPS* yang kurang presisi, konektivitas *GSM*, dan terkait instalasi pada kendaraan atau kondisi aki.
2. Aplikasi web untuk melakukan manajemen dan pemantauan kendaraan secara *real-time* telah dirancang dan dikembangkan menggunakan *framework* Laravel, memanfaatkan *Maps API* untuk berinteraksi dengan data secara interaktif melalui peta, metode *client-polling* untuk mendapatkan data secara *real-time*, serta membuat fitur manajemen kendaraan, manajemen pengemudi, dan manajemen perangkat untuk memudahkan admin menggunakan sistem dan mengelolanya. Pengujian secara *black box* mendapatkan hasil tingkat keberhasilan pemenuhan sesuai spesifikasi sebesar 100%.

5.2. Saran

Berdasarkan evaluasi yang telah dilakukan terhadap pengujian, yaitu pada subbab 4.7 terkait evaluasi saran terhadap pengujian yang dilakukan, penelitian ini

masih memiliki kekurangan dan saran untuk penelitian selanjutnya, di antaranya sebagai berikut.

1. Mekanisme atau algoritma untuk membedakan titik lokasi yang keluar dari garis peta, sehingga data dapat dibuang jika memang jarak dari titik lokasi dengan garis terlalu besar atau dikoreksi jika perbedaannya hanya sedikit dan data yang tampil pada peta lebih bagus.
2. Menggunakan cara lain untuk menentukan status pergerakan kendaraan selain dari pengapian misalnya berdasarkan waktu, apabila instalasi kabel pada *GPS Tracker* sudah benar beserta pemasangannya pada kendaraan.
3. Menggunakan teknologi *websocket* jika penyebabnya karena apabila pada metode *client polling* jika interval untuk memperbaharui datanya kecil yang menyebabkan performa menjadi lambat sehingga *request* menjadi tinggi; atau mencoba optimasi *query SQL* atau bahkan sampai relokasi tabel jika permasalahannya pada struktur tabel, sehingga tidak perlu menggunakan teknologi *websocket* dan hanya perlu mengecilkan interval *client polling* untuk memperbaharui data.

DAFTAR PUSTAKA

[1] Y. W. Riyadi Putra, F. Nur Styaningsih, dan W. H. Herviana, "Analisis Perkembangan Transportasi Online di Indonesia di Era 4.0 Dengan Metode Penelitian Deskriptif," *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 4, no. 1, hlm. 162–170, Jan 2022, doi: 10.47233/jteksis.v4i1.389.

[2] N. B. Puspitasari, W. Budiawan, dan V. Hurisandi, "The Identification of Variables of Quality Influence Mobile Location-Based Service (m-LBS) (A Case Study: Go-Food Services in Semarang City)," dalam *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Sep 2019. doi: 10.1088/1757-899X/598/1/012105.

[3] M. M. Pathan dan U. A. Bongale, "Design And Implementation Of Real-Time Web-Based Vehicle Tracking System Using Sim," *Journal of Pharmaceutical Negative Results ¦*, vol. 13, no. 10, hlm. 2822–2828, Des 2022, doi: 10.47750/pnr.2022.13.S10.338.

[4] A. Wiyanda, Suswanto, dan F. V. A. S. Prasetya, "Development Tourism Geographic Information System of Samarinda," *TEPIAN*, vol. 3, no. 1, hlm. 13–23, Mar 2022, doi: 10.51967/tepian.v3i1.689.

[5] D. P. Nurdin, S. Karim, dan N. Kurniadin, "Geographic Information System for Mapping Agricultural Land in North Samarinda District," *TEPIAN*, vol. 3, no. 4, hlm. 165–172, Des 2022, doi: 10.51967/tepian.v3i4.701.

[6] Edy Irwansyah, *Sistem Informasi Geografis: Prinsip Dasar dan Pengembangan Aplikasi*, 1 ed. Yogyakarta: Penerbit Digibooks, 2013.

[7] U. F. Kurniawati *dkk.*, "Pengolahan Data Berbasis Sistem Informasi Geografis (SIG) Untuk Kebutuhan Penyusunan Profil di Kecamatan Sukolilo," *Jurnal Pengabdian kepada Masyarakat-DRPM ITS*, vol. 4, no. 3, hlm. 190–196, Apr 2020, Diakses: 27 Maret 2024. [Daring]. Tersedia pada: https://journal.its.ac.id/index.php/sewagati/article/view/363

[8] M. Arif Budiman, A. Zatulo Harefa, dan D. Virgian Shaka, "Perancangan Sistem Pelacak Gps Dan Pengendali Kendaraan Jarak Jauh Berbasis

Arduino," *Proceeding SENDIU 2020*, hlm. 356–363, Jul 2020, Diakses: 22 Maret 2024. [Daring]. Tersedia pada: https://www.unisbank.ac.id/ojs/index.php/sendi_u/article/view/8006

[9] F. Anugrah, "Sistem Kontrol dan Pengamanan Kotak Penyimpanan Uang Menggunakan Aplikasi Telegram Berbasis Arduino," *SNASTIKOM 2020*, vol. 2, no. 1, hlm. 150–156, Des 2023, Diakses: 17 Maret 2024. [Daring]. Tersedia pada: https://prosiding.snastikom.com/index.php/SNASTIKOM2020/article/view/64

[10] E. Gomes, F. Costa, C. De Rolt, P. Plentz, dan M. Dantas, "A Survey from Real-Time to Near Real-Time Applications in Fog Computing Environments," *Telecom*, vol. 2, no. 4, hlm. 489–517, Des 2021, doi: 10.3390/telecom2040028.

[11] M. Dudjak dan G. Martinović, "An API-first methodology for designing a microservice-based backend as a service platform," *Information Technology and Control*, vol. 49, no. 2, hlm. 206–223, Sep 2020, doi: 10.5755/j01.itc.49.2.23757.

[12] J. Ofoeda, R. Boateng, dan J. Effah, "Application programming interface (API) research: A review of the past to inform the future," *International Journal of Enterprise Information Systems*, vol. 15, no. 3, hlm. 76–95, Jul 2019, doi: 10.4018/IJEIS.2019070105.

[13] K. S. Bhandari dan P. K. Gupta, "Data Transfer Using Tcp Socket Over Http Application," Waknaghat, 2023. Diakses: 27 Maret 2024. [Daring]. Tersedia pada: http://ir.juit.ac.in:8080/jspui/jspui/handle/123456789/9872

[14] R. Yadav, "Building a Blog Project using JavaScript, NodeJS and MongoDB," 2021. Diakses: 27 Maret 2024. [Daring]. Tersedia pada: https://www.theseus.fi/handle/10024/498173

[15] N. Bouraqadi dan D. Mason, "PharoJS: Transpiling Pharo Classes to JS ECMAScript 5 versus ECMAScript 6," dalam *IWST 2023: International*

Workshop on Smalltalk Technologies, S. Ducasse dan G. Rakic, Ed., Lyon, 2023, hlm. 1–20. [Daring]. Tersedia pada: https://nootrix.com/noury

[16] Michael Fogus, *Functional JavaScript: Introducing Functional Programming with Underscore.js*, 1 ed. O'Reilly Media, 2023. Diakses: 27 Maret 2024. [Daring]. Tersedia pada: https://www.google.co.id/books/edition/Functional_JavaScript/OIhuxqUO2TcC

[17] B. Basumatary dan N. Agnihotri, "Benefits and Challenges of Using NodeJS," *International Journal of Innovative Research in Computer Science & Technology*, vol. 10, no. 3, hlm. 67–70, Mei 2022, doi: 10.55524/ijircst.2022.10.3.13.

[18] A. D. Hardiansyah, D. C. Nugrahaeni, P. Dewi, dan M. Kom, "Perancangan Basis Data Sistem Informasi Perwira Tugas Belajar (Sipatubel) Pada Kementerian Pertahanan," *SENAMIKA*, vol. 1, no. 2, hlm. 222–233, 2020, Diakses: 28 Maret 2024. [Daring]. Tersedia pada: https://conference.upnvj.ac.id/index.php/senamika/article/view/529

[19] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, dan B. Luo, "SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review," *Big Data and Cognitive Computing*, vol. 7, no. 2, Jun 2023, doi: 10.3390/bdcc7020097.

[20] Shoykulova Dilorom Kudratovna dan Sh.Q. Shoyqulov, "PHP is one of the main tools for creating a Web page in computer science lessons," *Texa.Jour. of Engg. and Tech.*, vol. 9, hlm. 83–87, Jun 2022, Diakses: 28 Maret 2024. [Daring]. Tersedia pada: https://zienjournals.com/index.php/tjet/article/view/2000

[21] H. C. Lim, X. Kang, dan S. Debray, "Modeling code manipulation in JIT compilers," dalam *SOAP 2022 - Proceedings of the 11th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis, co-located with PLDI 2022*, Association for Computing Machinery, Inc, Jun 2022, hlm. 9–15. doi: 10.1145/3520313.3534656.

[22] O. Widodo Purbo, "A Systematic Analysisis: Website Development using Codeigniter and Laravel Framework," *Enrichment: Journal of Management*, vol. 12, no. 1, hlm. 1008–1014, Des 2021, doi: https://doi.org/10.35335/enrichment.v12i1.346.

[23] M. I. N. Saroni dan B. Mulyanti, "Hypertext preprocessor framework in the development of web applications," *IOP Conf Ser Mater Sci Eng*, vol. 830, no. 2, Mei 2020, doi: 10.1088/1757-899X/830/2/022096.

[24] P. R. Chavan dan S. Pawar, "Comparison Study Between Performance of Laravel and Other PHP Frameworks," *IJRESM*, vol. 4, no. 10, hlm. 27–29, Okt 2021, Diakses: 28 Maret 2024. [Daring]. Tersedia pada: https://journal.ijresm.com/index.php/ijresm/article/view/1420

[25] R. Saily, H. Maizir, D. Yasri, dan P. Studi Teknik Sipil Sekolah Tinggi Teknologi Pekanbaru, "Pembuatan Peta Tematik Menggunakan Sistem Informasi Geografis (SIG) Pada Desa Teluk Latak," *Indonesian Journal of Construction Engineering and Sustainable Development (CESD)*, vol. 4, no. 2, hlm. 99–107, Des 2021, doi: https://doi.org/10.25105/cesd.v4i2.12497.

[26] A. Hajar, I. Nabawi, L. Kartikawati, F. R. Yudana, S. Budi, dan N. Prasetiyantara, "Pengolahan Data Spasial-Geolocation untuk Menghitung Jarak 2 Titik Spatial-Geolocation Data Processing to Calculate 2 Point," *Citec Journal*, vol. 8, no. 1, hlm. 32–42, Jan 2021, doi: https://doi.org/10.24076/citec.2021v8i1.265.

[27] C. Husada, K. Dwi Hartomo, dan H. Prillysca Chernovita, "Implementasi Haversine Formula untuk Pembuatan SIG Jarak Terdekat ke RS Rujukan COVID-19," *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, vol. 4, no. 5, hlm. 874–883, Okt 2020, doi: https://doi.org/10.29207/resti.v4i5.2255.

[28] Ph. D. and B. R. M. P. D. Roger S. Pressman, *Software Engineering A Practitioner's Approach*, 7 ed. McGraw Hill: Raghothaman Srinivasan, 2012.

[29] A. N. Yusril, I. Larasati, dan P. Al Zukri, "Systematic Literature Review Analisis Metode Agile dalam Pengembangan Aplikasi Mobile," *Sistemasi: Jurnal Sistem Informasi*, vol. 10, no. 2, hlm. 369–380, 2021, doi: https://doi.org/10.32520/stmsi.v10i2.1237.

[30] N. Lutfiani, P. Harahap, Q. Aini, A. Dimas, A. R. Ahmad, dan U. Rahardja, "Inovasi Manajemen Proyek I-Learning Menggunakan Metode Agile Scrumban," *InfoTekJar*, vol. 5, no. 1, hlm. 96–101, 2020, doi: 10.30743/infotekjar.v5i1.2848.

[31] N. Abdullatif dan S. Kassem, "Modelling of agent-based vehicle routing problem using unified modelling language," *Journal Europeen des Systemes Automatises*, vol. 53, no. 6, hlm. 781–789, Des 2020, doi: 10.18280/jesa.530604.

[32] Ian Sommerville, *Software Engineering*, 10 ed. England: PEARSON, 2016.

[33] D. Bahar Muslimin, D. Kusmanto, K. Femi Amilia, M. Syamsul Ariffin, dan S. Mardiana, "Pengujian Black Box pada Aplikasi Sistem Informasi Akademik Menggunakan Teknik Equivalence Partitioning," *Jurnal Informatika Universitas Pamulang*, vol. 5, no. 1, hlm. 19–25, Mar 2020, doi: https://doi.org/10.32493/informatika.v5i1.3778.

[34] W. Yuliani dan N. Banjarnahor, "Metode Penelitian Pengembangan (Rnd) Dalam Bimbingan Dan Konseling," *QUANTA J. Kaji. Bimbing. dan Konseling dalam Pendidik*, vol. 5, no. 3, hlm. 111–118, Des 2021, doi: 10.22460/q.v2i1p21-30.642.

[35] A. Maydiantoro, "Model-Model Penelitian Pengembangan (Research And Development)," *Jurnal Pengembangan Profesi Pendidik Indonesia (JPPPI)*, vol. 1, no. 2, 2021, Diakses: 22 April 2024. [Daring]. Tersedia pada: http://repository.lppm.unila.ac.id/43959/

[36] Y. H. Rayanto dan Sugianti, *Penelitian Pengembangan Model Addie Dan R2D2: Teori & Praktek*, 1 ed., vol. 1. Pasuruan: Lembaga Academic & Research Institute, 2020. Diakses: 22 April 2024. [Daring]. Tersedia pada:

https://books.google.co.id/books?id=pJHcDwAAQBAJ&dq=penelitian+pengembangan&lr=&source=gbs_navlinks_s

LAMPIRAN

Lampiran 1: Hasil Wawancara

Nama:	Agus Prihanto
Jabatan:	*Vice President*
Tanggal:	Kamis, 07 November 2022
Jam:	13:00 – 14:30
Tempat:	Kantor PT. Swa Nusa Multimedia
Alamat:	Jalan RC Veteran 11A, Rempoa, Bintaro, Pesanggrahan, Kota Jakarta Selatan 12330

Daftar Pertanyaan:

1. Bagaimana kondisi perusahaan saat ini?
   Jawab: Baik-baik saja.

2. Setahu saya PT. Swa Nusa Multimedia merupakan perusahaan yang bergerak di bidang logistik, bagaimana proses pengantaran logistik yang ada saat ini?
   Jawab: Betul, untuk saat ini proses pemesanan dan pengantaran masih berlangsung secara manual via *smartphone*.

3. Apa saja yang menjadi kendala selama melakukan pengantaran?
   Jawab: Perlu dilakukan komunikasi secara berkala via *smartphone* untuk mengetahui kondisi pengantaran seperti sopir, lokasi, barang bawaan, kendaraan, dan kendala apabila ada kendala.

4. Apakah ada jarak tertentu dalam mengantarkan barang?

Jawab: Tidak ada, pengantaran dapat dilakukan untuk jarak yang relatif dekat seperti dalam kelurahan, dan selama ini jarak yang paling jauh sudah sampai ke Jawa Timur.

5. Apakah PT. Swa Nusa Multimedia membutuhkan sistem pelacakan kendaraan yang *real-time*?
   Jawab: Iya.

6. Sebelumnya, siapa saja yang terlibat dalam proses pengantaran?
   Jawab: Staf admin untuk melakukan hal administratif seperti menerima pesanan, pengecekan kondisi kendaraan, memantau proses pengantaran dari jauh, dan berhubungan dengan penerima barang pada tempat tujuan dari jauh. Kemudian ada sopir yang bertugas mengantarkan logistik.

7. Apakah PT. Swa Nusa Multimedia setuju jika masalah ini akan dijadikan riset untuk tugas akhir saya?
   Jawab: Boleh, tidak masalah.

Lampiran 2: Dokumen Protokol GT06 untuk *GPS Tracker*

https://www.traccar.org/protocol/5023-gt06/GT06_GPS_Tracker_Communication_Protocol_v1.8.1.pdf