BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pelaksanaan program magang dilakukan di PT. Agrolink Nusantara Indonesia, dengan penulis menempati peran sebagai *Full Stack Development*. Berbeda dengan skema magang yang umumnya dilaksanakan secara berkelompok, program magang ini dijalankan secara individual. Meskipun demikian, penulis tetap menjadi bagian dari tim pengembangan aplikasi perusahaan dan terlibat secara aktif dalam kegiatan pengembangan sistem.

Dalam struktur organisasi perusahaan, penulis berada di bawah koordinasi *Chief Technology Officer* yang bertanggung jawab atas perencanaan, pengembangan, dan implementasi teknologi perusahaan. Untuk mendukung pelaksanaan tugas tersebut, CTO dibantu oleh *VP of Engineering* yang memimpin tim teknis, yang terdiri atas *Software Engineer (Backend)*, *Software Engineer (Frontend Flutter)*, *QA Engineer*, *Data/AI Engineer*, serta *DevOps/SRE*. Tim teknis ini berperan dalam pengembangan, pemeliharaan, dan peningkatan kualitas sistem digital perusahaan, termasuk sistem *Fleet Management* berbasis web yang menjadi fokus utama kegiatan magang.

Secara operasional, penulis bekerja dalam tim pengembangan yang terdiri atas penulis sebagai *Full Stack Development*, seorang rekan dari divisi *UI/UX Design*, serta supervisor yang berperan sebagai pembimbing lapangan. Dalam pelaksanaan tugasnya, penulis berkoordinasi secara langsung dengan supervisor terkait perancangan dan pengembangan sistem, serta secara struktural berada di bawah pengawasan manajemen perusahaan yang dipimpin oleh Bapak **Yonatan Ripandra Sinaga** selaku *Chief Executive Officer* di PT. Agrolink Nusantara Indonesia.

Selama periode magang, penulis mengemban tanggung jawab utama dalam perancangan dan pengembangan sistem *Fleet Management* berbasis web, yaitu sistem internal yang digunakan perusahaan untuk mengelola armada operasional. Supervisor berperan aktif sebagai pemberi arahan teknis, evaluator capaian kerja, serta penyedia umpan balik yang konstruktif terkait pengembangan fitur dan peningkatan kualitas sistem. Keterlibatan supervisor secara berkelanjutan memastikan bahwa proses implementasi berjalan sesuai dengan standar teknis

perusahaan serta mendukung pencapaian tujuan pembelajaran program magang.

Koordinasi dan komunikasi selama kegiatan magang dilaksanakan secara daring, menyesuaikan dengan pola kerja fleksibel yang diterapkan oleh perusahaan. Setiap hari kerja, yaitu Senin hingga Jumat, pembaruan progres pekerjaan dan diskusi teknis dilakukan melalui aplikasi *WhatsApp* sebagai media komunikasi utama. Selain itu, pertemuan evaluasi dilaksanakan setiap hari Sabtu secara daring melalui platform seperti Google Meet atau Zoom. Pertemuan ini bertujuan untuk meninjau capaian pekerjaan, mengidentifikasi kendala teknis yang dihadapi, serta menyusun rencana kerja lanjutan agar proses pengembangan sistem dapat berjalan secara terarah, terukur, dan sistematis.

3.2 Tugas yang Dilakukan

Tugas utama yang diberikan kepada penulis adalah mengembangkan *website Fleet Management*, yaitu sistem internal yang dirancang untuk mendukung pengelolaan armada dan aktivitas operasional yang terkait dengan distribusi hasil pertanian. Secara konseptual, sistem ini memiliki kemiripan dengan layanan komersial seperti *Lacak.io* dan *Transtrack*, namun rancangan dan implementasinya disesuaikan secara spesifik dengan kebutuhan operasional PT. Agrolink Nusantara Indonesia yang bergerak di sektor agrikultur.

Fitur-fitur utama yang dikembangkan dalam sistem *Fleet Management* meliputi:

- **Manajemen Armada**
  Meliputi pendataan kendaraan, pemantauan status operasional (misalnya: *idle*, *in transit*, *maintenance*), serta pengelolaan informasi terkait kinerja dan kelayakan armada.

- **Pelacakan Perjalanan Truk secara *Real-Time***
  Menyediakan kemampuan pemantauan lokasi kendaraan, visualisasi rute perjalanan, dan estimasi waktu kedatangan (*Estimated Time of Arrival*) secara langsung.

- **Manajemen Trip dan Aktivitas Pengiriman**
  Berisi fungsi pengaturan dan pencatatan informasi perjalanan, termasuk tujuan pengiriman, penugasan sopir, waktu keberangkatan, dan waktu kedatangan.

- **Manajemen Maintenance**
  Mencakup penjadwalan perawatan kendaraan, pencatatan riwayat perbaikan, serta pemantauan status kelayakan dan kesiapan armada untuk beroperasi.

- **Dashboard Laporan (*Reports*)**
  Menyajikan ringkasan dan statistik operasional, seperti jumlah perjalanan, durasi rata-rata pengiriman, jumlah armada aktif, serta indikator performa kendaraan dan sopir.

Melalui pengembangan *website Fleet Management* ini, perusahaan diharapkan dapat meningkatkan efektivitas pengelolaan armada secara menyeluruh. Sistem yang terintegrasi memungkinkan pemantauan operasional dilakukan secara lebih akurat, mendukung pengambilan keputusan berbasis data, serta meningkatkan efisiensi distribusi hasil pertanian dalam kegiatan operasional sehari-hari. Pada skala yang lebih luas, pengembangan sistem ini juga berkontribusi terhadap pembentukan ekosistem digital di sektor pertanian yang lebih terintegrasi dan adaptif terhadap kebutuhan pemangku kepentingan di lapangan.

3.2.1 Technology Stack

Dalam proses perancangan dan pengembangan sistem *Fleet Management* berbasis web, digunakan beberapa teknologi yang dipilih untuk mendukung efisiensi, skalabilitas, serta kemudahan integrasi. Adapun *technology stack* yang digunakan adalah sebagai berikut:

- **Frontend**
  Flutter Web digunakan untuk membangun antarmuka pengguna (*user interface*) yang responsif dan konsisten pada berbagai ukuran layar. Pemanfaatan *Flutter* mendukung proses pengembangan yang cepat dengan satu basis kode (*single codebase*) untuk platform web.

- **Backend-as-a-Service (BaaS) dan Basis Data**
  Supabase digunakan sebagai layanan *Backend-as-a-Service* yang menyediakan fitur autentikasi, pengelolaan basis data, serta integrasi *API*. Supabase memanfaatkan *PostgreSQL* sebagai sistem manajemen basis data utama, sehingga rancangan skema dan implementasi penyimpanan data pada sistem ini selaras dengan judul laporan, yaitu penggunaan *PostgreSQL* sebagai basis data utama pada sistem *Fleet Management*.

- **Version Control**
  Git dan GitHub digunakan sebagai alat untuk pengelolaan versi kode sumber (*source code management*), dokumentasi perubahan, serta kolaborasi pengembangan dengan pihak lain apabila diperlukan.

- **Design Tools**
  Figma dimanfaatkan untuk perancangan antarmuka pengguna dan penyusunan alur pengalaman pengguna, sehingga tampilan *website* dapat dirancang secara sistematis sebelum diimplementasikan dalam bentuk kode.

Pemilihan *technology stack* tersebut didasarkan pada kebutuhan pengembangan aplikasi web modern yang mendukung pengembangan cepat, fleksibel, dan mudah diintegrasikan dengan infrastruktur yang digunakan perusahaan. Secara khusus, penggunaan *Flutter* dan *PostgreSQL* melalui Supabase menjadikan sistem ini relevan dengan fokus rancang bangun yang diangkat dalam laporan magang.

3.2.2 Environment dan Tools

Selama proses pengembangan sistem *Fleet Management*, digunakan berbagai *environment* dan *tools* yang dirancang untuk mendukung produktivitas, kolaborasi, serta kualitas hasil pengembangan. Pemilihan perangkat dan perangkat lunak disesuaikan dengan kebutuhan pengembangan *Full Stack* berbasis *Flutter Web* dan integrasi basis data *PostgreSQL* melalui Supabase. Adapun rincian *environment* dan *tools* yang digunakan adalah sebagai berikut:

- **Operating System**
  Windows 11 digunakan sebagai lingkungan utama pengembangan. Sistem operasi ini dipilih karena stabilitasnya dalam menjalankan Flutter SDK, Supabase CLI, serta berbagai tools pendukung lainnya.

- **Code Editor**
  Visual Studio Code (VS Code) berperan sebagai *Integrated Development Environment (IDE)* utama. VS Code menyediakan fitur seperti *code completion*, integrasi Flutter/Dart, *hot reload*, serta dukungan berbagai ekstensi untuk pengembangan web dan manajemen Git.

- **Browser untuk Pengujian**
  Google Chrome dan Microsoft Edge digunakan untuk menjalankan serta

menguji hasil *build* Flutter Web secara langsung. Pengujian dilakukan untuk memastikan tampilan dan fungsionalitas sistem berjalan dengan baik pada berbagai *browser*.

- **Command Line Interface (CLI)**
  Terminal/Command Prompt digunakan untuk menjalankan perintah seperti `flutter doctor`, `flutter run`, `flutter build`, serta perintah Git seperti `git add`, `git commit`, dan `git push`. Penggunaan CLI mempermudah otomasi dan pengelolaan alur kerja pengembangan.

- **Akses Dokumentasi dan Tools Tambahan**
  Browser juga dimanfaatkan untuk mengakses dokumentasi resmi *Flutter*, *Supabase*, dan *PostgreSQL*, referensi pemrograman, serta platform desain seperti *Figma* sebagai pendukung proses pengembangan.

Dengan kombinasi *environment* dan *tools* tersebut, rangkaian proses mulai dari perancangan antarmuka, penulisan kode, integrasi basis data, hingga pengujian dan *debugging* dapat dilakukan secara lebih terstruktur, efisien, dan sesuai dengan praktik pengembangan perangkat lunak di industri. Hal ini selaras dengan tujuan program magang, yaitu memberikan pengalaman nyata dalam rancang bangun sistem *Fleet Management* berbasis web dengan *Flutter* dan *PostgreSQL* di PT. Agrolink Nusantara Indonesia.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan magang berlangsung selama **16 minggu** dengan total waktu mencapai **640 jam**. Selama periode tersebut, penulis terlibat dalam rangkaian kegiatan teknis yang mencakup perencanaan, perancangan, implementasi, hingga pengujian aplikasi *Fleet Management* berbasis *Flutter* dan *Supabase*. Rincian kegiatan ditampilkan pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (16 Minggu)

Minggu ke-	**Pekerjaan yang dilakukan**
1	Mengidentifikasi kebutuhan sistem dan menentukan modul utama yang akan dikembangkan: *Dashboard*, *Fleet*, *Routes*, *Maintenance*, dan *Reports*. Melakukan studi literatur terkait arsitektur aplikasi berbasis *Flutter*.
2	Mempelajari *Flutter Framework* secara mendalam, termasuk *widget tree*, *state management*, serta navigasi multi-layer menggunakan *Navigator 2.0*.
3	Melakukan *setup* lingkungan pengembangan *(Flutter, Dart, Node.js)* serta mempelajari struktur proyek dan standar penulisan kode (*clean architecture*).
4	Mendesain skema database untuk tabel *users*, *drivers*, *vehicles*, *companies*, *trips*, dan *notifications*. Menyusun relasi antar tabel menggunakan *PostgreSQL*.
5	Mendesain antarmuka pengguna di *Figma* untuk modul *Dashboard*, *Fleet*, dan *Routes*. Membuat alur pengguna (*user flow*) untuk proses login, registrasi, dan autentikasi peran pengguna.
6	Mengimplementasikan antarmuka halaman Login, Register, dan OTP Verification dengan alur tiga langkah. Menyusun validasi formulir dan pengelolaan input pengguna.
7	Mengembangkan tampilan Dashboard dan Fleet, termasuk komponen statistik, *recent trips*, dan daftar kendaraan. Menghubungkan komponen UI dengan basis data.
8	Mengembangkan tampilan halaman Routes dan Maintenance serta menambahkan logika navigasi ke halaman detail kendaraan, rute, dan jadwal perawatan.
9	Mengimplementasikan sistem autentikasi menggunakan Supabase Auth, termasuk login email, registrasi, dan manajemen sesi. Melakukan pengujian koneksi *backend*.

Bersambung ke halaman berikutnya

Minggu ke-	Pekerjaan yang dilakukan
10	Mengembangkan CRUD untuk modul Assignment dan Fleet. Membuat fungsi *filtering* dan *sorting* data pada tabel *Fleet* serta memastikan logika *filter* berjalan secara dinamis.
11	Mengonfigurasi Supabase Storage untuk unggah foto profil pengguna dan gambar kendaraan. Mengimplementasikan fitur unggah foto dengan pratinjau sebelum disimpan.
12	Mengembangkan sistem notifikasi untuk pembaruan status kendaraan (Idle, In Transit, Maintenance) serta membuat endpoint untuk pengiriman notifikasi internal.
13	Mengintegrasikan data *real-time* dari Supabase ke halaman Dashboard dan Fleet menggunakan *stream listeners*. Menguji pengambilan data rute dan riwayat perjalanan kendaraan.
14	Melakukan pengujian menyeluruh terhadap modul Dashboard, Fleet, Routes, dan Maintenance. Melakukan perbaikan bug pada logika filter, autentikasi, dan penyimpanan gambar.
15	Melakukan optimasi performa aplikasi, termasuk perbaikan *load time*, optimalisasi query Supabase, serta pengurangan *rebuild widget*.
16	Melakukan finalisasi fitur, dokumentasi teknis, dan penyiapan laporan akhir magang. Menyusun kesimpulan dan rekomendasi untuk pengembangan fitur lanjutan.

3.3.1 Project Utama

Pada proyek utama, penulis bertanggung jawab dalam pengembangan website sistem *Fleet Management*, yang merupakan salah satu produk dalam platform *Agrotara*. Website ini dikembangkan menggunakan bahasa pemrograman *Dart* dengan framework *Flutter* sebagai teknologi *frontend*, serta menggunakan *PostgreSQL* sebagai *database server*. Proses pengembangan dilakukan di lingkungan kerja *Visual Studio Code* dengan penerapan struktur kode yang modular dan terorganisasi.

Website *Fleet Management* dirancang untuk mendukung kebutuhan manajemen armada secara terintegrasi, meliputi pengelolaan data kendaraan, pemantauan aktivitas operasional, serta penyajian informasi armada secara

real-time melalui tampilan antarmuka yang informatif dan mudah digunakan. Dalam pengembangannya, penulis terlibat sejak tahap awal perancangan hingga tahap implementasi, termasuk membangun komponen antarmuka pengguna (*user interface*), mengatur alur navigasi antar halaman, serta memastikan konsistensi tampilan sesuai dengan rancangan *UI/UX* yang telah disediakan oleh tim desain.

Selain pengembangan sisi antarmuka, penulis juga bertanggung jawab pada sisi *backend*, khususnya dalam perancangan dan pengelolaan basis data menggunakan *PostgreSQL*. Tanggung jawab tersebut mencakup pembuatan skema basis data, perancangan relasi antar tabel, serta integrasi data antara *frontend* dan *backend* agar sistem dapat berjalan secara optimal.

3.3.2 Perancangan Sistem

Perancangan sistem fleet manajemen disusun dalam bentuk flowchart yang menjelaskan alur kerja proses secara menyeluruh, serta dilengkapi dengan struktur tabel yang digunakan dalam pengolahan data fleet.

A. Flowchart

Flowchart merupakan diagram yang digunakan untuk merepresentasikan urutan logika atau langkah-langkah dalam suatu proses secara sistematis dan terstruktur. Diagram ini memanfaatkan berbagai simbol grafis untuk menunjukkan aktivitas, keputusan, input, output, maupun proses lain yang terjadi di dalam sistem. Berdasarkan Gambar 3.1, alur sistem dimulai ketika pengguna membuka website dan sistem menampilkan halaman *login*. Pada tahap awal terdapat keputusan apakah pengguna sudah memiliki akun. Jika pengguna telah memiliki akun, pengguna melakukan input *email* dan *password*, kemudian sistem melakukan validasi format input serta autentikasi kredensial. Apabila kredensial valid, sistem membentuk sesi login dan mengarahkan pengguna menuju halaman *dashboard*. Sebaliknya, jika kredensial tidak valid maka sistem menampilkan pesan kesalahan.

Apabila pengguna belum memiliki akun, sistem mengarahkan pengguna ke halaman *register*. Pengguna mengisi data registrasi yang dibutuhkan, lalu sistem melakukan validasi dan menyimpan data tahap awal (*Step 1*). Jika data valid, proses registrasi dilanjutkan ke tahap berikutnya yang mencakup informasi perusahaan (*company*) serta kendaraan/armada (*vehicle/fleet*), hingga akun berhasil dibuat. Setelah akun terbentuk, pengguna diarahkan kembali ke halaman *login*

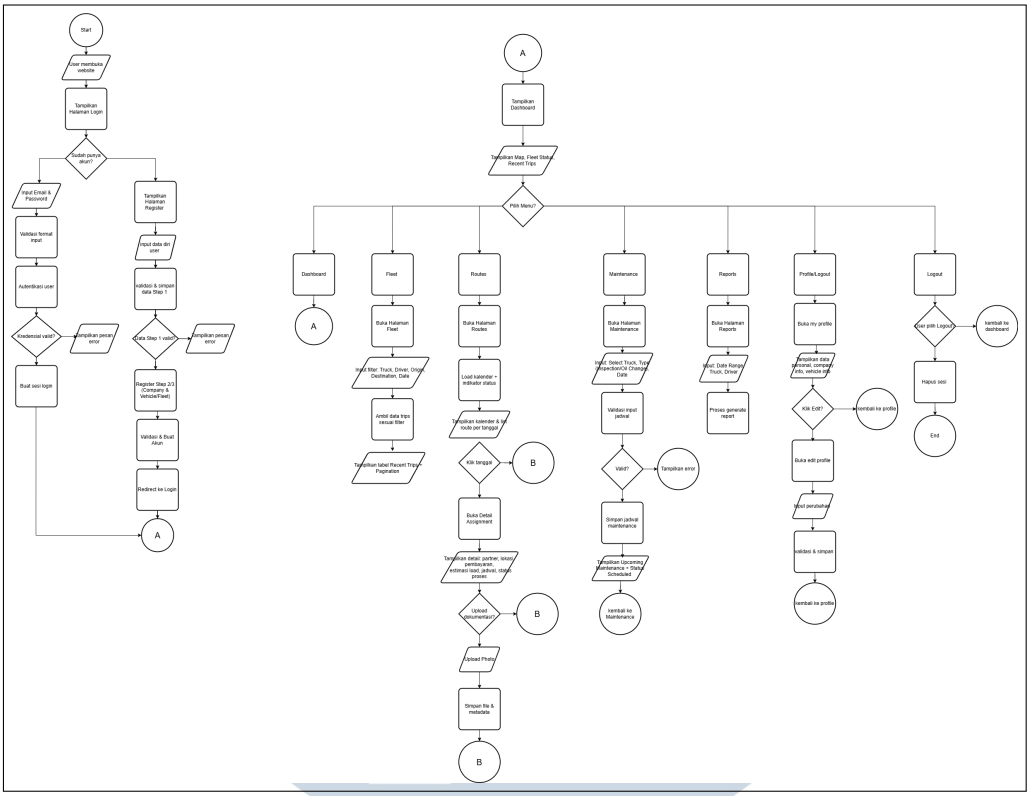
untuk melakukan autentikasi.

Setelah berhasil masuk, sistem menampilkan *dashboard* yang memuat ringkasan informasi operasional, seperti peta lokasi, status armada, dan *recent trips*. Dari dashboard, pengguna dapat memilih menu untuk mengakses modul utama, yaitu *Fleet*, *Routes*, *Maintenance*, *Reports*, serta *Profile*. Pada modul *Fleet*, pengguna dapat memasukkan parameter filter (misalnya *truck*, *driver*, asal, tujuan, dan tanggal) untuk menampilkan daftar perjalanan terbaru beserta navigasi halaman.

Pada modul *Routes*, sistem menampilkan kalender beserta indikator status armada pada tanggal tertentu. Pengguna dapat memilih tanggal untuk melihat rute atau penugasan dan melanjutkan ke halaman *Detail Assignment*. Pada halaman tersebut, sistem menampilkan detail penugasan seperti mitra, lokasi, metode pembayaran, estimasi muatan, jadwal, serta status proses. Pengguna juga dapat mengunggah dokumentasi, kemudian sistem menyimpan berkas dan metadata sebagai bukti aktivitas operasional.

Pada modul *Maintenance*, pengguna dapat menjadwalkan perawatan dengan memilih kendaraan, jenis perawatan, dan tanggal. Sistem memvalidasi input, menyimpan jadwal perawatan, serta menampilkan daftar *upcoming maintenance* dengan status *scheduled*. Sementara itu, modul *Reports* memungkinkan pengguna memilih rentang waktu serta parameter kendaraan atau pengemudi untuk menghasilkan rekapitulasi data dan ringkasan laporan.

Modul *Profile* digunakan untuk menampilkan data profil pengguna dan memberikan fasilitas *edit* profil. Sistem akan memvalidasi perubahan yang dilakukan dan menyimpan data terbaru sebelum kembali menampilkan halaman profil. Proses penggunaan sistem dapat diakhiri melalui fitur *logout* yang akan menghapus sesi pengguna dan mengembalikan tampilan ke kondisi awal.



Gambar 3.1. Flowchart Sistem *Fleet Management*.

B. Skema Basis Data

Bagian ini menjelaskan rancangan struktur basis data yang diterapkan dalam sistem *fleet management* pada PT. Agrolink Nusantara Indonesia. Perancangan basis data disusun untuk mendukung proses pengelolaan armada secara terintegrasi, mulai dari pendataan entitas utama hingga pencatatan aktivitas operasional pengemudi di lapangan. Skema basis data menggunakan pendekatan relasional, di mana setiap tabel memiliki fungsi yang jelas serta dihubungkan melalui relasi kunci primer (*primary key*) dan kunci asing (*foreign key*). Hubungan antar tabel pada sistem ini ditunjukkan pada Gambar 3.2.

Secara umum, struktur basis data dikelompokkan menjadi tabel master dan tabel transaksi. Tabel master menyimpan data dasar yang relatif stabil, meliputi data pengguna dan otorisasi melalui tabel *users*, *roles*, dan *user_roles*. Informasi organisasi dan mitra kerja disimpan pada tabel *companies* dan *partners*. Sumber daya operasional inti direpresentasikan melalui tabel *drivers* (pengemudi) dan *vehicles* (kendaraan), yang kemudian direlasikan dengan tabel *trucks* sebagai identitas armada operasional. Untuk menstandarkan nilai kategori dan mencegah

pengulangan data, sistem juga menyediakan tabel referensi, yaitu *vehicle_types*, *photo_types*, dan *notification_types*.

Tabel transaksi digunakan untuk mencatat aktivitas operasional harian. Pengelolaan rute direpresentasikan melalui tabel *routes*, sedangkan penugasan kerja dicatat pada tabel *assignments* yang terhubung dengan *routes*, *trucks*, *drivers*, dan *partners*. Dokumentasi lapangan pada penugasan dipisahkan ke tabel *assignment_photos*. Realisasi perjalanan direkam pada tabel *trips* untuk menyimpan informasi waktu berangkat, waktu tiba, dan status perjalanan. Pemantauan kondisi armada dilakukan melalui tabel *maintenance*. Kebutuhan ringkasan kinerja direkap pada tabel *reports*, sedangkan penyampaian informasi sistem kepada pengguna difasilitasi melalui tabel *notifications* yang diklasifikasikan berdasarkan *notification_types*.

Pembuktian normalisasi hingga Bentuk Normal Ketiga (3NF) berbasis ketergantungan fungsional.

Skema basis data dirancang mengikuti kaidah normalisasi hingga Bentuk Normal Ketiga (3NF) untuk meminimalkan redundansi serta mencegah anomali *insert*, *update*, dan *delete*. Pembuktian dilakukan melalui evaluasi bentuk normal dan ketergantungan fungsional (*functional dependency*) pada beberapa tabel representatif.

(1) Bentuk Normal Pertama (1NF). Seluruh atribut pada setiap tabel bersifat atomik (satu kolom menyimpan satu nilai) dan tidak terdapat *repeating group*. Relasi bernilai jamak dimodelkan sebagai tabel terpisah, misalnya: (i) relasi banyak-ke-banyak pengguna–peran melalui *user_roles*; (ii) dokumentasi foto pengemudi pada *driver_photos*; dan (iii) dokumentasi foto penugasan pada *assignment_photos*. Dengan pemisahan ini, tidak ada atribut yang menyimpan daftar nilai dalam satu kolom.

(2) Bentuk Normal Kedua (2NF). Setiap tabel menggunakan kunci utama tunggal berupa *id* (UUID). Karena tidak terdapat kunci utama komposit, maka ketergantungan parsial (*partial dependency*) tidak terjadi. Secara formal, untuk setiap tabel T, seluruh atribut non-kunci A memenuhi $id \rightarrow A$. Contoh ketergantungan fungsional ditunjukkan pada Tabel 3.2.

Tabel	Functional Dependency (FD)
vehicles	`vehicles.id -> {brand, type_id, plate_number, driver_id, company_id, created_at, updated_at}`
routes	`routes.id -> {origin, destination, route_date, truck_id, driver_id, status, created_at}`

Tabel 3.2. Contoh ketergantungan fungsional (2NF) pada tabel representatif.

Dengan demikian, seluruh atribut non-kunci bergantung penuh pada kunci utama tabel masing-masing.

(3) Bentuk Normal Ketiga (3NF). Syarat 3NF menyatakan tidak boleh ada ketergantungan transitif, yaitu atribut non-kunci bergantung pada atribut nonkunci lain. Pada skema ini, atribut-atribut non-kunci dirancang agar bergantung langsung pada kunci utama tabel, sementara informasi deskriptif entitas lain dirujuk melalui *foreign key*. Pembuktian ditunjukkan melalui beberapa contoh berikut.

a) Normalisasi otorisasi pengguna: *users–roles–user_roles*. Pada tabel *users*, atribut yang disimpan hanya identitas pengguna; sedangkan atribut peran disimpan pada *roles*. Relasi ditangani oleh *user_roles*. Contoh *Functional Dependency*:

- $users.id \rightarrow \{full\_name, phone\_number, created\_at, updated\_at\}$.
- $roles.id \rightarrow \{role\_name\}$ (dengan *unique constraint* pada *role_name*).
- $user\_roles.id \rightarrow \{user\_id, role\_id\}$.

Jika *role_name* disimpan langsung pada *users*, maka terjadi redundansi dan anomali pembaruan (misalnya perubahan nama peran harus memperbarui banyak baris). Dengan pemisahan ini, atribut non-kunci pada *users* tidak bergantung pada atribut non-kunci lain.

b) Normalisasi kategori melalui tabel referensi. Nilai kategori yang berulang distandardisasi melalui tabel referensi: *vehicle_types*, *photo_types*, dan *notification_types*. Contoh *Functional Dependency*:

- $vehicle\_types.id \rightarrow \{name\}$, dan pada *vehicles* hanya disimpan $type\_id$.
- $photo\_types.id \rightarrow \{name\}$, dan pada *driver_photos* hanya disimpan $photo\_type\_id$.

Pola ini mencegah ketergantungan transitif seperti *vehicles.id* $\rightarrow$ *type_id* $\rightarrow$ *vehicle_types.name* di dalam tabel yang sama, karena *name* tidak disalin ke *vehicles*. Dengan demikian, atribut deskriptif kategori tidak menjadi atribut non-kunci pada tabel transaksi/master yang tidak semestinya.

c) Tabel transaksi tidak menyalin atribut turunan dari master. Sebagai contoh, tabel *assignments* menyimpan relasi penugasan melalui *foreign key* tanpa menyalin nama pengemudi, kode truk, atau nama mitra. Ketergantungan fungsionalnya dapat dituliskan sebagai berikut.

Tabel	**Functional Dependency (FD)**
assignments	`assignments.id -> {route_id, truck_id, driver_id, partner_id, working_description, department, duration_hours, plantation_location, payment_method, estimated_load_kg, actual_load_ton, scheduled_time_start, scheduled_time_end, assignment_status, created_at}`

Tabel 3.3. Contoh ketergantungan fungsional pada tabel *assignments*.

Jika tabel *assignments* menyimpan atribut seperti *partner_name* atau *truck_code*, maka akan muncul ketergantungan transitif dan redundansi, misalnya `assignments.id -> partner_id -> partners.partner_name` yang seharusnya diperoleh melalui operasi *join*, bukan disimpan ulang. Desain saat ini memastikan seluruh atribut non-kunci pada *assignments* bergantung langsung pada *assignments.id*.

d) Pemisahan identitas armada: *vehicles* dan *trucks*. Pemodelan memisahkan data kendaraan (atribut fisik/registrasi) pada *vehicles* dari identitas armada operasional pada *trucks* (misalnya *truck_code*). Contoh FD:

- $vehicles.id \rightarrow \{plate\_number, brand, type\_id, driver\_id, company\_id\}$.
- $trucks.id \rightarrow \{vehicles\_id, truck\_code\}$ (dengan *unique constraint* pada *truck_code*).

Tabel transaksi (*routes*, *assignments*, *trips*, *maintenance*) merujuk *truck_id* agar identitas armada konsisten dan tidak terjadi duplikasi *truck_code* atau *plate_number* pada tabel transaksi.

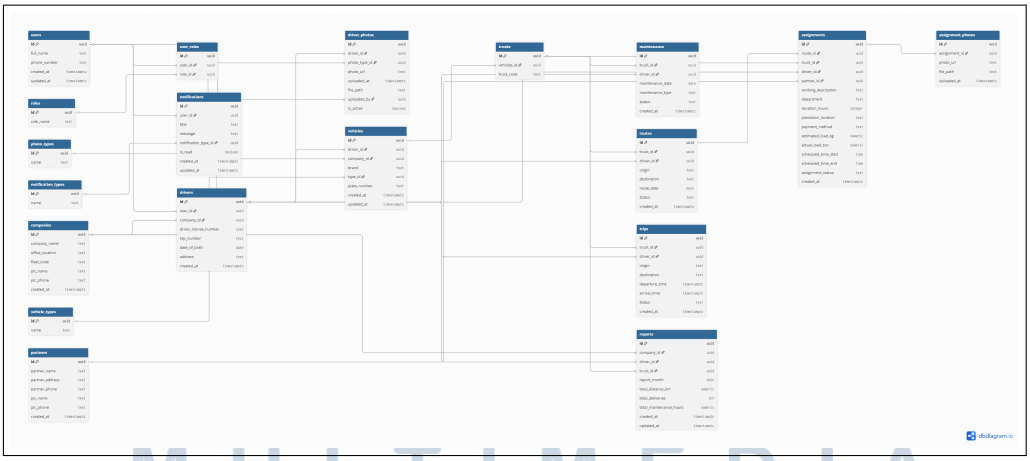
e) Ringkasan laporan sebagai data agregat terpisah. Tabel *reports* berisi hasil agregasi untuk kebutuhan analitik periodik, misalnya:

- $reports.id \rightarrow \{company\_id, driver\_id, truck\_id, report\_month, total\_distance\_km, total\_deliveries, total\_maintenance\_hours, created\_at, updated\_at\}$.

Pemisahan ini menjaga tabel transaksi tetap menyimpan data rinci, sedangkan tabel laporan menyimpan nilai ringkasan untuk kebutuhan pelaporan. Dengan relasi berbasis *foreign key*, sistem menjaga integritas referensial sekaligus menghindari pengulangan data deskriptif.

Selain itu, penerapan *unique constraint* pada atribut tertentu (misalnya *role_name*, *fleet_code*, *plate_number*, dan *truck_code*) bertindak sebagai kunci kandidat untuk menjamin keunikan data serta memperkuat integritas tanpa menambah redundansi.

Berdasarkan evaluasi bentuk normal dan contoh ketergantungan fungsional di atas, skema basis data memenuhi 3NF karena: (i) setiap atribut bersifat atomik (1NF), (ii) tidak terdapat ketergantungan parsial karena kunci utama tunggal (2NF), dan (iii) tidak terdapat ketergantungan transitif pada atribut non-kunci karena atribut deskriptif dipisahkan ke tabel master/referensi dan dihubungkan melalui *foreign key* (3NF). Dengan demikian, rancangan basis data mendukung integritas referensial, meminimalkan redundansi, dan memudahkan pengolahan serta penyajian informasi operasional.



Gambar 3.2. Skema Basis Data Sistem *Fleet Management*.

3.3.3 Proses Slicing UI dari Figma ke Flutter

Salah satu tahapan krusial dalam pengembangan antarmuka pengguna adalah proses yang dikenal sebagai *slicing*, yaitu menerjemahkan desain visual

yang dibuat dengan perangkat seperti Figma menjadi komponen antarmuka aplikasi dalam bentuk implementasi kode program. Proses ini menjembatani tahap perancangan dan implementasi teknis, sehingga antarmuka yang dihasilkan tidak hanya memiliki tampilan yang menarik, tetapi juga fungsional dan siap diterapkan dalam aplikasi.

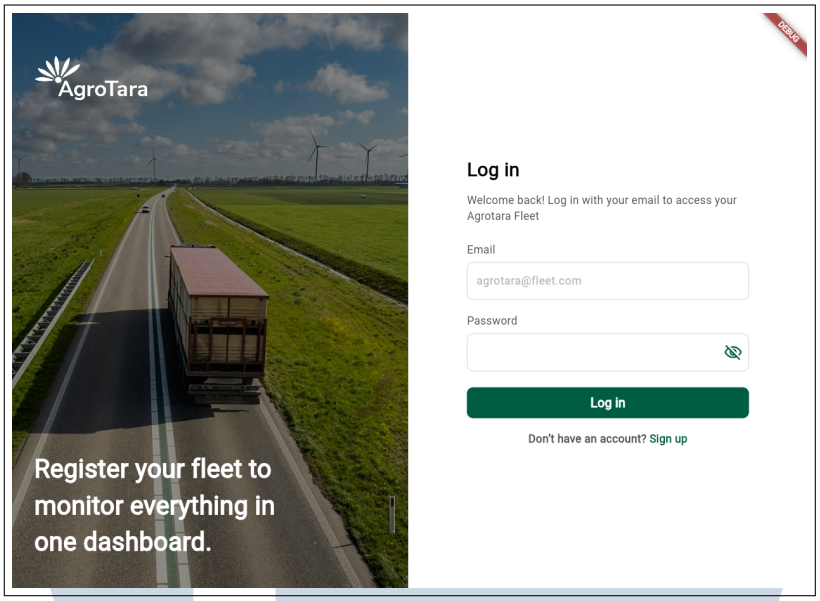
Menurut Białkowski dan Smołka [8], proses *slicing* antarmuka ke dalam *framework* seperti Flutter dinilai lebih efisien dari sisi waktu pengerjaan maupun performa dibandingkan pendekatan tradisional. Hal ini karena Flutter menyediakan arsitektur berbasis *widget* yang sesuai untuk merepresentasikan komponen visual pada desain digital. Pada praktiknya, *slicing* dilakukan dengan mengidentifikasi elemen desain pada Figma misalnya tombol, ikon, dan teks kemudian menerjemahkannya ke dalam *widget* Flutter seperti `Text`, `Container`, `Row`, `Column`, `Stack`, dan `Image`.

Selain itu, menurut penelitian yang dilakukan oleh Li et al. [9], desain antarmuka pengguna yang baik harus diintegrasikan ke dalam pola kode yang konsisten agar pengalaman pengguna tetap optimal selama transisi dari prototipe ke aplikasi. Selain itu, proses slicing ini mempertimbangkan layout, responsivitas, dan aksesibilitas.

Seluruh antarmuka pengguna awalnya dirancang dalam Figma saat mengembangkan sistem *Fleet Management*. Tim pengembang melakukan slicing, yang mencerminkan setiap bagian desain ke dalam struktur kode Flutter. Misalnya, halaman login yang berisi kolom input dan tombol login dengan menggunakan kombinasi `Form`, `TextField`, dan `ElevatedButton`. Hasil *slicing* memastikan bahwa situs web tidak hanya selaras dengan rancangan desain, tetapi juga menampilkan tampilan serta performa yang konsisten pada berbagai jenis perangkat.

A. Hasil Slicing Tampilan Login

Tampilan *Login* dirancang sebagai gerbang akses utama ke sistem *Fleet Management*. Pada halaman ini, pengguna diminta memasukkan *email* dan *password* untuk proses autentikasi. Selain itu, sistem menyediakan tombol **Log in** sebagai aksi utama serta tautan *Sign up* bagi pengguna yang belum memiliki akun.

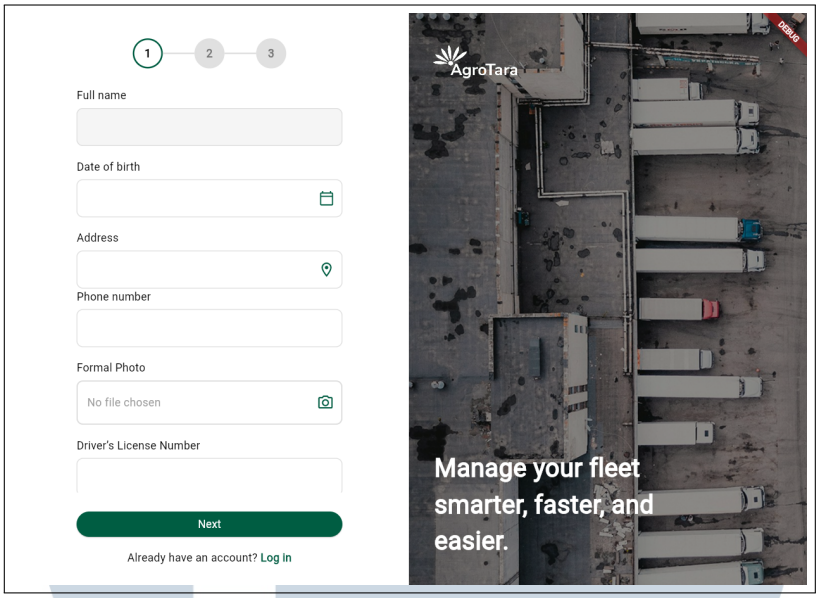


Gambar 3.3. Tampilan *Login*.

Gambar 3.3 menampilkan antarmuka *login* dengan struktur yang sederhana dan fokus pada kebutuhan autentikasi, sehingga memudahkan pengguna untuk melakukan akses secara cepat dan konsisten.

B. Hasil Slicing Tampilan Register

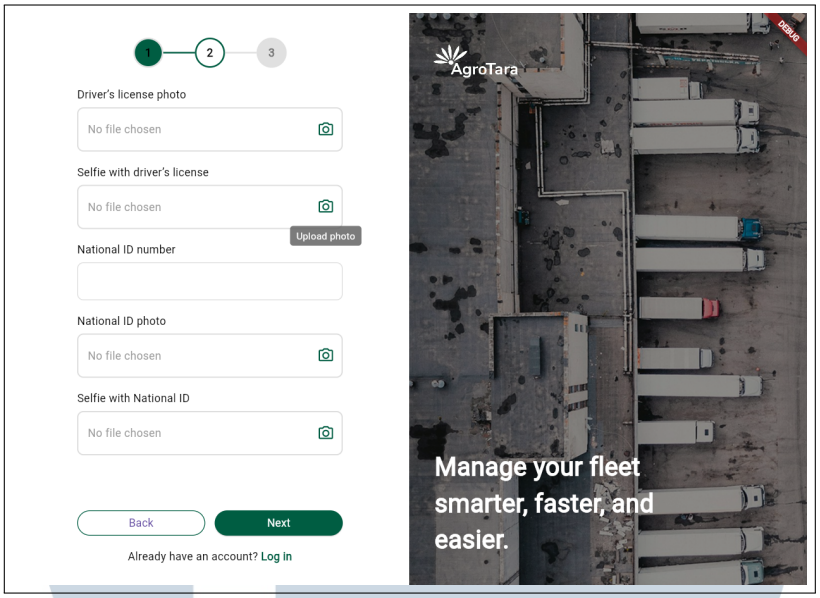
Tampilan *Register Step 1* digunakan untuk mengumpulkan informasi dasar pengguna/driver sebagai data identitas awal. Data yang diinput meliputi nama lengkap, tanggal lahir, alamat, nomor telepon, unggah foto formal, serta nomor SIM. Halaman ini juga dilengkapi indikator tahapan (1-3) dan tombol **Next** untuk melanjutkan ke tahap berikutnya setelah validasi input terpenuhi.



Gambar 3.4. Tampilan Registrasi Tahap Pertama.

Gambar 3.4 memperlihatkan rancangan form yang ringkas dan terstruktur, sehingga membantu pengguna melengkapi data personal secara sistematis sebelum masuk ke proses verifikasi lanjutan.

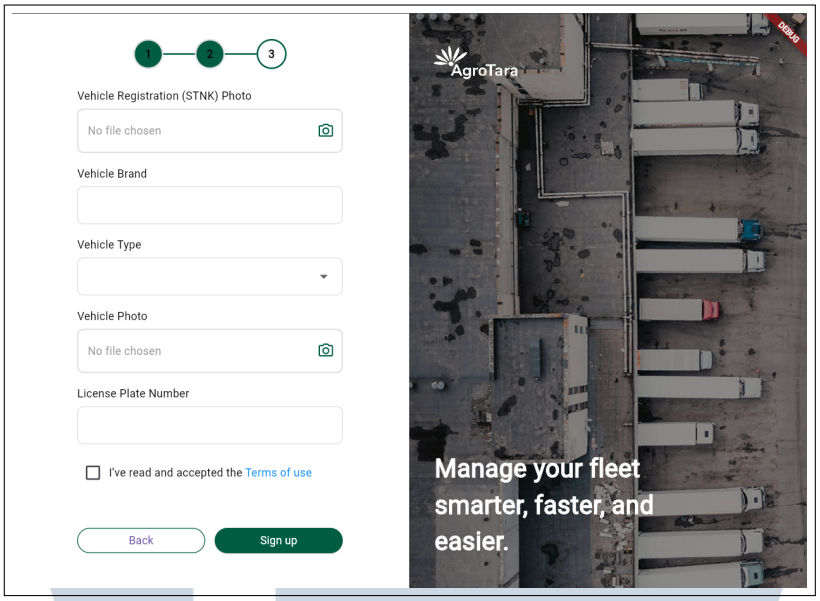
Tampilan *Register Step 2* berfokus pada proses verifikasi identitas pengguna/driver untuk meningkatkan validitas data. Pada tahap ini, pengguna diminta mengunggah foto SIM, *selfie* dengan SIM, mengisi nomor identitas nasional (NIK), mengunggah foto KTP, serta *selfie* dengan KTP. Tombol **Next** digunakan untuk melanjutkan proses setelah sistem melakukan validasi kelengkapan data.



Gambar 3.5. Tampilan Registrasi Tahap Kedua.

Berdasarkan Gambar 3.5, penyusunan komponen input dan unggah dokumen dirancang agar pengguna dapat memenuhi kebutuhan verifikasi secara bertahap, sekaligus meminimalkan kesalahan pengisian.

Tampilan *Register Step 3* digunakan untuk melengkapi data kendaraan yang akan dikelola dalam sistem. Informasi yang dikumpulkan meliputi unggah foto STNK, merek kendaraan, jenis kendaraan (melalui *dropdown*), unggah foto kendaraan, serta nomor polisi. Pada tahap akhir, pengguna diwajibkan menyetujui *Terms of use* sebelum menekan tombol **Sign up** sebagai konfirmasi pembuatan akun.

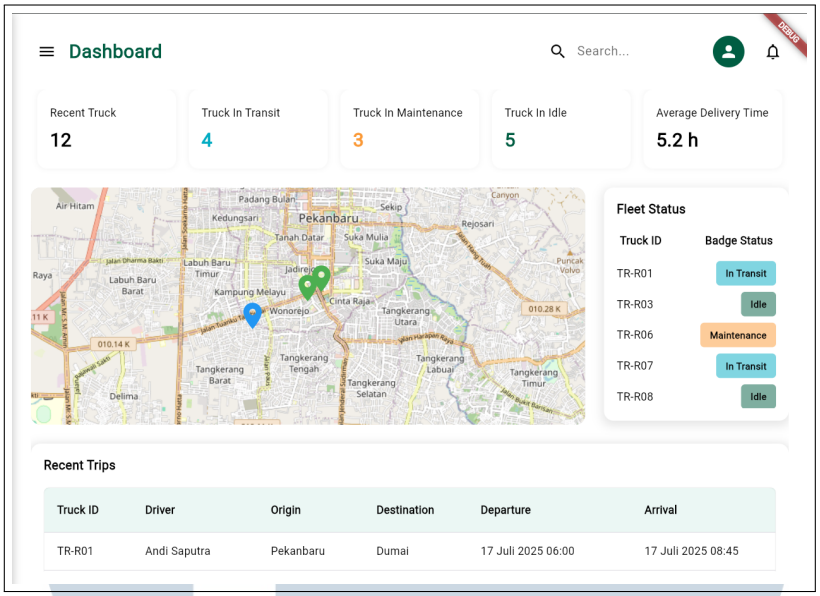


Gambar 3.6. Tampilan Registrasi Tahap Ketiga.

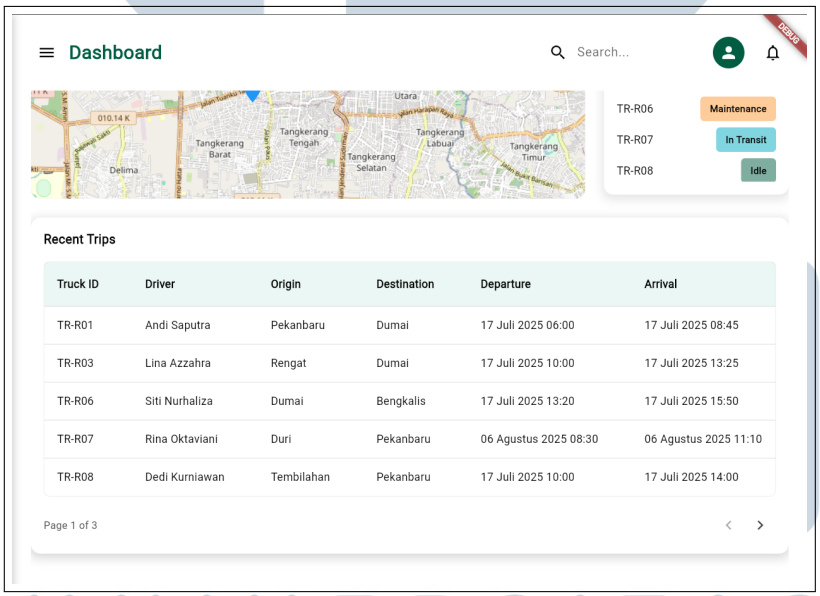
Gambar 3.6 menunjukkan bahwa proses pendaftaran dibangun secara berurutan agar data pengguna dan kendaraan tervalidasi dengan baik sebelum akun diaktifkan dan digunakan pada modul operasional sistem.

C. Hasil Slicing Tampilan Dashboard

Tampilan *Dashboard* berfungsi sebagai halaman utama untuk memantau kondisi armada secara ringkas dan terpusat. Pada bagian atas, sistem menampilkan kartu ringkasan (*summary cards*) yang memuat indikator operasional, seperti jumlah armada terbaru, armada dalam perjalanan (*in transit*), armada dalam perawatan (*maintenance*), armada dalam kondisi *idle*, serta rata-rata waktu pengiriman (*average delivery time*). Selain itu, tersedia komponen peta untuk visualisasi lokasi armada dan tabel *Fleet Status* yang menampilkan status setiap unit kendaraan melalui *badge*. Bagian *Recent Trips* disajikan dalam bentuk tabel yang berisi informasi perjalanan terbaru, meliputi *Truck ID*, nama pengemudi, asal, tujuan, waktu keberangkatan, dan waktu tiba, serta dilengkapi navigasi halaman (*pagination*) untuk memudahkan penelusuran data.



Gambar 3.7. Tampilan *Dashboard*.

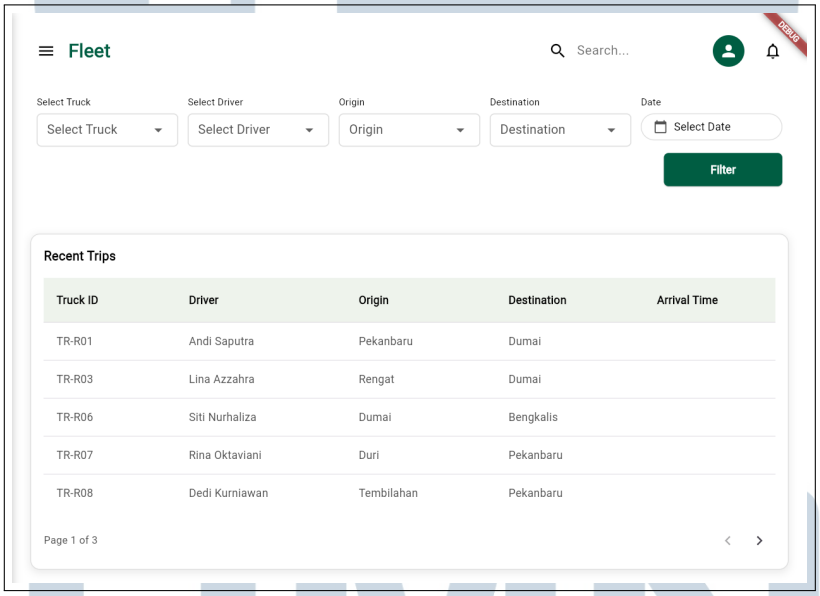


Gambar 3.8. Tampilan *Dashboard*.

Berdasarkan Gambar 3.7, penyajian informasi pada dashboard disusun secara hierarkis untuk mempercepat identifikasi kondisi armada dan aktivitas operasional terkini, sehingga mendukung proses monitoring dan pengambilan keputusan secara lebih efektif.

D. Hasil Slicing Tampilan Fleet

Tampilan *Fleet* dirancang untuk menampilkan data perjalanan armada secara lebih spesifik melalui fitur penyaringan (*filtering*). Pengguna dapat memilih parameter seperti kendaraan (*Select Truck*), pengemudi (*Select Driver*), asal (*Origin*), tujuan (*Destination*), dan tanggal (*Date*). Setelah parameter dipilih, pengguna menekan tombol **Filter** untuk menampilkan hasil sesuai kriteria. Hasil penyaringan ditampilkan pada tabel *Recent Trips* yang memuat identitas armada, pengemudi, asal, tujuan, dan waktu tiba, serta dilengkapi *pagination* untuk memfasilitasi penelusuran data.



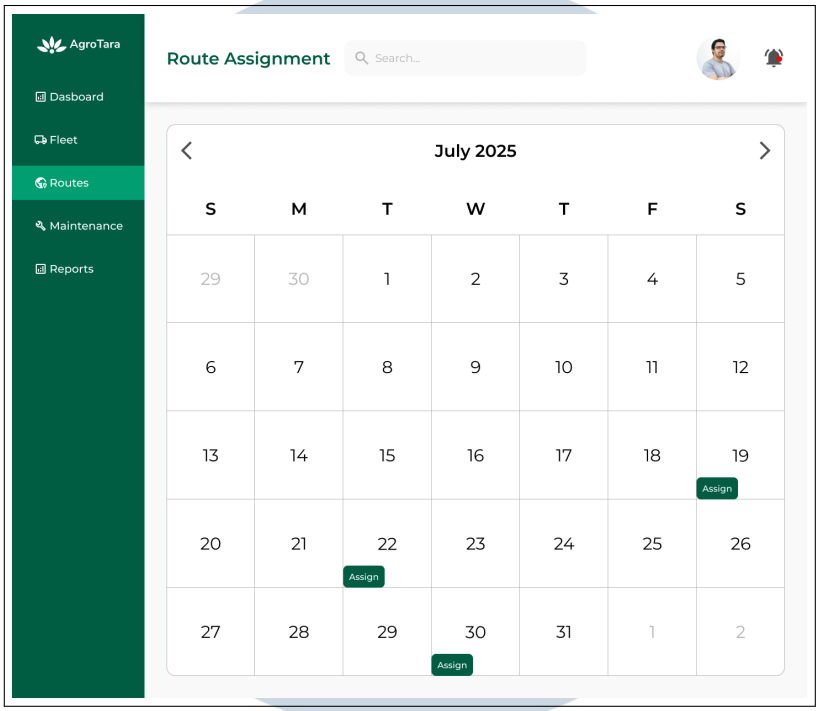
Gambar 3.9. Tampilan *Fleet*.

Berdasarkan Gambar 3.9, modul *Fleet* memudahkan pengguna dalam melakukan pencarian data perjalanan berdasarkan atribut tertentu, sehingga proses monitoring dan penelusuran histori perjalanan armada menjadi lebih efektif dan efisien.

E. Hasil Slicing Tampilan Routes

Tampilan *Routes* dirancang untuk membantu pengguna memantau aktivitas armada berbasis kalender. Halaman ini menampilkan kalender bulanan yang dilengkapi indikator status (misalnya *in transit*, *idle*, dan *maintenance*) pada tanggal tertentu. Pada bagian bawah, sistem menampilkan daftar rute/aktivitas pada tanggal

yang dipilih, termasuk informasi *Truck ID*, nama pengemudi, arah perjalanan (asal-tujuan), serta *badge* status untuk memudahkan identifikasi kondisi armada.

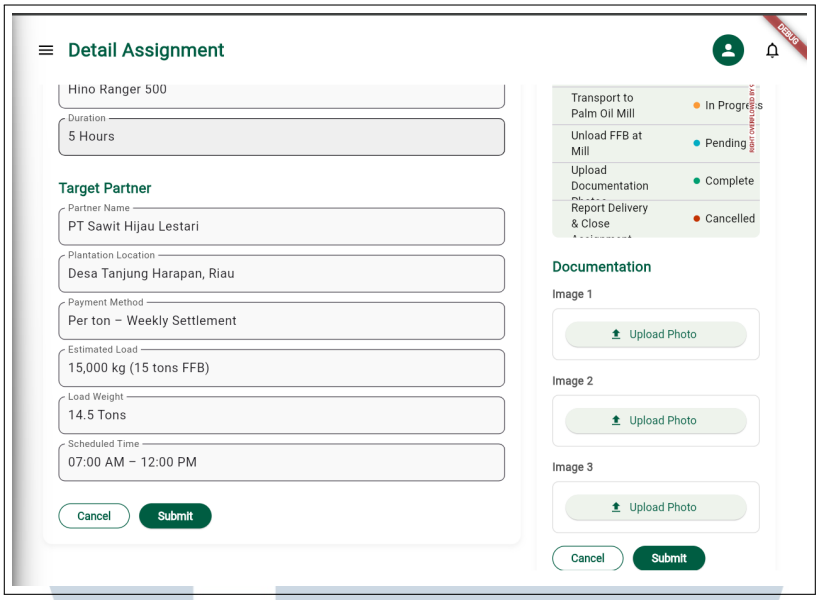


Gambar 3.10. Tampilan *Routes*.

Berdasarkan Gambar 3.10, visualisasi kalender memberikan gambaran cepat mengenai distribusi aktivitas armada dalam satu periode, sehingga mendukung perencanaan dan pengawasan operasional.

F. Hasil Slicing Tampilan Detail Assignment

Tampilan *Detail Assignment* berfungsi untuk menampilkan rincian penugasan secara lengkap, mulai dari informasi kendaraan, durasi, hingga data mitra tujuan. Informasi mitra (*Target Partner*) mencakup nama mitra, lokasi perkebunan, metode pembayaran, estimasi muatan, berat muatan, serta jadwal pelaksanaan. Selain itu, halaman ini menyediakan komponen *Documentation* untuk unggah bukti pelaksanaan (foto) pada beberapa slot, serta menampilkan tahapan proses penugasan beserta statusnya untuk mendukung pemantauan progres.

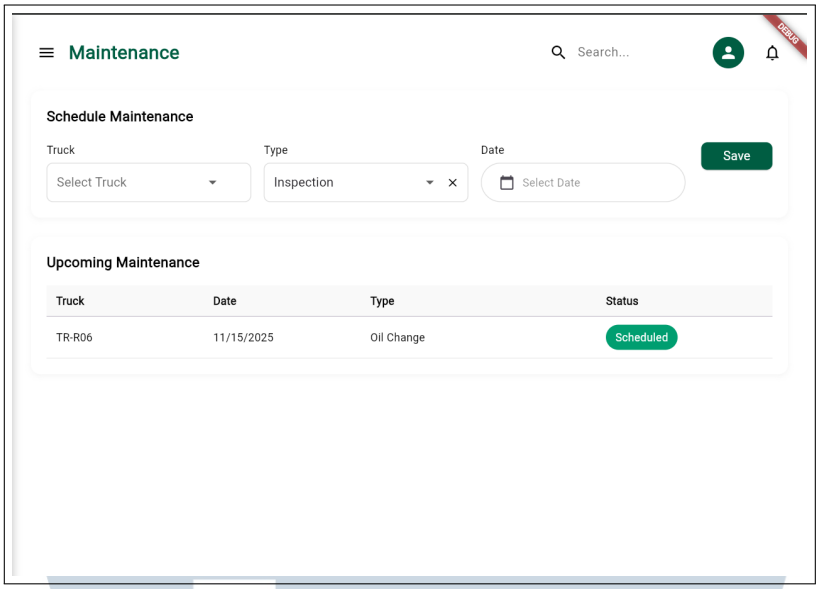


Gambar 3.11. Tampilan *Assingment*.

Gambar 3.11 menunjukkan bahwa halaman ini dirancang sebagai pusat informasi penugasan, sehingga pengguna dapat melakukan verifikasi data dan pendokumentasian aktivitas operasional secara terintegrasi.

G. Hasil Slicing Tampilan Maintenance

Tampilan *Maintenance* dirancang untuk mengelola penjadwalan perawatan kendaraan secara terstruktur. Pada bagian *Schedule Maintenance*, pengguna dapat memilih kendaraan, jenis perawatan, dan tanggal pelaksanaan, kemudian menyimpan jadwal melalui tombol **Save**. Selanjutnya, sistem menampilkan daftar *Upcoming Maintenance* yang memuat informasi kendaraan, tanggal, jenis perawatan, serta status penjadwalan (*scheduled*) sebagai bentuk kontrol terhadap rencana perawatan armada.



Gambar 3.12. Tampilan *Maintenance*.

Gambar 3.12 menunjukkan bahwa modul *maintenance* membantu memastikan perawatan armada terdokumentasi dan terjadwal, sehingga mendukung keandalan kendaraan serta menekan risiko gangguan operasional.

H. Hasil Slicing Tampilan Reports

Tampilan *Reports* merupakan modul analitik operasional yang menyajikan ringkasan kinerja armada dalam bentuk indikator statistik (kartu metrik) dan visualisasi tren (grafik). Informasi pada modul ini dibentuk melalui alur data *end-to-end* yang melibatkan tiga komponen utama, yaitu *admin* sebagai penginisiasi penugasan, *driver* sebagai pelaksana sekaligus penyedia dokumentasi lapangan, serta *system* sebagai komponen yang melakukan pencatatan, konsolidasi, dan agregasi data menjadi keluaran laporan.

Alur pembentukan data laporan. Pertama, *admin* menghasilkan data operasional dengan menyusun rute dan membuat penugasan melalui modul *Routes*. Proses ini tersimpan pada basis data melalui tabel *routes* dan *assignments*. Pada tahap ini, *admin* menetapkan atribut inti penugasan, antara lain *truck_id*, *driver_id*, *partner_id*, jadwal pelaksanaan (*scheduled_time_start* dan *scheduled_time_end*), serta parameter pendukung lainnya (misalnya *department*, *plantation_location*, *estimated_load_kg*, dan *actual_load_ton*). Setelah data tersimpan, penugasan akan tersedia pada daftar *Routes* dan dapat diakses oleh *driver* sesuai mekanisme otorisasi sistem.

Kedua, *driver* melakukan pembaruan pada bagian yang menjadi kewenangannya, terutama terkait dokumentasi lapangan dan pembaruan status pelaksanaan. Dokumentasi aktivitas diunggah dan disimpan pada tabel *assignment_photos* yang berelasi langsung dengan *assignments*. Pembaruan status (misalnya progres penugasan) berfungsi untuk menjaga keterlacakan (*traceability*) dan konsistensi rekam jejak aktivitas operasional.

Ketiga, *system* melakukan konsolidasi data operasional dengan mencatat realisasi perjalanan pada tabel *trips* serta aktivitas perawatan pada tabel *maintenance*. Seluruh data tersebut kemudian direkap menjadi ringkasan periodik pada tabel *reports*. Dengan demikian, halaman *Reports* tidak menampilkan data mentah, melainkan menampilkan hasil olahan (*aggregated view*) yang telah terintegrasi melalui relasi kunci primer–kunci asing pada entitas *truck_id*, *driver_id*, dan *company_id*.

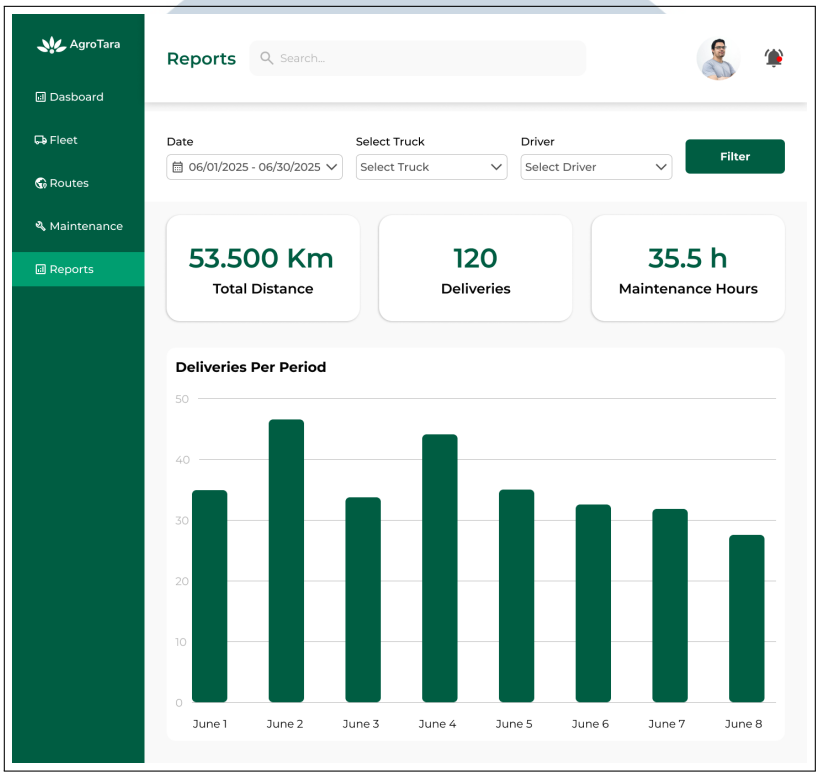
Mekanisme statistik dan visualisasi. Pada halaman *Reports*, pengguna (khususnya *admin*) dapat menerapkan *filtering* berdasarkan rentang tanggal, kendaraan (*truck*), dan pengemudi (*driver*). Parameter ini membatasi himpunan data yang diproses sehingga indikator yang ditampilkan merepresentasikan periode dan entitas yang dipilih. Setelah *filter* diterapkan, sistem melakukan komputasi agregat untuk menghasilkan:

- *Total Distance*: akumulasi jarak tempuh pada periode terpilih.
- *Deliveries*: jumlah aktivitas pengiriman/perjalanan yang tercatat pada periode terpilih.
- *Maintenance Hours*: akumulasi durasi perawatan armada pada periode terpilih.

Selain metrik agregat, sistem membentuk grafik *Monthly Summary* melalui agregasi periodik per bulan, sehingga tren performa dapat diamati dari waktu ke waktu. Dengan pendekatan ini, grafik berperan sebagai instrumen analisis tren dan bukan sekadar tampilan data statis, karena nilai yang ditampilkan mengikuti parameter *filter* yang digunakan.

Implikasi terhadap evaluasi operasional. Rancangan modul *Reports* memastikan bahwa keluaran statistik bersifat terukur, konsisten, dan dapat ditelusuri (*auditable*). Setiap nilai pada kartu metrik dan grafik memiliki keterkaitan langsung dengan transaksi operasional pada tabel *assignments*, realisasi perjalanan pada *trips*, serta aktivitas perawatan pada *maintenance*. Dengan demikian, modul

ini mendukung evaluasi kinerja operasional dan pengambilan keputusan berbasis data melalui penyajian informasi ringkas (indikator) dan analisis tren (grafik) yang sesuai dengan data operasional sistem.

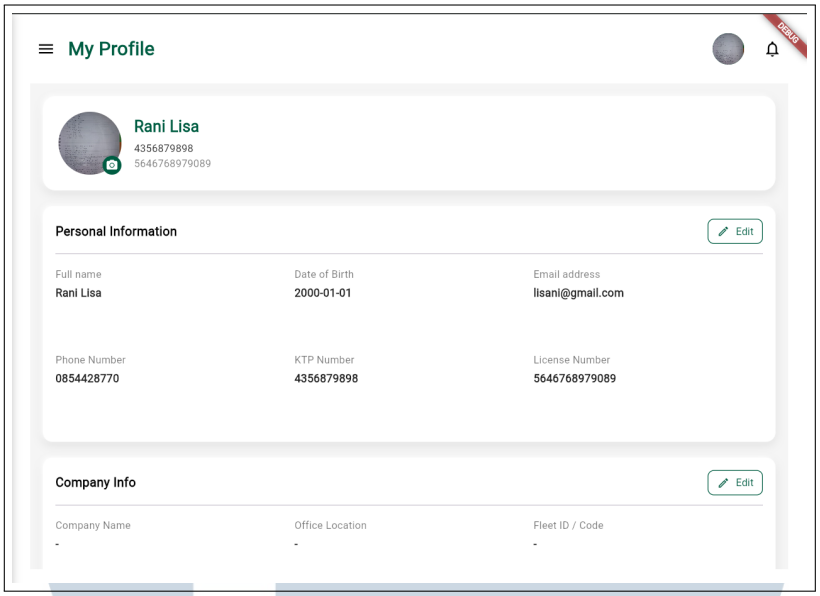


Gambar 3.13. Tampilan *Reports*.

Berdasarkan Gambar 3.13, modul *Reports* memfasilitasi pemantauan kinerja armada melalui penyajian indikator dan visualisasi yang informatif, sehingga data operasional dapat diinterpretasikan secara cepat dan mendukung proses evaluasi secara periodik.

I. Hasil Slicing Tampilan Profile

Tampilan *My Profile* berfungsi untuk menampilkan informasi akun pengguna secara terstruktur, mencakup data personal (nama lengkap, tanggal lahir, email, nomor telepon, nomor KTP, dan nomor SIM) serta informasi perusahaan (*company info*) apabila tersedia. Halaman ini juga menyediakan tombol **Edit** yang digunakan untuk melakukan pembaruan data ketika diperlukan.

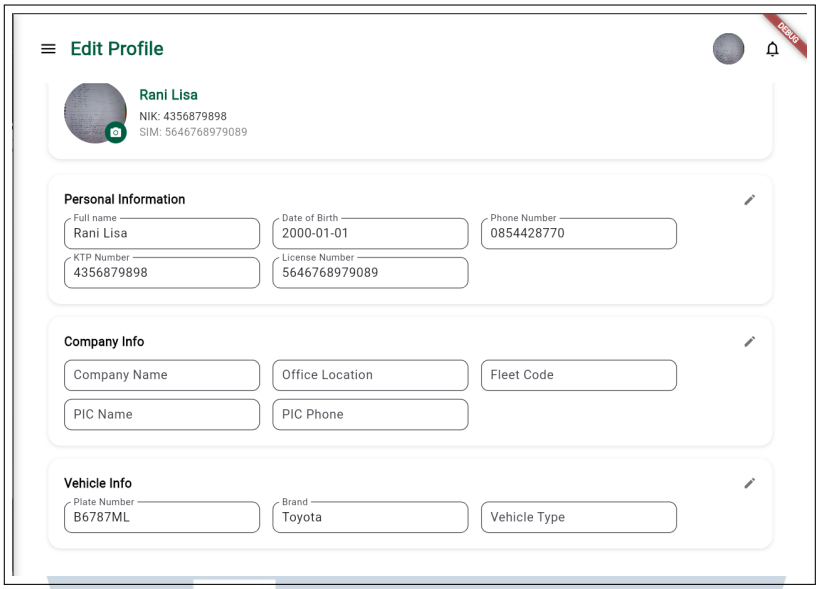


Gambar 3.14. Tampilan *Profile User*.

Gambar 3.14 menunjukkan bahwa informasi disajikan dalam bentuk blok kategori untuk meningkatkan keterbacaan dan memudahkan pengguna memverifikasi data yang tersimpan pada sistem.

J. Hasil Slicing Tampilan Edit Profile

Tampilan *Edit Profile* dirancang untuk memfasilitasi pembaruan data pengguna secara mandiri. Form pengeditan dikelompokkan berdasarkan kategori, yaitu *Personal Information*, *Company Info*, dan *Vehicle Info*. Pengguna dapat memperbarui data pada masing-masing kategori sesuai kebutuhan, kemudian sistem melakukan validasi sebelum menyimpan perubahan agar konsistensi dan keakuratan data tetap terjaga.

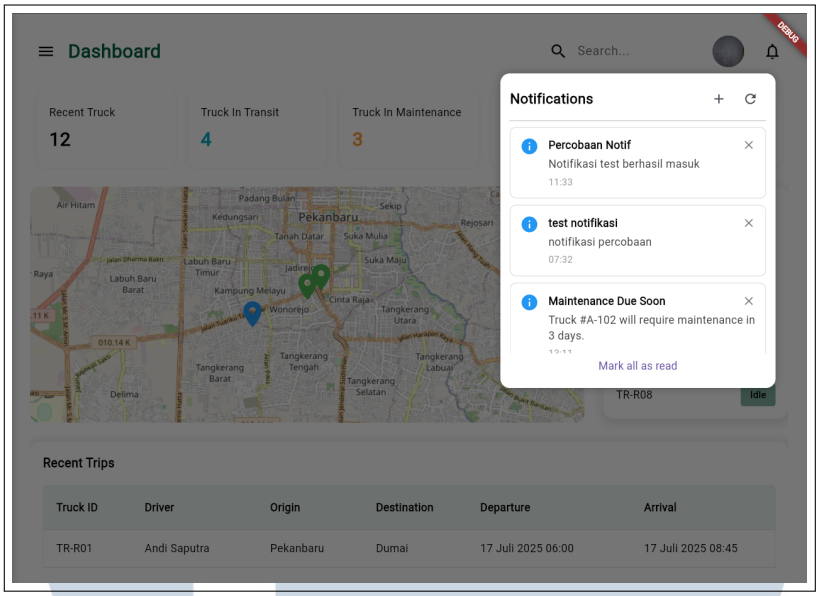


Gambar 3.15. Tampilan *Edit Profile*.

Berdasarkan Gambar 3.15, rancangan form yang terstruktur membantu pengguna memperbarui informasi secara lebih efisien dan mengurangi potensi kesalahan pengisian.

K. Hasil Slicing Tampilan Notification

Tampilan Notifikasi dirancang untuk menyajikan informasi pembaruan sistem secara *real-time* kepada pengguna, seperti notifikasi uji coba maupun peringatan operasional (contohnya pengingat jadwal perawatan yang akan segera jatuh tempo). Notifikasi ditampilkan dalam bentuk daftar (*list*) yang memuat judul, deskripsi singkat, dan waktu notifikasi. Selain itu, sistem menyediakan fitur pengelolaan notifikasi, seperti menandai semua notifikasi sebagai telah dibaca (*mark all as read*) serta menghapus notifikasi tertentu.



Gambar 3.16. Tampilan Notifikasi

Berdasarkan Gambar 3.16, fitur notifikasi membantu pengguna memperoleh informasi penting secara cepat sehingga mendukung respons operasional dan pengambilan keputusan yang lebih tepat waktu.

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kegiatan magang yang dilaksanakan secara daring, penulis menghadapi beberapa kendala yang memengaruhi kelancaran dan efektivitas proses kerja. Kendala tersebut mencakup aspek teknis maupun non-teknis, yang selanjutnya diatasi melalui berbagai upaya penyesuaian dan strategi pemecahan masalah. Adapun uraian kendala dan solusi yang ditempuh selama kegiatan magang adalah sebagai berikut.

Kendala Teknis

1. **Adaptasi terhadap penggunaan *framework* Flutter.**
   Penggunaan *framework* Flutter merupakan pengalaman pertama bagi penulis dalam mengembangkan antarmuka aplikasi berbasis web. Pada tahap awal magang, penulis mengalami kesulitan dalam memahami struktur widget, konsep pemrograman deklaratif, serta pengelolaan tata letak dan navigasi antarkomponen. Kondisi ini berdampak pada proses *slicing* antarmuka dan pengembangan fitur yang memerlukan waktu relatif lebih lama.

Solusi: Untuk mengatasi kendala tersebut, penulis mempelajari dokumentasi resmi Flutter secara mandiri, mengikuti tutorial dan referensi daring yang relevan, serta melakukan diskusi dengan pembimbing dan tim teknis perusahaan. Melalui proses pembelajaran berkelanjutan dan praktik langsung, penulis mampu meningkatkan pemahaman terhadap Flutter sehingga proses pengembangan antarmuka dapat berjalan lebih efektif pada tahap selanjutnya.

2. **Adaptasi terhadap penggunaan platform Supabase.**
   Penulis juga menghadapi kendala dalam beradaptasi dengan penggunaan Supabase sebagai platform *Backend-as-a-Service*, khususnya dalam memahami konsep autentikasi, pengelolaan basis data berbasis *PostgreSQL*, serta integrasi antara frontend Flutter dan layanan backend. Kurangnya pengalaman sebelumnya dalam menggunakan Supabase menyebabkan proses integrasi sistem memerlukan waktu penyesuaian.

   Solusi: Penulis mengatasi kendala ini dengan mempelajari dokumentasi resmi Supabase, melakukan eksplorasi fitur secara bertahap, serta menerapkan pendekatan *trial and error* dalam lingkungan pengembangan. Selain itu, penulis memanfaatkan diskusi dengan supervisor untuk memastikan implementasi backend dan basis data telah sesuai dengan kebutuhan sistem dan standar perusahaan.

3. **Keterbatasan koneksi internet selama pelaksanaan magang daring.**
   Pelaksanaan magang secara daring menyebabkan ketergantungan tinggi terhadap kualitas koneksi internet. Koneksi yang kurang stabil menjadi kendala dalam proses sinkronisasi kode dengan repositori, pengujian aplikasi berbasis web, serta pelaksanaan rapat koordinasi secara daring.

   Solusi: Untuk meminimalkan dampak kendala tersebut, penulis menyiapkan jaringan alternatif seperti *tethering* dari perangkat seluler serta mengatur waktu kerja agar aktivitas yang membutuhkan koneksi stabil dilakukan pada jam-jam tertentu. Penulis juga mengunduh dokumentasi dan materi pendukung untuk dipelajari secara luring sehingga produktivitas tetap terjaga.

Kendala Non-Teknis

1. **Koordinasi dan komunikasi tim dalam lingkungan kerja daring.**
   Pelaksanaan magang secara daring menyebabkan terbatasnya interaksi

langsung antaranggota tim. Perbedaan waktu respons dan keterbatasan komunikasi non-verbal terkadang menimbulkan keterlambatan dalam penyampaian informasi maupun klarifikasi tugas.

Solusi: Penulis mengatasi kendala ini dengan menjaga komunikasi yang aktif dan terstruktur melalui media daring yang digunakan perusahaan, seperti grup pesan instan dan pertemuan virtual. Selain itu, penulis secara proaktif menyampaikan perkembangan pekerjaan dan kendala yang dihadapi agar koordinasi dengan pembimbing dan tim tetap berjalan dengan baik.

2. **Manajemen waktu dan disiplin kerja mandiri.**
   Bekerja secara daring menuntut tingkat kemandirian dan disiplin yang tinggi dalam mengatur waktu kerja. Pada awal pelaksanaan magang, penulis memerlukan penyesuaian untuk membagi waktu antara pengerjaan tugas magang, pembelajaran mandiri, dan aktivitas pribadi.

   Solusi: Penulis menyusun jadwal kerja harian dan mingguan secara mandiri serta menetapkan prioritas tugas berdasarkan tenggat waktu yang ditentukan. Dengan penerapan manajemen waktu yang lebih terstruktur, penulis dapat menyelesaikan tugas magang secara tepat waktu dan tetap menjaga produktivitas kerja.

PT. Agrolink Nusantara Indonesia

Dokumen Pengujian Manual Website – Sistem *Fleet Management*

Jenis Pengujian : Manual Testing (*Black-Box*)
Objek Uji : Website *Fleet Management*
Tanggal Pelaksanaan : 11 Januari 2026
Disusun oleh : Shella Faulina
Peran Penguji : *Chief Executive Officer*
Supervisor : Yonatan Ripandra Sinaga

---

1. Tujuan

Dokumen ini disusun sebagai laporan resmi pelaksanaan pengujian manual pada website sistem *Fleet Management* PT. Agrolink Nusantara Indonesia. Pengujian dilakukan untuk memverifikasi bahwa fungsi-fungsi utama sistem berjalan sesuai spesifikasi fungsional, terutama terkait alur operasional *admin–driver–system* pada modul rute, penugasan, dokumentasi lapangan, perawatan, dan laporan (*reports*).

2. Ruang Lingkup Pengujian

Ruang lingkup pengujian manual mencakup:

1. Autentikasi dan otorisasi berbasis peran (*admin* dan *driver*).
2. Modul *Routes* dan pembuatan *Assignment*.
3. Akses tugas oleh *driver* dan unggah dokumentasi (*assignment photos*).
4. Modul *Maintenance*.
5. Modul *Reports*: *filter* (tanggal, truk, driver), kartu metrik, dan grafik *monthly summary*.
6. Notifikasi sistem.

3. Metode Pengujian

Metode yang digunakan adalah **manual *black-box testing***, yaitu pengujian berorientasi pada perilaku sistem berdasarkan masukan (*input*) dan keluaran

(*output*) tanpa meninjau struktur kode. Setiap skenario uji mendokumentasikan prasyarat, langkah uji, data uji, keluaran yang diharapkan, keluaran aktual, dan status kelulusan.

4. Lingkungan Pengujian (*Test Environment*)

Perangkat : Laptop
Sistem Operasi : Windows 11
Peramban : Google Chrome
***Backend*/DB** : Supabase – PostgreSQL
Penyimpanan File : Supabase Storage
Koneksi : WiFi Rumah

5. Data Uji (*Test Data*)

Pengujian menggunakan data minimum berikut:

1. 1 akun *admin* aktif dan 1 akun *driver* aktif.
2. Minimal 1 data *truck*, 1 data *driver*, dan 1 data *partner*.
3. Minimal 1 rute dan 1 penugasan yang dibuat oleh *admin*.
4. Minimal 1 unggahan dokumentasi oleh *driver*.
5. Minimal 1 data perawatan (*maintenance*) untuk periode tertentu.
6. Data multi-periode (lebih dari 1 bulan) untuk validasi grafik *monthly summary*.

6. Daftar Skenario Uji dan Hasil

Tabel 3.4. Hasil Manual Testing Website *Fleet Management*

ID	Fitur	Langkah Uji (Ringkas)	Keluaran yang Diharapkan	Status
MT-01	Login Admin	Masukkan kredensial admin → klik *Login*	Berhasil masuk dan diarahkan ke *Dashboard* admin	*Pass*

Lanjutan pada halaman berikutnya

ID	Fitur	Langkah Uji (Ringkas)	Keluaran yang Diharapkan	Status
MT-02	Login Driver	Masukkan kredensial driver → klik *Login*	Berhasil masuk dan menu sesuai peran driver	*Pass*
MT-03	Validasi Login	Kredensial salah → *Login*	Muncul pesan error autentikasi, tidak masuk sistem	*Pass*
MT-04	Kontrol Akses Driver	Driver mencoba akses fitur admin (buat assignment)	Akses ditolak/halaman tidak tersedia (*unauthorized*)	*Pass*
MT-05	Create Route	Admin buat *route* (origin, destination, tanggal, truck, driver)	Data *routes* tersimpan dan tampil di daftar	*Pass*
MT-06	Create Assignment	Admin pilih route → *New Assignment* → isi form	Data *assignments* tersimpan, status awal terbentuk	*Pass*
MT-07	Validasi Form	Admin kosongkan field wajib → simpan	Sistem menolak dan menampilkan validasi field wajib	*Pass*
MT-08	Driver View Task	Driver buka daftar tugas → pilih assignment	Detail tugas tampil sesuai input admin	*Pass*
MT-09	Upload Dokumentasi	Driver unggah foto → submit	File tersimpan; entri *assignment_photos* terbentuk	*Pass*
MT-10	Validasi Upload	Driver unggah format/ukuran tidak valid	Sistem menolak unggah dan menampilkan pesan error	*Pass*
MT-11	Update Status	Ubah status (mis. *in progress*/*completed*)	Status tersimpan konsisten di detail dan daftar	*Pass*

Lanjutan pada halaman berikutnya

ID	Fitur	Langkah Uji (Ringkas)	Keluaran yang Diharapkan	Status
MT-12	Create Maintenance	Admin input perawatan (tgl, jenis, status, truck, driver)	Data *maintenance* tersimpan dan tampil	*Pass*
MT-13	Reports Filter (Tanggal)	Pilih *date range* → *Filter*	Kartu metrik menyesuaikan periode terpilih	*Pass*
MT-14	Reports Filter (Truck/Driver)	Pilih truck+driver → *Filter*	Metrik dan grafik hanya untuk truck/driver tersebut	*Pass*
MT-15	Grafik Monthly Summary	Rentang multi-bulan → *Filter*	Grafik menampilkan ringkasan per bulan (tren)	*Pass*
MT-16	Notifikasi	Buka notifikasi → lihat daftar/ubah terbaca	Notifikasi tampil; status terbaca berubah	*Pass*

7. Pernyataan

Berdasarkan pelaksanaan pengujian manual pada tanggal 11 Januari 2026, sistem *Fleet Management* telah diuji sesuai skenario pada dokumen ini. Status kelulusan setiap skenario tercantum pada Tabel 3.4 dan menjadi dasar evaluasi fungsionalitas sistem.

Tangerang, 11 Januari 2026

Disetujui oleh,

Yonatan Ripandra Sinaga

Chief Executive Officer