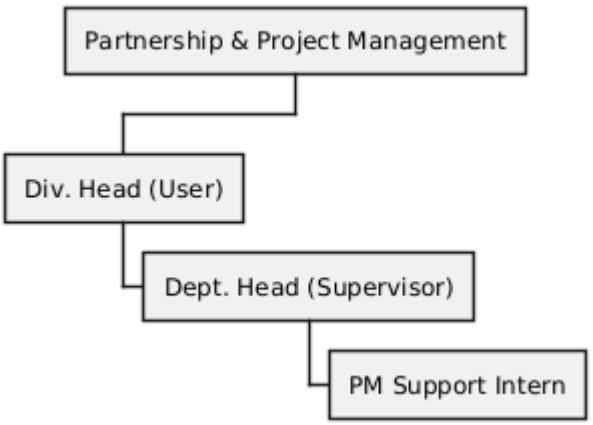
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

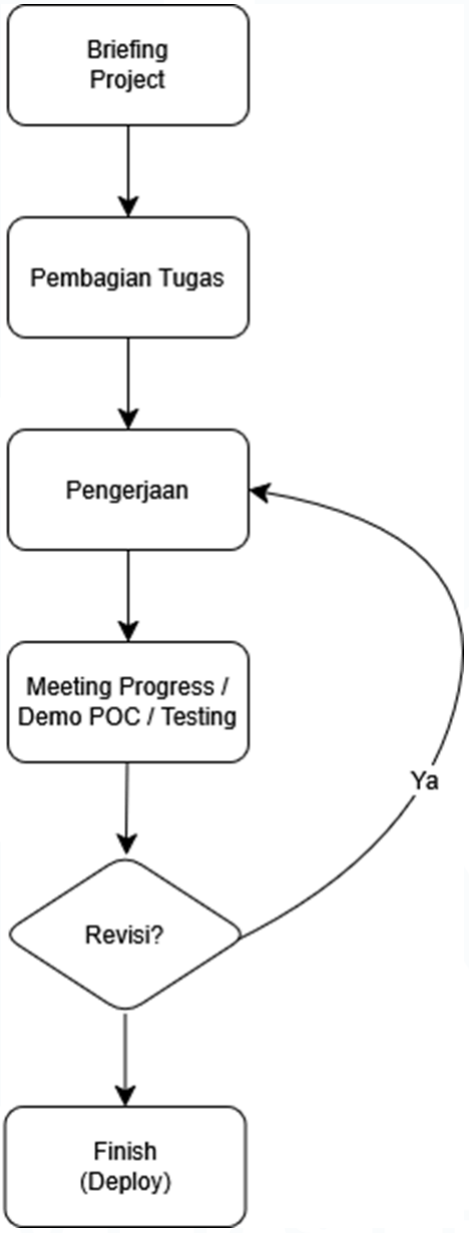


Gambar 3.1 Struktur divisi *Partnership & Project Management*

Mahasiswa magang di PT. Samakta Mitra (OneSmartServices) yaitu sebuah *strategic business unit* dibawah PT. Sinarmas Land yang berfokus ke *tech advisory* untuk Sinarmas Land, JV, dan *customer* eksternal. Mahasiswa ditempatkan pada divisi *Partnership & Project Management* sebagai *Project Manager Support Intern*, yang dipimpin oleh Bapak Efriansyah selaku *user*, dengan Ibu Rheny sebagai *supervisor*.

Pelaksanaan magang di PT. Samakta Mitra mengadopsi model berbasis proyek (*project-based*), dimana tugas yang diberikan kepada mahasiswa sangat bervariasi dan disesuaikan dengan kebutuhan dari berbagai divisi. Hal ini memungkinkan untuk memperoleh pengalaman yang lebih luas dalam berbagai aspek pekerjaan. Tugas yang diberikan tidak selalu berasal dari *user* utama, tetapi juga dapat diberikan oleh penanggung jawab proyek dari divisi berbeda. Dengan demikian, kami tidak hanya menjalankan tugas dari satu pihak, tetapi juga berkesempatan untuk berkolaborasi dengan peserta magang dari divisi lain dalam

sebuah tim proyek. Kolaborasi ini mendorong kerja sama tim, memperluas wawasan mengenai berbagai aspek operasional perusahaan, serta meningkatkan kemampuan dalam beradaptasi dengan lingkungan kerja yang dinamis.



Gambar 3.2 Alur kerja magang

Selama periode magang, mahasiswa terlibat dalam pengerjaan empat proyek utama yang hasilnya didokumentasikan dalam laporan ini. Alur kerja

untuk setiap proyek umumnya mengikuti prosedur yang terstruktur dapat dilihat pada gambar 3.2. Sebuah proyek diawali dengan penyampaian arahan oleh *user* dalam *meeting briefing project*. Dalam pertemuan tersebut atau pada *meeting* lanjutan, pembagian tugas akan dilakukan untuk memastikan setiap anggota tim memahami peran dan tanggung jawabnya. Setelah tugas dibagikan, pengerjaan proyek pun dimulai sesuai dengan arahan yang telah diberikan. Selama proses pengerjaan, diadakan *meeting* mingguan untuk melaporkan perkembangan proyek, di mana mahasiswa mempresentasikan progres yang telah dicapai serta menerima revisi, kritik, dan saran dari *user* untuk memastikan proyek berjalan sesuai harapan.

Menjelang tenggat waktu (*deadline*), akan diadakan *meeting final* atau demo *Proof of Concept (POC)* sekaligus sesi pengujian bersama *user*. Tujuan dari pertemuan ini adalah untuk memastikan bahwa proyek yang telah dikerjakan sesuai dengan spesifikasi dan tujuan yang telah ditetapkan. Setelah proyek dinyatakan selesai dan memenuhi standar yang diharapkan, langkah terakhir adalah proses *deployment* atau penyerahan proyek kepada user agar dapat digunakan sesuai kebutuhan operasional.

Dalam merancang arsitektur teknis untuk aplikasi-aplikasi ini, salah satu pertimbangan utama adalah memenuhi tuntutan akan *user experience* yang terasa instan dan responsif, dengan meminimalkan atau bahkan menghilangkan *state loading* yang mengganggu. Tantangan untuk mencapai hal ini sering kali terletak pada ukuran *bundle* JavaScript yang besar dan ketergantungan pada kondisi jaringan pengguna yang tidak selalu ideal. Oleh karena itu, pendekatan *Server-Side Rendering* (SSR) lebih diutamakan dibandingkan *Client-Side Rendering* (CSR), karena lebih andal dalam kondisi jaringan yang bervariasi dan tidak membebani *browser* dengan *bundle* yang besar di awal. Pendekatan ini menjadi landasan dalam pemilihan *framework* pengembangan yang akan digunakan.

Untuk mengeksekusi visi tersebut, pendekatan teknis yang dipilih adalah pengembangan aplikasi web dengan memanfaatkan *framework* modern, yaitu Remix. *Framework* ini dipilih karena keunggulan fundamentalnya, yaitu arsitektur yang mengintegrasikan *backend-frontend* dan implementasi *Server-Side Rendering* (SSR) secara bawaan. Mekanisme SSR memastikan konten di *render* di sisi server, menghasilkan *loading time* yang sangat cepat dan pengalaman pengguna yang responsif. Kombinasi antara siklus pengembangan yang cepat dan performa aplikasi yang optimal ini sangat sesuai dengan kebutuhan untuk menghasilkan solusi fungsional dan aman dalam kerangka waktu program magang, tanpa mengorbankan kualitas jangka panjang.

Keunggulan performa yang disebutkan di atas berakar pada mekanisme *Server-Side Rendering* (SSR) yang secara fundamental berbeda dari pendekatan *Client-Side Rendering* (CSR). Pada model SSR yang diimplementasikan Remix, server telah menyiapkan dokumen HTML yang utuh dan lengkap dengan konten sebelum mengirimkannya ke peramban pengguna. Sebaliknya, pada model CSR yang umum pada banyak *framework* lain, peramban hanya menerima kerangka HTML minimal beserta *file* JavaScript besar yang kemudian bertanggung jawab untuk merender seluruh konten di sisi klien. Perbedaan arsitektural ini memiliki implikasi langsung dan signifikan pada kecepatan muat awal sebuah halaman dan keseluruhan pengalaman pengguna. [1]

Secara teknis, efektivitas SSR digambarkan sebagai sebuah "pola arsitektur yang pivotal" dalam pengembangan web modern karena kemampuannya dalam mengurangi metrik performa krusial seperti *Time to First Paint* (TTP) dan *Time to Interactive* (TTI). Pengurangan waktu pada kedua metrik ini berarti pengguna dapat melihat konten visual lebih cepat dan mulai berinteraksi dengan halaman lebih awal, yang secara langsung mendukung klaim "*loading time* yang sangat cepat" dan "pengalaman pengguna yang responsif". Lebih lanjut, *framework* Remix mengimplementasikan kapabilitas canggih yang dikenal sebagai *streaming SSR*. Dengan teknik ini, server dapat mulai

mengirimkan potongan-potongan HTML ke *browser* bahkan sebelum seluruh halaman selesai di *render*. Pendekatan ini secara dramatis meningkatkan persepsi kecepatan dan menegaskan kematangan teknologi yang dipilih untuk proyek-proyek ini. [2]

3.2 Tugas dan Uraian Kerja Magang

3.2.1 Timeline

Selama masa magang, telah dikerjakan beberapa proyek yang telah ditugaskan. Rincian proyek yang dikerjakan dapat dilihat pada tabel 3.1 berikut:

Tabel 3.1 Timeline kerja magang

Bulan	**Minggu**	**Pekerjaan**
Februari	1	Onboarding, perkenalan dengan semua tim. Briefing jobdesc.
Februari	2	Belajar mandiri *business process* di PT. Samakta Mitra
Februari	3	Membuat flowchart *business process*, koordinasi dengan tim admin.
Februari	4	Discussing dan boilerplating backend OneSmartDraw.
Maret	1	Mulai backend development, mengimplementasi core system.

Maret	2	Membuat validasi input form dengan zod.
Maret	3	Mengimplementasi kesuluruhan sistem, implementasi socket.io untuk realtime.
Maret	4	Bug fix dan menambahkan fitur yang di request user. Deploy di server production, dan mengoperasikan control di acara. Diskusi mengenai SLA dan membuat skrip POC di python.
April	1	Konversi logic POC python ke typescript untuk perhitungan SLA. Mencari strategi melalui API Aruba Central.
April	2	Membuat kalkulasi multi-site berdasarkan API,
April	3	Develop remix-auth ke OAuth 2.0 HPE dan integrasi front-end dan logika back-end. Memulai boilerplating proyek OneSmartDMS.
April	4	Melanjutkan fitur deteksi intermittent devices untuk Aruba SLA Calculator.

		Develop autentikasi MSAL OAuth 2.0 ke remix-auth untuk OneSmartDMS.
Mei	1	Membuat report page untuk Aruba SLA Calculator. Mulai develop RBAC untuk OneSmartDMS.
Mei	2	Serah terima dan deployment docker Aruba SLA Calculator ke tim devops. Melanjutkan fitur Document Tracker di OneSmartDMS.
Mei	3	Membuat fitur approval untuk Letter Numbering di OneSmartDMS dan membuat ulang layout front-end.
Mei	4	Melakukan bug fixing RBAC, membuat fitur document preview di OneSmartDMS
Mei	5	Melakukan bug fix timestamping di Aruba SLA Calculator dan deploy ulang.
Juni	1	Mulai boilerplating SAM, merencanakan routing dan fitur yang akan digunakan.
Juni	2	Membuat core system peminjaman kendaraan, membuat fitur extension dan reimbursements, membuat fitur multi-company di SAM.

Juni	3	Membuat admin panel dan dashboard untuk summary di SAM dan approval extensions dan reimbursements di SAM.
Juni	4	Mengintegrasikan API Google Maps dan bug fixing core system di SAM
Juli	1	Melakukan bug fixing dan penambahan field di page peminjaman dan management vehicle. Demo POC SAM ke user.
Juli	2	Deployment SAM ke production.
Juli	3	Bug fixing Aruba SLA dan SAM.

3.2.2 Onboarding dan Belajar Mandiri

Kegiatan magang secara resmi dimulai pada 3 Februari 2025. Minggu pertama diisi dengan perkenalan kepada sesama peserta magang, jajaran staf, dan *vendor* perusahaan. Pada periode ini mendapatkan penjelasan mengenai struktur organisasi serta difasilitasi untuk kebutuhan administratif, seperti pembuatan kartu akses, pendaftaran parkir, dan pengenalan pada sistem presensi berbasis *web* (IMAGES).

Sesuai arahan pada *briefing* awal tanggal 24 Januari 2025, tugas pertama adalah memahami alur *business process* yang diterapkan perusahaan. Untuk itu, diperlukan untuk diskusi dengan admin untuk memahami alur kerja dan berkonsultasi dengan Bapak Efriansyah mengenai aspek teknisnya.

Sebagai verifikasi pemahaman, kemudian membuat sebuah *flowchart* untuk memvisualisasikan keseluruhan proses tersebut.

3.2.3 OneSmartDraw

Dalam proyek ini, mahasiswa ditugaskan mengembangkan aplikasi undian *doorprize* di bawah arahan Bapak Ari selaku *Person in Charge (PIC)*. Tujuan utamanya adalah menggantikan proses manual dengan sistem efisien yang terdiri dari dua komponen: panel *admin* untuk mengelola peserta dan hadiah secara rahasia, serta tampilan visual undian yang menarik bagi audiens. Proses pengembangan, pengujian *Proof of Concept (POC)*, hingga *deployment* ke *Azure Cloud App VM* diselesaikan dalam waktu tiga minggu.

Dengan kebebasan memilih teknologi, aplikasi ini dibangun menggunakan *tech stack* modern yang berfokus pada performa dan keamanan. Rincian implementasinya adalah sebagai berikut:

- ***Backend***: Dibangun menggunakan Express.js dengan *runtime* Bun. Pengelolaan *database* menggunakan Prisma *ORM*, sementara validasi data ditangani oleh Zod.
- ***Frontend***: Antarmuka visual dikembangkan dengan React.js, serta dipercantik menggunakan Tailwind CSS dan ShadcnUI.
- **Keamanan & Autentikasi**: Menerapkan *stateless authentication* menggunakan *JSON Web Token (JWT)* dan proteksi *Captcha* dari Cloudflare Turnstile untuk mencegah celah keamanan seperti *SQL Injection*, *XSS*, dan *session hijacking*.

Fitur utama dari aplikasi ini adalah sinkronisasi *real-time* yang diimplementasikan menggunakan protokol *WebSocket* dengan *library* Socket.IO. Mekanisme ini memungkinkan operator mengendalikan undian dari panel *admin*, sementara tampilan visual (*wheel spin*) pada layar utama dan perangkat pribadi peserta akan berputar dan menampilkan pemenang

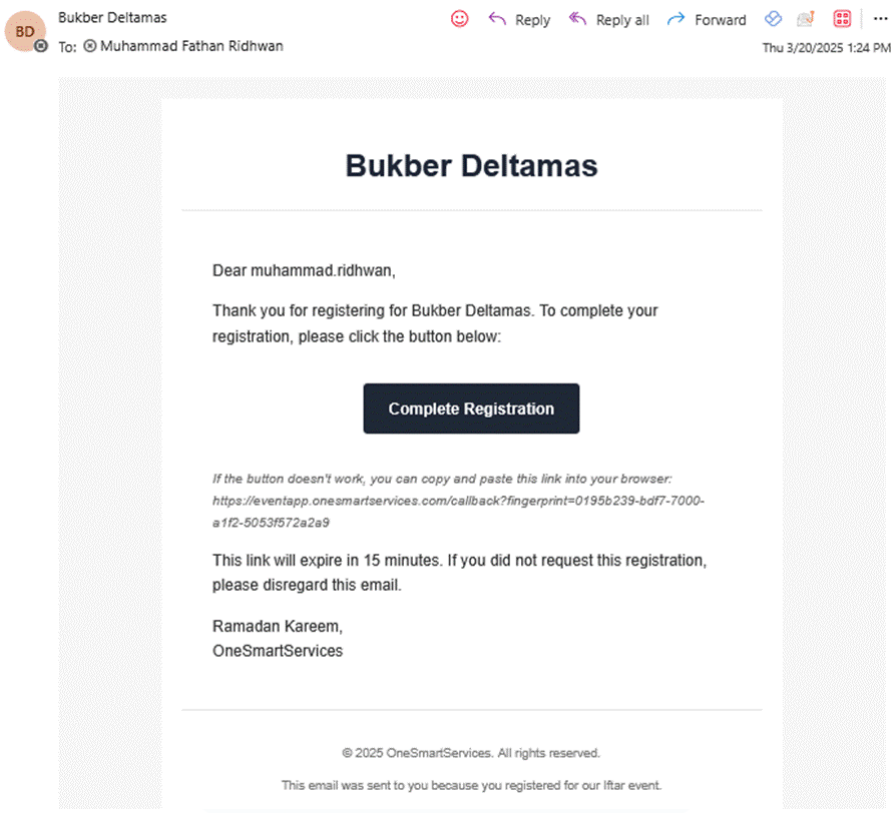
secara serentak. Pendekatan ini memastikan semua peserta, bahkan yang berada jauh dari panggung, mendapatkan pengalaman yang adil dan imersif.

Untuk meningkatkan efisiensi manajemen data dan fleksibilitas operasional, dikembangkan beberapa fitur tambahan. Fitur-fitur ini mencakup:

- **Operasi Massal (*Bulk Operations*)**: Mengimplementasikan fitur *bulk insert* melalui *file CSV* untuk menambahkan daftar peserta terblokir secara massal, serta *bulk update* di sisi *backend* untuk mempercepat pembaruan data.
- **Kontrol Fitur (*Feature Toggle*)**: Sebuah mekanisme untuk mengaktifkan atau menonaktifkan fungsionalitas tertentu secara dinamis, seperti membuka dan menutup akses registrasi peserta sesuai jadwal acara.

Guna memastikan keadilan dan mencegah pendaftaran ganda, sistem ini juga mengimplementasikan mekanisme identifikasi unik berbasis *fingerprint* perangkat. *Fingerprint* ini dihasilkan secara acak menggunakan UUIDv7, kemudian disimpan di localStorage pada *browser* peserta. Meskipun validasi awal menggunakan *email* korporat, *fingerprint* ini berfungsi sebagai lapisan keamanan tambahan untuk mendeteksi dan mencegah perangkat yang sama mendaftar lebih dari satu kali, sehingga menjaga integritas proses undian.

Untuk mengatasi pendaftaran ganda, perlu diterapkan sistem Magic Link sebagai metode login setelah registrasi. Setelah peserta mendaftar, mereka akan menerima tautan login unik yang hanya berlaku selama 15 menit. *Link* ini dikirim melalui email menggunakan NodeMailer, yang mengirimkan permintaan SMTP ke *server* Outlook. Implementasi dapat dilihat pada gambar 3.4.



Gambar 3.3 Format email yang dikirimkan ke email corporate

Saat peserta membuka tautan tersebut, *controller* akan melakukan *set* properti loginValidUntil menjadi NULL, menandakan bahwa peserta telah berhasil login dan mencegah penggunaan ulang tautan tersebut.

Proses undian terjadi di backend yang menjalankan langkah-langkah berikut: Controller drawWinners akan menerima *array* Id dari hadiah, kemudian melakukan *query* untuk mendapatkan daftar peserta yang memenuhi syarat sebagai pemenang, yang disebut *candidates*.

Kriteria peserta yang masuk dalam daftar candidates adalah:

- **isBlocked = false**, Peserta tidak diblokir.
- **isAlreadyWon = false**, Peserta belum pernah memenangkan hadiah sebelumnya.

- **isNotAvailable = false**, Peserta tidak termasuk dalam daftar yang sudah dipanggil tetapi tidak hadir.
- **loginValidUntil = NULL**, Menandakan peserta telah menyelesaikan proses registrasi.

Secara default, setiap peserta memiliki weight sebesar 1.0, yang dapat disesuaikan untuk mengatur peluang menang. Proses pemilihan pemenang dilakukan dengan cara berikut:

1. Weight dari setiap kandidat dikalikan dengan RNG diantara 0 dan 1.
2. Hasil perkalian tersebut diurutkan dari nilai terbesar ke terkecil.
3. Sejumlah n peserta dengan nilai weight tertinggi dipilih sebagai pemenang.

Untuk detail implementasi dapat dilihat pada gambar 3.4.

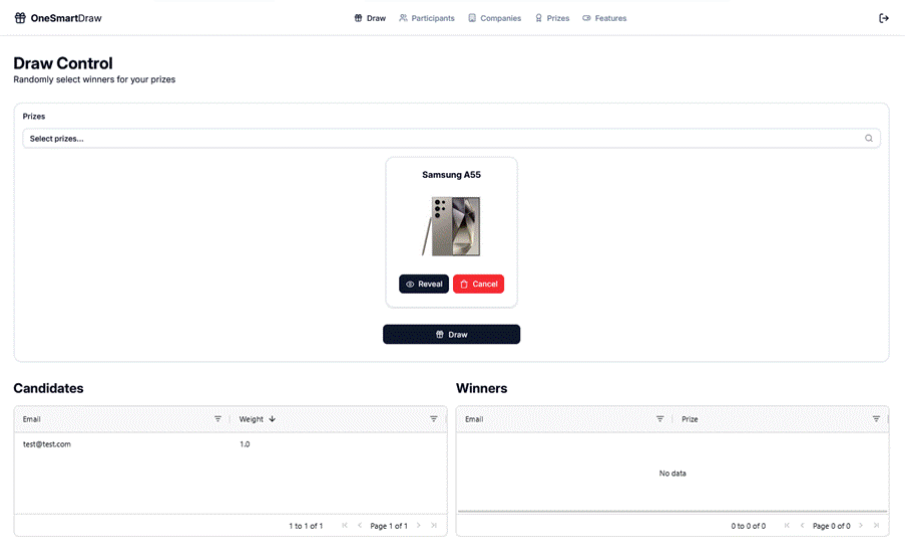
```
const drawWinners = (
  candidates: { id: number; weight: number }[],
  numWinners: number
) => {
  const shuffled = candidates
    .map((c) => ({
      ...c,
      rand: Math.random() * c.weight, // multiply by weight
    }))
```

NUSANTARA

```
    .sort((a, b) => b.rand - a.rand); // pick highest random
values



  return shuffled.slice(0, numWinners).map((winner) =>
winner.id);

};
```

Gambar 3.4 Potongan kode untuk menentukan pemenang.

Admin Panel mengirimkan permintaan ke *backend* untuk memulai proses undian. Setelah pemenang ditentukan, data pemenang dikirimkan ke frontend melalui Socket.IO, dan frontend akan menjalankan animasi pemutaran *spin wheel*.



Gambar 3.5 Tampilan kontrol utama

Setelah proses undian pemenang berjalan, data pemenang dikirimkan ke frontend melalui Socket.IO, yang kemudian akan memulai animasi pemutaran *spinwheel*. Background dari tampilan utama juga merupakan

animasi yang dibuat di Rive dan dijadikan background dengan library react-canvas-lite.

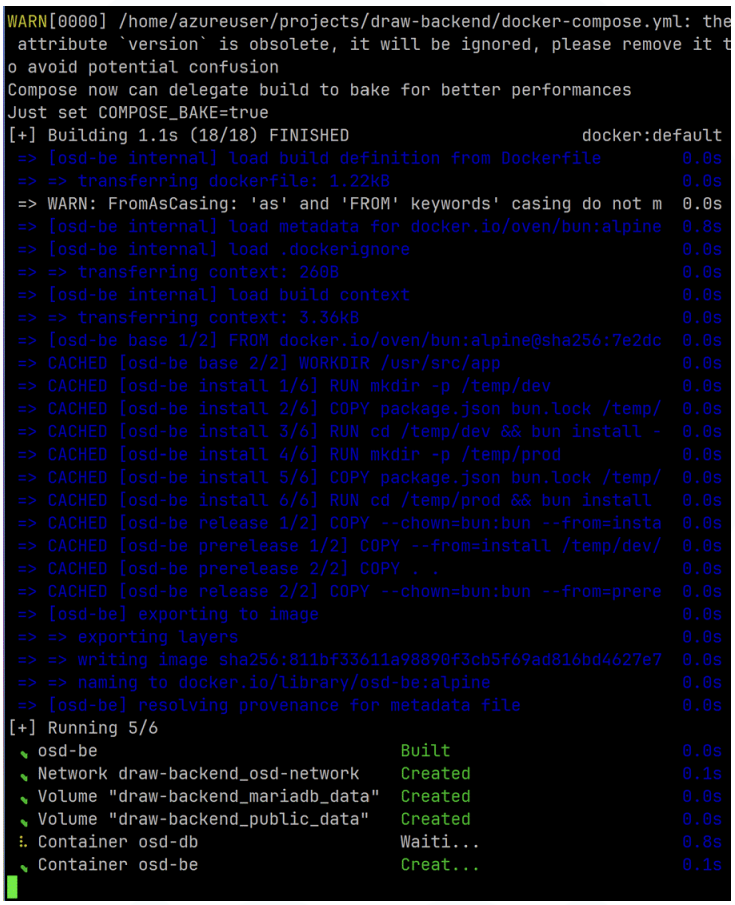
Gambar 3.6 Tampilan utama di layar besar dan pengguna

Arsitektur aplikasi ini memisahkan antara logika dan presentasi. Penentuan pemenang sepenuhnya dilakukan di *backend* dan dikontrol melalui Panel *Admin*. Oleh karena itu, *frontend* hanya berfungsi sebagai lapisan visual untuk menciptakan euforia, menampilkan animasi *spin wheel* selama lima detik serta efek visual seperti *confetti* saat pemenang diumumkan.

Untuk proses *deployment*, diterapkan pendekatan modern yang efisien dan otomatis:

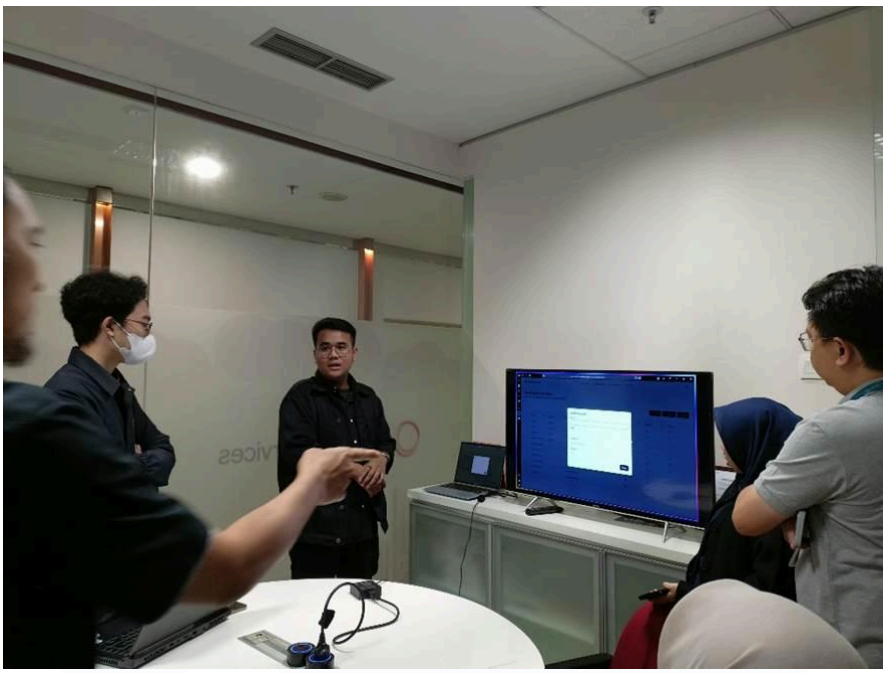
- ***Frontend***: Di-*deploy* menggunakan Cloudflare Pages yang terhubung dengan *repository* GitHub. Setiap perubahan pada kode secara otomatis memicu proses *deployment* (CI/CD), sehingga aplikasi sisi klien selalu *up-to-date*.
- ***Backend***: Menerapkan strategi *containerization* dengan Dockerfile untuk membungkus aplikasi. Selain itu, docker-compose.yml digunakan untuk mengorkestrasi *container backend* bersama *database*

MariaDB, menyederhanakan proses *deployment* di *VM* Azure CloudApps. Proses *deployment* dapat dilihat pada gambar 3.7.



Gambar 3.7 Proses deployment dengan docker compose

Beberapa hari sebelum acara pertama dimulai, mahasiswa melakukan demo POC dan pengujian bersama berdasarkan test case yang diberikan oleh Pak Ari dan Team IFTAR 2025. Test case ini dirancang untuk mengidentifikasi celah atau error dalam aplikasi, sehingga mahasiswa dapat memperbaiki bug yang ditemukan dan melakukan *redeployment* sebelum acara berlangsung.



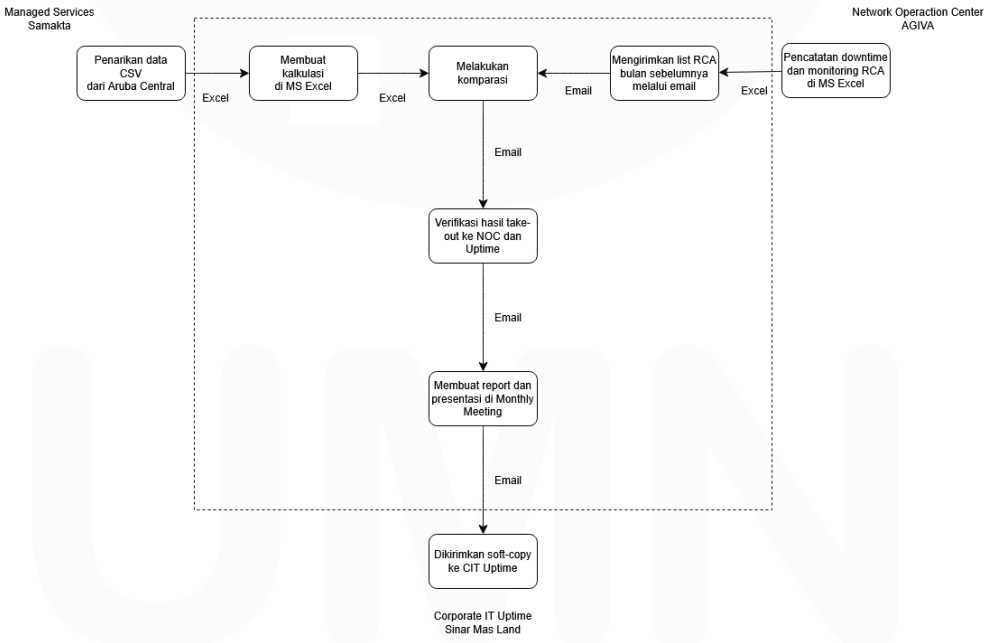
Gambar 3.8 Demo POC OneSmartDraw

Aplikasi yang dikembangkan oleh mahasiswa telah digunakan dalam dua event berbeda tanpa mengalami kendala besar. Jika aplikasi ini akan digunakan kembali di event mendatang, mahasiswa akan melakukan handover untuk memastikan kelancaran penggunaannya.

3.2.4 Aruba SLA Calculator

Sebuah proyek untuk pengembangan dan otomatisasi kalkulasi *Service Level Agreement* (SLA) perangkat *access point* Aruba diinisiasi pada tanggal 21 Maret 2024. Proyek ini berada di bawah koordinasi Bapak Ari, Bapak Putra dan *vendor* AGIVA sebagai *Person in Charge* (PIC). Tujuan utama dari proyek ini untuk mengukur *uptime* seluruh perangkat yang tersebar di area Sinarmas Land. Fungsionalitas inti dari sistem yang akan dikembangkan adalah menghasilkan laporan secara otomatis apabila terdapat wilayah dengan performa SLA di bawah ambang batas yang telah ditentukan, yaitu 99,50%.

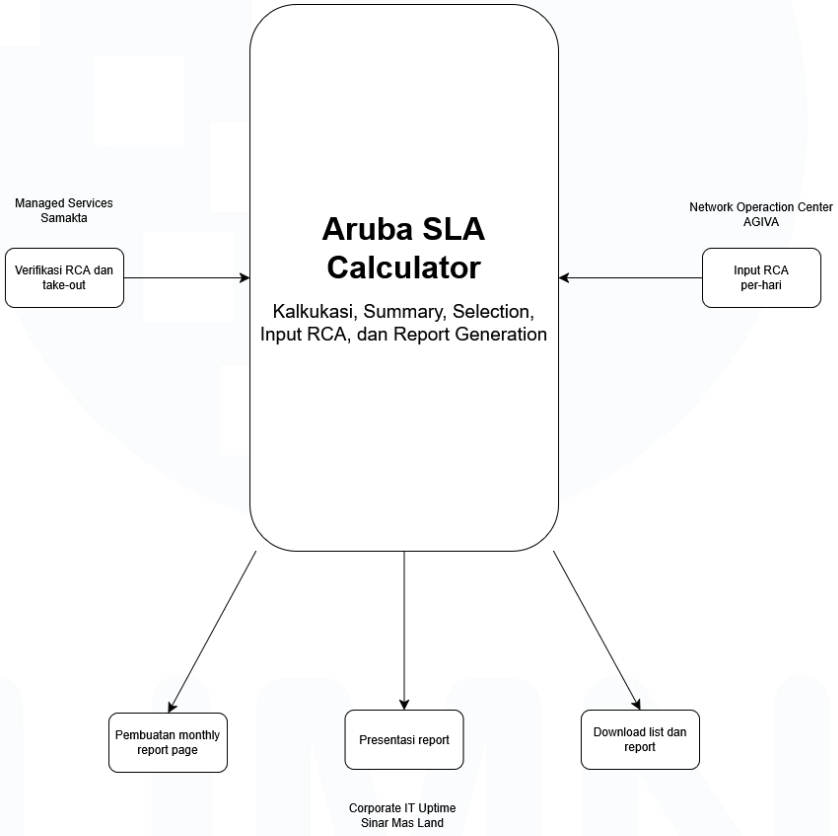
Hingga kini, kalkulasi SLA masih dilakukan secara manual menggunakan Excel. Meskipun metode perhitungan manual tersebut dapat menghasilkan angka SLA, alur kerja yang menyertainya sangat tidak efisien dan terfragmentasi. Setelah perhitungan selesai, hasilnya didistribusikan melalui email setiap bulan. Di sisi lain, pihak NOC (*Network Operation Center*) harus mengelola dan menyimpan historis *Root Cause Analysis* (RCA) serta *downtime* untuk setiap *access point* dalam sebuah dokumen terpisah. *File* ini kemudian diserahkan secara manual kepada tim *Managed Services*, yang selanjutnya harus melakukan seleksi data secara manual berdasarkan catatan RCA tersebut sebelum membuat laporan terpisah dalam format PowerPoint untuk dipresentasikan ke tim CIT Uptime. Keseluruhan proses yang berlapis dan manual ini dapat diobservasi pada gambar 3.9.



Gambar 3.9 *Flow* sebelum ada Aruba SLA Calculator

Berdasarkan permasalahan tersebut, tujuan utama dari pengembangan Aruba SLA Calculator adalah menjadi sebuah aplikasi terpusat yang merombak total alur kerja ini. Harapannya, platform ini menjadi *single source of truth* di mana semua pihak terkait, baik tim NOC maupun *Managed*

Services, dapat langsung mengakses, memvalidasi, dan melakukan perubahan data di satu tempat yang sama. Dengan demikian, koordinasi melalui email dapat dihilangkan sepenuhnya, yang secara signifikan mengurangi potensi *human error* dan mempermudah pekerjaan tim *Managed Services* dalam menyusun laporan. Alur kerja ideal yang diharapkan melalui aplikasi ini dapat diobservasi pada gambar 3.10.



Gambar 3.10 *Flow* sesudah ada Aruba SLA Calculator

Total waktu layanan (*total service minutes*) sebagai contoh dihitung dengan rumus berikut:

$$Total\ Service\ Minutes = Jumlah\ Perangkat \times Jam \times Menit \times Hari$$

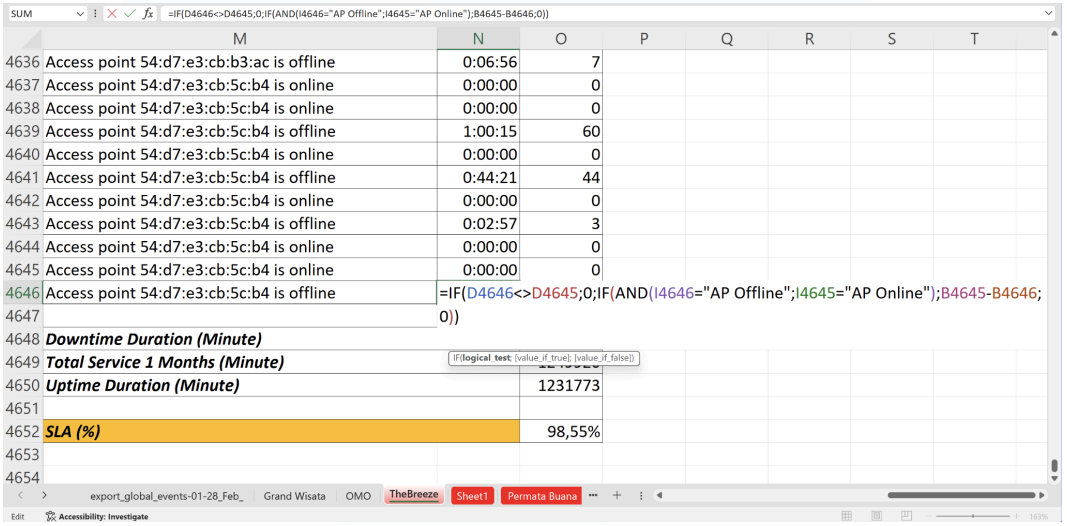
Kemudian, total waktu *uptime* (*total uptime minutes*) diperoleh dengan mengurangi total waktu layanan dengan total *downtime minutes*:

$$Total\ Uptime\ Minutes = Total\ Service\ Minutes - Total\ Downtime\ Minutes$$

Dari hasil tersebut, SLA (%) dapat dihitung dengan rumus:

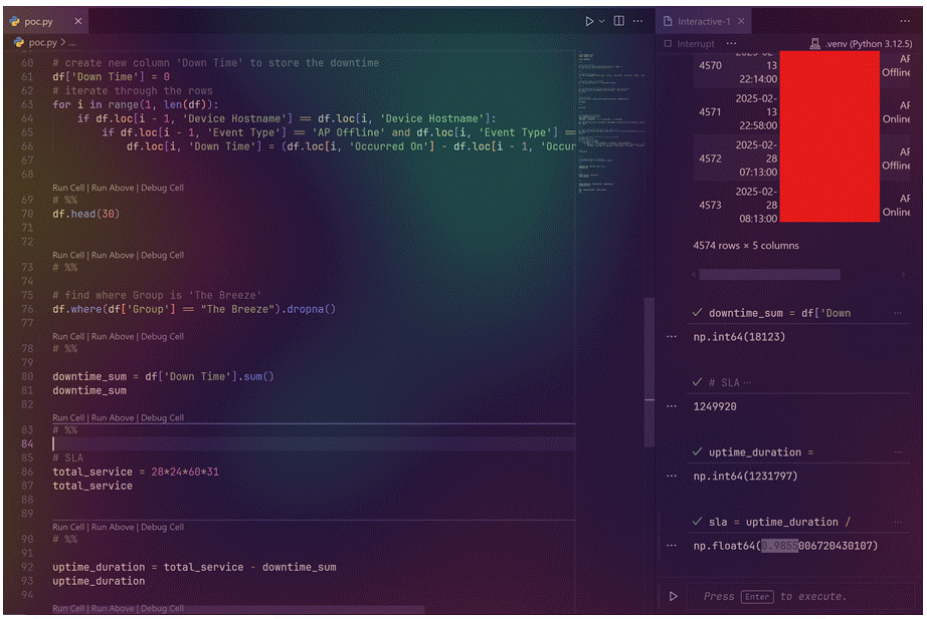
$$SLA\% = (\frac{Total\ Uptime\ Minutes}{Total\ Service\ Minutes}) \times 100\%$$

Proses perhitungan manual ini dapat dilihat pada gambar 3.11.



Gambar 3.11 Kalkulasi SLA manual dengan Excel

Untuk mengatasi keterbatasan metode manual, dikembangkanlah *Proof of Concept* (POC) berupa *script* Python yang mampu membaca, memfilter, dan menghitung SLA secara otomatis. Skrip ini menerapkan prinsip perhitungan yang sama dengan metode manual di Excel, sebagaimana ditunjukkan pada gambar 3.12.



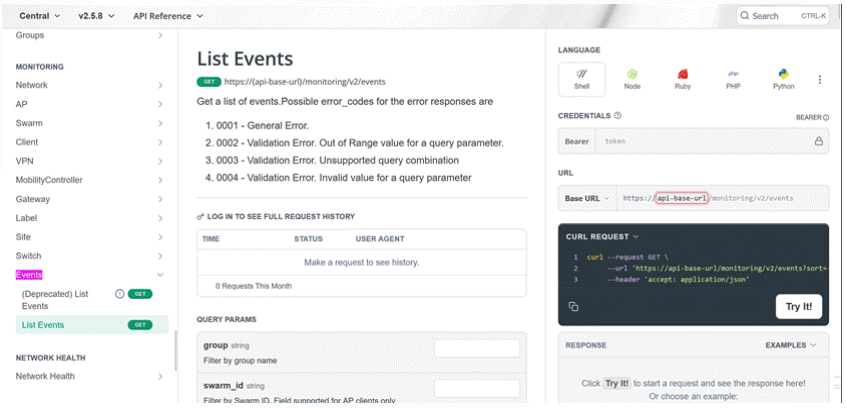
Gambar 3.12 POC Python script untuk kalkulasi SLA

Setelah tahap POC fungsional dari kalkulator SLA berhasil dikembangkan dan didemonstrasikan kepada para pemangku kepentingan. Sesi demonstrasi ini menjadi titik balik penting dalam kelanjutan proyek.

Sesi *demo* awal mendapat tanggapan positif dari Bapak Teguh (tim *Network Operation Center - NOC*) dan Bapak Putra, yang menilai alat ini sangat membantu. Dalam diskusi yang sama, Bapak Teguh memberikan masukan krusial, yaitu perlunya justifikasi *downtime* berdasarkan *Root Cause Analysis (RCA)*. Beliau menekankan bahwa tidak semua gangguan, seperti masalah dari *Internet Service Provider (ISP)*, dapat dihitung sebagai pelanggaran *Service Level Agreement (SLA)*.

Berdasarkan masukan tersebut, ruang lingkup proyek diperluas menjadi sebuah *platform web* komprehensif. Solusi teknis untuk ini ditemukan melalui API HPE Aruba Central yang memungkinkan penarikan data *log* via *endpoint List Events* yang dapat dilihat pada gambar 3.13. Fitur utama yang disepakati untuk dikembangkan adalah:

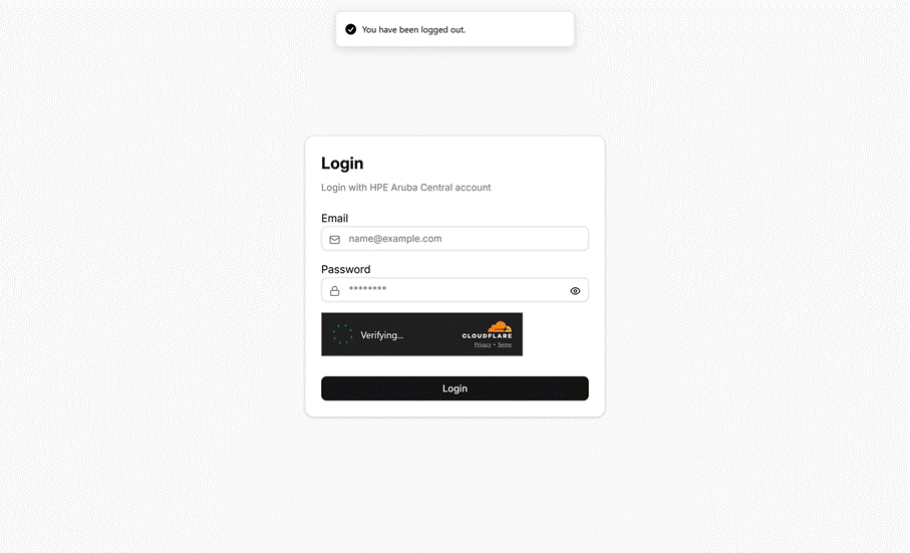
- **Kalkulasi *SLA* Otomatis**: Menghitung pencapaian *SLA* berdasarkan data *uptime* perangkat.
- **Manajemen Justifikasi**: Halaman khusus bagi tim *NOC* untuk memberikan justifikasi *RCA* pada setiap *event downtime*, yang secara otomatis akan dikecualikan (*excluded*) dari perhitungan.
- **Laporan Bulanan**: Fitur untuk menghasilkan laporan (*report*) *SLA* untuk keperluan audit dan pelaporan manajemen.



Gambar 3.13. API *documentation* HPE Aruba Central

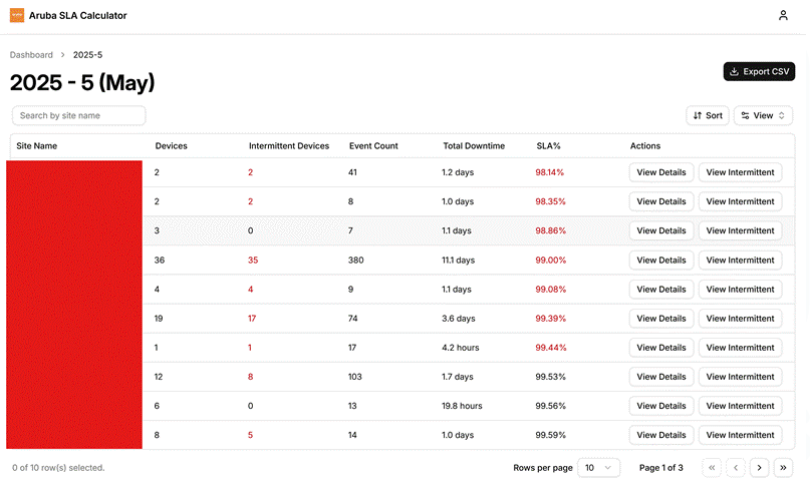
Langkah teknis pertama dan paling fundamental untuk dapat menarik data adalah mengimplementasikan mekanisme autentikasi yang aman sesuai standar yang ditetapkan oleh HPE Aruba Central. API HPE Aruba Central diamankan menggunakan protokol OAuth 2.0, yang merupakan standar industri untuk proses otorisasi yang aman.

Setelah autentikasi berhasil, data historis perangkat ditarik secara langsung dari *Events & Logs API* yang disediakan oleh Aruba Central. Data tersebut diterima dalam format JSON (*JavaScript Object Notation*), sebuah format yang terstruktur dan ringan, yang membuat proses pengolahan dan parsing di sisi aplikasi menjadi jauh lebih efisien dibandingkan dengan format lain.



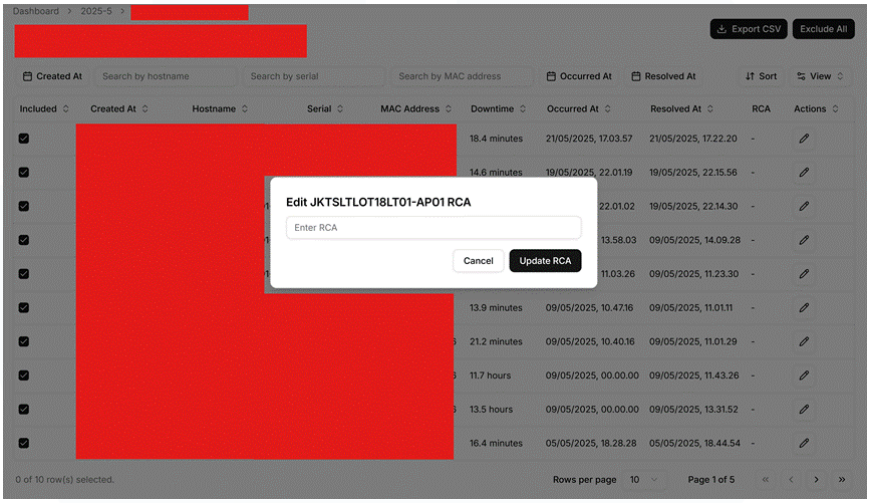
Gambar 3.14 *Authentication* menggunakan OAuth 2.0

Setelah itu di-implementasi kalkulasi yang sama berdasarkan POC berbasis Python sebelumnya menjadi website berdasarkan *framework* Remix dengan ShadcnUI dan *datatable* yang menyajikan data *events* yang telah dikelompokkan per-bulan dan per-*site*. Tampilan utama dapat dilihat pada gambar 3.15.



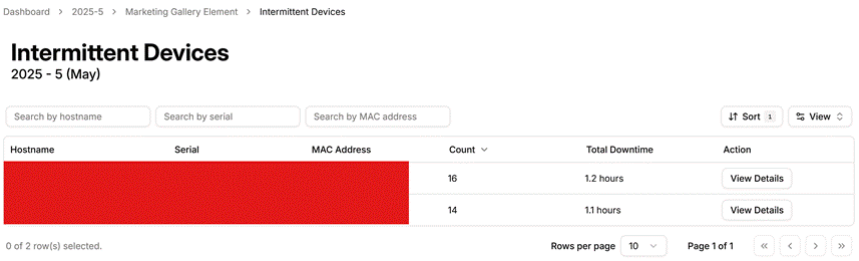
Gambar 3.15 Tampilan detail per-*site* di bulan yang dipilih

Lalu, pihak NOC dapat menambahkan detail RCA dan *toggle* apakah *events* tersebut dapat di *include* dalam kalkulasi SLA.



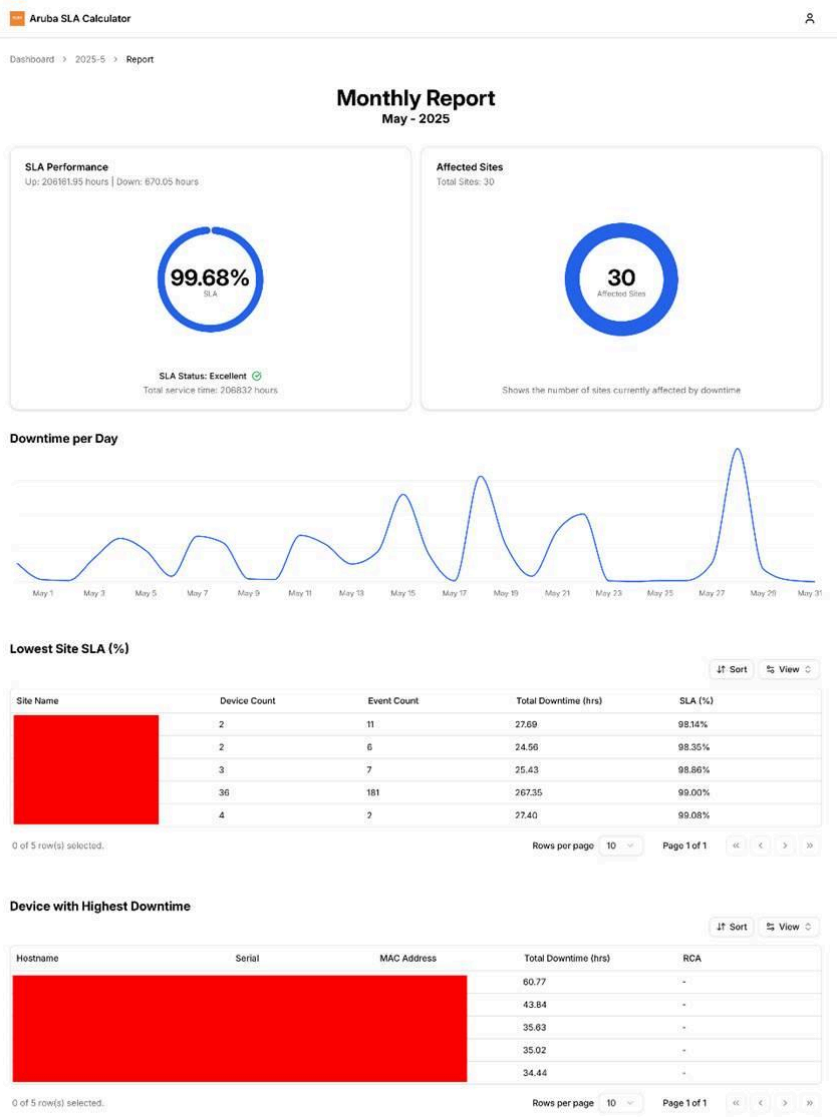
Gambar 3.16 Tampilan penambahan RCA pada AP

Adapun apabila *downtime* AP dibawah 10 menit, termasuk ke golongan *intermittent*. Hal ini dapat terjadi apabila AP mengirimkan *event* AP *Online* dan AP *Offline* secara terus menerus. Hal ini dapat dijadikan acuan bagi report untuk menindaklanjuti kejadian tak terduga tersebut di *monthly meeting*.



Gambar 3.17 Tampilan halaman *intermittent device* pada *site*

Untuk menyajikan data yang telah diolah menjadi informasi yang mudah dipahami dan dapat ditindaklanjuti, platform *Aruba SLA Calculator* dilengkapi dengan sebuah halaman *report*. Halaman ini berfungsi sebagai pusat visualisasi data yang memungkinkan tim manajemen dan NOC untuk memantau kondisi kesehatan jaringan secara efisien serta mengidentifikasi area bermasalah dengan cepat.



Gambar 3.18 Tampilan untuk *monthly report*

Dasbor utama dirancang untuk memberikan pandangan operasional yang komprehensif melalui beberapa komponen visual utama:

- **Grafik *SLA* Keseluruhan**: Menampilkan persentase rata-rata *SLA* dari seluruh perangkat dalam bentuk *gauge chart* untuk memberikan gambaran performa jaringan secara sekilas.
- **Grafik Tren *Downtime* Harian**: Memetakan total durasi *downtime* per hari untuk membantu identifikasi pola atau anomali insiden.
- **Tabel Peringkat Situs *SLA* Terendah**: Menyajikan daftar lokasi dengan performa terburuk untuk membantu manajemen memprioritaskan tindakan perbaikan.
- **Tabel Peringkat Perangkat *Downtime* Tertinggi**: Menyajikan data *access point (AP)* individual dengan *downtime* tertinggi sebagai acuan investigasi teknis bagi tim *NOC*.

Mengingat data yang diolah bersifat sensitif, tahap *deployment* aplikasi ditangani dengan prosedur keamanan yang ketat. Proses perilisan diserahterimakan kepada tim Azure *DevOps* untuk memastikan aplikasi ditempatkan pada infrastruktur *cloud* yang aman dan terisolasi. Nantinya, *platform* ini akan di-*host* agar hanya dapat diakses melalui jaringan internal perusahaan, sehingga kerahasiaan dan integritas data operasional terjaga dengan membatasi akses hanya untuk pihak yang berwenang.

3.2.5 OSD - OneSmartDMS

Pada tanggal 22 April 2025, sebuah proyek untuk mendigitalisasi alur kerja (*workflow*) pada divisi *Project Management and Partnership* telah di inisiasi di bawah arahan Bapak Rian selaku *Division Head*. Proyek ini dilatarbelakangi oleh adanya beberapa proses kerja krusial yang masih bergantung pada metode manual, seperti proses pembuatan nomor surat resmi

yang rentan kesalahan, kesulitan dalam pelacakan (*tracking*) dokumen-dokumen legal, serta pengelolaan dokumen kemitraan misalnya NDA, MoU, dan sertifikat yang belum terpusat.

Pengembangan OneSmartDMS merupakan sebuah inisiatif digitalisasi jangka panjang. Oleh karena itu, pengembangannya tidak dilakukan secara monolitik, melainkan menggunakan pendekatan modular. Artinya, fungsionalitas-fungsionalitas utama akan dirancang, dikembangkan, dan dirilis secara bertahap dalam bentuk "modul" yang saling terintegrasi. Strategi ini memungkinkan perusahaan untuk dapat merasakan manfaat dari setiap fitur lebih cepat tanpa harus menunggu keseluruhan sistem selesai dibangun.

Pada tahap pengembangan saat ini, yang bertepatan dengan periode magang, telah berhasil dikembangkan tiga modul fundamental sebagai fondasi awal sistem. Ketiga modul tersebut adalah:

1. **User Management:** Untuk pengelolaan pengguna dan hak akses.
2. **Document Numbering:** Untuk otomatisasi pembuatan nomor surat resmi.
3. **Document Tracking:** Untuk pelacakan dokumen legal dan kemitraan.

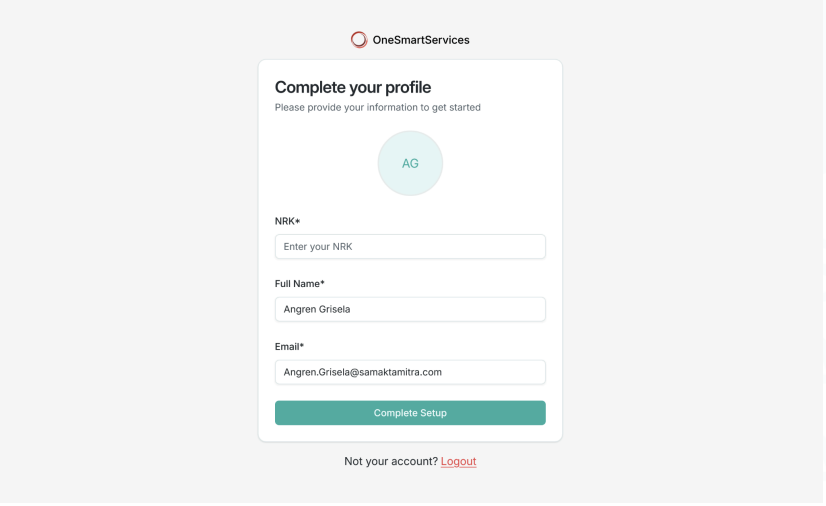
Platform OneSmartDMS dikembangkan sebagai aplikasi *web* dengan arsitektur dan tumpukan teknologi (*tech stack*) modern sebagai berikut:

- **Framework & Akses**: Dibangun menggunakan Remix dengan standar *Progressive Web App (PWA)*. Hal ini memungkinkan aplikasi untuk di-*install* di perangkat *desktop* maupun *mobile* demi akses yang lebih cepat, diimplementasikan menggunakan *library* remix-pwa.
- **Antarmuka Pengguna (*UI*)**: Menggunakan *library* komponen ShadcnUI di atas Tailwind CSS, dengan pendekatan desain *mobile-first* yang responsif.

- **Autentikasi & Keamanan**: Mengadopsi protokol *OAuth 2.0* yang terintegrasi dengan akun Microsoft perusahaan, untuk memastikan hanya karyawan internal yang dapat mengakses sistem.
- **Penyimpanan *File***: Terhubung dengan sistem penyimpanan objek yang kompatibel dengan S3 (*S3-compatible storage*) seperti Cloudflare R2 untuk mengelola dokumen (*PDF* dan lainnya).

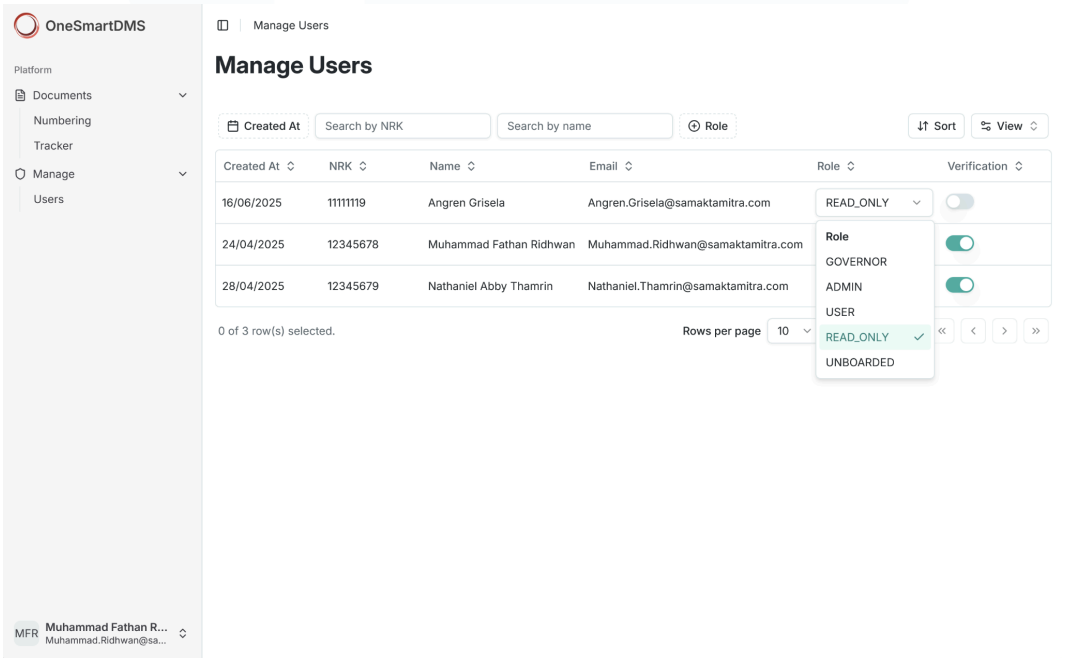
Sebagai bagian dari *roadmap* ke depan, direncanakan pengembangan modul *Project Tracker*. Modul ini akan berfungsi untuk mengelola data proyek yang berjalan maupun yang telah diarsipkan (*archived*), dengan fitur kunci berupa notifikasi otomatis untuk jadwal penagihan termin pembayaran proyek.

Bagi pengguna yang pertama kali *login* ke dalam aplikasi, setelah proses autentikasi OAuth 2.0 berhasil, diwajibkan untuk melalui tahap *Onboarding*. Tahap ini bertujuan untuk melengkapi data profil yang tidak tercakup dalam *claim* standar dari Microsoft, yang hanya menyediakan nama lengkap dan email. Pengguna akan diminta untuk memasukkan informasi penting lainnya seperti Nomor Registrasi Karyawan (NRK) dapat dilihat pada Gambar 3.19, guna menyelesaikan proses registrasi dan mendapatkan akses penuh ke fitur-fitur aplikasi.



Gambar 3.19 Tampilan halaman *onboarding*

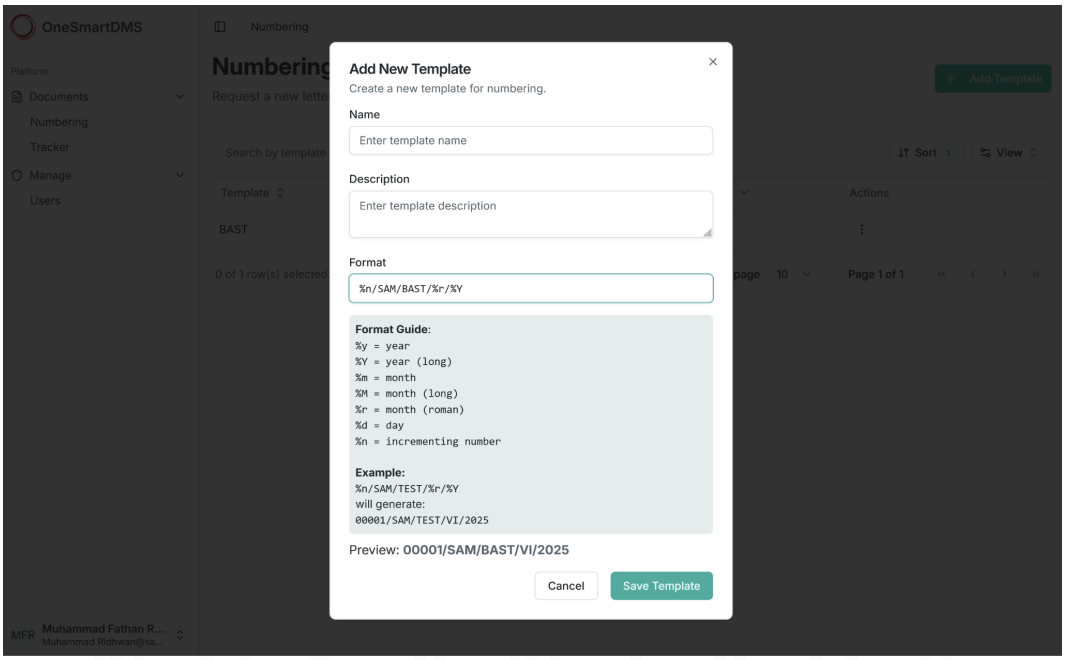
Setelah seorang pengguna menyelesaikan proses *onboarding*, akunnya tidak langsung aktif sepenuhnya. Diperlukan satu langkah verifikasi manual yang hanya dapat dilakukan oleh pengguna dengan peran tertinggi, yaitu *Governor*. Seorang Governor bertugas untuk memverifikasi validitas data pengguna baru tersebut. Setelah verifikasi berhasil, hanya *Governor* yang memiliki wewenang untuk menetapkan *role* yang sesuai untuk pengguna. Fitur administratif tingkat tinggi ini sengaja dibatasi hanya untuk peran *Governor* guna menjaga integritas dan keamanan data.



Gambar 3.20 Fitur *user management*

Untuk mengelola hak akses secara sistematis, aplikasi ini mengimplementasikan model keamanan *Role-Based Access Control* (RBAC). Prinsip RBAC memastikan bahwa setiap pengguna hanya dapat mengakses sumber daya dan fitur yang sesuai dengan wewenang dan tanggung jawabnya. Terdapat beberapa peran yang telah didefinisikan di dalam sistem:

- **GOVERNOR**: Peran dengan wewenang tertinggi yang memiliki akses penuh ke seluruh fitur aplikasi, termasuk manajemen pengguna dan konfigurasi sistem.
- **ADMIN**: Diberikan kepada admin perusahaan yang bertugas melakukan perubahan atau mutasi data, serta memberikan persetujuan (*approval*) dalam alur kerja tertentu.
- **USER**: Peran standar untuk karyawan umum yang menggunakan aplikasi untuk kebutuhan operasional sehari-hari.
- **READ_ONLY**: Peran dengan hak akses terbatas yang hanya dapat melihat data (*view-only*) tanpa bisa mengubahnya, berguna untuk keperluan audit atau tinjauan apabila user melakukan kesalahan.
- **UNBOARDED**: Sebuah peran transisional yang secara otomatis diberikan kepada pengguna baru. Peran ini menandakan bahwa pengguna tersebut belum menyelesaikan proses *onboarding* atau belum diverifikasi oleh *Governor*.



Gambar 3.21 Tampilan pembuatan *numbering template*

Salah satu modul inti yang dikembangkan dalam platform OneSmartDMS adalah Document Numbering. Modul ini berfungsi untuk mengotomatiskan proses pembuatan nomor surat atau dokumen resmi agar konsisten, terstandarisasi, dan bebas dari kesalahan manusia. Sesuai dengan prinsip RBAC yang telah diimplementasikan, fitur untuk menambah dan mengelola *template* format nomor dokumen ini dibatasi. Hanya pengguna dengan peran admin atau lebih tinggi yang memiliki wewenang untuk mengakses dan mengkonfigurasi fungsionalitas ini.

Setiap *template* format nomor dokumen disimpan sebagai sebuah *string* teks yang dapat dikustomisasi dengan menggunakan kode-kode penanda (*placeholders*) khusus. Saat pengguna membuat nomor dokumen baru, sistem akan secara otomatis mengganti kode penanda ini dengan nilai yang sesuai pada saat itu.

Berikut adalah panduan kode penanda yang didukung oleh sistem:

- `%y`: Tahun dalam format dua digit (misalnya, 25).
- `%Y`: Tahun dalam format empat digit (misalnya, 2025).
- `%m`: Bulan dalam format angka (misalnya, 06).
- `%M`: Nama bulan dalam format teks (misalnya, Juni).
- `%r`: Bulan dalam format angka Romawi (misalnya, VI).
- `%d`: Tanggal dalam format angka (misalnya, 16).
- `%n`: Nomor urut yang akan bertambah secara otomatis (*auto-incrementing*) setiap kali nomor baru dibuat dalam periode per-tahun.

Dengan mekanisme ini, proses penomoran dokumen yang sebelumnya manual kini menjadi sepenuhnya otomatis, cepat, dan dapat diandalkan. Implementasi *template* format ini dapat dilihat di potongan kode pada gambar 3.22.

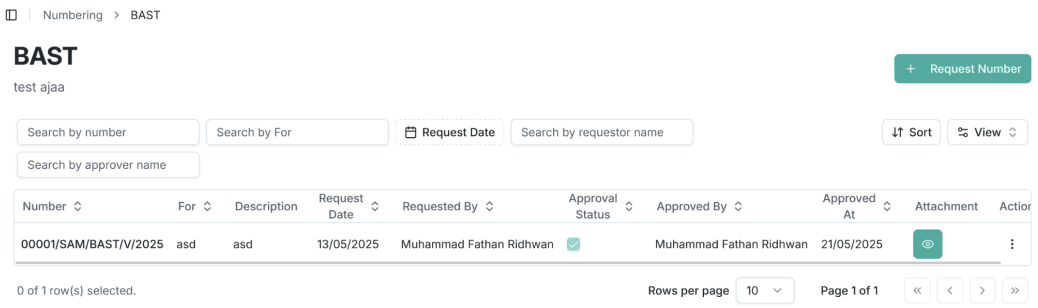
```
export const parseNumberingTemplateFormat = (
  template: string,
  n: number = 1,
  date = new Date()
) => {
  const year = date.getFullYear();
  const month = date.getMonth() + 1;
  const monthLong = date.toLocaleString("default", { month: "long"
});
  const monthRoman = [
    "I",
    "II",
    "III",
    "IV",
    "V",
    "VI",
    "VII",
    "VIII",
    "IX",
    "X",
    "XI",
    "XII",
  ][month - 1];

  const day = date.getDate();
  const incrementingNumber = n.toString().padStart(5, "0");

  const formattedTemplate = template
    .replace(/%y/g, year.toString().slice(-2))
    .replace(/%Y/g, year.toString())
    .replace(/%m/g, month.toString().padStart(2, "0"))
    .replace(/%M/g, monthLong)
    .replace(/%r/g, monthRoman)
    .replace(/%d/g, day.toString().padStart(2, "0"))
    .replace(/%n/g, incrementingNumber);

  return formattedTemplate;
};
```

Gambar 3.22 Potongan kode untuk *formatting template*

Setelah *template* format nomor dokumen dibuat oleh admin, alur kerja untuk pengguna pun dapat dimulai. Pengguna dengan peran *user* dapat

mengajukan permohonan untuk membuat nomor surat baru berdasarkan *template* yang tersedia. Permohonan ini akan masuk ke dalam sebuah antrian persetujuan yang dapat ditinjau oleh pengguna dengan peran *admin*.

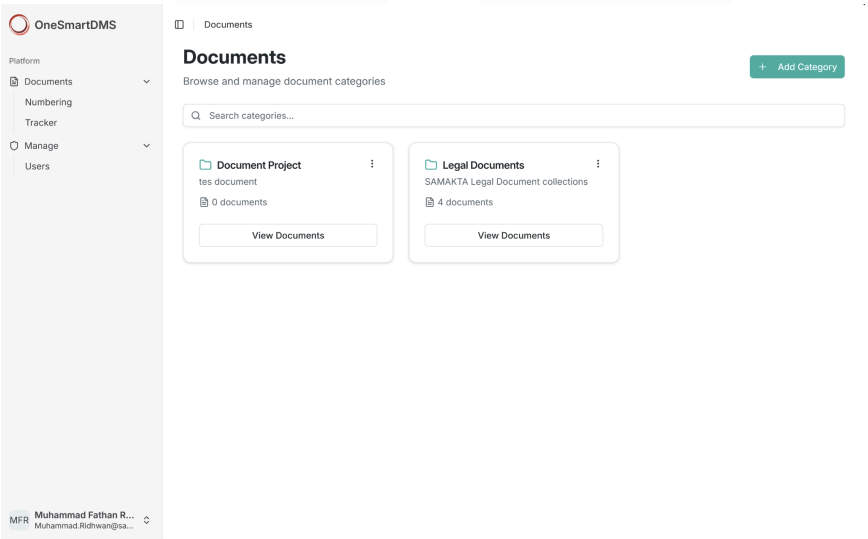
Seorang admin bertugas untuk memvalidasi setiap permohonan yang masuk. Jika tujuan dari pembuatan nomor surat tersebut dianggap valid dan sesuai dengan kebutuhan, admin dapat memberikan persetujuan. Setelah disetujui dan nomor surat berhasil dibuat, proses tidak berhenti di situ. Sebagai langkah terakhir untuk memastikan akuntabilitas, pengguna yang mengajukan permohonan diwajibkan untuk melampirkan dokumen final yang telah menggunakan nomor surat tersebut. Langkah ini berfungsi sebagai *proof of use* dan untuk keperluan pengarsipan, sehingga setiap nomor yang dikeluarkan memiliki rekam jejak dan dokumen rujukan yang jelas.



Gambar 3.23 Tampilan *datatable* nomor surat

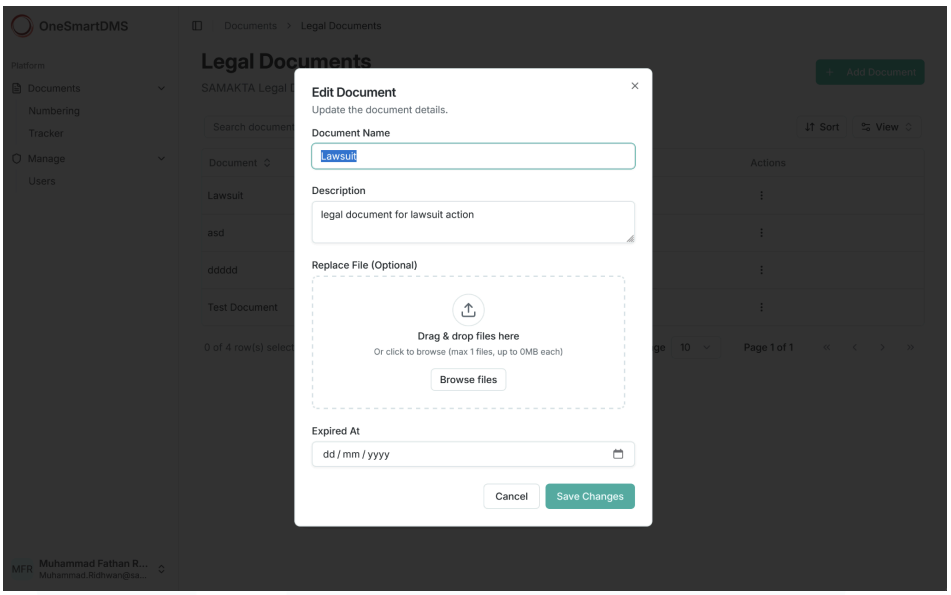
Modul penting kedua yang menjadi bagian dari platform OneSmartDMS adalah *Document Tracking*. Modul ini dirancang secara spesifik untuk menjawab tantangan dalam mengelola dan melacak dokumen-dokumen bernilai tinggi, seperti dokumen legal dan dokumen kemitraan strategis (misalnya NDA, MoU), yang sebelumnya tersebar dan sulit untuk dipantau.

Fitur ini menyediakan sebuah repositori digital terpusat di mana setiap dokumen dapat disimpan dengan aman. Dengan demikian, proses pencarian dokumen dan pemantauan statusnya menjadi jauh lebih efisien.



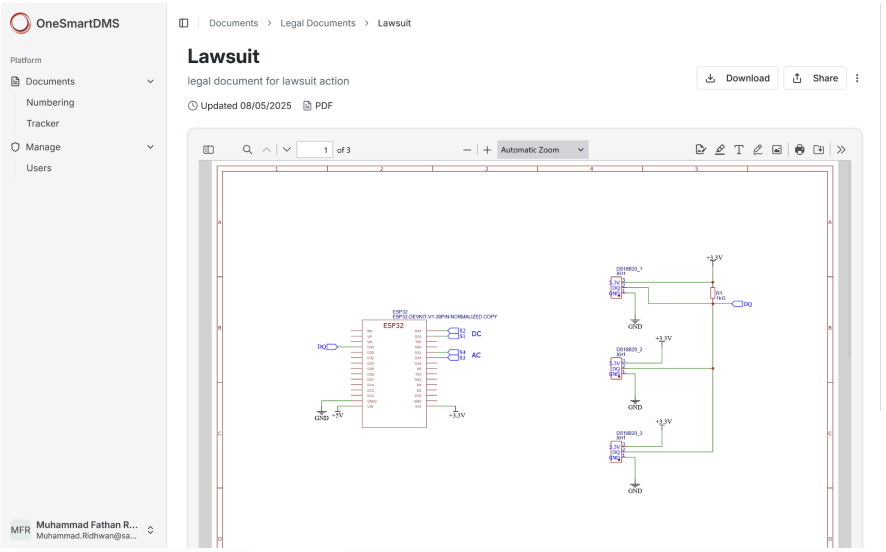
Gambar 3.24 Tampilan utama modul *Document Tracker*

Hanya pengguna dengan peran *governor* dan *admin* yang memiliki hak untuk membuat kategori dokumen baru. Selain itu, semua tindakan yang bersifat mengubah data seperti menambah dokumen baru, memperbarui informasi dokumen, dan menghapus dokumen dari sistem juga hanya dapat dilakukan oleh kedua peran tersebut, menu dapat dilihat pada gambar 3.25. Sementara itu, peran lainnya seperti *user* atau *read_only* memiliki akses yang lebih terbatas, misalnya hanya untuk melihat atau mencari dokumen sesuai dengan hak yang diberikan pada kategori tersebut



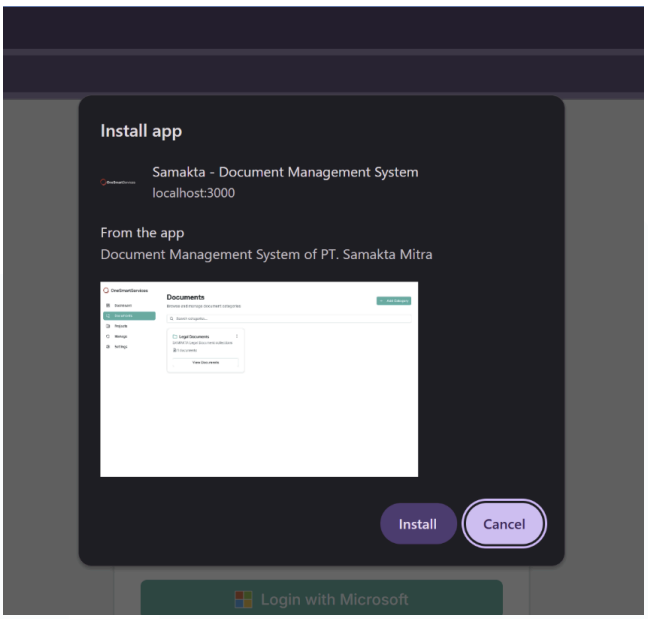
Gambar 3.25 Menu untuk menambahkan atau mengubah dokumen

Selain fitur administratif untuk pengelolaan data, modul *Document Tracking* juga dilengkapi dengan berbagai fitur interaksi guna memudahkan pengguna dalam mengakses dan mendistribusikan dokumen. Pengguna dapat melihat konten dokumen (seperti PDF) secara langsung di dalam aplikasi melalui fitur *preview*, sehingga tidak perlu mengunduhnya terlebih dahulu. Opsi untuk *download* dokumen ke perangkat lokal juga tetap tersedia.



Gambar 3.26 Tampilan *preview* PDF dokumen yang dipilih

Untuk mempermudah proses kolaborasi dan distribusi, diimplementasikan juga fitur *share*. Fitur ini memanfaatkan Web Share API, sebuah antarmuka *browser* modern yang memungkinkan aplikasi untuk menggunakan mekanisme *native sharing* dari sistem operasi perangkat. Saat pengguna memilih opsi ini, akan muncul dialog berbagi bawaan dari perangkatnya (misalnya, opsi untuk berbagi ke WhatsApp, Email, atau aplikasi lainnya), menciptakan pengalaman yang intuitif dan terintegrasi, khususnya pada perangkat *mobile*.



Gambar 3.27 PWA dapat di install di perangkat

Sesuai dengan kebutuhan proyek untuk membuat aplikasi yang dapat di-*install*, *platform* OneSmartDMS mengadopsi teknologi *Progressive Web App (PWA)* untuk memberikan pengalaman pengguna yang menyerupai aplikasi *native*. Implementasi teknisnya berpusat pada penyiapan *Web Manifest*, yaitu sebuah *file* yang mendefinisikan identitas aplikasi seperti nama, ikon, dan tema kepada *browser*.

Dalam *framework* Remix, *file manifest* ini disajikan secara dinamis melalui *route loader* untuk pengelolaan yang lebih fleksibel. Dengan adanya *manifest* yang valid (bersama komponen PWA lain seperti *Service Worker*), *browser* secara otomatis akan memberikan opsi kepada pengguna untuk meng-*install* aplikasi langsung ke *home screen* perangkat *desktop* maupun *mobile*.

3.2.6 SAM - Smart Auto Mobile

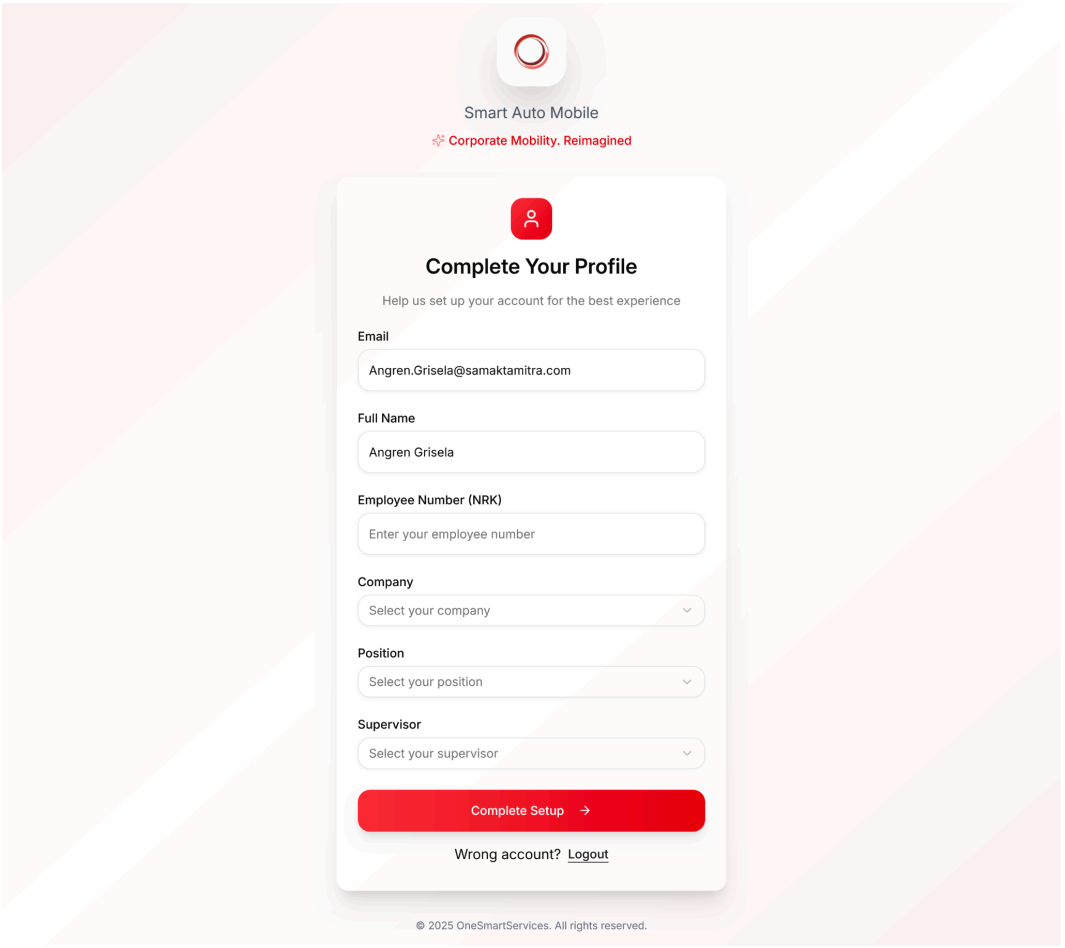
Proyek pengembangan selanjutnya adalah SAM (*Smart Auto Mobile*), sebuah *platform* digital di bawah arahan Bapak Rian selaku *Person in Charge*

(PIC). Tujuan utamanya adalah mendigitalisasi proses peminjaman dan pemantauan kendaraan operasional dengan pendekatan *mobile-first*. *Platform* ini juga dirancang dengan arsitektur *multi-company* agar dapat diadopsi oleh berbagai *Business Unit (BU)* lain. Pengembangannya direncanakan dalam dua fase:

- **Fase 1**: Fokus pada fungsi administrasi dasar, meliputi sistem pemesanan (*booking*), persetujuan, serta pencatatan jarak tempuh dan riwayat perawatan.
- **Fase 2**: Mengimplementasikan teknologi *Internet of Things (IoT)* untuk pelacakan posisi *GPS* secara *real-time* dan pembacaan data diagnostik mesin dari *port OBD-II*.

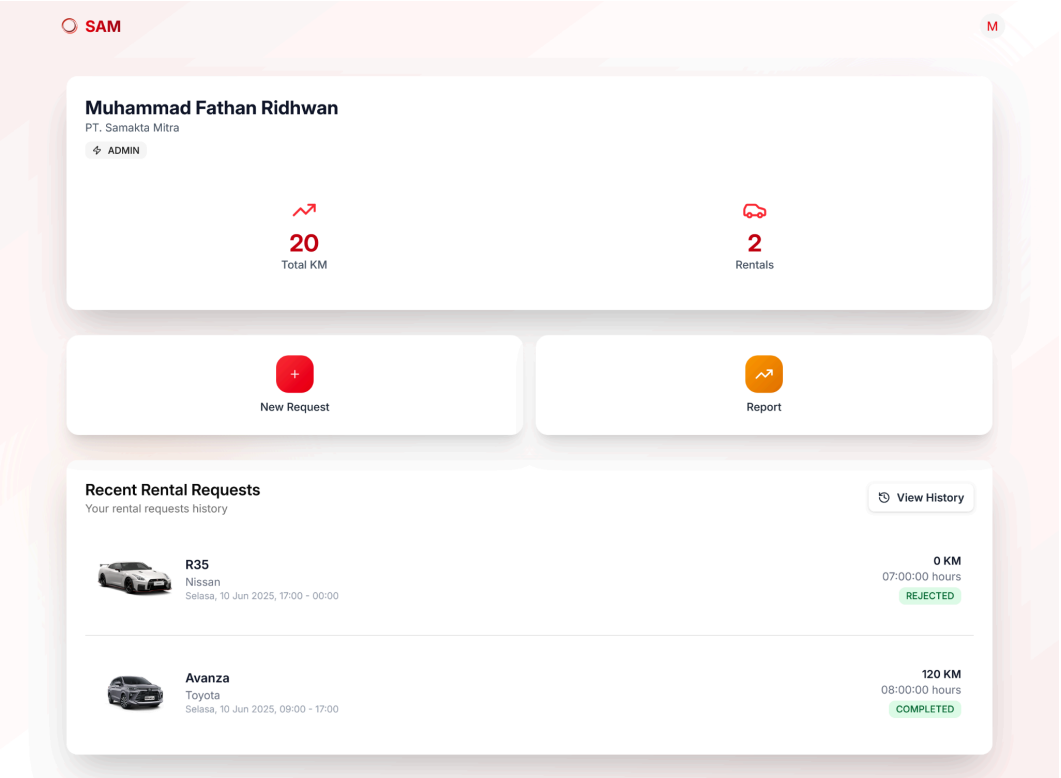
Secara teknis, aplikasi SAM dikembangkan sebagai *Progressive Web App (PWA)* untuk memberikan kemudahan instalasi di berbagai perangkat. *Platform* ini dibangun menggunakan *framework* React dengan React Router v7. Sementara itu, untuk antarmuka pengguna (*UI*), digunakan *library* komponen ShadcnUI untuk memastikan tampilan yang bersih, responsif, dan konsisten.

Untuk aspek keamanan dan manajemen akses, aplikasi SAM mengadopsi mekanisme autentikasi yang konsisten dengan proyek sebelumnya, yaitu menggunakan standar OAuth 2.0. Secara spesifik, proses *login* diintegrasikan dengan platform Microsoft, mengingat seluruh akun karyawan di lingkungan Sinarmas Land terkelola dalam ekosistem tersebut. Pendekatan ini secara efektif menciptakan batasan akses yang aman, di mana hanya pengguna dengan akun resmi dalam ruang lingkup Sinarmas Land yang dapat melakukan autentikasi dan masuk ke dalam aplikasi.



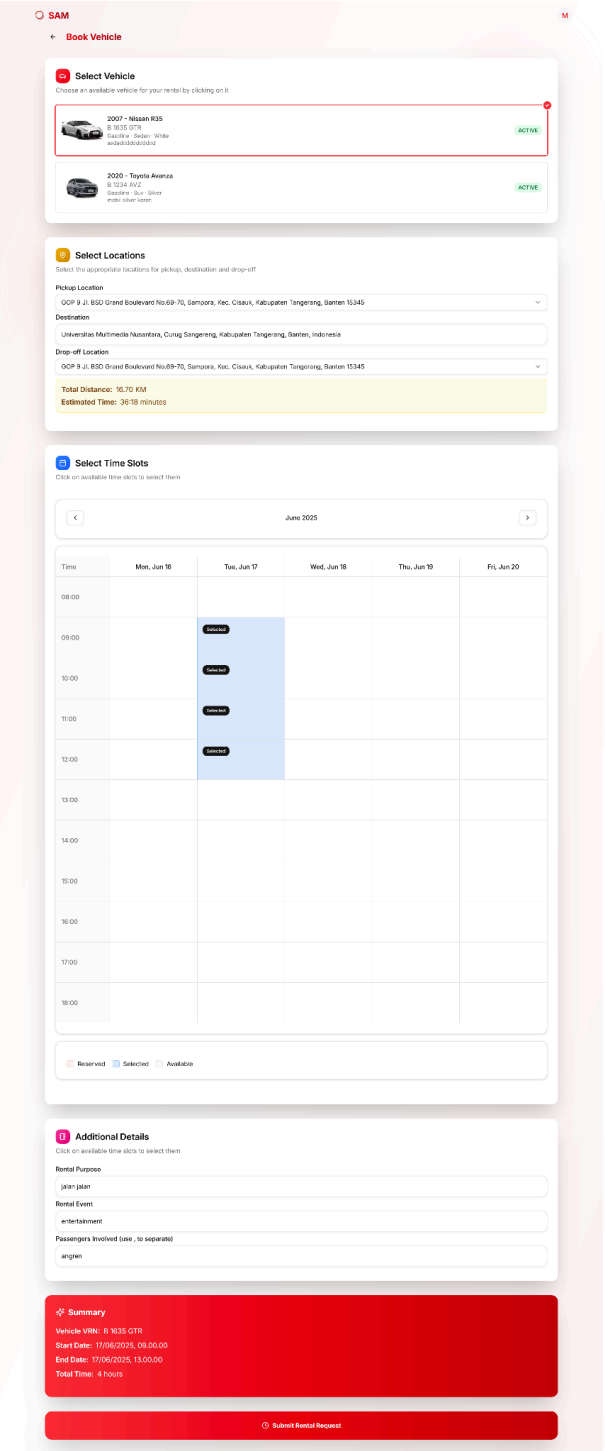
Gambar 3.28 Tampilan halaman *onboarding*

Sama seperti sistem sebelumnya, bagi pengguna yang baru pertama kali mengakses aplikasi SAM, akan diarahkan ke alur proses *onboarding* setelah berhasil melakukan autentikasi seperti pada gambar 3.28. Proses ini esensial untuk melengkapi data profil pengguna, karena informasi yang didapatkan dari *claim* autentikasi Microsoft hanya terbatas pada nama lengkap dan alamat email. Pada tahap ini, pengguna diwajibkan untuk mengisi beberapa data penting lainnya seperti NRK (Nomor Registrasi Karyawan), *Business Unit* tempat mereka bekerja, posisi atau jabatan, dan nama supervisor. Kelengkapan data ini menjadi fondasi agar sistem dapat berfungsi dengan baik sesuai arsitektur *multi-company*.



Gambar 3.29 Tampilan utama SAM untuk pengguna

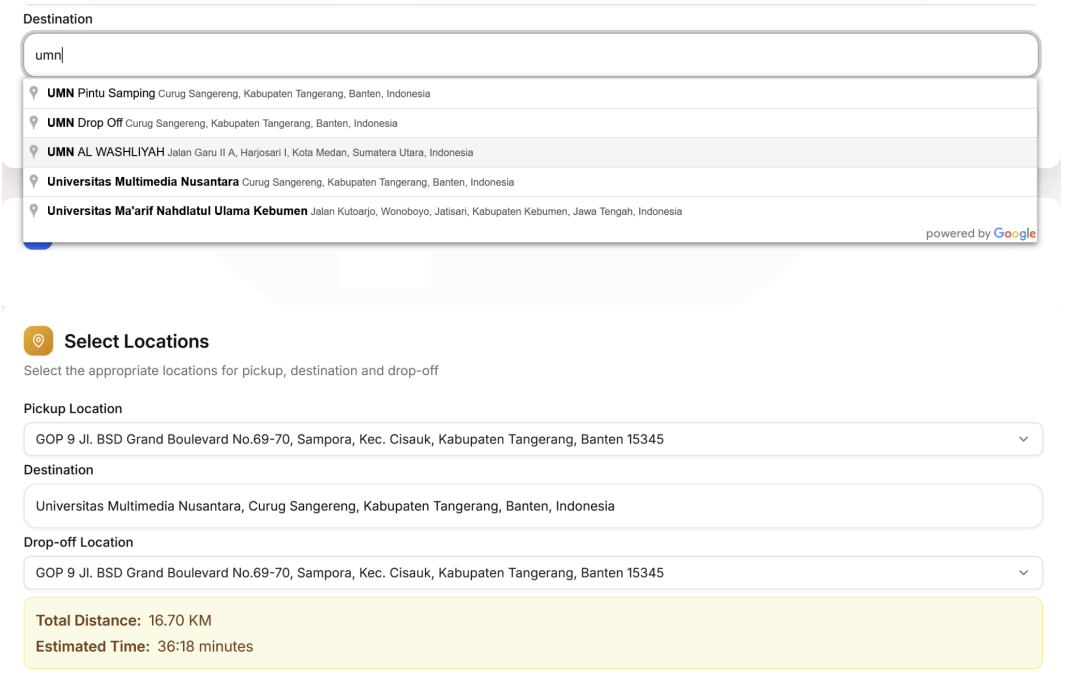
Antarmuka utama aplikasi SAM dirancang sebagai sebuah dasbor yang mengutamakan kemudahan dan kecepatan akses bagi pengguna. Halaman ini menyajikan beberapa komponen kunci: sebuah area untuk statistik penggunaan yang memberikan wawasan singkat terkait aktivitas kendaraan, serta dua tombol aksi utama (*call-to-action*) untuk fungsi yang paling sering digunakan, yaitu memesan kendaraan dan menghasilkan laporan bulanan untuk *reimbursement*. Selain itu, untuk kemudahan referensi, disajikan pula daftar singkat riwayat peminjaman terakhir, sehingga pengguna dapat dengan cepat meninjau kembali perjalanan mereka tanpa harus masuk ke menu yang lebih dalam.



Gambar 3.30 Tampilan halaman *New Booking*

Proses untuk membuat pemesanan baru dirancang agar mudah dan terstruktur. Pada halaman *Request New Booking*, langkah pertama bagi

pengguna adalah memilih kendaraan yang tersedia dari daftar kendaraan. Selanjutnya, pengguna diwajibkan untuk mengisi beberapa rincian penting seperti lokasi tujuan, rentang waktu peminjaman (tanggal dan jam), serta tujuan peminjaman. Seluruh informasi ini dicatat oleh sistem untuk keperluan pelaporan penggunaan kendaraan. Satu aspek penting dalam proses ini adalah penentuan lokasi pengambilan dan pengembalian. Untuk menjaga konsistensi, lokasi-lokasi ini tidak diisi secara bebas, melainkan dipilih dari daftar titik lokasi (*predefined locations*) yang telah ditentukan sebelumnya oleh *governor* dari masing-masing perusahaan.



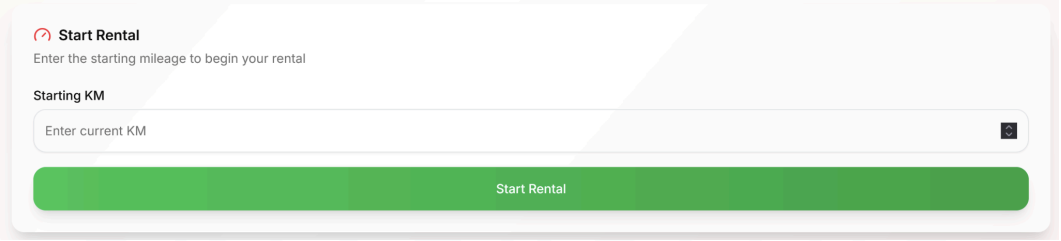
Gambar 3.31 Fitur yang mengimplementasi Google Maps API

Untuk lebih meningkatkan kemudahan dan akurasi dalam proses pemesanan, halaman *New Booking* diintegrasikan dengan beberapa layanan dari Google Maps API. Seperti yang dapat dilihat pada Gambar 3.31, pada kolom isian lokasi tujuan, diimplementasikan fitur Place Autocomplete API. Fitur ini memberikan saran lokasi secara *real-time* saat pengguna mengetik,

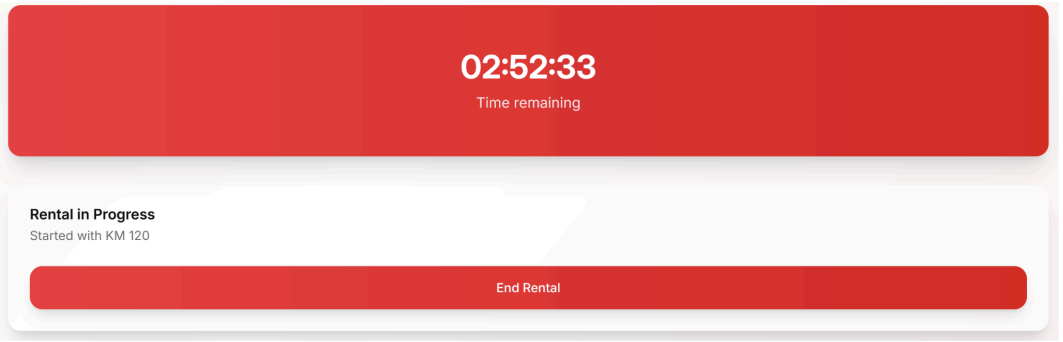
sehingga sangat memudahkan proses pencarian dan mengurangi risiko kesalahan penulisan alamat.

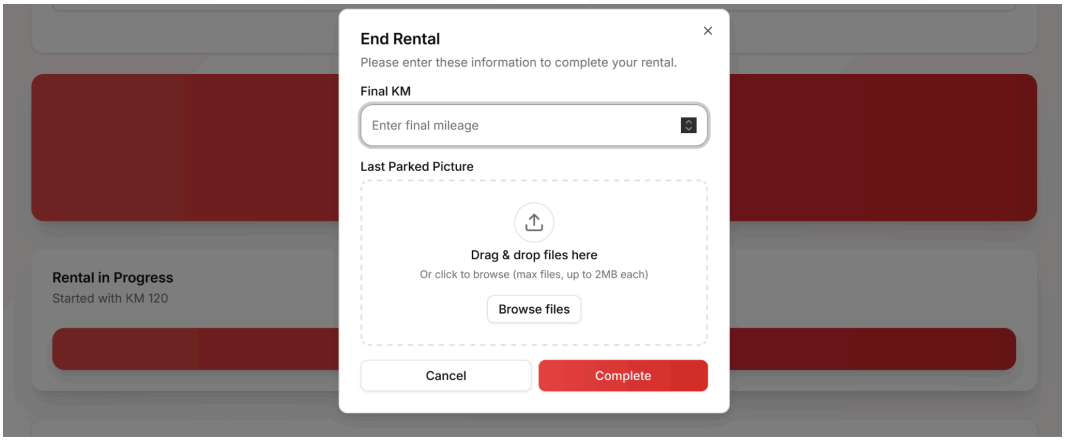
Setelah lokasi tujuan dipilih, aplikasi kemudian memanfaatkan Distance Matrix API untuk secara otomatis menghitung estimasi waktu tempuh dari titik penjemputan ke lokasi tujuan tersebut. Informasi estimasi ini kemudian ditampilkan kepada pengguna. Tujuannya adalah untuk memberikan kesadaran (*awareness*) mengenai durasi perjalanan yang diperlukan, sehingga pengguna dapat menentukan rentang waktu peminjaman kendaraan secara lebih akurat dan realistis, yang pada akhirnya dapat mengurangi kebutuhan untuk meminta perpanjangan waktu.

Dari sisi implementasi dan biaya, pemilihan kedua layanan Google Maps API ini juga didasarkan pada pertimbangan efisiensi. Baik *Place Autocomplete API* maupun *Distance Matrix API* termasuk dalam paket "Essentials" yang menyediakan kuota penggunaan gratis (*free tier*). Meskipun demikian, terdapat *rate limit* untuk kuota gratis ini, yaitu sebanyak 10.000 *requests* per bulan. Jumlah ini dianggap lebih dari cukup untuk kebutuhan operasional pada fase awal peluncuran aplikasi, namun pemantauan penggunaan secara berkala akan diperlukan jika di masa depan aplikasi ini diadopsi secara luas. [3]



Gambar 3.32 Tampilan menu mulai peminjaman



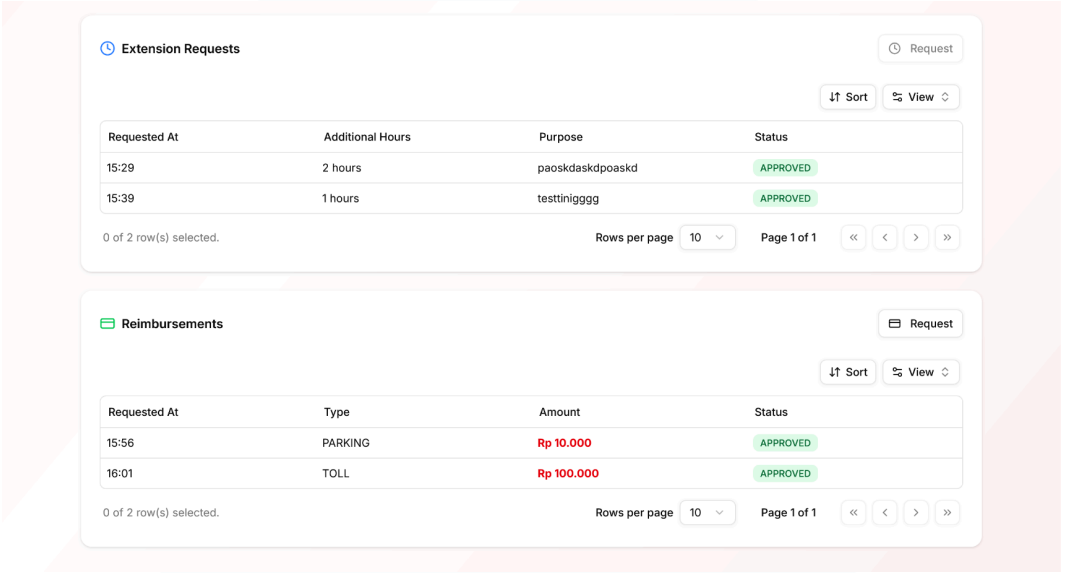


Gambar 3.33 Menu untuk menghentikan peminjaman

Setelah sebuah pemesanan disetujui, pengguna akan berinteraksi dengan perjalanan mereka melalui halaman Booking Detail. Halaman ini dirancang sebagai pusat kendali selama periode peminjaman dan mengadopsi sebuah konsep interaktif yaitu tombol *Start* dan *Stop*.

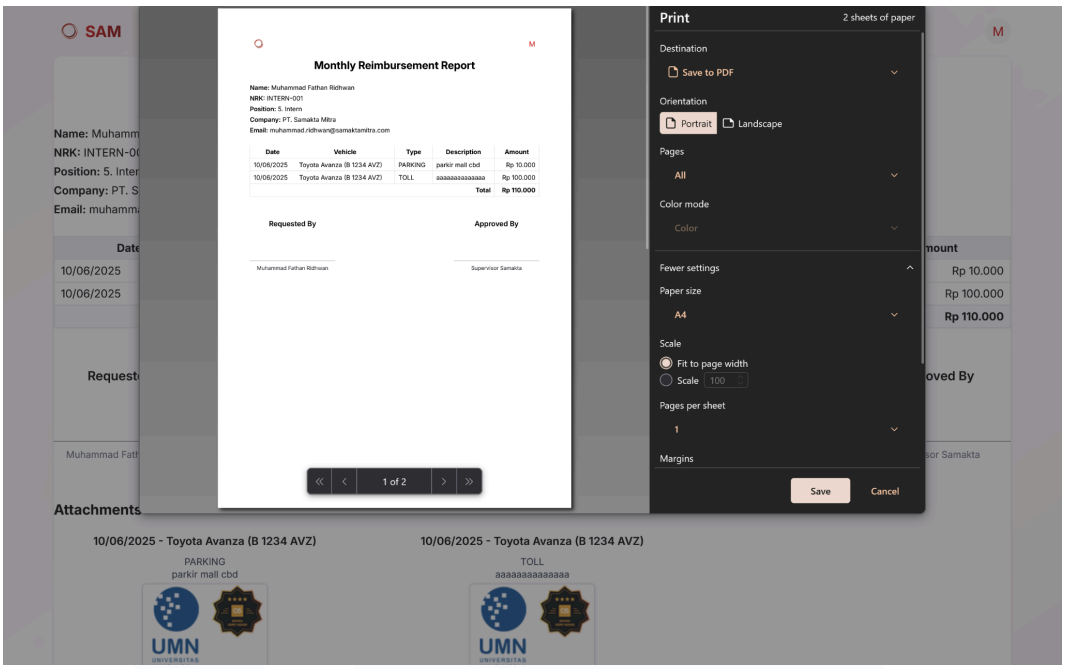
Tujuan utama dari mekanisme ini adalah untuk mendapatkan pencatatan waktu penggunaan yang akurat dan sesuai kondisi nyata (*real-time*), bukan hanya berdasarkan jadwal pemesanan awal. Saat pengguna akan memulai perjalanan, mereka menekan tombol *Start*, yang secara otomatis mengubah status peminjaman menjadi *Ongoing* dan mencatat waktu keberangkatan aktual. Sebaliknya, tombol *Stop* ditekan saat perjalanan telah usai dan kendaraan dikembalikan. Pendekatan ini sangat meningkatkan

integritas dan akurasi data penggunaan kendaraan untuk keperluan pelaporan dan analisis.



Gambar 3.34 Menu pengajuan penambahan waktu dan pengajuan *reimbursement*

Selain mekanisme *start-stop*, halaman detail ini juga memfasilitasi kebutuhan lain yang mungkin muncul. Pengguna dapat mengajukan perpanjangan waktu peminjaman apabila durasi perjalanan ternyata melebihi estimasi awal. Terdapat pula fitur untuk menambahkan klaim *reimbursement*, di mana pengguna dapat mengunggah bukti pembayaran untuk biaya-biaya yang terkait dengan perjalanan dinas, seperti biaya tol atau parkir, untuk diproses lebih lanjut.



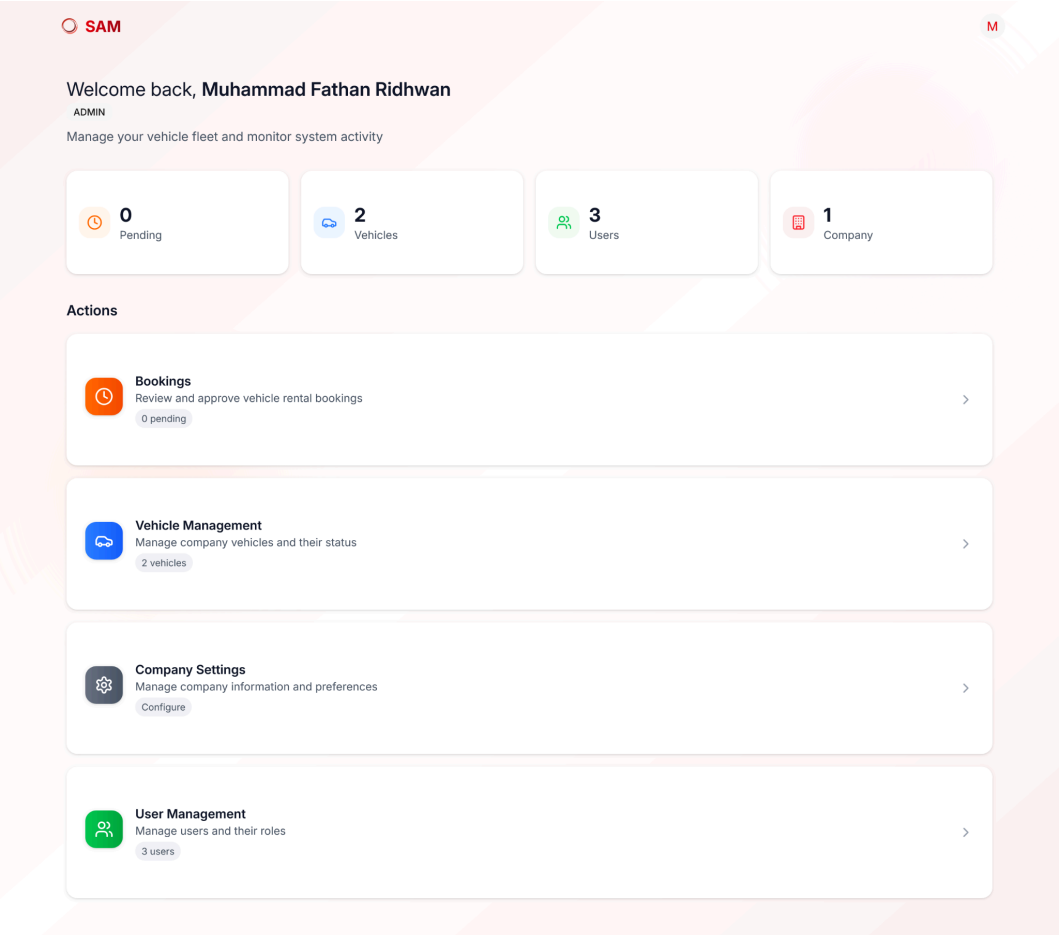
Gambar 3.35 Fitur *print out reimbursement*

Untuk memfasilitasi proses penggantian biaya perjalanan, pengguna dapat mengakses fitur Laporan *Reimbursement* Bulanan langsung dari *dashboard* utama. Saat fitur ini digunakan, sistem akan menarik semua data klaim *reimbursement* yang telah dicatat oleh pengguna pada halaman detail peminjaman selama periode satu bulan. Data tersebut kemudian disusun menjadi sebuah dokumen laporan yang rapi dan siap cetak, berisikan rincian setiap pengeluaran beserta totalnya. Laporan terstandarisasi ini dapat diserahkan sebagai bukti pengajuan *reimbursement*, membuat proses klaim menjadi lebih cepat, transparan, dan mudah dilacak.

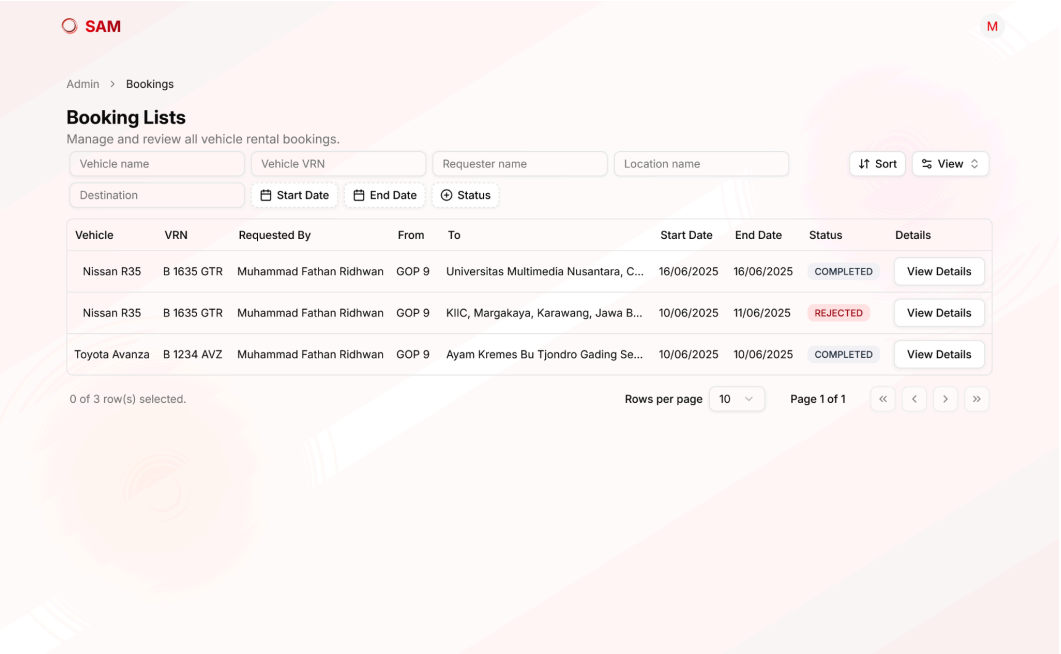
Dalam aplikasi SAM, pengguna dengan peran *administrator* atau *governor* memegang tanggung jawab utama dalam mengelola dan mengawasi seluruh aktivitas operasional yang terjadi di dalam sistem. Tugas inti mereka adalah bertindak sebagai validator, di mana mereka harus meninjau dan memberikan persetujuan untuk setiap permohonan yang diajukan oleh pengguna, seperti permintaan peminjaman kendaraan atau perpanjangan waktu.

Untuk menjalankan tugas-tugas tersebut, mereka diberikan akses ke sebuah Panel Administrasi yang terpusat. Panel ini terbagi menjadi empat menu utama untuk memudahkan pengelolaan:

1. **Bookings (Pemesanan):** Halaman untuk melihat, mengelola, dan menyetujui seluruh permintaan pemesanan kendaraan yang diajukan oleh pengguna di perusahaan mereka.
2. **Vehicle Management (Manajemen Kendaraan):** Menu untuk menambah, memperbarui, dan mengelola data seluruh armada kendaraan operasional, termasuk informasi mengenai status, jadwal *maintenance*, dan ketersediaan.
3. **Company Settings (Pengaturan Perusahaan):** Area khusus, terutama bagi Governor, untuk mengkonfigurasi pengaturan spesifik perusahaan, seperti mendaftarkan titik lokasi penjemputan dan pengembalian (*predefined locations*).
4. **User Management (Manajemen Pengguna):** Fitur untuk mengelola pengguna yang terdaftar di bawah perusahaan tersebut, termasuk melihat detail dan mengatur status atau peran mereka.



UNIVERSITAS MULTIMEDIA NUSANTARA

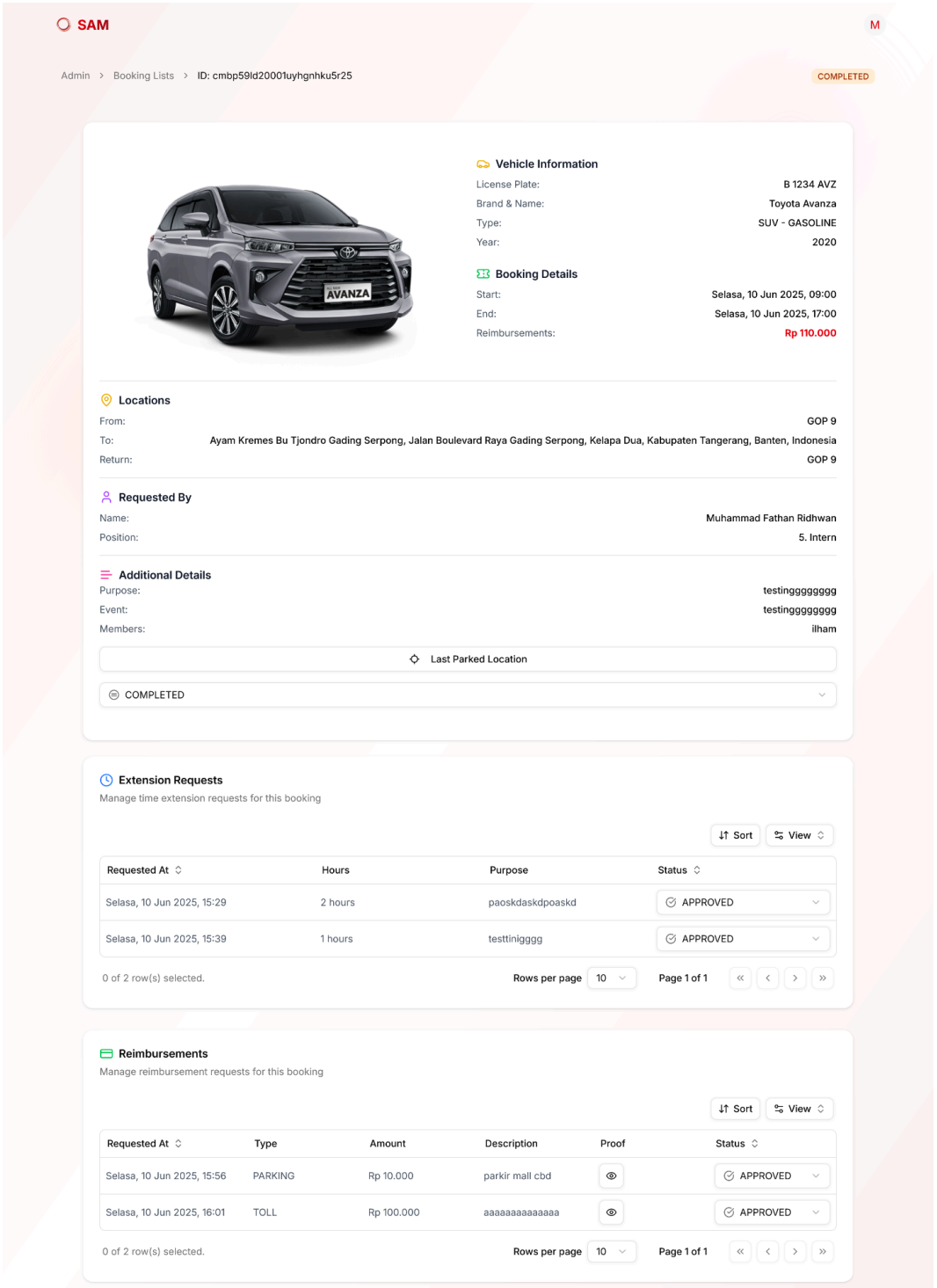


Gambar 3.36 Tampilan *booking lists*

Di dalam menu *Bookings*, admin atau governor akan disajikan sebuah halaman daftar pemesanan (*Booking Lists*) yang berfungsi sebagai *dashboard* pemantauan utama. Halaman ini menampilkan seluruh data pemesanan dalam format *datatable* yang informatif, memungkinkan admin untuk melakukan pencarian dan penyaringan dengan mudah. Informasi kunci yang ditampilkan pada setiap baris mencakup nomor kendaraan, nama pemohon (*requestor*), lokasi tujuan, dan status pemesanan saat ini (misalnya, *Pending, Approved, On-Trip*).

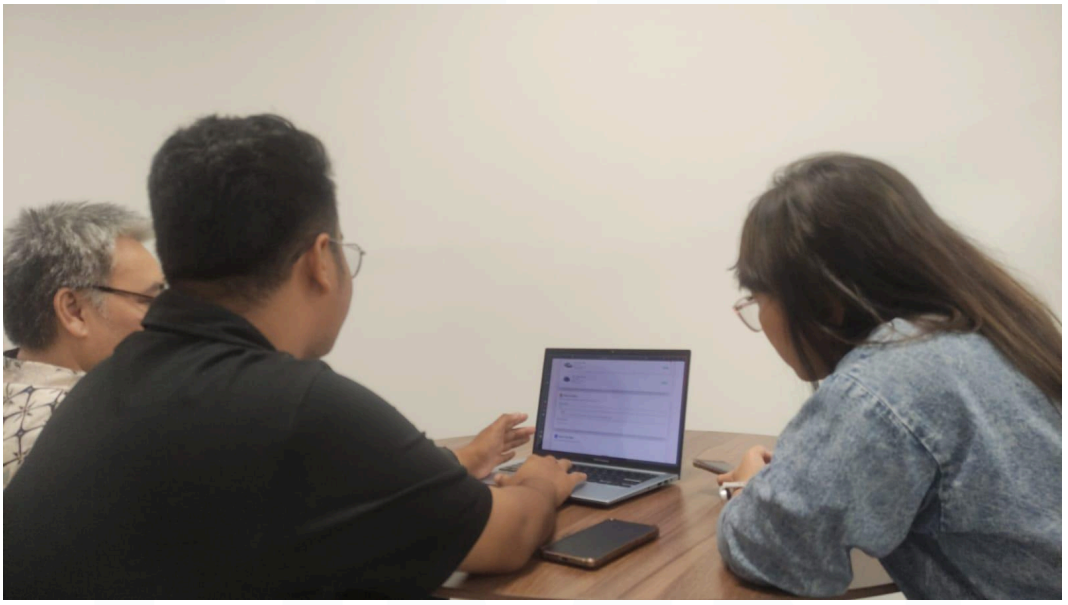
Ketika seorang admin memilih salah satu entri dari daftar tersebut, mereka akan diarahkan ke halaman Detail Pemesanan (*Booking Detail*) pada gambar 3.37. Halaman ini merupakan pusat kendali di mana admin atau governor dapat mengambil tindakan spesifik. Jika status pemesanan masih 'Pending', admin dapat memberikan persetujuan atau menolak permintaan peminjaman tersebut. Apabila peminjaman sedang berlangsung atau telah selesai, halaman ini akan menampilkan opsi untuk meninjau dan menyetujui

permintaan perpanjangan waktu atau klaim *reimbursement* yang diajukan oleh pengguna.



Gambar 3.37 Halaman *booking details* pada admin

Sebagai tahap validasi sebelum proses *deployment*, dilaksanakan sebuah sesi demonstrasi *Proof of Concept* (POC) untuk aplikasi SAM. Dalam sesi ini, dipresentasikan fitur-fitur utama yang telah berhasil diimplementasikan hingga saat itu. Selain itu, dijelaskan juga beberapa penyesuaian dan evolusi fungsionalitas yang berbeda dari *backlog* atau rencana awal proyek. Demonstrasi ini bertujuan untuk memastikan keselarasan pemahaman dan mendapatkan persetujuan akhir dari Pak Rian sebelum aplikasi dirilis ke lingkungan *production*.



Gambar 3.38 Demonstrasi POC SAM

3.3 Kendala yang Ditemukan

Selama periode magang, terdapat beberapa kendala utama yang dihadapi dalam proses pengembangan berbagai proyek. Kendala-kendala ini menjadi pelajaran penting dalam hal manajemen waktu, sumber daya, dan komunikasi di lingkungan kerja yang dinamis. Berikut adalah rincian dari setiap tantangan yang ditemukan:

1. **Keterbatasan Sumber Daya Manusia**

Proses pengembangan proyek sering kali dilakukan secara mandiri, mengingat tidak adanya personel khusus lain yang dapat berfokus pada *web development* ataupun desain *UI/UX*. Kondisi ini mengharuskan untuk merancang desain antarmuka dari awal sekaligus melakukan implementasi teknisnya. Akibatnya, alur kerja menjadi lebih panjang karena tanggung jawab desain dan *development* harus ditangani hanya oleh satu orang.

2. **Manajemen Waktu dan Prioritas**

   Adanya proyek yang berjalan secara bersamaan (*overlapping*) serta aktivitas di luar lingkup pengembangan seperti *meeting* mendadak atau pengerjaan tugas lainnya menyebabkan fokus terbagi. Hal ini berdampak pada jadwal pengerjaan, di mana beberapa fitur dalam *backlog* utama mengalami keterlambatan implementasi karena harus menyesuaikan dengan prioritas lain yang lebih mendesak.

3. **Kesenjangan Informasi Awal**

   Pada beberapa sesi *planning* dan *brainstorming*, terdapat detail fungsionalitas yang terlewat atau tidak tersampaikan secara menyeluruh. Kesenjangan informasi ini baru teridentifikasi di tengah proses pengembangan, sehingga mengakibatkan adanya pekerjaan ulang dan revisi tambahan yang tidak diantisipasi sebelumnya.

4. **Penambahan Fitur di Tengah Pengembangan (*Scope Creep*)**

   Selama proses pengembangan berlangsung, beberapa kali muncul permintaan fitur tambahan dari *user* atau pihak terkait. Meskipun bertujuan baik untuk menyempurnakan produk, permintaan ini secara langsung menambah beban pada *backlog* pengerjaan. Hal ini menjadi tantangan tersendiri, terutama ketika fitur-fitur inti yang telah direncanakan sebelumnya bahkan belum selesai diimplementasikan.

3.4 Solusi atas Kendala yang Ditemukan

Untuk mengatasi berbagai kendala yang ditemukan selama proses magang, diterapkan beberapa strategi dan pendekatan proaktif. Solusi-solusi ini tidak hanya bertujuan untuk menyelesaikan masalah yang ada, tetapi juga untuk meningkatkan efisiensi dan kualitas kerja secara keseluruhan. Berikut adalah solusi yang diimplementasikan untuk setiap kendala:

1. **Mengatasi Keterbatasan Sumber Daya dengan Efisiensi**

   Untuk mempercepat proses kerja yang dilakukan secara mandiri, mahssiwa memaksimalkan penggunaan *tools* AI dan sumber daya yang ada. Dalam tahap desain, mahasiswa memanfaatkan *library* komponen *UI Kit* (seperti ShadcnUI) dan membuat desain aplikasi dengan AI untuk membuat *mockup* fungsional. Pendekatan ini secara drastis mengurangi waktu yang dibutuhkan pada fase desain dan memungkinkan untuk lebih cepat masuk ke tahap *development*.

2. **Penerapan Manajemen Tugas yang Terstruktur**

   Untuk mengatasi jadwal yang tumpang tindih (*overlapping*), mahasiswa menerapkan sistem manajemen tugas pribadi menggunakan papan kanban (*kanban board*) untuk memvisualisasikan seluruh pekerjaan dan tenggat waktunya.

3. **Proses Konfirmasi Proaktif sebelum Pengembangan**

   Guna menghindari kesenjangan informasi, mahasiswa mengubah pendekatan dari pasif menunggu instruksi menjadi aktif meminta konfirmasi. Setelah sesi *planning* atau *brainstorming* mahasiswa akan membuat notulen.

4. **Pengelolaan *Backlog* untuk Mengendalikan *Scope Creep***

Untuk menangani permintaan fitur tambahan di tengah jalan, setiap permintaan baru tidak langsung dikerjakan. Sebaliknya, permintaan tersebut didokumentasikan dan dimasukkan ke dalam *backlog* proyek. Kemudian, bersama *Person in Charge (PIC)*, kami akan meninjau dan memprioritaskan fitur tambahan tersebut, menimbangnya dengan beban kerja yang ada. Pendekatan ini memastikan bahwa ruang lingkup proyek (*scope*) tetap terkendali dan setiap penambahan fitur dilakukan secara strategis tanpa mengorbankan target utama.