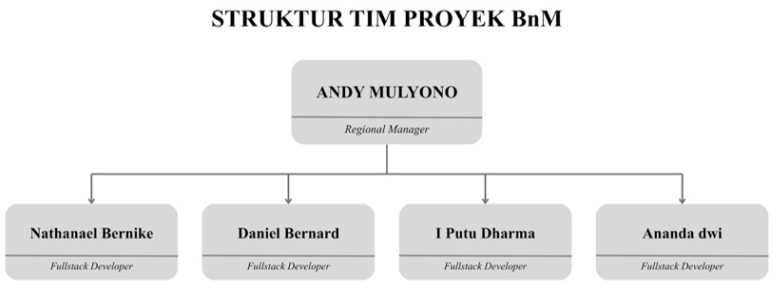
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kerja praktek magang di PT Sumber Alfaria Trijaya Tbk, mahasiswa ditempatkan pada Departemen *Building Management and Maintenance* (BMM) sebagai *Fullstack Developer Intern*. Dalam kegiatan magang ini, mahasiswa bertanggung jawab untuk merancang dan membangun aplikasi berbasis web yang berfungsi sebagai sistem pencatatan progres pembangunan toko baru Alfamart. Proyek ini bersifat mandiri, di mana mahasiswa secara langsung menangani seluruh aspek pengembangan sistem tanpa adanya tim pengembang internal dari pihak BMM.

Mahasiswa tidak memiliki pembimbing atau mentor khusus dalam pelaksanaan magang, sehingga seluruh koordinasi dan pengarahan dilakukan langsung oleh Manager Departemen BMM selaku *Person In Charge* (PIC) proyek digitalisasi ini. Mahasiswa melakukan *brainstorming* bersama manager untuk memahami kebutuhan sistem, menentukan rancangan antarmuka, serta menyusun strategi pengembangan fitur sesuai dengan kebutuhan operasional departemen.



Gambar 3.1. Struktur Proyek BnM

Kedudukan dan struktur koordinasi dalam proyek dapat dijelaskan sebagai berikut:

- **Manager (PIC Proyek):**
  - Memberikan arahan, penjelasan kebutuhan sistem, serta menentukan prioritas fitur yang akan dikembangkan.

  - Menjadi penghubung antara pihak manajemen dengan mahasiswa untuk memastikan hasil pengembangan sesuai standar perusahaan.
  - Melakukan evaluasi hasil kerja dan memberikan masukan terhadap sistem yang telah dikembangkan.

- **Mahasiswa Magang (*Fullstack Developer Intern*):**
  - Bertanggung jawab penuh terhadap perancangan, pengembangan, integrasi, dan pengujian *system*.
  - Melaksanakan seluruh tahapan pengembangan aplikasi secara mandiri mulai dari analisis kebutuhan, pembuatan *frontend*, *backend*, hingga *deployment*.
  - Melakukan pelaporan progres kerja dan hasil implementasi secara langsung kepada manager departemen.

- **Teknologi dan Infrastruktur yang Digunakan:**
  - ***Frontend*:** React.js, HTML, CSS, dan JavaScript untuk pembuatan antarmuka pengguna.
  - ***Backend*:** Node.js, Python untuk pengelolaan logika server dan komunikasi dengan *database*.
  - ***Database*:** Google Spreadsheet sebagai penyimpanan data utama yang diintegrasikan melalui Google Apps Script dan API.
  - ***Hosting*:** Render sebagai *hosting backend* dan Vercel sebagai *hosting frontend* untuk keperluan akses sistem secara daring.

- **Alur Komunikasi dan Koordinasi:**
  - Komunikasi dilakukan langsung antara mahasiswa dan manager melalui pertemuan tatap muka, WhatsApp.
  - Setiap kali terdapat arahan atau perubahan kebutuhan sistem, mahasiswa akan mendiskusikannya bersama manager untuk memastikan hasil sesuai dengan harapan.
  - Evaluasi dilakukan secara berkala berdasarkan hasil implementasi fitur dan performa aplikasi yang telah diuji coba di lapangan.

- **Pengerjaan Tugas:**

- Tugas langsung dikerjakan setelah mahasiswa mendapatkan penjelasan dan arahan dari manager Departemen BMM mengenai sistem atau fitur yang perlu dikembangkan. Setiap fitur yang dikerjakan disesuaikan dengan kebutuhan departemen dan diarahkan untuk mendukung proses digitalisasi pencatatan progres pembangunan toko baru Alfamart.
- Bertanggung jawab untuk mengimplementasikan seluruh tahapan pengembangan sistem secara mandiri, mulai dari pembuatan antarmuka (*frontend*) menggunakan React.js, HTML, CSS, dan JavaScript, hingga pembuatan logika server (*backend*) dengan Node.js yang terhubung ke Google Spreadsheet melalui Google Apps Script API.
- Infrastruktur sistem di-*hosting* menggunakan Render untuk *backend* dan Vercel untuk *frontend*, sehingga aplikasi dapat diakses secara daring oleh pihak internal BMM. Mahasiswa juga memastikan bahwa setiap fungsi berjalan dengan baik dan dapat menyajikan data secara *real-time*.
- Setiap fitur yang telah selesai dikembangkan akan diuji secara lokal terlebih dahulu sebelum di-*deploy*. Pengujian dilakukan untuk memastikan tidak ada kesalahan koneksi antara *frontend*, *backend*, dan basis data Spreadsheet.
- Dalam pelaksanaan tugas, mahasiswa diharapkan untuk bekerja secara maksimal dan disiplin terhadap tenggat waktu yang telah disepakati bersama manager. Apabila terjadi kendala teknis atau *error* dalam sistem (seperti *bug*, *deployment failure*, atau kesalahan pada API), mahasiswa akan melakukan analisis mandiri untuk mencari solusi, serta melaporkan hasil temuan dan langkah perbaikannya kepada manager sebagai bentuk transparansi kerja.

- **Konsultasi dan Revisi:**
  - Setelah setiap tahap pengembangan sistem selesai dikerjakan, mahasiswa melakukan konsultasi atau asistensi dengan manager Departemen BMM untuk mendapatkan *feedback* terhadap hasil kerja. Evaluasi ini dilakukan untuk memastikan fitur yang dikembangkan sudah sesuai dengan kebutuhan pengguna dan standar sistem internal perusahaan.
  - Mahasiswa dapat melakukan komunikasi dan konsultasi tambahan di luar sesi evaluasi, baik melalui WhatsApp, maupun pertemuan langsung

di kantor, apabila terdapat kendala teknis atau revisi mendesak yang perlu segera diselesaikan.

- Apabila terdapat revisi atau masukan terhadap sistem, mahasiswa akan segera melakukan perbaikan berdasarkan arahan manager. Perbaikan diharapkan dapat diselesaikan dalam waktu yang telah ditentukan, namun apabila revisi bersifat kompleks atau membutuhkan pengujian tambahan, maka waktu pengerjaan dapat disesuaikan dengan kondisi proyek.
- Setiap revisi yang telah diselesaikan akan dikonsultasikan kembali kepada manager untuk memastikan bahwa hasil perbaikan sudah sesuai dengan kebutuhan dan tidak menimbulkan kendala baru pada sistem. Proses ini dilakukan secara berulang hingga fitur dianggap stabil dan siap digunakan dalam lingkungan operasional BMM.

3.2 Tugas yang Dilakukan

Periode magang dimulai pada 17 Juni 2025 hingga 16 Desember 2025 di PT Sumber Alfaria Trijaya Tbk. Penulis ditempatkan pada Departemen *Building Management and Maintenance* (BMM) dengan peran sebagai *Fullstack Developer Intern*. Fokus utama penulis dalam kegiatan magang ini adalah pengembangan modul digitalisasi sistem pencatatan progres pembangunan toko (Sistem Opname) yang menjadi bagian integral dari *Dashboard* BMM.

Dalam pelaksanaan magang ini, penulis bekerja secara kolaboratif sebagai bagian dari tim pengembang (*development team*) bersama rekan magang lainnya. Meskipun penulis memiliki tanggung jawab utama (*ownership*) pada pengembangan fitur Opname, proses perancangan arsitektur sistem dilakukan secara terintegrasi. Penulis aktif berkoordinasi dengan rekan tim untuk menyelaraskan *frontend* berbasis React.js dan *backend* Node.js, memastikan modul Opname dapat berjalan harmonis dengan modul lain (seperti Form RAB dan SPK) yang dikerjakan oleh anggota tim lainnya. Basis data Google Spreadsheet digunakan sebagai penyimpanan terpusat yang diakses bersama oleh berbagai modul dalam sistem.

Sepanjang pelaksanaan magang, penulis berkoordinasi secara intensif baik dengan rekan satu tim maupun langsung dengan Manager Departemen BMM selaku *Person In Charge* (PIC). Koordinasi dengan tim dilakukan untuk mengatasi kendala teknis bersama, seperti integrasi *hosting* dan penyelesaian *bug* pada server.

Sementara itu, koordinasi dengan Manager berfokus pada validasi kebutuhan sistem (*requirements*), pelaporan progres mingguan, serta revisi fitur agar sesuai standar operasional perusahaan. Seluruh proses magang dijalankan dengan disiplin tinggi mengikuti tenggat waktu tim yang telah disepakati.

3.3 Uraian Pelaksanaan Magang

Rincian aktivitas mingguan penulis selama magang dapat dilihat pada Tabel 3.1 berikut:

Tabel 3.1. Detail Pekerjaan yang Dilakukan

Minggu ke-	**Pekerjaan yang dilakukan**
1	• Pengenalan lingkungan kerja, rekan tim, dan penyelesaian administrasi magang. • Mempelajari dasar-dasar *framework* Lit.js (membaca dokumentasi, membuat contoh sederhana). • Eksplorasi fitur komponen web menggunakan Lit.js.
2	• Mulai *coding* kerangka dasar *frontend website* pengganti. • Menerima *feedback* mentor terkait struktur kode dan melakukan revisi agar sesuai standar tim. • Mempelajari dan mengimplementasikan *Shadow DOM* pada Lit.js. • Mempelajari konsep *reactive update* (*requestUpdate*, *signals*, *Preact signals*).
3	• Melakukan *refactoring* kode agar lebih modular dan rapi. • Menerapkan *Context* pada Lit.js untuk manajemen data antar komponen. • Melakukan revisi berulang pada implementasi *Context* untuk memastikan struktur kode mudah dipelihara (*maintainable*).

Tabel 3.1 (Lanjutan)

Minggu ke-	Pekerjaan yang dilakukan
4	• Beralih dari *Context* ke *Signals* sesuai arahan mentor untuk efisiensi alur data. • Mengatasi tantangan minimnya referensi Lit.js dengan eksperimen mandiri. • Menyerahkan hasil proyek (*handover*) kepada mentor dan melakukan revisi akhir. • Penutupan tugas proyek pembelajaran Lit.js.
5	• Bergabung dengan rekan tim untuk mengerjakan Form RAB. • Memahami struktur kode rekan dan membantu proses *debugging*. • Menangani *error* pada *backend* berbasis Google Spreadsheet (masalah *write data*). • Memutuskan migrasi *backend* ke Python untuk mengatasi keterbatasan Spreadsheet dan mulai melakukan penyesuaian kode.
6	• Menjalankan tes internal untuk memvalidasi *backend* Python dan integrasi *frontend*. • Melakukan uji coba dengan staf departemen. • Menemukan dan memperbaiki *bug* kritis terkait token pada server *cloud* (Render). • Memastikan stabilitas sistem setelah perbaikan.
7	• Memulai tahap *testing* dengan pengguna di cabang (eksternal departemen). • Memantau performa aplikasi di lingkungan nyata. • Mengumpulkan *feedback* terkait *error* dan kendala teknis dari staf cabang.
8	• Memperbaiki *bug* input *double* pada form dan masalah server yang muncul kembali. • Mengimplementasikan validasi tambahan pada form. • Melakukan *re-testing* dengan staf cabang untuk memastikan perbaikan efektif.

Tabel 3.1 (Lanjutan)

Minggu ke-	Pekerjaan yang dilakukan
9	• Menerima tugas baru untuk membuat Form Opname. • Membangun antarmuka (*frontend*) Form Opname menggunakan React. • Membuat fitur *Login/Logout* dengan pemisahan peran (PIC dan Kontraktor) menggunakan data *dummy*.
10	• Mengembangkan fitur PDF Generator untuk mencetak hasil opname. • Mempelajari dan mengimplementasikan Google Apps Script (GAS) sebagai *backend* integrasi data RAB. • Membangun *backend* Node.js untuk menangani autentikasi dan pencatatan log *login*.
11	• Membuat logika perhitungan otomatis (selisih volume & harga) untuk *user* PIC. • Mengimplementasikan fitur *freeze data* setelah disimpan. • Mengembangkan *Dashboard* Kontraktor untuk fitur *Approval* (Persetujuan). • Merapikan format hasil unduhan PDF agar sesuai standar laporan resmi.
12	• Menambahkan fitur pemilihan No. ULOK dan Lingkup Pekerjaan. • Melakukan *Deployment*: *Frontend* ke Vercel, *Backend* ke Render. • Menggabungkan (integrasi) Form Opname ke dalam *Dashboard* Web Utama (bersama Form RAB & SPK). • Memperbaiki *bug* halaman *blank* pasca integrasi.
13	• Melakukan tes mandiri (*self-debugging*) pada fitur web opname. • Membantu rekan tim memperbaiki *bug* pada *hosting backend* (Render). • Berdiskusi dan membantu perbaikan masalah pada *website* RAB utama.

Tabel 3.1 (Lanjutan)

Minggu ke-	Pekerjaan yang dilakukan
14	• Melakukan sesi uji coba (*trial*) aplikasi dengan pihak internal Alfamart. • Memastikan alur kerja aplikasi berjalan lancar saat digunakan pengguna internal.
15	• Menerima masukan untuk menambahkan fitur *Reject* bagi Kontraktor. • Mengimplementasikan fitur *Reject* dan kolom Catatan Alasan (*Notes*). • Melakukan tes fitur baru tersebut bersama PIC dan Kontraktor.
16	• Menangani *bug* web *blank* secara tiba-tiba. • Membantu merapikan Form RAB dan memperbaiki *bug* pada Form SPK. • (Catatan: Ada kegiatan *Employee Gathering* pada 29/09).
17	• Fokus utama pada perbaikan masalah *backend* di Render yang menyebabkan instabilitas. • Memastikan web kembali normal dan melanjutkan proses *trial*.
18	• Mengerjakan revisi tambahan pada fitur RAB dan SPK. • Membuat form khusus untuk pematerain dokumen RAB. • Menghadiri rapat koordinasi (Rakor).
19	• Melakukan *trial* internal yang melibatkan PIC dan Manager. • Memperbaiki *bug* yang ditemukan secara langsung saat sesi *trial* berlangsung.
20	• Menerima dan membahas daftar revisi akhir dari Manager. • Menangani insiden masalah server (*downtime*) yang membuat aplikasi tidak dapat diakses selama beberapa hari (29–31 Okt).
21	• Memastikan server kembali normal. • Melanjutkan pengerjaan revisi yang tertunda akibat masalah server.

Memasuki minggu ke-5 hingga akhir periode magang, proses pembelajaran beralih dari pengenalan teknis dasar menuju penerapan praktis dalam proyek nyata

(*real-world project*). Pada fase ini, keterlibatan aktif difokuskan pada kolaborasi tim, pengembangan fitur mandiri, serta pemecahan masalah teknis yang kompleks. Fokus pembelajaran selama periode ini meliputi:

- **Kolaborasi Tim dan Adaptasi Arsitektur Sistem**
  Pada minggu ke-5 hingga ke-8, dipelajari pentingnya adaptabilitas dalam pengembangan perangkat lunak melalui kolaborasi pada proyek Form RAB. Pemahaman alur logika kode yang ditulis oleh rekan tim lain (*code comprehension*) dilakukan untuk membantu proses *debugging*. Selain itu, diperoleh pembelajaran krusial mengenai keterbatasan teknis (*technical limitations*) saat menangani integrasi Google Spreadsheet sebagai basis data. Dilakukan analisis akar masalah pada kegagalan proses tulis data (*write data error*) serta eksplorasi solusi alternatif, mulai dari percobaan menggunakan Python hingga akhirnya menetapkan arsitektur yang lebih stabil.

- **Pengembangan *Fullstack* Modern (React.js & Node.js)**
  Memasuki minggu ke-9 hingga ke-12, pembelajaran berfokus pada pembangunan modul Sistem Opname secara menyeluruh (*end-to-end*). Penguasaan React.js diperdalam untuk membangun antarmuka yang dinamis, termasuk pembuatan fitur *Login* berbasis peran (*Role-based Access Control*) untuk membedakan akses PIC dan Kontraktor. Di sisi *backend*, dilakukan pengembangan server menggunakan Node.js yang terintegrasi dengan Google Apps Script (GAS). Teknik ini dipelajari untuk menjembatani komunikasi data antara aplikasi web dan Spreadsheet secara efisien, serta untuk menangani logika bisnis yang kompleks seperti kalkulasi otomatis selisih biaya (*variance*).

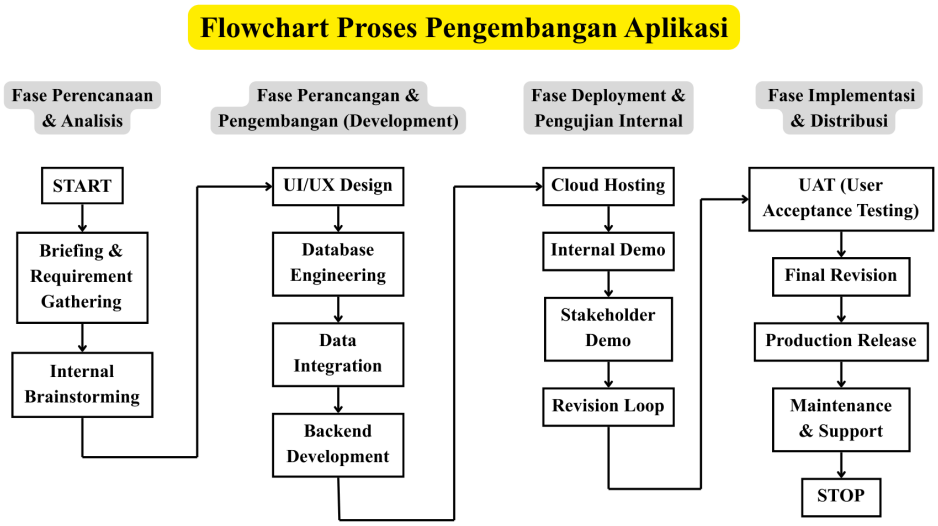
- **Integrasi Sistem dan *Cloud Deployment***
  Pada minggu ke-12, dipelajari proses *deployment* aplikasi ke lingkungan produksi berbasis *cloud*. Layanan Vercel digunakan untuk *hosting frontend* dan Render untuk *backend*. Dalam proses ini, dilakukan konfigurasi *environment variables*, penanganan masalah *Cross-Origin Resource Sharing* (CORS), serta pemastian koneksi antara klien dan server berjalan stabil. Selain itu, teknik integrasi sistem juga dipelajari dengan menggabungkan modul Opname ke dalam *Dashboard* Web Utama perusahaan guna memastikan konsistensi navigasi dan pengalaman pengguna antar-modul.

- **Manajemen Pengujian dan Pemeliharaan (*Testing & Maintenance*)**

Dari minggu ke-13 hingga ke-21, fokus pembelajaran bergeser ke siklus hidup pengujian perangkat lunak. Berbagai metode pengujian dipraktikkan, mulai dari *internal testing*, *User Acceptance Testing* (UAT) dengan manajerial, hingga *live trial* dengan pengguna di cabang. Umpan balik (*feedback*) pengguna dikelola untuk iterasi perbaikan fitur, seperti penambahan fitur *Reject* dan catatan revisi. Selain itu, diperoleh pengalaman nyata dalam penanganan insiden (*incident response*) saat terjadi kendala *downtime* server, yang melatih kemampuan analisis dan pemecahan masalah di bawah tekanan waktu.

3.3.1 Proses Pelaksanaan

Secara umum, pelaksanaan kerja praktik magang berfokus pada transformasi proses administrasi operasional di Departemen *Building Management and Maintenance* (BMM) dari metode konvensional menjadi digital. Tanggung jawab utama difokuskan sebagai pengembang utama (*Lead Developer*) untuk modul Sistem Opname dan Renovasi Toko. Pekerjaan ini meliputi analisis alur kerja opname manual, perancangan arsitektur sistem *fullstack*, pengembangan antarmuka pengguna (*frontend*) yang interaktif, pembangunan logika server (*backend*) yang stabil, hingga integrasi dengan basis data terpusat menggunakan Google Spreadsheet. Selain aspek teknis, dilakukan koordinasi intensif dengan tim pengembang internal untuk memastikan modul Opname terintegrasi secara mulus ke dalam *dashboard* utama departemen, serta pelaksanaan pendampingan (*trial*) kepada pengguna akhir (PIC dan Kontraktor) untuk memastikan sistem berjalan sesuai kebutuhan operasional lapangan.



Gambar 3.2. Flowchart Proses Pengembangan Aplikasi

Flowchart Proses Pengembangan Aplikasi

Proses pengembangan ”Aplikasi Pencatatan Opname Toko Baru” dilakukan melalui pendekatan sistematis yang terbagi ke dalam empat fase utama. Hal ini dilakukan untuk memastikan bahwa aplikasi yang dibangun sesuai dengan kebutuhan operasional di lapangan dan kebijakan perusahaan.

I. **Fase Perencanaan & Analisis**

Pada tahap awal, dilakukan *Briefing & Requirement Gathering* melalui koordinasi langsung dengan Manager untuk memahami kendala dalam pencatatan progres pembangunan toko. Berdasarkan arahan tersebut, dilakukan *Internal Brainstorming* dengan tim intern untuk memetakan alur kerja sistem, menentukan aktor yang terlibat, serta menyusun daftar fitur prioritas yang harus ada di dalam aplikasi.

II. **Fase Perancangan & Pengembangan (Development)**

Fase ini berfokus pada pembangunan teknis aplikasi:

- **UI/UX Design:** Tahap pertama adalah merancang antarmuka pengguna, dengan fokus utama pada halaman *Dashboard* sebagai pusat informasi progres pembangunan.
- **Database Engineering:** Menggunakan Google Sheets sebagai basis data utama. Pemilihan ini didasari atas pertimbangan kemudahan

aksesibilitas bagi staf operasional (orang awam) agar data tetap dapat dipantau secara manual jika diperlukan.

- **Data Integration:** Melakukan integrasi data dari formulir RAB (Rencana Anggaran Biaya) yang sudah ada ke dalam *database* sistem opname agar data pembangunan bersifat sinkron.
- **Backend Development:** Membangun logika server menggunakan Node.js/Express untuk memproses data, mengatur keamanan, dan menjalankan fungsi-fungsi aplikasi sesuai permintaan manajerial.

III. **Fase Deployment & Pengujian Internal**

Sebelum dirilis secara luas, aplikasi melalui tahap validasi:

- **Cloud Hosting:** Aplikasi dideploy menggunakan layanan *free tier* (Vercel untuk *Frontend* dan Render untuk *Backend*) sesuai dengan permintaan efisiensi biaya (*zero cost*).
- **Demo & Revision Loop:** Dilakukan demonstrasi aplikasi secara bertahap, mulai dari tingkat tim intern hingga ke tingkat pemangku kepentingan (*Stakeholder Demo*) yaitu tim BMM di Head Office. Jika terdapat masukan atau kekurangan, sistem akan masuk ke dalam *Revision Loop* (kembali ke tahap pengembangan) untuk dilakukan perbaikan hingga dinyatakan layak.

IV. **Fase Implementasi & Distribusi**

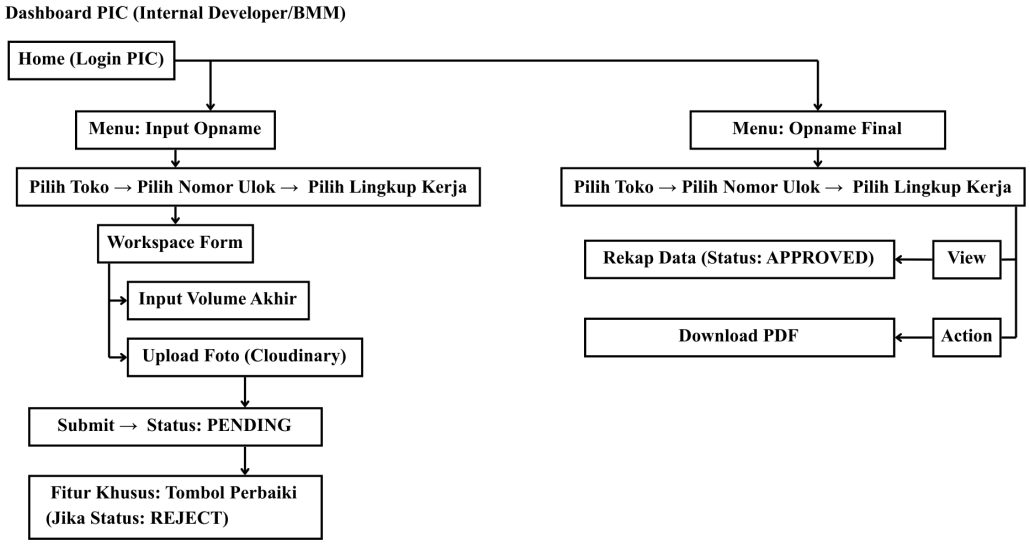
Tahap akhir adalah peluncuran aplikasi secara resmi:

- **UAT (User Acceptance Testing):** Pengujian langsung di lapangan yang dilakukan di beberapa cabang area provinsi Banten untuk melihat performa aplikasi di kondisi nyata.
- **Final Revision:** Penyesuaian akhir berdasarkan umpan balik (*feedback*) dari pengguna di lapangan guna meminimalkan *bug* atau kesulitan penggunaan.
- **Production Release:** Aplikasi mulai di-*broadcast* secara nasional kepada seluruh *Branch Manager* di Indonesia untuk digunakan secara resmi.
- **Maintenance & Support:** Tahap pemeliharaan rutin untuk menjaga stabilitas sistem dan melakukan pembaruan jika terdapat kebutuhan revisi di masa mendatang.

A Proyek 1

SITEMAP PROYEK 1

Sitemap merupakan representasi visual dari struktur navigasi aplikasi yang memetakan seluruh hierarki menu dan alur kerja sistem. Fungsi utama dari *sitemap* ini adalah untuk memberikan gambaran logis mengenai bagaimana pengguna berinteraksi dengan fitur-fitur digitalisasi opname, memastikan proses bisnis dari tahap *input* hingga pelaporan berjalan secara sistematis.



Gambar 3.3. Sitemap Dashboard PIC (PIC/Support)

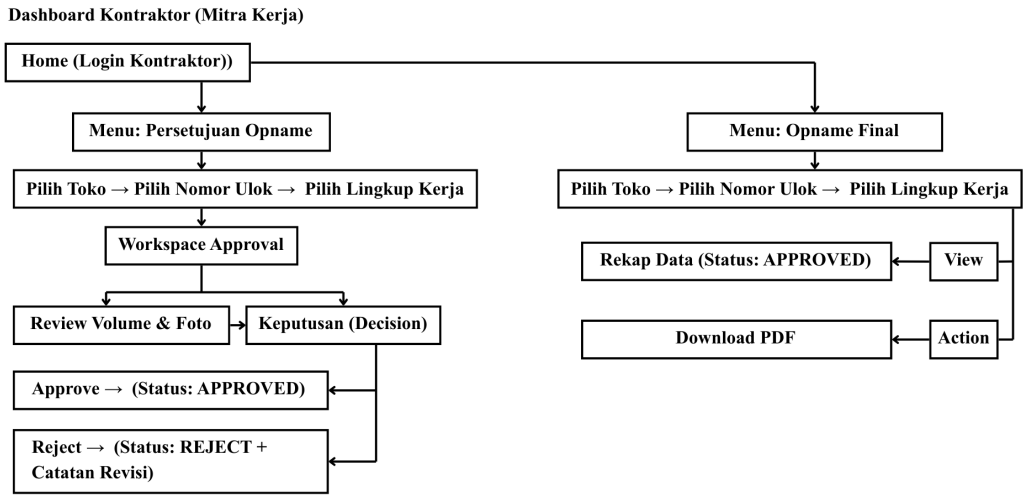
Dashboard PIC (Internal Developer)

Dashboard PIC merupakan antarmuka utama yang dirancang khusus bagi staf operasional internal Departemen BMM. Fungsi utamanya adalah sebagai pusat kendali untuk melakukan manajemen data progres pembangunan toko secara harian. Melalui portal ini, PIC memiliki wewenang untuk menginisiasi pencatatan realisasi pekerjaan, mengunggah bukti fisik, serta memantau status persetujuan dari pihak mitra. Di halaman utama PIC, terdapat dua alur kerja utama:

- **Menu: Input Opname**
  - **Flow Navigasi:** (Pilih Toko → Pilih Nomor ULOK → Pilih Lingkup Kerja).
  - **Workspace (Form Opname):**

    - *Input Volume Akhir*: Memasukkan angka realisasi pekerjaan di lapangan.
    - *Upload Foto*: Mengunggah bukti fisik pekerjaan (sinkronisasi ke Cloudinary).
    - *Action Submit*: Mengirim data ke database Spreadsheet.
    - *Status Tracking*: Data berubah menjadi status ”PENDING” (Menunggu Persetujuan Kontraktor).
    - *Fitur Tambahan*: Tombol ”Perbaiki” (hanya muncul jika kontraktor melakukan *Reject*).

- **Menu: Opname Final**
  - **Flow Navigasi:** (Pilih Toko → Pilih Nomor ULOK → Pilih Lingkup Kerja).
  - **Data View:** Menampilkan rekapitulasi pekerjaan yang sudah berstatus ”APPROVED” (Status: APPROVED).
  - **Action:** *Download* PDF (Menghasilkan dokumen Berita Acara resmi dengan kalkulasi pajak otomatis).



Gambar 3.4. Sitemap Dashboard Kontraktor (Mitra Kerja)

Dashboard Kontraktor (Mitra Kerja)

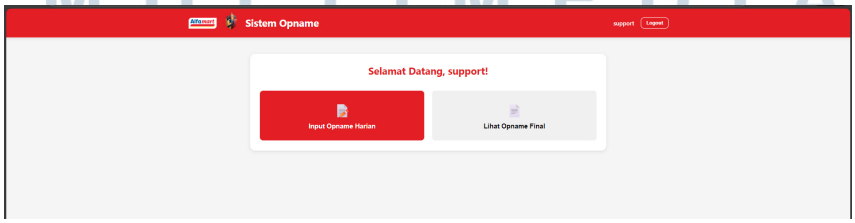
Dashboard Kontraktor berfungsi sebagai portal verifikasi digital bagi pihak

eksternal atau mitra kerja Alfamart. Berbeda dengan PIC, fokus utama dari *dashboard* ini adalah akuntabilitas dan validasi, di mana kontraktor berperan sebagai pemeriksa (verifikator) untuk memastikan bahwa data realisasi yang diinput oleh PIC sudah sesuai dengan kondisi nyata di lapangan dan perjanjian RAB. Sebagai pihak yang melakukan verifikasi data yang diinput oleh PIC:

- **Menu: Persetujuan Opname**
  - **Flow Navigasi:** (Pilih Toko → Pilih Nomor ULOK → Pilih Lingkup Kerja).
  - **Workspace (Approval Page):**
    - *Review*: Memeriksa Volume Akhir dan Foto Bukti yang diunggah PIC.
    - *Keputusan*:
      - ***Approve***: Mengubah status menjadi ”APPROVED” (Data akan muncul di menu Opname Final).
      - ***Reject***: Memberikan catatan revisi (Data dikembalikan ke PIC untuk diperbaiki).
- **Menu: Opname Final (Histori)**
  - **Flow Navigasi:** (Pilih Toko → Pilih Nomor ULOK → Pilih Lingkup Kerja).
  - **Action:** *Download* PDF (Untuk arsip penagihan pihak kontraktor).

Dalam merealisasikan sistem opname (proyek utama) digital ini dengan diterapkan metode pengembangan bertahap (*step-by-step development*) untuk memastikan setiap modul berfungsi optimal sebelum melangkah ke fitur berikutnya. Berikut adalah rincian tahapan teknis yang dilakukan, mulai dari inisiasi antarmuka hingga pengembangan fitur lanjutan:

1. **Perancangan Antarmuka Utama (*Dashboard*) untuk PIC**

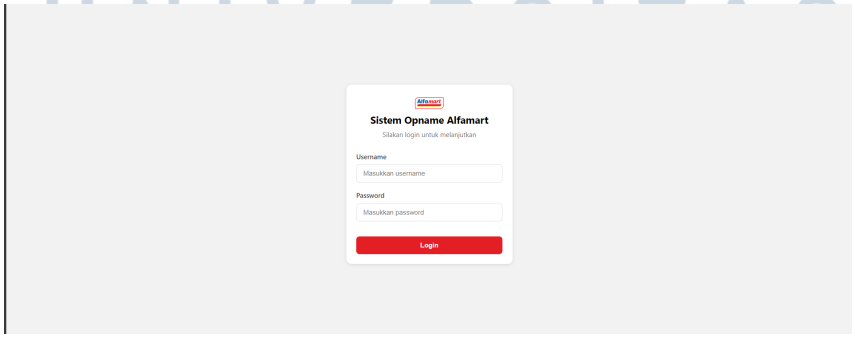


Gambar 3.5. Dashboard PIC

Seperti yang terlihat pada Gambar 3.5 tahap awal pengembangan difokuskan pada pembuatan halaman landas (*landing page*) bagi pengguna internal atau PIC. Dilakukan perancangan komponen `PICDashboard.js` yang berfungsi sebagai pusat navigasi utama. Pada tahap ini, disusun tata letak (*layout*) yang responsif dan intuitif untuk memastikan bahwa PIC dapat dengan mudah mengakses menu-menu krusial segera setelah masuk ke dalam sistem, serta dipersiapkan kerangka *routing* untuk navigasi ke halaman pemilihan toko, Berikut merupakan fitur-fiturnya :

- **Menu Navigasi "Input Opname Harian":** Fitur ini dirancang sebagai gerbang utama bagi PIC untuk memulai alur kerja pencatatan progres. Saat menu ini dipilih, sistem akan mengarahkan pengguna ke halaman pemilihan toko untuk melakukan input data volume realisasi pekerjaan harian secara *real-time*.
- **Menu Navigasi "Lihat Opname Final":** Opsi ini berfungsi untuk memisahkan aktivitas pelaporan dari aktivitas input. Melalui menu ini, PIC dapat mengakses modul rekapitulasi untuk memantau status persetujuan (*approval*) dari kontraktor serta mengakses fitur unduh dokumen Berita Acara (PDF) tanpa risiko mengubah data yang sedang berjalan.
- **Pemisahan Alur Kerja (*Workflow Separation*):** Diterapkan pemisahan navigasi secara visual di *dashboard* untuk meningkatkan pengalaman pengguna (*User Experience*). Dengan membedakan akses antara input data dan tinjauan laporan, sistem dapat meminimalisir kesalahan operasional dan mempercepat akses ke fitur spesifik yang dibutuhkan oleh pengguna.

2. **Implementasi Sistem Autentikasi dan Integrasi Basis Data**

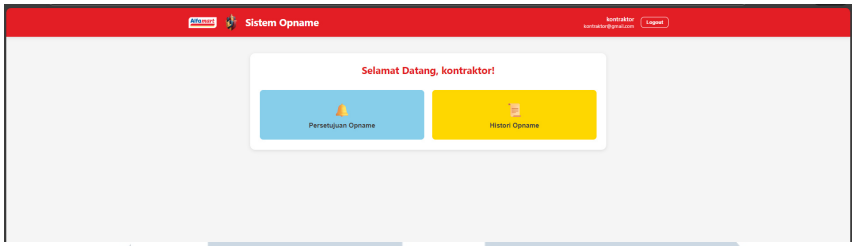


Gambar 3.6. LoginPage

Dikembangkan halaman *Login* yang terhubung langsung dengan *backend* Node.js untuk memproses validasi pengguna. Pada tahap ini, diterapkan logika percabangan di sisi server untuk memverifikasi kredensial dari dua sumber data berbeda pada Google Spreadsheet, yaitu sheet ”users” untuk PIC dan sheet ”data kontraktor” untuk mitra kerja. Sistem autentikasi ini juga dilengkapi dengan manajemen sesi menggunakan *Context API* (`AuthContext.js`) untuk menjaga keamanan akses selama pengguna berinteraksi dengan aplikasi, sebagaimana yang ditampilkan pada Gambar 3.6 yang memiliki fitur sebagai berikut :

- **Mekanisme *Login* Tersegregasi (*Multi-User Login*):** Sistem dirancang untuk membedakan metode input kredensial berdasarkan peran. Untuk pengguna internal (PIC), autentikasi dilakukan menggunakan Akun Email Karyawan SAT. Sedangkan untuk mitra kerja (Kontraktor), akses dilakukan menggunakan alamat email yang terdaftar. Logika percabangan di sisi server akan memverifikasi input ini terhadap dua sumber data berbeda pada Google Spreadsheet, yaitu sheet ”users” dan ”data kontraktor”.
- **Validasi Akses Berbasis Cabang:** Sebagai standar keamanan operasional dalam proyek ini, kata sandi (*password*) yang digunakan dikonfigurasi mengikuti Kode Cabang atau identitas lokasi kerja masing-masing pengguna. Mekanisme ini memudahkan pengelolaan akses sekaligus memastikan bahwa pengguna yang *login* benar-benar memiliki otorisasi terhadap cabang yang bersangkutan.
- **Isolasi Data Spesifik (*Data Scoping*):** Setelah proses *login* berhasil, sistem menerapkan filter data otomatis di sisi *backend*. Aplikasi hanya akan menampilkan daftar toko dan data opname yang relevan dengan cabang atau wilayah kerja pengguna tersebut. Fitur ini mencegah PIC atau Kontraktor mengakses data proyek di luar tanggung jawab mereka, sehingga menjaga kerahasiaan dan integritas data antar-cabang.

3. **Pengembangan *Dashboard* Khusus Kontraktor**

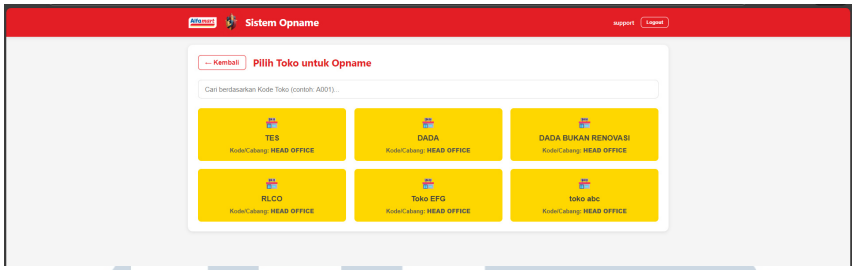


Gambar 3.7. Dashboard Kontraktor

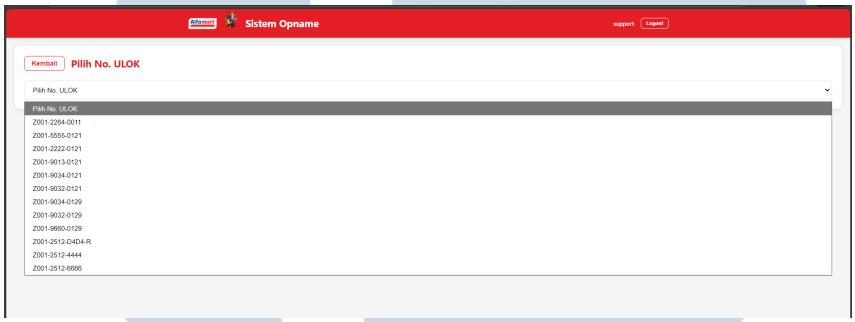
Berbeda dengan tampilan PIC, dikembangkan *dashboard* khusus untuk pengguna Kontraktor seperti pada Gambar 3.7 yaitu dengan antarmuka yang cukup sama dengan *dashboard* PIC. Fokus utama pada halaman ini adalah menu Persetujuan Opname, di mana kontraktor dapat langsung melihat daftar pekerjaan yang memerlukan verifikasi. Pengembangan ini bertujuan untuk memisahkan kepentingan (*concern*) antara penginput data (PIC) dan verifikator (Kontraktor) agar alur kerja menjadi lebih terarah dengan fitur sebagai berikut.

- **Menu Navigasi "Persetujuan Opname":** Fitur ini merupakan gerbang utama bagi kontraktor untuk melakukan tugas verifikasi. Saat menu ini dipilih, sistem akan mengarahkan pengguna ke halaman pemilihan toko, yang kemudian berlanjut ke modul `ApprovalPage`. Di sini, kontraktor dapat melihat daftar pekerjaan yang berstatus "Pending" dan melakukan tindakan persetujuan (*Approve*) atau penolakan (*Reject*).
- **Menu Navigasi "Lihat Opname Final" (Histori):** Disediakan menu khusus untuk akses histori dan pelaporan. Melalui fitur ini, kontraktor dapat meninjau kembali pekerjaan-pekerjaan yang statusnya sudah selesai diproses (baik disetujui maupun ditolak). Fitur ini juga memberikan akses kepada kontraktor untuk mengunduh dokumen Berita Acara (PDF) yang sah sebagai arsip administrasi mereka.
- **Pemisahan Peran Fungsional:** Pengembangan antarmuka ini bertujuan untuk memisahkan kepentingan (*concern*) antara penginput data (PIC) dan verifikator (Kontraktor). Dengan navigasi yang terarah, kontraktor dapat fokus sepenuhnya pada tugas validasi tanpa terganggu oleh fitur input data, sehingga alur kerja menjadi lebih efisien dan minim kesalahan.

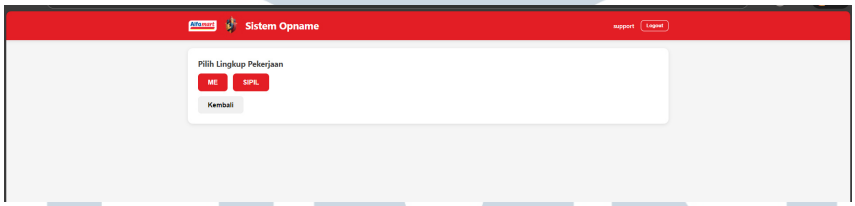
4. **Penerapan Logika Navigasi Bertingkat (Filter Hirarkis) untuk PIC**



Gambar 3.8. Filter Toko Sesuai Cabang



Gambar 3.9. Filter No Ulok Sesuai Toko



Gambar 3.10. Lingkup Kerja Sesuai No Ulok

Sebelum PIC dapat menginput data, diimplementasikan mekanisme validasi bertingkat untuk mencegah kesalahan input lokasi. Disusun alur di mana PIC wajib memilih Kode Toko terlebih dahulu, kemudian sistem akan menyaring Nomor ULOK yang relevan, dan terakhir memilih Lingkup Pekerjaan (seperti Sipil atau ME). Logika ini dibangun dengan memanfaatkan *query parameters* pada API untuk memanggil data yang spesifik sesuai dengan pilihan sebelumnya, dengan fitur yaitu :

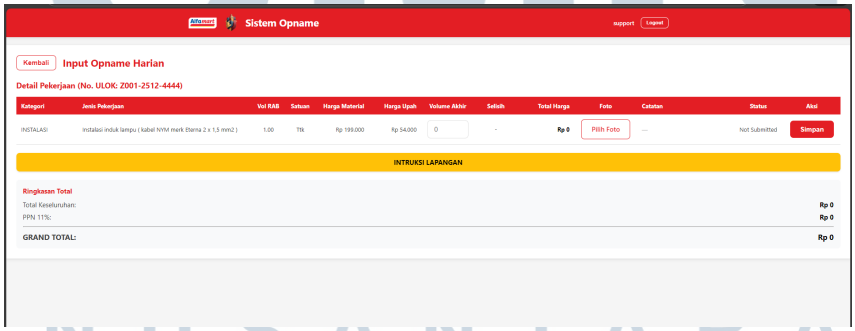
- **Pemilihan Toko (*Store Selection*):** Setelah menekan tombol menu di *dashboard* (baik ”Input Opname” untuk PIC atau ”Persetujuan Opname” untuk Kontraktor), sistem terlebih dahulu menampilkan daftar toko yang relevan seperti pada Gambar 3.8. Pengguna wajib memilih satu toko spesifik untuk membatasi konteks data yang akan diproses.

- **Pemilihan Nomor ULOK (*Project Identification*):** Setelah toko dipilih, sistem akan menyaring dan menampilkan daftar Nomor ULOK (Usulan Lokasi) yang tersedia di toko tersebut seperti pada Gambar 3.9. Langkah ini krusial karena satu toko dapat memiliki beberapa proyek renovasi yang berjalan bersamaan. Pengguna harus memilih nomor ULOK yang tepat agar input data atau persetujuan tidak salah sasaran.
- **Pemilihan Lingkup Pekerjaan (*Scope Filtering*):** Pada Gambar 3.10 merupakan tahap terakhir yaitu pemilihan lingkup kerja (misalnya: Sipil, ME, atau Renovasi). Fitur ini berfungsi sebagai filter akhir untuk memastikan bahwa formulir opname atau halaman persetujuan hanya menampilkan item pekerjaan yang sesuai dengan bidang keahlian atau kontrak yang sedang dikerjakan.
- **Tujuan Validasi Data:** Alur filter yang ketat ini diterapkan untuk meminimalisir kesalahan manusia (*human error*). Bagi PIC, mekanisme ini memastikan data diinput ke proyek yang benar. Bagi Kontraktor, hal ini memastikan persetujuan (*approval*) hanya diberikan pada dokumen yang valid dan spesifik.

5. **Penerapan Logika Navigasi Bertingkat untuk Kontraktor**

   Logika navigasi yang sama diadaptasi untuk sisi Kontraktor. Mekanisme ini memastikan bahwa kontraktor tidak dapat mengakses atau mengintervensi data proyek yang bukan menjadi tanggung jawabnya, sehingga integritas dan kerahasiaan data antarproyek tetap terjaga.

6. **Konstruksi Formulir Opname Cerdas dan Kalkulasi Otomatis**



Gambar 3.11. Input Opname Page PIC

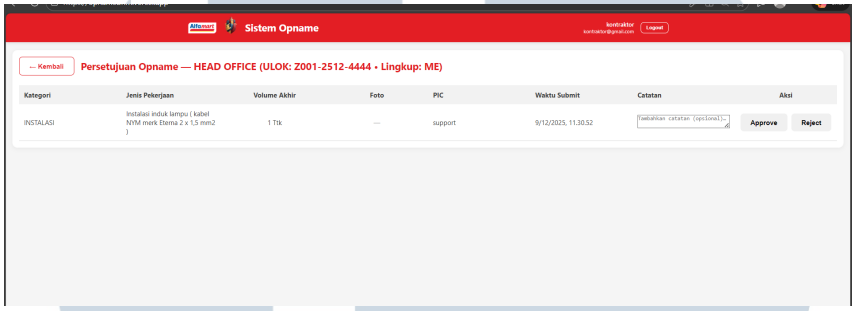
Ini merupakan tahap pengembangan inti, di mana dilakukan pembangunan komponen `OpnameForm.js`. Formulir ini dirancang untuk menarik data

referensi (Kategori Pekerjaan, Jenis Pekerjaan, Volume RAB, Satuan, Harga Material, Harga Upah) secara otomatis dari basis data, sehingga PIC hanya perlu mengisi Volume Akhir dan mengunggah Bukti Foto. Diterapkan algoritma kalkulasi di sisi *client* yang secara *real-time* menghitung selisih biaya (*variance*) dan total harga, serta memberikan indikator visual jika terjadi kelebihan anggaran (*overbudget*) untuk membantu PIC dalam pengambilan keputusan berikut merupakan fitur-fiturnya.

- **Integrasi Data RAB Eksternal:** Pada Gambar 3.11 sistem secara otomatis menarik data referensi (Kategori Pekerjaan, Jenis Pekerjaan, Volume RAB, Satuan, Harga Material, dan Harga Upah) dari sheet ”data rab”. Data ini bersumber dari sistem ”Penawaran Final Kontraktor” (aplikasi eksternal) yang telah disinkronisasi ke dalam database Spreadsheet proyek ini, guna memastikan acuan data yang digunakan PIC selalu akurat dan konsisten.
- **Input Realisasi & Kalkulasi Selisih Otomatis:** PIC hanya perlu menginput Volume Akhir hasil pengukuran lapangan. Sistem menerapkan algoritma di sisi *client* yang secara *real-time* menghitung Selisih Volume (Volume RAB – Volume Akhir) dan Total Harga Akhir. Fitur ini menghilangkan kebutuhan perhitungan manual dan meminimalisir kesalahan aritmatika.
- **Indikator Visual Anggaran (*Overbudget Alert*):** Sistem memberikan umpan balik visual instan. Jika nilai selisih biaya menunjukkan angka negatif, teks akan otomatis berubah warna menjadi merah, dan jika selisih biaya menunjukkan angka positif, teks akan otomatis berubah menjadi warna hijau. Indikator ini berfungsi sebagai peringatan dini bagi PIC untuk meninjau kembali pekerjaan sebelum melakukan submisi.
- **Unggah Bukti Dokumentasi:** Terdapat fitur Unggah Foto pada setiap baris item pekerjaan. PIC wajib melampirkan foto kondisi lapangan sebagai bukti fisik. Foto ini kemudian diproses dan disimpan ke layanan penyimpanan *cloud* agar tidak membebani server aplikasi utama.
- **Mekanisme Penyimpanan & Status Persetujuan:** Setelah data lengkap, PIC menekan tombol Simpan. Data yang dikirim (*submit*) akan tersimpan ke sheet ”opname final” dan secara otomatis diberi

status ”PENDING”. Status ini menandakan bahwa data telah masuk ke dalam antrean verifikasi dan menunggu tindakan selanjutnya dari pihak Kontraktor (*Approve/Reject*).

7. **Pengembangan Sistem Verifikasi Digital (*Approval System*)**



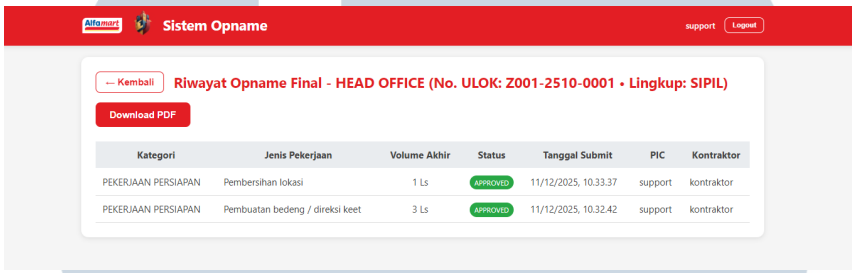
Gambar 3.12. Approval Page Kontraktor

Pada Gambar 3.12 dilakukan pembangunan modul persetujuan pada halaman `ApprovalPage.js` yang memungkinkan kontraktor meninjau hasil input PIC. Pada tahap ini, dirancang antarmuka tabel yang menampilkan rincian item pekerjaan beserta fotonya, dilengkapi dengan tombol aksi *Approve* dan *Reject*. Setiap interaksi pada tombol ini akan mengirimkan permintaan (*request*) ke API untuk memperbarui status pekerjaan di basis data secara instan, berikut fitur fiturnya :

- **Tinjauan Detail Pekerjaan:** Saat Kontraktor mengakses halaman persetujuan, sistem menyajikan tabel komprehensif yang memuat Kategori Pekerjaan, Jenis Pekerjaan, serta perbandingan antara Volume Akhir (hasil input PIC) dengan data RAB. Sistem juga menampilkan nilai Selisih secara transparan agar Kontraktor dapat langsung mendeteksi jika terjadi ketidaksesuaian kuantitas.
- **Verifikasi Bukti Visual & Identitas Penginput:** Untuk validasi fisik, Kontraktor dapat melihat Foto Dokumentasi yang diunggah PIC langsung dari tabel (jika tersedia). Selain itu, demi akuntabilitas data, sistem menampilkan metadata berupa Nama PIC penginput dan Waktu Pengiriman (*timestamp*), sehingga Kontraktor mengetahui siapa dan kapan data tersebut dilaporkan.
- **Fitur Catatan Alasan Penolakan (*Rejection Notes*):** Diterapkan fitur interaktif berupa kolom Catatan (*text area*) pada setiap baris item.

Fitur ini sangat krusial ketika Kontraktor memilih aksi *Reject* (Tolak). Dengan fitur ini, Kontraktor dapat menuliskan alasan penolakan secara spesifik, sehingga PIC mendapatkan umpan balik yang jelas untuk melakukan perbaikan data.

8. **Tampilan *Dashboard* Opname Final**



Gambar 3.13. Tampilan Dashboard Opname Final

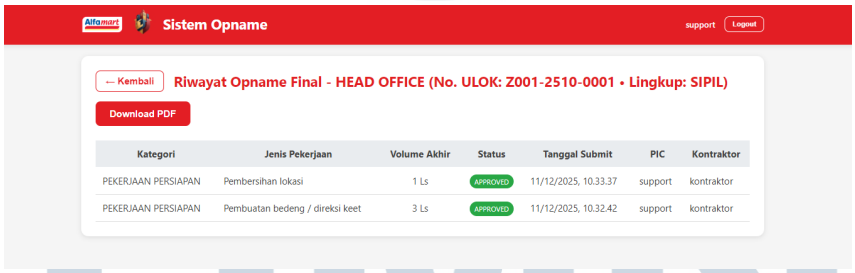
Tampilan pada Gambar 3.13 ini dirancang sebagai pusat rekapitulasi data bagi pengguna PIC untuk meninjau status akhir pekerjaan yang telah melalui proses verifikasi. Berbeda dengan formulir input, antarmuka ini berfungsi khusus untuk menampilkan data yang telah divalidasi dan disetujui (*Approved*) oleh Kontraktor guna memastikan bahwa informasi yang ditampilkan adalah data final yang siap untuk pelaporan. Secara teknis, modul ini mengimplementasikan logika navigasi bertingkat yang konsisten dengan modul lainnya. Pemilihan Nomor ULOK dan Lingkup Pekerjaan diwajibkan terlebih dahulu melalui komponen navigasi interaktif. Setelah parameter terpilih, sistem melakukan pemanggilan API (*fetch*) ke *endpoint* khusus (`/api/opname/final`) yang menyaring data berdasarkan status persetujuan, sehingga hanya menampilkan item pekerjaan yang telah valid secara administratif berikut beberapa fiturnya :

- **Fitur Unduhan Dokumen Resmi (*PDF Download*):** Terdapat tombol aksi interaktif ”Download PDF” yang terintegrasi dengan fungsi generator laporan. Saat tombol ditekan, sistem mengompilasi seluruh data opname yang tampil menjadi dokumen Berita Acara format PDF yang legal dan siap cetak, lengkap dengan kop surat serta tanda tangan elektronik.
- **Klasifikasi Item Pekerjaan:** Tabel data menyajikan kolom Kategori Pekerjaan dan Jenis Pekerjaan secara terstruktur. Fitur ini memudahkan

PIC dalam mengidentifikasi kelompok pekerjaan dan detail spesifik item yang telah diselesaikan.

- **Validasi Volume Realisasi:** Sistem menampilkan kolom Volume Akhir yang memuat angka kuantitas pekerjaan yang telah disetujui oleh kontraktor. Angka ini merupakan data final yang digunakan sebagai dasar perhitungan pembayaran termin.
- **Transparansi Status dan Waktu:** Untuk keperluan audit, tabel dilengkapi dengan kolom Tanggal Submit yang mencatat waktu pengiriman data secara presisi, serta label Status (misalnya: *Approved*) yang memberikan kepastian hukum bahwa data tersebut telah valid.
- **Identitas Penanggung Jawab:** Guna menjaga akuntabilitas data, antarmuka ini menampilkan secara eksplisit informasi Nama PIC yang melakukan input dan Nama Kontraktor yang melakukan persetujuan. Fitur ini memastikan bahwa setiap baris data dapat ditelusuri kembali kepada personel yang bertanggung jawab.

9. **Pembuatan Mesin Generator Dokumen PDF (Berita Acara)**



Gambar 3.14. PDF Generator Opname Final

Pada Gambar 3.14 untuk memenuhi kebutuhan administrasi formal, dikembangkan fitur cetak laporan menggunakan pustaka `jspdf`. Fitur ini memungkinkan pengguna (PIC atau Kontraktor) untuk mengunduh (*download*) data opname yang telah disetujui menjadi dokumen digital Berita Acara Opname dalam format PDF. Disusun kode program untuk mengatur tata letak (*layout*) tabel secara dinamis, melakukan perhitungan otomatis PPN 11% dan *Grand Total*, serta menyusun lampiran foto proyek agar tercetak rapi dalam format dokumen baku yang siap untuk ditandatangani, berikut fitur fiturnya :

- **Unduhan Langsung ke Perangkat (*Direct Download*):** Diimplementasikan fitur unduhan instan yang terintegrasi pada tombol ”Unduh PDF”. Saat PIC atau Kontraktor menekan tombol tersebut, sistem akan memproses data di sisi *client* dan secara otomatis menyimpan berkas PDF ke dalam penyimpanan perangkat (*device storage*) pengguna. Mekanisme ini memastikan dokumen dapat segera digunakan untuk keperluan arsip atau tanda tangan tanpa proses pemuatan (*loading*) halaman tambahan.
- **Kalkulasi Biaya dan Pajak Otomatis:** Dalam proses generasi dokumen, sistem menjalankan algoritma kalkulasi untuk menjumlahkan seluruh item pekerjaan yang disetujui. Disusun kode program yang secara otomatis menghitung nilai PPN 11% dan *Grand Total* dari sub-total proyek, sehingga angka yang tertera di dokumen dijamin akurat dan sinkron dengan basis data (*database*).
- **Penyusunan Tata Letak Dinamis (*Dynamic Layout*):** Generator PDF dirancang mampu menyesuaikan panjang tabel berdasarkan jumlah item pekerjaan. Jika data melebihi satu halaman, sistem secara otomatis akan membuat halaman baru dengan tetap mempertahankan *header* (kop surat) dan *footer* yang konsisten. Selain itu, fitur ini juga menyusun lampiran Foto Bukti Pekerjaan secara rapi di halaman lampiran, lengkap dengan keterangan jenis pekerjaannya.

10. **Integrasi Visualisasi Catatan Kontraktor pada Sisi PIC**

Gambar 3.15. Visualisasi Reject dan Catatan

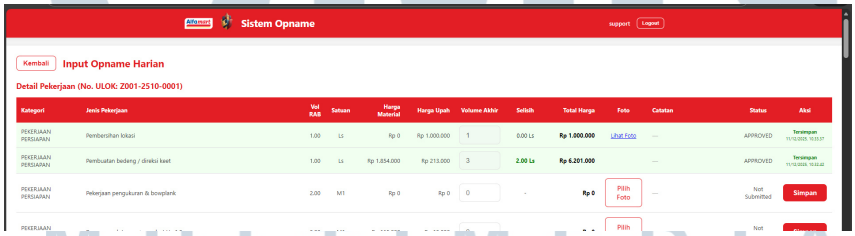
Pada Gambar 3.15 dilakukan pembaruan pada tampilan antarmuka PIC agar dapat menampilkan catatan yang ditulis oleh kontraktor. Fitur ini menciptakan siklus komunikasi dua arah yang efektif, di mana PIC dapat langsung mengetahui alasan spesifik mengapa suatu pekerjaan ditolak (misalnya: ”Foto kurang jelas” atau ”Volume tidak sesuai”) tanpa perlu berkomunikasi secara manual di luar *system*, berikut beberapa fiturnya :

- **Visualisasi Status *Rejected*:** Saat Kontraktor menolak suatu pekerjaan,

status pada tabel *dashboard* PIC akan berubah menjadi ”REJECTED” (ditandai dengan label berwarna merah). Indikator ini memberikan sinyal visual instan kepada PIC bahwa terdapat item pekerjaan yang memerlukan perhatian atau tindak lanjut segera.

- **Tampilan Catatan Alasan (*Rejection Notes*):** Di sebelah status penolakan, sistem memunculkan Catatan Revisi yang ditulis oleh Kontraktor. Fitur ini memungkinkan PIC membaca alasan spesifik penolakan tersebut (misalnya: ”Foto buram” atau ”Volume realisasi tidak sesuai RAB”) tanpa perlu menghubungi Kontraktor secara manual, sehingga komunikasi menjadi lebih efisien dan terdokumentasi.
- **Tombol Aksi Perbaiki (*Fix Button*):** Khusus untuk item yang berstatus *Rejected*, sistem memunculkan tombol interaktif ”Perbaiki”. Saat tombol ini ditekan, kolom input yang sebelumnya terkunci (*frozen*) akan terbuka kembali. Fitur ini memberikan akses kepada PIC untuk merevisi data, baik berupa input ulang Volume Akhir yang benar maupun pengunggahan foto bukti baru.
- **Reset Siklus Persetujuan (*Re-Approval Process*):** Setelah perbaikan selesai dilakukan dan tombol simpan ditekan, sistem secara otomatis mengubah status pekerjaan kembali menjadi ”PENDING”. Logika ini mengulang proses *approval* dari awal, di mana data revisi tersebut akan masuk kembali ke *dashboard* Kontraktor untuk diperiksa dan disetujui ulang.

11. **Tampilan *Approved***



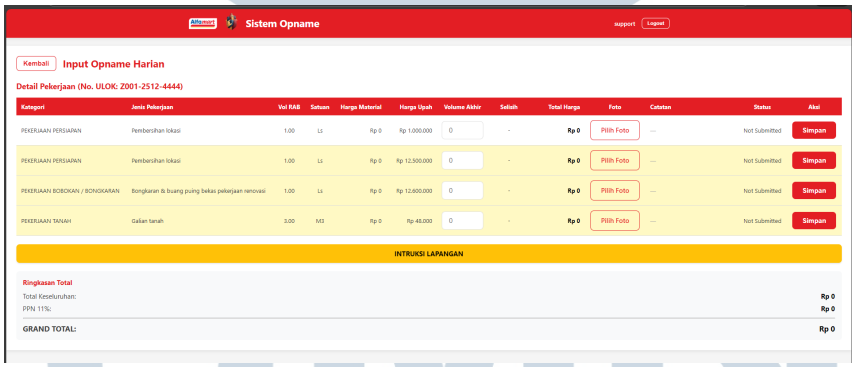
Gambar 3.16. Tampilan Approved

Tampilan pada Gambar 3.16 merupakan *Dashboard* Opname Final yang secara khusus menyajikan rekapitulasi data pekerjaan yang telah berhasil divalidasi dan mendapatkan status disetujui (*Approved*) oleh pihak Kontraktor. Pada antarmuka ini, dapat ditinjau rincian item pekerjaan secara

komprehensif, mulai dari kategori, jenis pekerjaan, hingga volume akhir realisasi, yang ditampilkan lengkap dengan identitas PIC penginput serta nama Kontraktor penanggung jawab. Halaman ini juga dilengkapi dengan fitur unduhan dokumen (*Download PDF*) yang memungkinkan pengguna untuk mengekspor data opname terverifikasi tersebut menjadi Berita Acara resmi, guna memastikan bahwa seluruh informasi yang tersaji adalah data final yang siap digunakan untuk keperluan administrasi dan pelaporan, berikut fitur nya :

- **Indikator Visual Status Final:** Setiap baris data yang telah disetujui oleh Kontraktor ditandai dengan blok warna hijau pada kolom status. Penanda visual ini berfungsi sebagai konfirmasi mutlak bahwa item pekerjaan tersebut telah valid. Selain itu, data yang tampil di halaman ini bersifat terkunci (*read-only*) dan tidak dapat diubah kembali guna menjamin integritas data laporan akhir.

12. **Pengembangan Fitur Penanganan Instruksi Lapangan (IL)**



Gambar 3.17. Fitur Intruksi Lapangan

Tahap terakhir adalah pengembangan fitur untuk menangani pekerjaan tambahan di luar RAB, yang disebut Instruksi Lapangan (IL) seperti yang dapat dilihat pada Gambar 3.17. Diterapkan logika khusus di mana item pekerjaan baru yang diinput sebagai IL ditandai dengan blok warna kuning pada antarmuka tabel. Indikator visual ini berfungsi agar PIC dan Kontraktor dapat dengan mudah membedakan antara pekerjaan kontraktual (RAB asli) dan pekerjaan tambahan (*addendum*) yang terjadi di lapangan, berikut beberapa fiturnya:

- **Integrasi Modul Eksternal (*External Link Integration*):** Disediakan tombol interaktif ”Instruksi Lapangan” pada antarmuka PIC. Saat tombol tersebut ditekan, sistem akan mengarahkan pengguna (*redirect*) ke aplikasi web khusus (”Web Instruksi Lapangan”). Fitur ini berfungsi sebagai sarana bagi PIC untuk menambahkan item atau jenis pekerjaan baru yang sebelumnya tidak tercantum atau terlewat pada data ”Web Penawaran Final Kontraktor”.
- **Visualisasi Identitas Pekerjaan (*Yellow Block Indicator*):** Setelah PIC menambahkan item baru pada aplikasi IL, data tersebut akan tersinkronisasi dan muncul kembali di formulir input opname. Diterapkan logika *styling* khusus di mana baris jenis pekerjaan IL tersebut secara otomatis tertampil dengan blok warna kuning. Penanda visual ini sangat krusial agar PIC dan Kontraktor dapat membedakan dengan cepat antara pekerjaan kontraktual (RAB Asli) dan pekerjaan tambahan (*addendum*).
- **Standardisasi Proses Input:** Meskipun berstatus sebagai item tambahan (IL), antarmuka dirancang agar PIC dapat mengisi data pada baris berwarna kuning tersebut dengan alur yang sama. PIC tetap dapat memasukkan Volume Akhir dan mengunggah Foto Bukti melalui mekanisme yang serupa dengan pengisian item RAB reguler, guna menjaga konsistensi pengalaman pengguna (*user experience*) tanpa perlu mempelajari alur baru.

B Proyek 1 *Backend* (API Server)

Bagian *backend* dikembangkan menggunakan Node.js sebagai fondasi utama dalam membangun *Application Programming Interface* (API) yang terhubung langsung dengan Google Spreadsheet sebagai basis data. Berkas (*file*) utama yang digunakan adalah `server.mjs`. Teknologi yang digunakan mencakup Node.js, Google Auth Service Account JWT, Cloudinary SDK, dan Google Apps Script.

Berikut adalah rincian *endpoint* dan struktur kode yang dibangun:

1. Autentikasi dan Log Aktivitas

```
// --- Endpoint Login (PIC + KONTRAKTOR via data_kontraktor) ---
app.post("/api/login", async (req, res) => {
  try {
    await doc.loadInfo();

    const { username, password } = req.body;
    if (!username || !password) {
      return res
        .status(400)
        .json({ message: "Username dan password diperlukan." });
    }

    const inputUsername = String(username).trim();
    const inputPassword = String(password).trim();

    // 1) Coba login sebagai PIC lewat sheet 'users' (tetap seperti semula)
    const usersSheet = doc.sheetsByTitle["users"];
    if (usersSheet) {
      const rows = await usersSheet.getRows();
      const userRow = rows.find((row) => {
        const sheetUsername = row.get("username")?.trim();
        const sheetPassword = row.get("password")?.toString().trim();
        return (
          sheetUsername?.toLowerCase() === inputUsername.toLowerCase() &&
          sheetPassword === inputPassword
        );
      });

      if (userRow) {
        await logLoginAttempt(inputUsername, "SUCCESS(PIC)");
        const userData = {
          id: userRow.get("id"),
          username: userRow.get("username"),
          name: userRow.get("name"),
          role: userRow.get("role"),
          ...(userRow.get("role") === "pic" && {
            kode_toko: userRow.get("kode_toko"),
            no_ulok: userRow.get("no_ulok"),
          }),
          ...(userRow.get("role") === "kontraktor" && {
            company: userRow.get("company"),
          }),
        };
        return res.status(200).json(userData);
      }
    }
    // Catatan: jika sheet 'users' tidak ada, kita lanjut ke cek kontraktor tanpa error 500.

    // 2) Coba login sebagai KONTRAKTOR via sheet 'data_kontraktor'
    const kontraktorSheet = doc.sheetsByTitle["data_kontraktor"];
    if (!kontraktorSheet) {
      // Jika sheet ini pun tidak ada, benar-benar gagal auth
      await logLoginAttempt(inputUsername, "FAILED(NO_SHEETS)");
      return res.status(401).json({ message: "Username atau password salah" });
    }

    const kRows = await kontraktorSheet.getRows();
    const kRow = kRows.find((row) => {
      const namaKontraktor = row.get("nama_kontraktor")?.trim() || "";
      const namaCabang = row.get("nama_cabang")?.trim() || "";
      return (
        namaKontraktor.toLowerCase() === inputUsername.toLowerCase() &&
        namaCabang.toLowerCase() === inputPassword.toLowerCase()
      );
    });

if (kRow) {
  const status = (kRow.get("status_kontraktor") || "")
    .toString()
    .trim()
    .toUpperCase();
  if (status && status !== "AKTIF") {
    await logLoginAttempt(inputUsername, "FAILED(NOT_ACTIVE)");
    return res.status(403).json({ message: "Akun kontraktor tidak aktif." });
  }

  // Ambil username standar dari kolom E (kontraktor_username)
  const kontraktorUsername = (kRow.get("kontraktor_username") || "")
    .toString()
    .trim();
  const namaKontraktor = (kRow.get("nama_kontraktor") || "").toString().trim();

  await logLoginAttempt(inputUsername, "SUCCESS(KONTRAKTOR)");
return res.status(200).json({
  id: kRow.get("no") || "",
  // pakai username standar dari kolom E
  username: kontraktorUsername || namaKontraktor || inputUsername,
  name: kontraktorUsername || namaKontraktor || inputUsername,
  kontraktor_username: kontraktorUsername || "",
  role: "kontraktor",
  company: namaKontraktor, // kolom C = nama perusahaan
  cabang: kRow.get("nama_cabang") || "",
  status_kontraktor: kRow.get("status_kontraktor") || "",
});
}


    // 3) Jika tidak cocok di keduanya
    await logLoginAttempt(inputUsername, "FAILED");
    return res.status(401).json({ message: "Username atau password salah" });
  } catch (error) {
    console.error("Error di /api/login:", error);
    res.status(500).json({ message: "Terjadi kesalahan pada server." });
  }
});
```

Kode 3.1 API Login dan Log Login

Kode 3.1 `api/login` dan `log_login`

- `/api/login`: *Endpoint* untuk autentikasi pengguna (PIC dan kontraktor) berdasarkan data pada *sheet* `users` dan `data_kontraktor`.
- `log_login`: *Sheet* untuk mencatat setiap aktivitas *login* (berhasil atau gagal).

2. Upload dan Manajemen Berkas

```
// --- Endpoint Upload Foto (dengan konversi ke JPEG) ---
app.post("/api/upload", upload.single("file"), async (req, res) => {
  try {
    if (!req.file) {
      return res
        .status(400)
        .json({ message: "Tidak ada file yang di-upload." });
    }
    const jpegBuffer = await sharp(req.file.buffer)
      .jpeg({ quality: 90 })
      .toBuffer();
    const uploadStream = (buffer) => {
      return new Promise((resolve, reject) => {
        const stream = cloudinary.uploader.upload_stream(
          { folder: "opname_alfamart" },
          (error, result) => {
            if (error) reject(error);
            else resolve(result);
          }
        );
        stream.end(buffer);
      });
    };
    const result = await uploadStream(jpegBuffer);
    res.status(200).json({ link: result.secure_url });
  } catch (error) {
    console.error("Error saat upload ke Cloudinary:", error);
    res.status(500).json({ message: "Gagal meng-upload file." });
  }
});
```

Kode 3.2 API Upload

Kode 3.2 `api/upload`

- `/api/upload`: Berfungsi mengunggah foto ke *Cloudinary*, dikonversi otomatis ke format JPEG, lalu menghasilkan tautan URL.

3. Data Toko dan Nomor ULOK

Kode 3.3 API Toko

Kode 3.3 `api/toko`

- `/api/toko`: Mengambil daftar toko berdasarkan cabang atau PIC yang melakukan *login* (bersumber dari *sheet* `data_rab` dan `users`).

```
// --- ENDPOINT BARU: Mengambil daftar unik no_ulok untuk kode_toko tertentu ---
app.get("/api/uloks", async (req, res) => {
  try {
    const { kode_toko } = req.query;
    if (!kode_toko)
      return res.status(400).json({ message: "Kode toko diperlukan." });
    await doc.loadInfo();
    const rabSheet = doc.sheetsByTitle["data_rab"];
    if (!rabSheet)
      return res
        .status(500)
        .json({ message: "Sheet 'data_rab' tidak ditemukan." });
    const rows = await rabSheet.getRows();
    const uloksSet = new Set();
    rows.forEach((row) => {
      if (row.get("kode_toko") === kode_toko && row.get("no_ulok")) {
        uloksSet.add(row.get("no_ulok"));
      }
    });
    res.status(200).json(Array.from(uloksSet));
  } catch (error) {
    console.error("Error di /api/uloks:", error);
    res.status(500).json({ message: "Terjadi kesalahan pada server." });
  }
});
```

Kode 3.4 API Uloks

Kode 3.4 `api/uloks`

- `/api/uloks`: Mengambil daftar nomor ULOK unik untuk setiap kode toko.

```
// --- ENDPOINT BARU: Ambil daftar LINGKUP (ME/SIPIL) untuk kode_toko + no_ulok ---
app.get("/api/lingkups", async (req, res) => {
  try {
    const { kode_toko, no_ulok } = req.query;
    if (!kode_toko || !no_ulok) {
      return res
        .status(400)
        .json({ message: "Kode toko dan No. ULOK diperlukan." });
    }

    await doc.loadInfo();
    const rabSheet = doc.sheetsByTitle["data_rab"];
    if (!rabSheet) {
      return res
        .status(500)
        .json({ message: "Sheet 'data_rab' tidak ditemukan." });
    }

    const rows = await rabSheet.getRows();
    const setLingkup = new Set();

    rows.forEach((row) => {
      const sameToko =
        (row.get("kode_toko") || "").toString().trim() === kode_toko;
      const sameUlok = (row.get("no_ulok") || "").toString().trim() === no_ulok;
      if (sameToko && sameUlok) {
        const lk = (row.get("lingkup_pekerjaan") || "").toString().trim();
        if (lk) setLingkup.add(lk.toUpperCase());
      }
    });

    // Jika di sheet hanya ada satu lingkup, frontend bisa menampilkan langsung / auto-select
    return res.status(200).json({ lingkups: Array.from(setLingkup) });
  } catch (error) {
    console.error("Error di /api/lingkups:", error);
    return res.status(500).json({ message: "Terjadi kesalahan pada server." });
  }
});
```

Kode 3.5 API Lingkups

Kode 3.5 `api/lingkups`

- `/api/lingkups`: Mengambil daftar lingkup pekerjaan (misalnya ME atau SIPIL) untuk toko dan nomor ULOK tertentu.

```
// --- ENDPOINT BARU: PIC & KONTRAKTOR dari tab opname_final (by no_ulok) ---
app.get("/api/pic-kontraktor-opname", async (req, res) => {
  try {
    const { no_ulok } = req.query;
    if (!no_ulok)
      return res.status(400).json({ message: "No. Ulok diperlukan." });

    await doc.loadInfo();
    const finalSheet = doc.sheetsByTitle["opname_final"];
    if (!finalSheet) {
      return res
        .status(500)
        .json({ message: "Sheet 'opname_final' tidak ditemukan." });
    }

    const rows = await finalSheet.getRows();

    // Normalisasi komparasi
    const norm = (v) => (v ?? "").toString().toUpperCase().replace(/\s+/g, "");
    const getAny = (row, keys) => {
      for (const k of keys) {
        const val = row.get(k);
        if (val !== undefined && val !== null && String(val).trim() !== "") {
          return String(val).trim();
        }
      }
      return "";
    };

    // Ambil SEMUA baris utk no_ulok tsb (sering ada banyak)
    const matches = rows.filter(
      (r) => norm(r.get("no_ulok")) === norm(no_ulok)
    );

    if (matches.length) {
      // Cari yang terisi lebih dulu; kalau ada beberapa, ambil yg tanggal_submit terbaru
      const withValues = matches
        .map((r) => ({
          pic: getAny(r, ["pic_username", "PIC_USERNAME", "pic", "PIC"]),
          name: getAny(r, ["name", "Name", "PIC_NAME"]), // ← ambil kolom name
          kontr: getAny(r, [
            "kontraktor_username",
            "KONTRAKTOR_USERNAME",
            "kontraktor",
            "KONTRAKTOR",
          ]),
          tgl: r.get("tanggal_submit") || "",
        }))
        .sort((a, b) => String(b.tgl).localeCompare(String(a.tgl)));

      const best =
        withValues.find((x) => x.pic || x.kontr || x.name) || withValues[0];
      return res.status(200).json({
        pic_username: best.pic || "N/A",
        kontraktor_username: best.kontr || "N/A",
        name: best.name || "", // ← kirim nama PIC
      });
    }

    return res
      .status(200)
      .json({ pic_username: "N/A", kontraktor_username: "N/A" });
  } catch (e) {
    console.error("Error di /api/pic-kontraktor-opname:", e);
    return res
      .status(500)
      .json({ pic_username: "N/A", kontraktor_username: "N/A" });
  }
});
```


Kode 3.6 API PIC Kontraktor Opname

Kode 3.6 `api/pic-kontraktor-opname`

- `/api/pic-kontraktor-opname`: Mengambil data PIC dan kontraktor dari

sheet `opname_final` berdasarkan nomor ULOK.

```
// --- ENDPOINT BARU: daftar unik PIC (by no_ulok + lingkup, opsi filter kode_toko) ---
app.get("/api/pic-list", async (req, res) => {
  try {
    const { no_ulok, lingkup, kode_toko } = req.query;
    if (!no_ulok) return res.status(400).json({ message: "No. Ulok diperlukan." });

    await doc.loadInfo();
    const finalSheet = doc.sheetsByTitle["opname_final"];
    if (!finalSheet) {
      return res.status(500).json({ message: "Sheet 'opname_final' tidak ditemukan." });
    }

    const rows = await finalSheet.getRows();
    const norm = (v) => (v ?? "").toString().trim().toUpperCase();

    let filtered = rows.filter((r) => norm(r.get("no_ulok")) === norm(no_ulok));

    if (lingkup) {
      filtered = filtered.filter((r) => norm(r.get("lingkup_pekerjaan")) === norm(lingkup));
    }
    if (kode_toko) {
      filtered = filtered.filter((r) => norm(r.get("kode_toko")) === norm(kode_toko));
    }

    // Ambil yang APPROVED saja biar akurat
    filtered = filtered.filter((r) => norm(r.get("approval_status")) === "APPROVED");

    // Kumpulkan nama dari kolom 'name' (fallback ke 'pic_username' kalau kolom name kosong)
    const names = new Set();
    for (const r of filtered) {
      const display =
        (r.get("name") || "").toString().trim() ||
        (r.get("pic_username") || "").toString().trim();
      if (display) names.add(display);
    }

    res.status(200).json({ pic_list: Array.from(names) });
  } catch (e) {
    console.error("Error di /api/pic-list:", e);
    res.status(500).json({ message: "Gagal mengambil daftar PIC." });
  }
});
```

Kode 3.7 API PIC List

Kode 3.7 `api/pic-list`

- `/api/pic-list`: Mengambil daftar nama PIC dari data opname yang sudah disetujui berdasarkan nomor ULOK dan lingkup pekerjaan.

4. Modul Opname (PIC dan Kontraktor)

Kode 3.8 API Opname

Kode 3.8 `api/opname`

- `/api/opname`: Mengambil daftar pekerjaan opname dari *sheet* `data_rab` dan hasil *input* dari *sheet* `opname_final`.

Kode 3.9 API Opname Submit

Kode 3.9 `api/opname/item/submit`

- `/api/opname/item/submit`: *Endpoint* bagi PIC untuk mengirimkan hasil opname, termasuk unggahan foto dan volume akhir pekerjaan.

```
// --- Ambil item Pending untuk persetujuan ---
app.get("/api/opname/pending", async (req, res) => {
  try {
    const { kode_toko, no_ulok } = req.query;
    if (!kode_toko || !no_ulok) {
      return res
        .status(400)
        .json({ message: "Kode toko dan No. ULOK diperlukan." });
    }

    await doc.loadInfo();
    const finalSheet = doc.sheetsByTitle["opname_final"];
    if (!finalSheet) {
      return res
        .status(500)
        .json({ message: "Sheet 'opname_final' tidak ditemukan." });
    }

    const rows = await finalSheet.getRows();
    const norm = (v) => (v ?? "").toString().trim().toUpperCase();

    const pending = rows.filter((row) => {
      const sameToko = norm(row.get("kode_toko")) === norm(kode_toko);
      const sameUlok = norm(row.get("no_ulok")) === norm(no_ulok);
      const status = norm(row.get("approval_status"));
      // anggap kosong sebagai Pending juga
      return sameToko && sameUlok && (status === "" || status === "PENDING");
    });

    const result = pending.map((row) => ({
      kategori_pekerjaan: row.get("kategori_pekerjaan") || "",
      item_id: row.get("item_id") || null,
      jenis_pekerjaan: row.get("jenis_pekerjaan") || "",
      volume_akhir: row.get("volume_akhir") || "",
      pic_username: row.get("pic_username") || "",
      tanggal_submit: row.get("tanggal_submit") || "",
      // informasi tambahan kalau kamu ingin tampilkan
      vol_rab: row.get("vol_rab") || "",
      satuan: row.get("satuan") || "",
      foto_url: row.get("foto_url") || "",
      name: (row.get("name") || "").toString().trim() || fallbackName, // ✅ tambahkan ini
    }));

    return res.status(200).json(result);
  } catch (error) {
    console.error("Error di /api/opname/pending:", error);
    return res.status(500).json({ message: "Terjadi kesalahan pada server." });
  }
});
```

Kode 3.10 API Opname Pending

Kode 3.10 `api/opname/pending`

- `/api/opname/pending`: Menampilkan daftar pekerjaan dengan status *Pending* (belum disetujui oleh kontraktor).

```
// --- Approve item opname ---
app.patch("/api/opname/approve", async (req, res) => {
  try {
    const { item_id, kontraktor_username, catatan } = req.body || {}; // ← tambahkan catatan
    if (!item_id) {
      return res.status(400).json({ message: "item_id diperlukan." });
    }

    await doc.loadInfo();
    const finalSheet = doc.sheetsByTitle["opname_final"];
    if (!finalSheet) {
      return res
        .status(500)
        .json({ message: "Sheet 'opname_final' tidak ditemukan." });
    }

    const rows = await finalSheet.getRows();
    const target = rows.find((r) => (r.get("item_id") || "") === item_id);

    if (!target) {
      return res.status(404).json({ message: "Item tidak ditemukan." });
    }

    target.set("approval_status", "APPROVED"); // konsisten
if (kontraktor_username && String(kontraktor_username).trim()) {
  // Ubah input apa pun (email/nama perusahaan/username) menjadi username standar dari data_kontraktor (kolom E)
  const storedUsername = await resolveContractorUsername(kontraktor_username);
  target.set("kontraktor_username", storedUsername);

  // Isi kolom 'kontraktor' dengan NAMA PERUSAHAAN dari data_kontraktor (kolom C)
  const companyName = await resolveContractorCompanyName(storedUsername);
  if (companyName) {
    target.set("kontraktor", companyName);
  }
}


    await finalSheet.loadHeaderRow();
    const headers = finalSheet.headerValues || [];
    const CAT_KEY = headers.includes("catatan")
      ? "catatan"
      : headers.includes("Catatan")
      ? "Catatan"
      : headers.includes("CATATAN")
      ? "CATATAN"
      : headers.includes("")
      ? ""
      : "catatan";

    if (typeof catatan === "string" && catatan.trim()) {
      const prev = target.get(CAT_KEY) || "";
      const stamp = new Date().toLocaleString("id-ID", {
        timeZone: "Asia/Jakarta",
      });
      const appended = prev
        ? `${prev}\n[APPROVE ${stamp}] ${catatan.trim()}`
        : `[APPROVE ${stamp}] ${catatan.trim()}`;
      target.set(CAT_KEY, appended);
    }

    await target.save();

    return res.status(200).json({ message: "Berhasil di-approve." });
  } catch (error) {
    console.error("Error di /api/opname/approve:", error);
    return res.status(500).json({ message: "Terjadi kesalahan pada server." });
  }
});
```

Kode 3.11 API Opname Approve

Kode 3.11 `api/opname/approve`

- `/api/opname/approve`: Digunakan oleh kontraktor untuk menyetujui pekerjaan opname dengan mengubah status menjadi *Approved* serta menambahkan catatan.

```
// --- Endpoint REJECT item opname ---
app.patch("/api/opname/reject", async (req, res) => {
  try {
    const { item_id, kontraktor_username, catatan } = req.body; // ← tambahkan catatan
    if (!item_id) {
      return res.status(400).json({ message: "item_id diperlukan." });
    }

    await doc.loadInfo();
    const finalSheet = doc.sheetsByTitle["opname_final"];
    if (!finalSheet) {
      return res
        .status(500)
        .json({ message: "Sheet 'opname_final' tidak ditemukan." });
    }

    const rows = await finalSheet.getRows();
    const row = rows.find((r) => r.get("item_id") === item_id);
    if (!row) {
      return res.status(404).json({ message: "Item tidak ditemukan." });
    }

    // Update status jadi REJECTED
    row.set("approval_status", "REJECTED");
if (kontraktor_username && String(kontraktor_username).trim()) {
  const storedUsername = await resolveContractorUsername(kontraktor_username);
  row.set("kontraktor_username", storedUsername);

  const companyName = await resolveContractorCompanyName(storedUsername);
  if (companyName) {
    row.set("kontraktor", companyName);
  }
}


    //  ✨ Tambah catatan
    await finalSheet.loadHeaderRow();
    const headers = finalSheet.headerValues || [];
    const CAT_KEY = headers.includes("catatan")
      ? "catatan"
      : headers.includes("Catatan")
      ? "Catatan"
      : headers.includes("CATATAN")
      ? "CATATAN"
      : headers.includes("")
      ? ""
      : "catatan";

    if (typeof catatan === "string" && catatan.trim()) {
      const prev = row.get(CAT_KEY) || "";
      const stamp = new Date().toLocaleString("id-ID", {
        timeZone: "Asia/Jakarta",
      });
      const appended = prev
        ? `${prev}\n[REJECT ${stamp}] ${catatan.trim()}`
        : `[REJECT ${stamp}] ${catatan.trim()}`;
      row.set(CAT_KEY, appended);
    }

    await row.save();

    return res.status(200).json({ message: "Item berhasil di-reject." });
  } catch (err) {
    console.error("Error di /api/opname/reject:", err);
    return res.status(500).json({ message: "Gagal reject item." });
  }
});
```

Kode 3.12 API Opname Reject

Kode 3.12 `api/opname/reject`

- `/api/opname/reject`: Digunakan oleh kontraktor untuk menolak pekerjaan opname dengan mengubah status menjadi *Rejected* serta menambahkan catatan revisi.

Kode 3.13 API Opname Final

Kode 3.13 `api/opname/final`

- `/api/opname/final`: Menampilkan seluruh pekerjaan opname yang telah disetujui (status *Approved*) dan siap diunduh menjadi laporan format PDF.

5. Data RAB dan Kontraktor

```
// --- Endpoint baru untuk mengambil data RAB dari data_rab ---
app.get("/api/rab", async (req, res) => {
  try {
    const { kode_toko, no_ulok, lingkup } = req.query;
    if (!kode_toko)
      return res.status(400).json({ message: "Kode toko diperlukan." });

    await doc.loadInfo();
    const rabSheet = doc.sheetsByTitle["data_rab"];
    if (!rabSheet)
      return res
        .status(500)
        .json({ message: "Sheet 'data_rab' tidak ditemukan." });

    const rows = await rabSheet.getRows();

    const norm = (v) => (v ?? "").toString().trim().toUpperCase();
    let rabItems = rows.filter(
      (row) => norm(row.get("kode_toko")) === norm(kode_toko)
    );

    // Filter berdasarkan no_ulok jika disediakan
    if (no_ulok) {
      rabItems = rabItems.filter(
        (row) => norm(row.get("no_ulok")) === norm(no_ulok)
      );
    }
    if (lingkup) {
      rabItems = rabItems.filter(
        (row) => norm(row.get("lingkup_pekerjaan")) === norm(lingkup)
      );
    }
    const result = rabItems.map((row) => ({
      kategori_pekerjaan: row.get("kategori_pekerjaan"),
      jenis_pekerjaan: row.get("jenis_pekerjaan"),
      satuan: row.get("satuan"),
      volume: row.get("vol_rab"),
      harga_material: row.get("harga_material"),
      harga_upah: row.get("harga_upah"),
      total_harga: row.get("total_harga"),
      pic_username: row.get("pic_username") || "",
      kontraktor_username:
        row.get("kontraktor") || row.get("kontraktor_username") || "",
      lingkup_pekerjaan: row.get("lingkup_pekerjaan") || "",
      is_il: (row.get("IL") || "").toString().trim().toLowerCase() === "ya",
    }));

    res.status(200).json(result);
  } catch (error) {
    console.error("Error di /api/rab:", error);
    res.status(500).json({ message: "Terjadi kesalahan pada server." });
  }
});
```

Kode 3.14 API RAB

Kode 3.14 `api/rab`

- `/api/rab`: Mengambil data RAB berdasarkan kode toko, nomor ULOK, dan lingkup pekerjaan.

```
// --- Endpoint untuk Kontraktor ---
app.get("/api/toko_kontraktor", async (req, res) => {
  try {
    // Terima email / nama_kontraktor / kontraktor_username → normalkan dulu
    const raw = (req.query.username || req.query.email || "").toString().trim();
    if (!raw) {
      return res
        .status(400)
        .json({ message: "Username Kontraktor diperlukan." });
    }
    const kontraktorUsername = await resolveContractorUsername(raw);

    await doc.loadInfo();
    const rabSheet = doc.sheetsByTitle["data_rab"];
    const kontrSheet = doc.sheetsByTitle["data_kontraktor"];
    if (!rabSheet) {
      return res
        .status(500)
        .json({ message: "Sheet 'data_rab' tidak ditemukan." });
    }

    const rabRows = await rabSheet.getRows();
    const norm = (v) => (v ?? "").toString().trim().toLowerCase();

    // 1) filter utama: kolom kontraktor_username di data_rab
    let assignedRows = rabRows.filter(
      (row) => norm(row.get("kontraktor_username")) === norm(kontraktorUsername)
    );

    // 2) fallback: pakai peta cabang dari data_kontraktor berbasis kontraktor_username (bukan nama)
    if (assignedRows.length === 0 && kontrSheet) {
      const kRows = await kontrSheet.getRows();
      const cabangs = new Set(
        kRows
          .filter(
            (r) =>
              norm(r.get("kontraktor_username")) === norm(kontraktorUsername) &&
              String(r.get("status_kontraktor") || "").toUpperCase() === "AKTIF"
          )
          .map((r) => (r.get("nama_cabang") || "").toString().trim())
      );

      if (cabangs.size > 0) {
        assignedRows = rabRows.filter((row) => {
          const kode = (row.get("kode_toko") || "").toString().trim();
          const nama = (row.get("nama_toko") || "").toString().trim();
          // sesuaikan jika kamu juga menyimpan 'nama_cabang' di data_rab
          return cabangs.has(kode) || cabangs.has(nama);
        });
      }
    }


    const storesMap = new Map();

    for (const row of assignedRows) {
      const kode_toko = (row.get("kode_toko") || "").toString().trim();
      const nama_toko = (row.get("nama_toko") || "").toString().trim();
      const no_ulok = (row.get("no_ulok") || "").toString().trim();
      const link_pdf = (row.get("link_pdf") || "").toString().trim();

      if (!kode_toko) continue;

      // KUNCI UNIK: Gabungan Kode dan Nama
      const compositeKey = `${kode_toko}__${nama_toko}`;

      if (!storesMap.has(compositeKey)) {
        storesMap.set(compositeKey, {
          kode_toko,
          nama_toko,
          link_pdf,
          no_uloks: new Set(),
        });
      }

      if (no_ulok) {
        storesMap.get(compositeKey).no_uloks.add(no_ulok);
      }
    }

    const result = Array.from(storesMap.values()).map((s) => ({
      kode_toko: s.kode_toko,
      nama_toko: s.nama_toko,
      link_pdf: s.link_pdf,
      no_uloks: Array.from(s.no_uloks),
    }));

    return res.status(200).json(result);
  } catch (error) {
    console.error("Error di /api/toko_kontraktor:", error);
    return res.status(500).json({ message: "Terjadi kesalahan pada server." });
  }
});
```

Kode 3.15 API Toko Kontraktor

Kode 3.15 `api/toko-kontraktor`

- `/api/toko-kontraktor`: Mengambil daftar toko dan nomor ULOK yang menjadi tanggung jawab kontraktor tertentu berdasarkan *username* atau nama perusahaan.

6. Debugging dan Monitoring

```
// --- ENDPOINT DEBUG untuk troubleshooting ---
app.get("/api/debug/opname-final", async (req, res) => {
  try {
    await doc.loadInfo();
    const finalSheet = doc.sheetsByTitle["opname_final"];
    if (!finalSheet) {
      return res
        .status(500)
        .json({ message: "Sheet 'opname_final' tidak ditemukan." });
    }

    const rows = await finalSheet.getRows();
    const allData = rows.map((row, index) => ({
      rowNumber: row.rowNumber,
      index: index,
      submission_id: row.get("submission_id") || "N/A",
      kode_toko: row.get("kode_toko") || "N/A",
      nama_toko: row.get("nama_toko") || "N/A",
      no_ulok: row.get("no_ulok") || "N/A",
      pic_username: row.get("pic_username") || "N/A",
      jenis_pekerjaan: row.get("jenis_pekerjaan") || "N/A",
      tanggal_submit: row.get("tanggal_submit") || "N/A",
      approval_status: row.get("approval_status") || "N/A",
    }));

    res.status(200).json({
      message: "Debug data dari opname_final",
      totalRows: rows.length,
      sheetTitle: finalSheet.title,
      data: allData,
    });
  } catch (error) {
    console.error("Error di /api/debug/opname-final:", error);
    res.status(500).json({
      message: "Error mengambil data debug",
      error: error.message,
    });
  }
});
```

Kode 3.16 API Debug Opname Final

Kode 3.16 `api/debug/opname-final`

- `/api/debug/opname-final`: Berfungsi untuk melihat seluruh data pada *sheet* `opname_final` guna keperluan *debugging*.

```
// --- ENDPOINT DEBUG untuk melihat headers sheet ---
app.get("/api/debug/sheet-headers", async (req, res) => {
  try {
    await doc.loadInfo();
    const finalSheet = doc.sheetsByTitle["opname_final"];
    if (!finalSheet) {
      return res
        .status(500)
        .json({ message: "Sheet 'opname_final' tidak ditemukan." });
    }

    await finalSheet.loadHeaderRow();

    res.status(200).json({
      message: "Headers dari sheet opname_final",
      headers: finalSheet.headerValues,
      sheetTitle: finalSheet.title,
    });
  } catch (error) {
    console.error("Error di /api/debug/sheet-headers:", error);
    res.status(500).json({
      message: "Error mengambil headers sheet",
      error: error.message,
    });
  }
});
```

Kode 3.17 API Debug Sheet Headers

Kode 3.17 `api/debug/sheet-headers`

- `/api/debug/sheet-headers`: Digunakan untuk melihat struktur atau *header* dari *sheet* `opname_final`.

```
// --- route default opsional ---
app.get("/", (req, res) => {
  res.json({ message: "Backend OK" });
});

// 6. Menjalankan server
app.listen(PORT, () => {
  console.log(`🚀 Server backend berjalan di port ${PORT}`);
});
```

Kode 3.18 Route Default

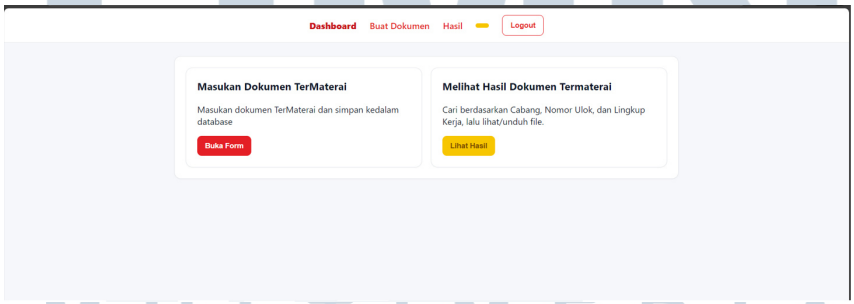
Kode 3.18 `/route default`

- /: *Route default* untuk memastikan *server* aktif dan berjalan dengan normal.

C Proyek 2 (Pengembangan Modul Digitalisasi Dokumen Legal)

Selain pengembangan sistem utama opname, dirancang dan dibangun pula modul khusus untuk digitalisasi dokumen legal (Dokumen Termeterai). Modul ini bertujuan memfasilitasi kontraktor dalam mengunggah dokumen administrasi, seperti RAB dan SPH yang telah disatukan.

1. **Perancangan Tampilan Awal (*Dashboard*)**



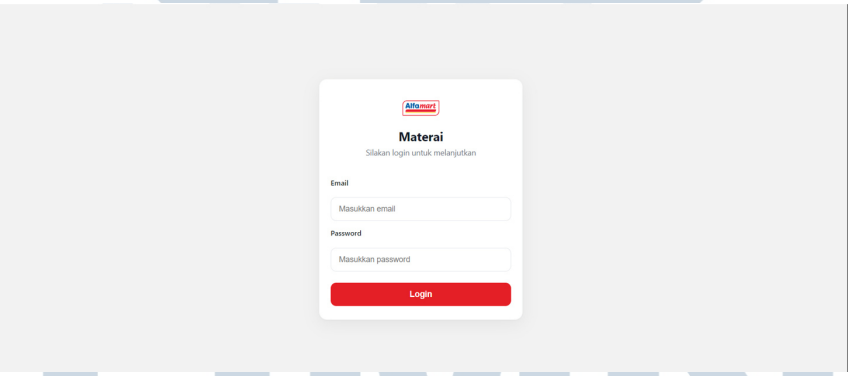
Gambar 3.18. Dashboard Dokumen Termaterai

Tahap pertama adalah menyiapkan antarmuka utama yang menjadi gerbang akses bagi pengguna seperti pada Gambar 3.18. Dilakukan modifikasi pada komponen `Dashboard.js` dengan menambahkan menu navigasi khusus, yaitu ”Masukan Dokumen Termeterai” dan ”Melihat Hasil Dokumen

Termeterai", agar pengguna (Kontraktor) dapat dengan mudah menemukan fitur ini setelah masuk ke dalam aplikasi, berikut beberapa fiturnya:

- **Menu "Masukan Dokumen Termeterai":** Fitur ini berfungsi sebagai pintu masuk utama bagi Kontraktor untuk mengakses formulir digital pengunggahan dokumen. Saat tombol "Buka Form" ditekan, pengguna akan diarahkan ke halaman *input* untuk menyimpan dokumen legal yang telah disahkan ke dalam basis data (*database*).
- **Menu "Melihat Hasil Dokumen Termeterai":** Opsi ini disediakan agar pengguna dapat memantau arsip dokumen yang telah tersimpan. Melalui menu ini, pengguna dapat mencari dokumen spesifik berdasarkan Cabang, Nomor ULOK, dan Lingkup Kerja, serta memiliki akses langsung untuk melihat atau mengunduh berkas (*file*) tersebut kembali.

2. **Implementasi Halaman *Login* Terintegrasi Basis Data**



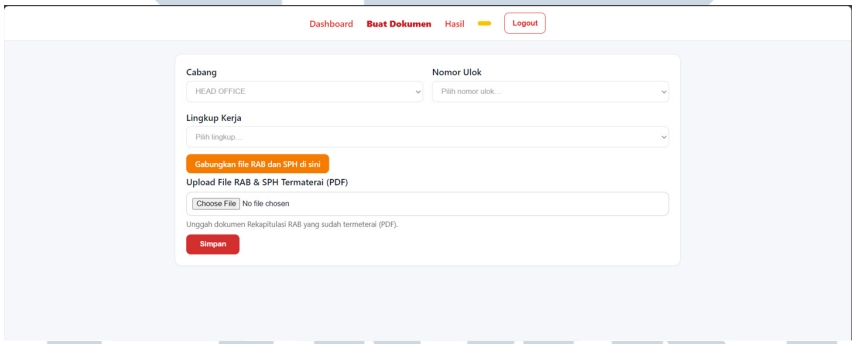
Gambar 3.19. LoginPage Dokumen Termaterai

Pada Gambar 3.19 dikembangkan halaman `Login.js` yang terhubung langsung dengan basis data Google Spreadsheet. Sistem autentikasi ini memverifikasi kredensial pengguna (PIC atau Kontraktor). Hal ini memastikan bahwa hanya pengguna yang terdaftar dalam *sheet database* yang dapat mengakses fitur pengunggahan dokumen.

- **Autentikasi Berbasis Email Kontraktor:** Untuk mempermudah identifikasi, kredensial *login* bagi pengguna eksternal (Kontraktor) menggunakan alamat email resmi yang terdaftar dalam sistem. Hal ini memastikan bahwa setiap aktivitas pengunggahan dokumen dapat dilacak kembali ke entitas bisnis yang bertanggung jawab.

- **Validasi Kata Sandi Berbasis Kode Cabang:** Sistem menerapkan standar keamanan operasional di mana kata sandi (*password*) yang digunakan dikonfigurasi sesuai dengan Kode Cabang atau lokasi kerja masing-masing kontraktor. Mekanisme ini mempermudah manajemen akses sekaligus memverifikasi bahwa kontraktor memiliki otorisasi sah untuk beroperasi di cabang tersebut.
- **Isolasi Akses Data (*Data Isolation*):** Setelah berhasil *login*, sistem secara otomatis menerapkan pembatasan akses data. Kontraktor hanya diizinkan untuk menginput dan melihat dokumen yang berkaitan dengan cabang tempat mereka bertugas. Fitur isolasi ini mencegah terjadinya lintas akses data antarcabang, menjaga kerahasiaan informasi proyek, dan meminimalisir kesalahan administratif (salah *upload* ke cabang lain).

3. **Pengembangan Halaman *Input* Dokumen (*Create Document*)**



Gambar 3.20. Tampilan Input Dokumen Termaterai

Dilakukan pembangunan halaman `CreateDocument.js` seperti pada Gambar 3.20 sebagai antarmuka utama bagi kontraktor untuk mengunggah berkas. Pada halaman ini, disediakan formulir yang mewajibkan kontraktor untuk memilih atribut proyek (Cabang, Nomor ULOK, dan Lingkup Kerja) serta mengunggah dokumen fisik yang telah dibubuhi meterai. Logika konversi berkas (*fileToBase64*) diterapkan agar dokumen dapat dikirim dan disimpan ke *server* secara digital.

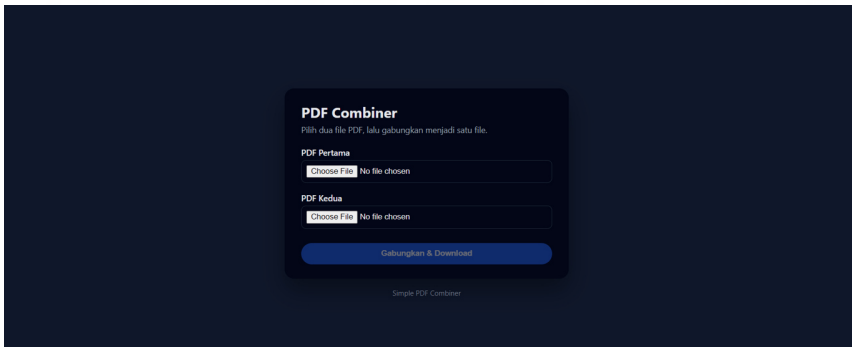
- **Pengisian Otomatis Cabang (*Auto-fill Branch*):** Kolom Cabang dirancang untuk terisi secara otomatis segera setelah halaman dimuat. Sistem mengambil data ini dari identitas *login* kontraktor (yang

merepresentasikan daerah/cabang mereka), sehingga kontraktor tidak perlu memilih cabang secara manual guna mencegah kesalahan *input* data ke cabang yang salah.

- **Pemilihan Nomor ULOK dan Lingkup Dinamis:** Sistem menerapkan logika *dropdown* bertingkat. Setelah cabang teridentifikasi, kolom Nomor ULOK hanya akan menampilkan daftar proyek yang tersedia di cabang tersebut. Selanjutnya, kolom Lingkup Kerja akan aktif setelah Nomor ULOK dipilih. Validasi alur ini memastikan data yang dikirimkan memiliki hierarki proyek yang valid.
- **Integrasi Alat Penggabungan PDF Eksternal:** Untuk mendukung kepraktisan administrasi, disediakan tombol pintasan berwarna oranye bertuliskan ”Gabungkan PDF (RAB + SPH)”. Tombol ini mengarahkan kontraktor ke aplikasi web utilitas terpisah untuk menggabungkan dua berkas dokumen (RAB dan SPH) menjadi satu berkas (*file*) PDF utuh sebelum diunggah ke sistem utama.
- **Mekanisme Unggah dan Simpan:** Setelah dokumen digabungkan, kontraktor dapat mengunggah berkas final tersebut melalui kolom *input* yang tersedia. Proses diakhiri dengan menekan tombol Simpan yang akan memicu konversi dokumen ke format digital (*Base64*) dan mengirimkannya ke *database server* untuk diarsipkan.

4. **Integrasi Utilitas Penggabungan Dokumen (*PDF Merger*)**

Gambar 3.21. Button Direct Pengabungan File

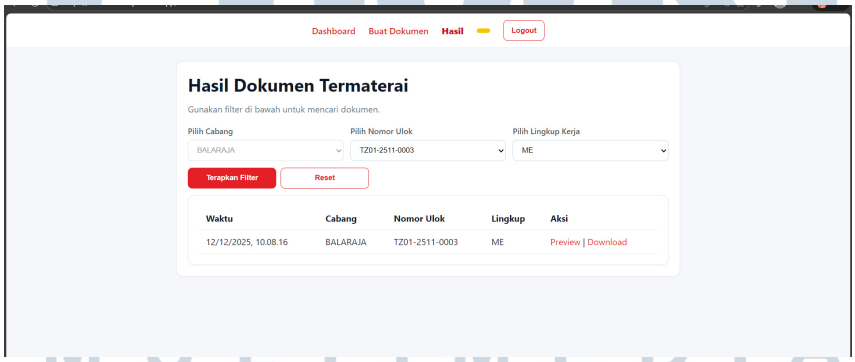


Gambar 3.22. Web untuk Mengabungkan dua File Menjadi satu

Untuk mempermudah administrasi, ditambahkan fitur pendukung berupa tombol tautan (*link button*) di bagian bawah formulir *input* seperti pada Gambar 3.21. Tombol ini mengarahkan pengguna ke aplikasi web terpisah seperti pada Gambar 3.22 yang berfungsi untuk menggabungkan dua berkas (*file*) PDF (RAB dan SPH) menjadi satu berkas utuh. Fitur ini memastikan dokumen yang diunggah ke dalam sistem sudah dalam format tunggal yang rapi sebelum diproses lebih lanjut, berikut kedua fiturnya:

- **Antarmuka *Input* Dua Kolom (*Dual Input Interface*):** Aplikasi penggabung ini menyediakan dua kolom unggahan yang terpisah. Kontraktor dapat memasukkan berkas (*file*) RAB pada kolom pertama dan berkas SPH (Surat Penawaran Harga) pada kolom kedua. Pemisahan ini memudahkan pengguna untuk memastikan urutan dokumen yang benar sebelum disatukan.
- **Proses Penggabungan & Unduhan Otomatis:** Setelah kedua berkas dipilih, sistem akan memproses penggabungan halaman PDF tersebut menjadi satu berkas utuh. Hasil dokumen yang telah digabungkan kemudian akan secara otomatis diunduh (*auto-download*) ke penyimpanan perangkat (*device*) pengguna, sehingga berkas siap untuk langsung diunggah ke dalam Sistem Dokumen Termeterai tanpa langkah tambahan yang rumit.

5. **Halaman ”Lihat Hasil Dokumen Termeterai”**



Gambar 3.23. Hasil Dokumen Termaterai

Halaman pada Gambar 3.23 ini dirancang sebagai pusat arsip digital yang memungkinkan pengguna untuk memantau dan mengakses kembali dokumen legal yang telah diunggah ke dalam sistem. Pada antarmuka ini,

pengguna dapat melakukan pencarian data secara spesifik menggunakan *filter* Nomor ULOK dan Lingkup Pekerjaan—dengan *filter* Cabang yang secara otomatis terkunci sesuai wilayah kerja pengguna—untuk menemukan serta mengunduh berkas PDF termeterai yang tersimpan di *database server*.

- ***Filter* Cabang Otomatis (*Auto-locked Branch*):** Saat halaman dimuat, kolom pencarian Cabang akan terisi dan terkunci secara otomatis sesuai dengan wilayah kerja pengguna yang sedang *login*. Fitur keamanan ini memastikan bahwa Kontraktor hanya dapat mengakses arsip dokumen milik cabangnya sendiri guna mencegah kebocoran data antarcabang.
- **Pencarian Spesifik (*Project Filtering*):** Untuk menemukan dokumen tertentu, pengguna diwajibkan memilih Nomor ULOK dan Lingkup Pekerjaan dari menu *dropdown*. Mekanisme ini membantu pengguna menyaring data proyek yang relevan dengan cepat tanpa harus mencari secara manual di seluruh basis data (*database*).
- **Akses Dokumen Fleksibel (*Preview & Download*):** Setelah data ditemukan, sistem menyediakan dua opsi akses bagi Kontraktor. Pengguna dapat memilih untuk melakukan Pratinjau (*Preview*) guna memeriksa isi dokumen di peramban (*browser*) terlebih dahulu, atau melakukan Unduhan Langsung (*Direct Download*) agar berkas PDF termeterai tersebut segera tersimpan ke dalam memori perangkat (*device*) lokal.

6. **Implementasi Notifikasi Otomatis (*Auto-Email*)**

Gambar 3.24. Hasil Auto Email Dokumen Termaterai

Sebagai tahap akhir dari alur kerja, diterapkan sistem automasi notifikasi. Setelah kontraktor menekan tombol ”Simpan” dan data berhasil terkirim ke *backend* (Google Apps Script), sistem secara otomatis memicu pengiriman email kepada Manager dan Koordinator proyek seperti pada Gambar 3.24. Notifikasi ini berfungsi sebagai pemberitahuan *real-time* bahwa terdapat dokumen termeterai baru yang telah masuk dan siap untuk ditinjau.

- **Pemicu Pengiriman *Real-time*:** Sistem dirancang untuk mengirimkan notifikasi secara instan segera setelah Kontraktor menekan tombol ”Simpan” pada halaman *input* dan data berhasil tervalidasi di *server*. Hal ini menjamin bahwa manajemen menerima informasi tanpa penundaan.
- **Distribusi Informasi ke Pemangku Kepentingan:** Email otomatis ditujukan langsung kepada Manager dan Koordinator Proyek. Di dalam badan email, sistem menyusun informasi metadata yang lengkap, mencakup Tanggal *Upload*, Nama Cabang, Kode ULOK, serta Lingkup Pekerjaan, sehingga penerima dapat langsung mengidentifikasi konteks proyek tanpa perlu membuka aplikasi terlebih dahulu.
- **Integrasi Tautan Dokumen dan Formulir SPK:** Untuk efisiensi kerja,

notifikasi ini dilengkapi dengan tautan/lampiran dokumen PDF (RAB & SPH) yang baru saja diunggah. Selain itu, disertakan pula tautan menuju Formulir SPK di dalam email tersebut, sehingga memungkinkan Manager atau Koordinator untuk langsung melanjutkan proses ke tahap pembuatan Surat Perintah Kerja (SPK) hanya dengan satu kali klik.

3.3.2 Kendala yang Ditemukan

Selama proses pengembangan dan implementasi sistem, ditemukan beberapa kendala teknis maupun nonteknis yang memengaruhi stabilitas aplikasi. Kendala-kendala tersebut dianalisis untuk ditemukan akar masalahnya, kemudian diterapkan solusi yang relevan agar sistem dapat berjalan optimal. Adapun kendala dan solusi tersebut adalah sebagai berikut:

1. **Duplikasi Data pada Proses Revisi (*Update Anomaly*):**
   Salah satu kendala *bug* paling kritis terjadi pada mekanisme penyimpanan data di Google Spreadsheet. Ketika data opname yang berstatus "*Rejected*" (ditolak oleh kontraktor) diperbaiki dan dikirim ulang oleh PIC, sistem gagal mengenali data lama tersebut. Akibatnya, alih-alih memperbarui (*update*) baris yang sudah ada, sistem justru membuat baris data baru (*insert*). Hal ini menyebabkan munculnya data ganda (*redundant*) dan jenis pekerjaan yang seharusnya tidak ada, sehingga mengacaukan rekapitulasi laporan akhir.

2. **Ketidaksesuaian Kalkulasi Total Harga:**
   Pada tahap awal pengembangan formulir opname, ditemukan ketidakakuratan pada fitur kalkulasi otomatis. Total harga yang tampil di antarmuka terkadang tidak sesuai dengan hasil perkalian antara volume dan harga satuan. Masalah ini disebabkan oleh kesalahan penanganan format data (*data type handling*) antara *string* dan *number* saat diproses oleh JavaScript di sisi *client*, serta perbedaan format desimal pada data yang ditarik dari RAB.

3. **Keterbatasan Infrastruktur *Cloud Hosting* (Isu *Runtime*):**
   Mengingat proyek ini menggunakan layanan *hosting* versi gratis (*Free Tier*) pada *platform* Render untuk *backend*, ditemukan kendala pembatasan kuota penggunaan (*runtime hours*). Akibat tingginya aktivitas akses saat uji coba, kuota *runtime server* habis sebelum bulan berakhir, yang mengakibatkan akun terkena *suspend* sementara. Dampaknya, aplikasi web tidak dapat

diakses oleh pengguna selama beberapa hari hingga kuota diatur ulang secara otomatis.

4. **Isolasi *Bug* Minor Antarmuka:**
   Selain kendala utama di atas, terdapat beberapa *bug* minor pada antarmuka pengguna (*frontend*), seperti elemen tampilan yang tidak responsif pada resolusi layar tertentu dan pesan kesalahan (*error handling*) yang kurang informatif saat terjadi gangguan koneksi internet.

3.3.3 Solusi atas Kendala yang Ditemukan

Untuk mengatasi kendala-kendala tersebut, diterapkan langkah-langkah penyelesaian sebagai berikut:

1. **Perbaikan Logika *Upsert* pada *Backend*:**
   Dilakukan revisi pada logika penyimpanan data dalam berkas `server.mjs`. Diterapkan mekanisme identifikasi unik (*unique identifier*) yang lebih ketat dengan menggabungkan variabel Kode Toko, Nomor ULOK, dan Jenis Pekerjaan sebagai kunci utama (*primary key*). Dengan logika ini, sebelum data disimpan, sistem akan memindai *spreadsheet* terlebih dahulu; jika data ditemukan, sistem akan melakukan pembaruan (*update*); jika tidak, baru dilakukan penambahan baris baru (*insert*). Solusi ini berhasil menghilangkan masalah duplikasi data saat proses revisi.

2. **Normalisasi Tipe Data untuk Kalkulasi Akurat:**
   Dilakukan perbaikan pada fungsi aritmetika di komponen `OpnameForm.js` dengan menambahkan fungsi utilitas normalisasi (`toNumber` atau `parseFloat`). Fungsi ini memastikan seluruh *input* angka dan harga dikonversi ke format numerik yang valid sebelum dilakukan operasi matematika, sehingga perhitungan total harga dan selisih biaya menjadi akurat dan konsisten.

3. **Implementasi Pembatasan Jam Operasional Sistem:**
   Sebagai solusi taktis untuk mengatasi keterbatasan kuota *runtime* pada Render, diterapkan kebijakan pembatasan akses aplikasi secara terprogram. Ditambahkan logika validasi waktu pada sistem *login* atau *middleware*, di mana aplikasi hanya dapat diakses pada hari kerja (Senin–Kamis) dan pada jam operasional tertentu (06.00–18.00 WIB). Strategi ini terbukti efektif

dalam menghemat penggunaan sumber daya *server* sehingga aplikasi tetap dapat berjalan stabil sepanjang bulan tanpa terkena penangguhan (*suspend*).

4. **Optimasi dan *Debugging* Berkala:**
   Untuk menangani *bug* minor, dilakukan proses *debugging* rutin berdasarkan laporan dari pengguna saat sesi uji coba (*trial*). Perbaikan meliputi penyesuaian CSS untuk tampilan yang lebih responsif serta penambahan indikator pemuatan (*loading*) dan pesan notifikasi yang jelas agar pengguna memahami status sistem saat terjadi proses pengiriman data.