BAB III

PELAKSANAAN KERJA

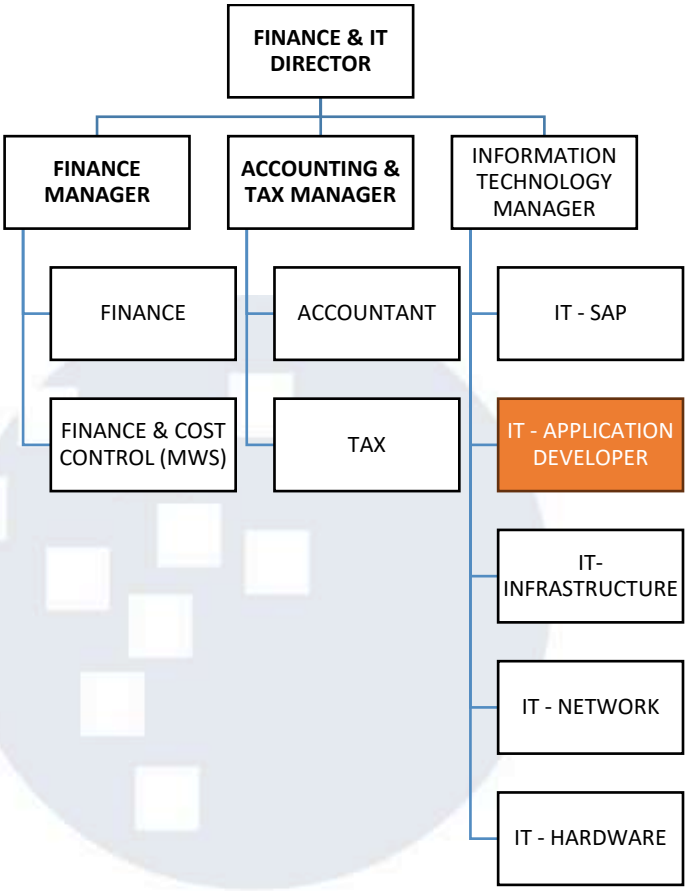
3.1 Kedudukan dan Koordinasi

Kontribusi selama program magang sangat dipengaruhi oleh kejelasan posisi dan alur kerja yang diterapkan. Oleh karena itu, bagian ini menguraikan secara rinci kedudukan **peserta magang** dalam struktur tim IT PT. Indonesia Morowali Industrial Park (IMIP), serta mekanisme koordinasi yang diterapkan bersama pembimbing lapangan dalam setiap pelaksanaan proyek..

3.1.1 Kedudukan

Sebagai mahasiswa Program Studi Sistem Informasi dengan peminatan akademis di bidang basis data (database), penempatan dilakukan pada Divisi IT, khususnya pada bagian IT – Application Developer. Penempatan tersebut disesuaikan dengan kompetensi yang dimiliki sehingga memungkinkan keterlibatan secara langsung dalam mendukung dan mengerjakan proyek-proyek pengembangan aplikasi yang sedang berjalan..

Berbeda dengan sistem rotasi, penempatan magang ini ditempatkan secara permanen di tim *Application Developer* untuk ikut terlibat dalam sejumlah proyek pengembangan secara menyeluruh, mulai dari tahap awal hingga tahap testing dan penyelesaian. Misalnya, saat mengerjakan proyek “Dashboard Pembelian” , yang difokuskan untuk menyelesaikan tugas-tugas terkait hingga proyek tersebut selesai, sebelumnya yang nantinya diahlikan ke proyek-proyek selanjutnya.Tugas yang diberikan akan terfokus pada riset teknologi baru, pembuatan *database,* dan pengembangan aplikasi yang relevan dengan agenda *technology refreshment* perusahaan. Rincian posisi peserta magang dalam struktur Divisi IT disajikan pada Gambar 3.1 berikut.



Gambar 3. 1 Struktur Organisasi IT Perusahaan

Secara organisasi, bagian IT - Application Developer berada di bawah *Information Technology Manager* dan dipimpin oleh *finance & IT director*. Untuk pengerjaan tugas dan koordinasi harian, anak magang bertanggung jawab langsung kepada pembimbing lapangan (*supervisor*) yang telah ditunjuk di dalam tim.

3.1.2 Koordinasi (SOP)

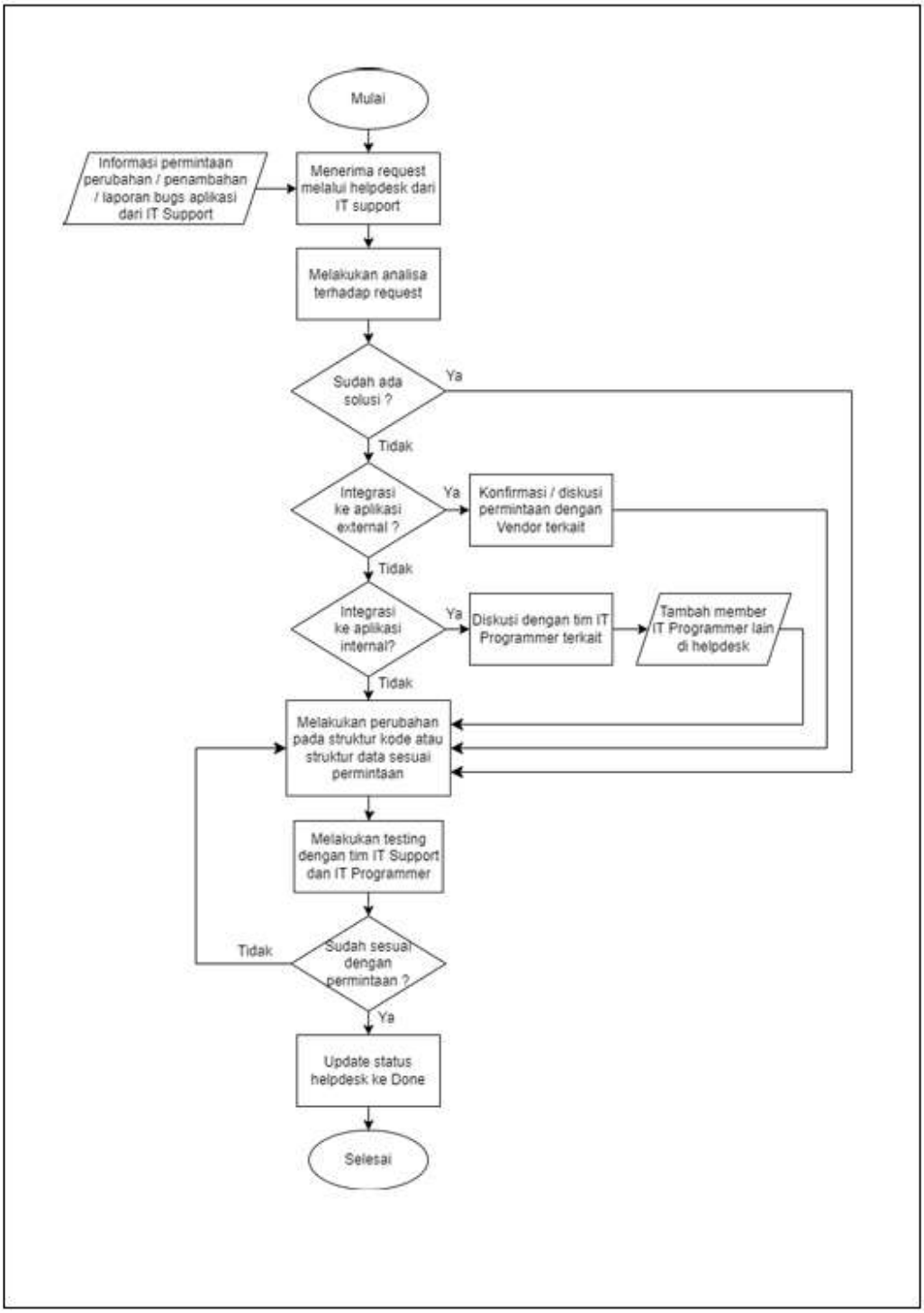
Setiap tugas yang dikerjakan atau yang diberikan selama periode magang, mengikuti alur kerja yang sistematis sesuai dengan Prosedur Operasional Standar (POS) yang sudah berlaku di departemen IT. Alur in dirancang untuk memastikan bahwa setiap adanya pengembangan, perubahan, atau perbaikan aplikasi berjalan secara efektif, terdokumentasi, dan sesuai dengan kebutuhan yang dibutuhkan oleh *end user*. Proses ini dimulai dari penerimaan tugas hingga penyelesaian dan implementasi.

Alur kerja diawali dengan penerimaan tugas dari *supervisor Application Development* atau pembimbing lapangan, yang biasanya disampaikan melalui sistem *ticketing* internal perusahaan, yaitu Aplikasi *IT Helpdesk.* Setelah tugas diterima, divisi IT melakukan analisis mendalam teradap permintaan tersebut untuk memahami ruang lingkup, tujuan, dan kebutuhan teknisnya. Tahap ini sering biasanya sering melibatkan riset terhadap teknologi atau metode-metode yang paling sesuai untuk menyelesaikan masalah.

Setelah analisis selesai, proses dilanjutkan ke tahap pengembangan atau implementasi, di mana divisi IT mulai mengerjakan solusi, baik dari penulisan kode, pembuatan skrip otomatisasi, desain antarmuka, dan pengujian aplikasi. Setelah tahap/versi awal selesai, divisi IT akan melakukan pengujian internal untuk memastikan fungsionalitas dasar berjalan sesuai harapan. Hasil dari pekerjaan ini kemudian diserahkan kepada *advisor* untuk proses *review*. Jika terdapat masukan atau perbaikan, maka akan dilakukan revisi hingga hasilnya menyesuaikan dengan apa yang diminta. Proses revisi ini merupakan siklus berulang yang *crucial* untuk *quality control.*

Apabila hasil pekerjaan telah disetujui oleh *advisor*, langkah selanjutnya adalah melakukan pengujian bersama dengan tim terkait, seperti tim IT *Support* atau tim *Quality Assurance* (QA), untuk memastikan solusi yang dibuat telah sesuai dengan permintaan awal pengguna. Setelah berhasil melewati tahap pengujian akhir, tugas dinyatakan selesai dan statusnya diperbarui pada sistem *IT Helpdesk.*

Secara visual, alur pelaksanaan pekerjaan dapat digambarkan pada gambar bagan 3.2 seperti berikut:



Gambar 3. 2 Bagan Alur Koordinasi
Sumber: IMIP/ICT/SOP-21

3.2 Tugas yang Dilakukan

Peserta magang di departemen IT Application Developer PT. Indonesia Morowali Industrial Park (IMIP) memiliki berbagai tugas yang bervariasi, tergantung pada kebutuhan proyek yang sedang berlangsung. Seluruh alur kerja-mulai dari permintaan pembuatan fitur baru dan perubahan sistem dikelola dan dimonitor secara terpusat melalui sistem ticketing internal yang disebut Aplikasi *IT Helpdesk.* Detail mengenai pelaksanaan kerja magang dan rincian tugas yang dikerjakan dapat dilihat dalam Tabel 3.1 berikut ini:

Tabel 3. 1 Detail Pekerjaan yang Dilakukan

No.	Minggu	Proyek	Pekerjaan yang dilakukan	Start Date	End Date
Tahap Awal Magang & Onboarding					
1	Minggu 1	Onboarding	Belajar Struktur dan SOP dalam perusahaan	**21-07-25**	**22-07-25**
2	Minggu 1	Onboarding	Learning and adjusting, belajar setup laravel project	**23-07-25**	**24-07-25**
3	Minggu 1	Onboarding	Membuat json responses simple di laravel	**24-07-25**	**25-07-25**
Laravel Purchasing Dashboard & Reporting Web Service					
4	Minggu 1	Dashboard Laravel	Migrasi seluruh data yang diperlukan	**24-07-25**	**25-07-25**
5	Minggu 1	Dashboard Laravel	Membuat routing system untuk menggabungkan semua dashboard view	**24-07-25**	**25-07-25**
6	Minggu 1-2	Dashboard Laravel	Membuat view dashboard	**28-07-25**	**01-08-25**

No.	Minggu	Proyek	Pekerjaan yang dilakukan	Start Date	End Date
7	Minggu 2	Dashboard Laravel	Membuat analytics dashboard	**28-07-25**	**01-08-25**
8	Minggu 2-3	Dashboard Laravel	Membuat view master vendors, projects, requests, dan purchases	**28-07-25**	**08-08-25**
9	Minggu 2-4	Dashboard Laravel	Membuat tampilan master lengkap dengan sistem CRUD Filtering data, dan export to excel	**08-08-25**	**15-08-25**
10	Minggu 3-5	Dashboard Laravel	membuat sistem import csv (Drag and drop) dengan template yang disedikan	**10-08-25**	**15-08-25**
11	Minggu 4-5	Dashboard Laravel	membuat nav bar didalam layout,	**15-08-25**	**22-08-25**
12	Minggu 4-5	Dashboard Laravel	Membuat navigation bar setting	**22-08-25**	**25-08-25**
13	Minggu 5	Dashboard Laravel	Penyerahan dan Github Submit	**25-08-25**	**26-08-25**
OCR System Web Service					
14	Minggu 5	OCR Service	membuat image to text dengan OCR (Optical Character Recognition)	**27-08-25**	**29-08-25**
15	Minggu 5	OCR Service	Melakukan testing menggunakan	**28-08-25**	**29-08-25**

No.	Minggu	Proyek	Pekerjaan yang dilakukan	Start Date	End Date
			TesseractOCR & EasyOCR		
16	Minggu 5	OCR Service	Membuat Python untuk logika OCR berjalan dan Laravel untuk menampilkan sistem berupa website.	**28-08-25**	**29-08-25**
17	Minggu 5-6	OCR Service	Memperbaiki tampilan website agar lebih proper, reviewing dan minor adjustment.	**29-08-25**	**01-09-25**
18	Minggu 6	OCR Service	Memisahkan Python (logic OCR) dengan file laravel	**01-09-25**	**03-09-25**
19	Minggu 8	OCR Service	mengintegrasikan OCR system kedalam web-service telegram	**25-09-25**	**26-09-25**
20	Minggu 8	OCR Service	Memasukkan OCR kedalam fitur manager-only.	**26-09-25**	**27-09-25**
Automasi Persetujuan Pembelian via Chatbot Telegram					
21	Minggu 6	Chatbot Telegram	Melakukan rancangan membuat bot telegram. Untuk stock purchasing update	**04-09-25**	**05-09-25**
22	Minggu 6	Chatbot Telegram	Mempelajari cara kerja bot telegram, dan	**04-09-25**	**05-09-25**

No.	Minggu	Proyek	Pekerjaan yang dilakukan	Start Date	End Date
			membuat bot telegram menggunakan botfather.		
23	Minggu 6-7	Chatbot Telegram	menambahkan command pada bot untuk cek report pebulannya dalam 1 tahun	**05-09-25**	**09-09-25**
24	Minggu 7	Chatbot Telegram	Membuat command baru pada bot untuk bisa melihat purchasing berdasarkan vendor	**09-09-25**	**10-09-25**
25	Minggu 7	Chatbot Telegram	Menambahkan fitur scheduler pada bot	**09-09-25**	**10-09-25**
26	Minggu 7	Chatbot Telegram	menambahkan fitur approval ketika ada request pembelian bisa di approve / direject melalui bot.	**10-09-25**	**12-09-25**
27	Minggu 7	Chatbot Telegram	Menambah fitur command menampilkan project report dan logistics.	**15-09-25**	**16-09-25**
28	Minggu 7-8	Chatbot Telegram	Membuat sistem role dan manager view pada telegram service web	**16-09-25**	**22-09-25**

No.	Minggu	Proyek	Pekerjaan yang dilakukan	Start Date	End Date
29	Minggu 8	Chatbot Telegram	membuat manager only notification dan integrasiin agar telegram web service bisa dipakai oleh banyak pengguna.	**22-09-25**	**24-09-25**
30	Minggu 15	Chatbot Telegram	Membuat fitur chat subscription agar *user* yang sudah tidak bekerja bisa dicabut aksesnya.	**03-11-25**	**05-11-25**
31	Minggu 15	Chatbot Telegram	Menambahkan fitur register by email, *user* bisa memliki banyak chat id sesuai dengan email yang sudah terdaftar	**06-11-25**	**07-11-25**
LLM Ticketing Sentence Similarity Microservice					
32	Minggu 9	LLM Ticketing	learning and testing hugging face model	**29-09-25**	**01-10-25**
33	Minggu 9	LLM Ticketing	migrate csv membuat tampilan sederhana ticket help desk	**02-10-25**	**03-10-25**
34	Minggu 9-10	LLM Ticketing	adjusting python script dan layouting web laravel	**06-10-25**	**09-10-25**
35	Minggu 10	LLM Ticketing	adjusting python script and adding	**10-10-25**	**13-10-25**

No.	Minggu	Proyek	Pekerjaan yang dilakukan	Start Date	End Date
			extra feature for ticketing		
36	Minggu 11	LLM Ticketing	melanjutkan membuat script similarity sentence	**13-10-25**	**14-10-25**
37	Minggu 11	LLM Ticketing	presentasi project	**14-10-25**	**14-10-25**
38	Minggu 11	LLM Ticketing	fine tuning script sentence similarites	**15-10-25**	**16-10-25**
39	Minggu 11	LLM Ticketing	melanjutkan fine tuning script sentence similarites	**16-10-25**	**17-10-25**
Designing And Adjusting IMIP APPS					
40	Minggu 12	Designing IMIP Apps	Merancang design IMIP Apps	**20-10-25**	**24-10-25**
41	Minggu 12	Designing IMIP Apps	Revisi design home page #1	**20-10-25**	**24-10-25**
42	Minggu 12	Designing IMIP Apps	Membuat Carousel	**20-10-25**	**21-10-25**
43	Minggu 12	Designing IMIP Apps	Membuat News block	**20-10-25**	**22-10-25**
44	Minggu 12	Designing IMIP Apps	Cari referensi	**20-10-25**	**23-12-25**
45	Minggu 13	Designing IMIP Apps	Membuat halaman news	**27-10-25-**	**27-10-25**
46	Minggu 13	Designing IMIP Apps	Revisi hero section	**28-10-25**	**29-10-25**
47	Minggu 13	Designing IMIP Apps	Mencari *icon* yang pas	**28-10-25**	**31-10-25**

No.	Minggu	Proyek	Pekerjaan yang dilakukan	Start Date	End Date
48	Minggu 14	Designing IMIP Apps	Membuat news detail	**03-11-25**	**05-11-25**
49	Minggu 14	Designing IMIP Apps	Membuat Videoplayer	**03-11-25**	**07-11-25**

3.3 Uraian Pelaksanaan Kerja

Pelaksanaan kerja magang di PT. Indonesia Morowali Industrial Park (IMIP), mengikuti alur pengembangan *software* yang terstruktur, dimuali dari tahap persiapan hingga finalisasi. Pada tahap awal, peserta magang menjalani proses orientasi untuk mengenal lingkungan kerja, standard operasional prosedur, dan *stack* teknologi yang digunakan. Untuk project yang dikerjakan menggunakan *framework* Laravel untuk aplikasi web dan *Python* untuk kebutuhan *scripting* atau pemrosessan data spesifik. Pemahaman ini dilanjutkan dengan menganalisis kebutuhan untuk setiap proyek melalui serangkaian diskusi bersama pembimbing lapangan dan tim. Seluruh penugasan proyek diberikan dan diarahkan langsung oleh *supervisor*, yang memastikan kesesuaian pekerjaan dengan tujuan tim dan agenda *technology refreshment* perusahaan. Fokus utama pekerjaan adalah pada kegiatan riset dan pengembangan (R&D), di mana peserta magang bertanggung jawab mulai dari tahap analisis kebutuhan hingga pengujian awal. Dalam melaksanakan tugas-tugas *coding* dan pengembangan, *Integrated Development Environment* (IDE) yang digunakan adalah Visual Studio Code (VS Code). Setiap *project* melibatkan penggunaan teknologi dan pendekatan yang berbeda, memberikan pengalaman praktis dalam pengembangan *full-stack*, integrasi sistem, serta pemanfaatan *library* dan API eksternal. Secara umum, pekerjaan sehari-hari meliputi analisis masalah sesuai arahan, penulisan kode di VS Code, *debugging*, koordinasi rutin dengan supervisor, serta dokumentasi teknis sederhana

3.3.1 Proses Pelaksanaan

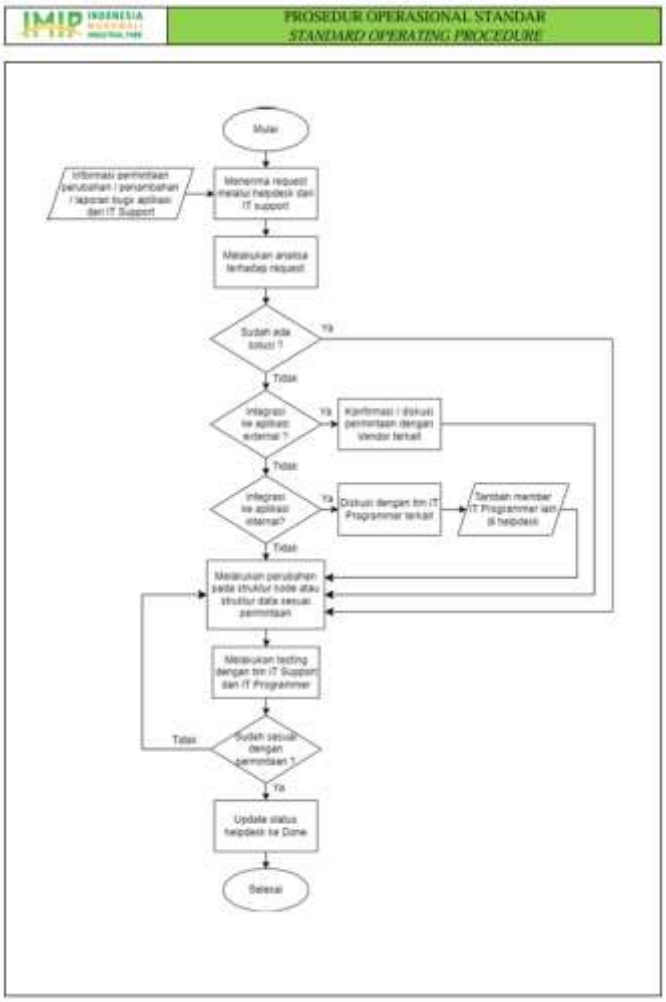
Bagian ini akan merincikan alur pengerjaan dari setiap proyek utama yang telah dilaksanakan selama proses magang, mulai dari perancangan hingga hasil akhir. Sesuai panduan dan penjelasan yang akan dilengkapi dengan *screenshot GUI* dan *code snipped* yang relevan untuk menggambarkan pekerjaan secara mendetail. Pekerjaan akan dipecah menjadi berbagai tahapan proses kerja utama yang paling signifikan, dimulai dengan proses *onboarding* dan penugasan proyek utama.

3.3.1.1 Tahap Awal Magang & Onboarding

Tahap awal pelaksanaan magang dimulai pada 21 Juli 2025 dengan proses onboarding berfokus pada adaptasi lingkungan dan pemahaman alur kerja, selama periode ini, peserta magang diperkenalkan dengan tim-tim yang IT Application Development dan struktur organisasi perusahaan, khususnya divisi IT. Seperti pada gambar 3.3.

Gambar 3. 3 Rapat tim IT App development

Seperti pada gambar 3.4 peserta magang lalu mempelajari *Standard Operating Procedures* (SOP) yang berlaku. SOP ini mencakup alur koordinasi harian, standar penulisan kode (*coding standard*), dan tools yang digunakan. Fase adaptasi ini sangat penting untuk menyesuaikan ritme kerja pribadi dengan tim *developer* di perusahaan.



Gambar 3. 4 Proedur Operasional Standard IT Application Developer

Setelah proses pengenalan non teknis, arahan pertama dari pembimbing adalah mempersiapkan *development environment* dan mempelajari *stack* teknologi utama yang digunakan, *framework* Laravel. Peserta magang secara mandiri mempelajari proses instalasi dan konfigurasi proyek Laravel baru menggunakan *dependency manager* Composer. Lalu melakukan pengaturan file .env untuk konfigurasi *environment* lokal, penyiapan koneksi ke *database* dan pemahaman mendalam mengenai struktur folder standard Laravel. Seperti pada gambar 3.5

Gambar 3. 5 Setup Laravel

Sebagai penugasan teknis pertama untuk memvalidasi *setup* lingkungan kerja, penulisan ditugaskan untuk membuat sebuah *endpoint* yang mengembalikan respon dalam format JSON. Proses ini melibatkan pendaftaran *route* baru pada file routes/api.php. Seperti pada gambar 3.6.

```
Route::prefix('pembelian')->controller(PembelianController::class)->group(function () {
    Route::get('/MostOrderMonth', 'mostOrdersMonth');
    Route::get('/Vendor', 'TotalVendor');
```

Gambar 3. 6 JSON Route di API.php

Route yang digunakan disini untuk memunculkan JSON response sesuai dengan controller yang dipanggil. Contohnya untuk /Vendor adalah alamat yang dipanggil *frontend* untuk meminta data Total Vendor. Ketika alamat ini diakses, Laravel akan menjalankan *function* TotalVendor di

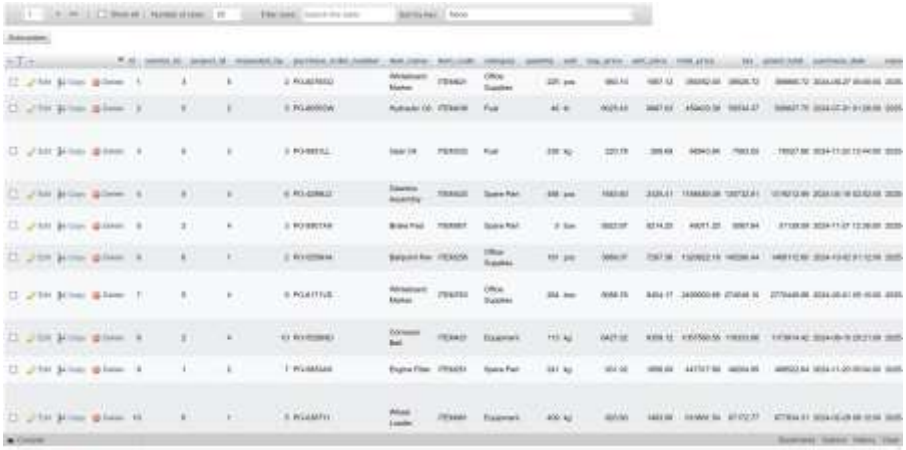
dalam PembelianController untuk menghitung data dan mengirimkannya kembali ke *dashboard*. Seperti pada gambar 3.7

Gambar 3. 7 JSON Response / Vendor

3.3.1.2 Laravel Purchasing Dashboard & Reporting Web Service

Project pertama yang dikerjakan adalah pengembangan *Purchasing Dashboard*, sebuah aplikasi web internal yang bertujuan untuk memvisualisasikan data dan aktivitas pembelian perusahaan. Tujuan utama dari project untuk mengubah data *purchasing* yang mentah dari *database* menjadi informasi analitik yang mudah dipahami oleh tim *management*, untuk membantu *decision making* untuk keperluan bisnis. Proses pengembangan proyek ini dibagi menjadi berbagai tahapan awal dari migration seluruh data yang diperlukan, membuat routing web dan API, Membuat view dashboard, membuat analytics dashboard, membuat view master, membuat CRUD master, membuat sistem import csv (Drag and drop), membuat nav bar didalam layout, navigation bar setting, dan terakhir pengumpulan ke github.

Hal pertama yang dilakukan adalah migrasi data yang diperlukan, tabel yang dipakai pada *project* ini adalah pembelian, vendors, projects, dan terkahir request. Masing-masing menyimpan data yang penting dan kegunaan spesifik untuk proses pelaporan dan analisis pada *dashboard*.



Gambar 3. 8 Tabel Pembelian

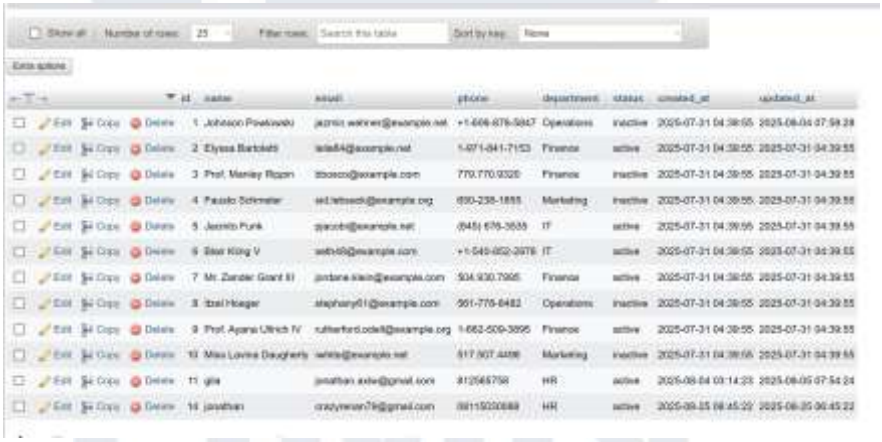
Seperti pada gambar 3.8, tabel ini berfungsi sebagai tabel utama dari *project* ini, menyimpan semua detail transaksi seperti tanggal, harga beli, harga jual (jika ada), status, dan identifikasi produk. Tabel ini menjadi sumber data mentah untuk semua kalkulasi analitik.

Gambar 3. 9 Tabel Vendor

Seperti pada gambar 3.9, tabel ini berfungsi untuk menyimpan data master vendor yang melakukan transaksi dengan perusahaan. Tabel ini akan dihubungkan dengan tabel utama yaitu tabel pembelian untuk menganalisis vendor dan melacak kinerja dari masing-masing vendor yang ada.

Gambar 3. 10 Tabel Projects

Seperti pada gambar 3.10, tabel ini digunakan untuk menyimpan *project-project* yang sedang dikerjakan di mana terdapat pembelian didalam project tersebut. Tabel ini memungkinkan *dashboard* untuk memfilter dan menyajikan data pembelian berdasarkan alokasi proyek yang sedang berjalan.



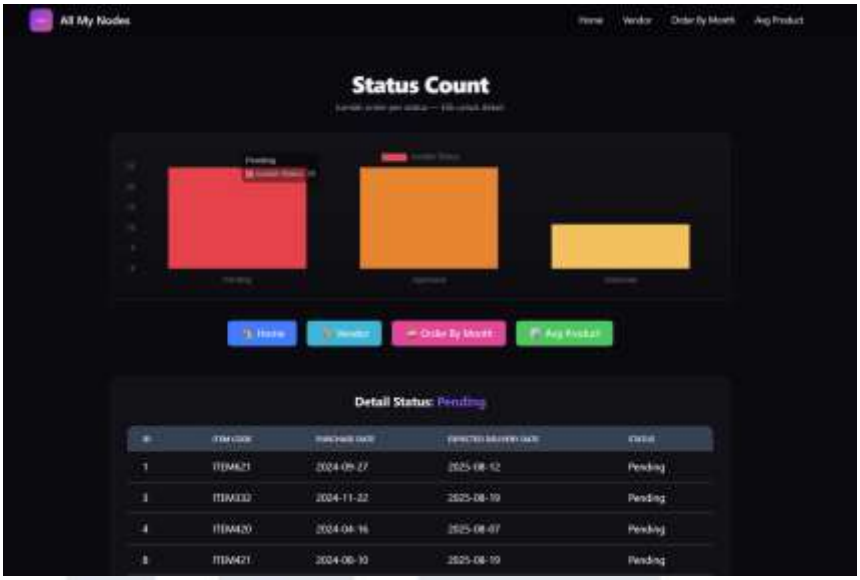
Gambar 3. 11 Tabel Request

Seperti pada gambar 3.11, tabel ini digunakan untuk menyimpan data pengajuan atau permintaan pembelian yang masuk. Tabel ini dihubungkan dengan tabel utama untuk melihat siapa aja *users* yang melakukan pembelian dan dari departemen apa yang melakukan pembelian. Setelah melakukan migrasi dan tabel-tabel ini sudah ada didalam database, mulai dengan membuat view-view dashboard didalam Laravel.Menggunakan blade sebagai *frontend* dan *PHP* untuk mengu(rus *controller* (backend). Hal pertama yang dilakukan adalah membuat routing untuk view dashboard utama yaitu dashboard jumlah *order* per

vendor, jumlah *order* per status, jumlah *order* per bulan , dan terakhir avg product.

Gambar 3. 12 Dashboard Order per Vendor

Seperti pada gambar 3.12 , kegunaan dari dasboard ini adalah untuk menjadi sebuah halaman laporan (*report*) spesifik yang didedikasikan untuk menganalisis kinerja dan aktivitas vendor. Kegunaan utamanya bagi manajemen adalah untuk menjawab pertanyaan-pertanyaan bisnis penting. Pada dashboard ini terdapat tampilan pie chart menggunakan *chart.js* dn juga terdapat Filter Berdasarkan Tanggal, yang dimana *user* dapat memilih tanggal ("From Date" dan "To Date") untuk menganalisis data dalam periode waktu spesifik. Setelah itu ada tabel untuk menampilkan daftar vendor yang sudah diolah. Dalam tabel juga terdapat *calculation* yaitu total belanja (jumlah total uang yang dibayarkan ke vendor) dan total proyek (jumlah *project* yang dikerjakan vendor)



Gambar 3. 13 Status Count Dashboard

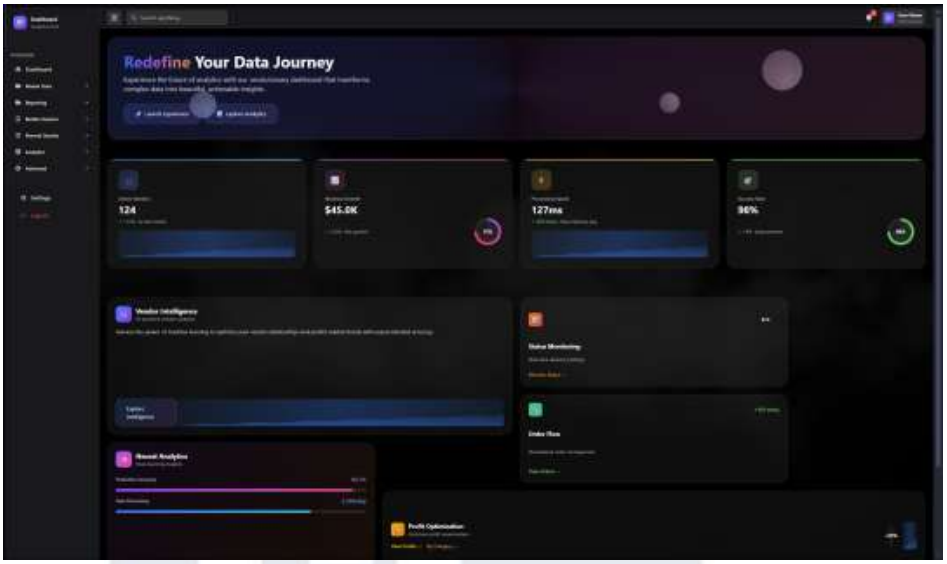
Seperti pada gambar 3.13, Dashboard view ini menampilkan halaman laporan visual yang didedikasikan untuk menganalisis distribusi status transaksi pembelian. Kegunaan utamanya bagi manajemen adalah untuk mendapatkan gambaran cepat (*snapshot*) mengenai alur kerja pembelian. Dashboard ini akan membantu manajemen melihat apakah ada jumlah transaksi yang tidak wajar yang tertahan pada status tertentu (misalnya, terlalu banyak yang "Pending" atau "Menunggu Persetujuan"). Lalu dari *bar chart* juga bisa melihat perbandingan antara transaksi yang sudah ”approved” , ”Pending” dan juga ”Delivered”. Disini data ditampilan menggunakan *google charts* membaca *array* data yang sudah masuk dan dijadikan visualisasi.

Gambar 3. 14 Order by Month Dashboard

Seperti pada gambar 3.14, dashboard ini bertujuan untuk memberikan laporan visualisasi yang dirancang khusus untuk melihat tren belanja harian. Kegunaan utama dari halaman ini adalah untuk memberikan wawasan kepada manajemen mengenai fluktuasi (naik-turunnya) total pengeluaran perusahaan perbulannya. Dashboard ini menampilkan *Line chart* yang memperlihatkan jumlah pembelian perbulannya. Dashboard ini dilengkapi dengan fitur *table view* dimana bisa melihat *item code* apa yang disorder pada bulan itu dan diisi dengan *purchase date* untuk menampilkan tanggal yang lebih detail. Melalui dashboard ini memudahkan untuk manajemen untuk secara cepat melihat jika terjadi lonjakan belanja yang tidak wajar pada hari tertentu, yang mungkin memerlukan investigasi lebih lanjut.

Gambar 3. 15 Average Product Dashboard

Seperti pada gambar 3.15, dashboard ini digunakan untuk menganalisis efisiensi dan kinerja logistik pengiriman yang dikelompokkan berdasarkan kategori produk. Fungi utama dari dashboard ini adalah untuk menghitung *rata-rata* jumlah hari yang dibutuhkan dari saat pemesanan hingga tanggal ekspektasi barang diterima untuk setiap *category*. Dashboard ini menampilkan data yang diurutkan sehingga manajemen dapat dengan sangat cepat melihat kategori produk mana yang memiliki waktu tunggu (lead time) pengiriman paling lama. Dashboard ini bisa digunakan untuk mengevaluasi apakah proses pengadaan atau vendor di *category* tertentu sering mengalami keterlambatan. Setelah membuat dashboard-dashboard sederhana itu tahap selanjutnya adalah membuat ”home” view untuk menampilkan atau sebagai *landing page* untuk project ini.



Gambar 3. 16 Home Page

Seperti yang ada pada gambar 3.16 ini adalah halaman utama yang akan dilihat oleh *user* setelah *login*. Halaman ini akan berfungsi sebagai *visual command center* yang memberikan gambaran ringkas dan analisis tingkat tinggi dari seluruh aktivitas pembelian yang terjadi. Halaman ini menampilkan empat "Kartu KPI" di bagian atas untuk data terpenting: Total Pembelian (*total transaction*), Total Vendor (*total vendor*), Total Proyek (jumlah *actiive project*), dan Total Permintaan (*total request*). Halaman ini juga menampilkan *bar chart* "Total Belanja (Monthly)". Ini yang digunakan untuk melihat kapan periode *peak purchasing* atau terendahnya terjadi dalam setahun.

Gambar 3. 17 CSS Style Code

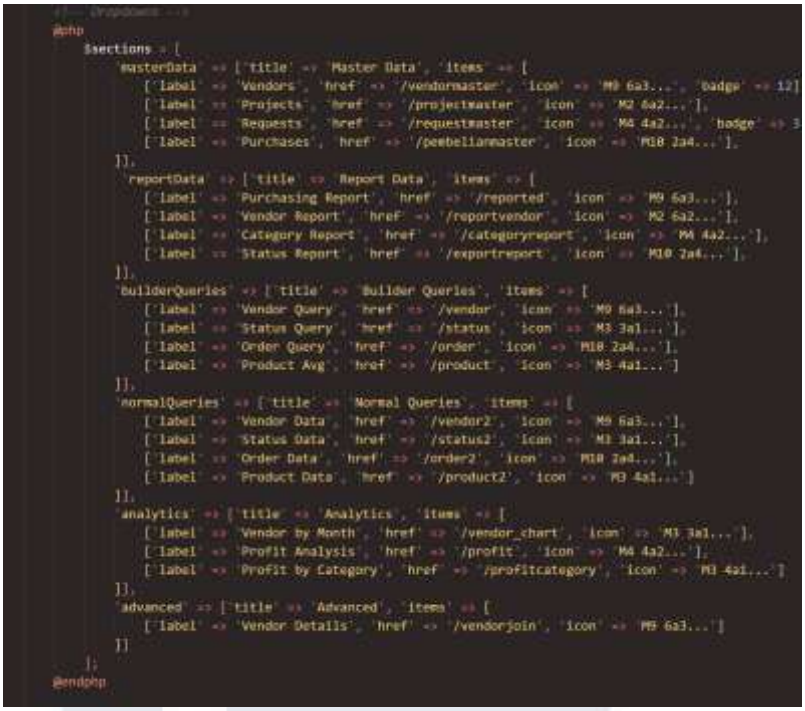
Seperti pada gambar 3.17 code snipped diatas adalah CSS yang digunakan untuk *home page* ini. Inspirasi pertama *design dashboard* adalah "Apple-style" yang modern. Ini menggunakan *dark mode* dan efek *glass morphism* (transparan dengan *blur*), seperti yang terlihat pada. apple-blur dan. bento-item.

```
.bento-item {
    background: rgba(28, 28, 30, 0.6);
    border: 1px solid rgba(255, 255, 255, 0.1);
    border-radius: 24px;
    padding: 1.5rem;
    position: relative;
    overflow: hidden;
    backdrop-filter: blur(20px);
    transition: all 0.4s cubic-bezier(0.4, 0, 0.2, 1);
    display: flex;
    flex-direction: column;
    justify-content: space-between;
}

.bento-item:hover {
    transform: translateY(-4px);
    border-color: rgba(255, 255, 255, 0.3);
    box-shadow: 0 20px 40px rgba(0, 0, 0, 0.3);
}
```

Gambar 3. 18 Bento-item CSS Code

Seperti pada gambar 3.18, tujuannya adalah menciptakan *interface* yang secara visual mengesankan dan intuitif, yang sejalan dengan tujuan perusahaan untuk pembaruan teknologi.



Gambar 3. 19 Sidebar Section

Code digambar 3.19 adalah code untuk menyimpan isi content" sidebar" yang masih ditulis secara *hardcode* kegunaan dari code ini untuk mengelompokkan semua halaman ke dalam kategori seperti "Master Data", "Reporting", dan "Builder Queries". *Dropdown* "Reporting" berfungsi sebagai menu yang mengarahkan *user* ke halaman-halaman laporan spesifik.

```
    // Function to animate number counters
    function animateValue(id, start, end, duration, prefix = '') {
        const obj = document.getElementById(id);
        if (!obj) return;

        let startTimestamp = null;
        const step = (timestamp) => {
            if (!startTimestamp) startTimestamp = timestamp;
            const progress = Math.min((timestamp - startTimestamp) / duration, 1);
            let value = Math.floor(progress * (end - start) + start);

            if (id === 'revenueCount') {
                obj.innerHTML = value.toFixed(1);
            } else {
                obj.innerHTML = value;
            }

            if (progress < 1) {
                window.requestAnimationFrame(step);
            }
        };
        window.requestAnimationFrame(step);
    }
</script>
```

Gambar 3. 20 Animation Script

Seperti pada gambar 3.20, halaman *dashboard* ini menampilkan empat kartu KPI utama, salah satunya adalah *Active Vendors*. Untuk memberikan *user experience* yang dinamis, angka pada kartu ini tidak muncul begitu saja, melainkan dianimasikan—dihitung naik dari 0 ke angka sebenarnya (misalnya, 124) saat halaman dimuat.

Gambar 3. 21 Jumlah Order per Vendor by Month Dashboard

Seperti pada gambar 3.22, kegunaan dari dashboard ini berfungsi sebagai alat analisis visual yang canggih untuk membedah kinerja vendor berdasarkan jumlah total pesanan (*total orders*). Kegunaan utamanya adalah untuk memberikan wawasan mendalam kepada manajemen mengenai vendor mana yang paling sering digunakan dan apa saja yang dibeli dari mereka. Dashboard ini menampilkan *pie chart* yang menunjukkan proporsi total pesanan untuk setiap vendor. Dengan *chart* ini, *user* dapat menarik kesimpulan vendor apa yang melakukan order terbanyak pada periode waktu yang sudah ditentukan. *User* tidak terbatas melihat data sepanjang waktu. Halaman ini menyediakan *form* "Start Date" dan "End Date". *User* dapat memilih periode spesifik (misalnya, "kuartal terakhir" atau "bulan lalu") dan menekan tombol "Apply" untuk memperbarui grafik secara langsung tanpa me-*refresh* halaman.

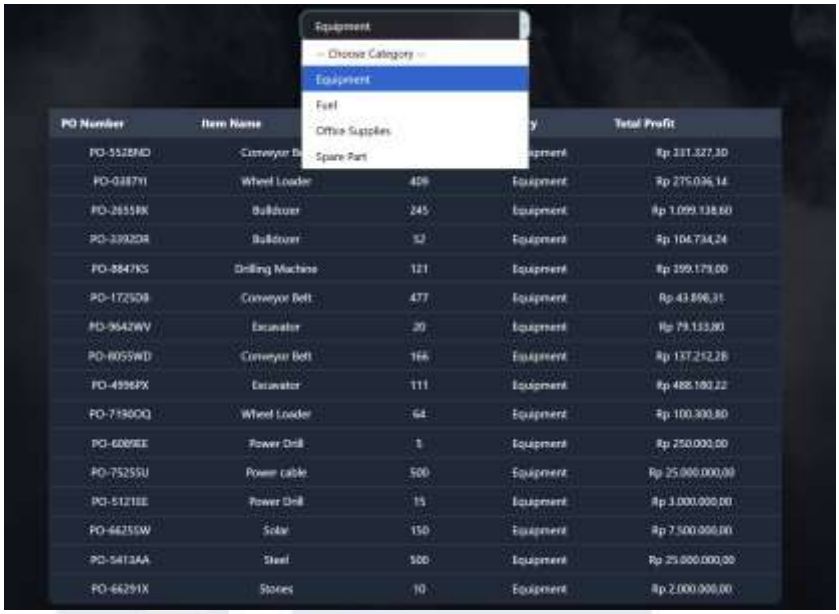
Gambar 3. 22 Drill Down Feature

Seperti pada gambar 3.22, pada dashboard ini terdapat fitur yang penting yaitu *drill-down* dimana, *user* dapat mengklik salah satu irisan

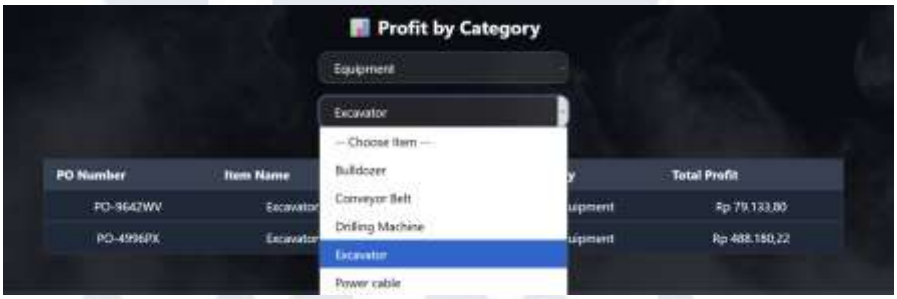
(*slice*) pada *Pie Chart*. Saat *user* menekan irisan *pie chart* misalnya, mengklik irisan "Vendor #2", sebuah tabel detail akan langsung muncul di bawah grafik. Tabel ini akan menampilkan daftar lengkap semua barang yang dibeli dari "Vendor #2" selama periode yang telah difilter, lengkap dengan nomor PO, nama barang, kode barang, *category* dan jumlahnya.

Gambar 3. 23 Profit by Category Dashboard

Seperti pada gambar 3.23, dashboard ini berfungsi untuk menganalisis profitabilitas (keuntungan) secara mendalam, dari level kategori produk hingga ke level per barang. Dari dashboard ini bisa menarik kesimpulan total profit dari seluruh transaksi di kategori apa. Dalam dashboard ini terdapat fitur *cascading dropdown* dimana pertama-tama memlih kategori produk dari *dropdown* pertama.



Gambar 3.23. 1 Cascading Filter

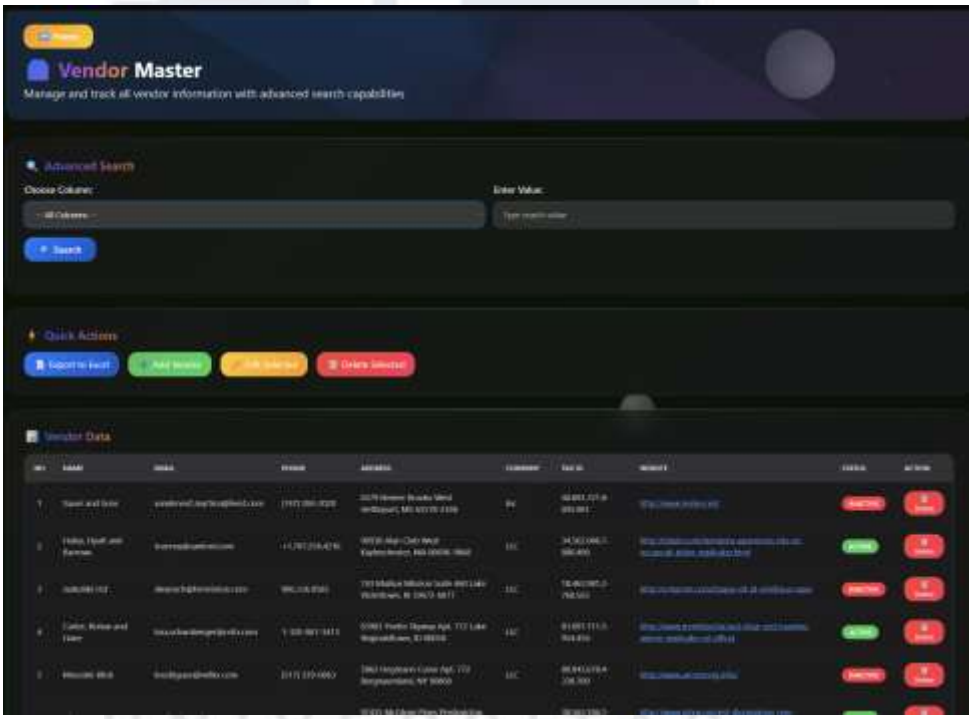


Gambar 3.23. 2 Cascading Filter #2

Setelah kategori dipilih, halaman ini secara otomatis melakukan dua hal , dashboardnya akan menampilkan *dropdown* kedua yang berisi daftar Nama Item yang spesifik *hanya* untuk kategori yang dipilih tadi. Seperti gambar 3.23.1 langsung menunjukkan tabel sesuai filter yang pertama. Setelah itu *user* dapat bih mempersempit pencarian dengan memilih Nama Item spesifik dari *dropdown* kedua seperti contoh pada gambar 3.23.2 (Excavator). Dengan kedua filter ini maka Tabel di bawahnya akan otomatis memuat ulang, kali ini hanya menampilkan data profitabilitas untuk "Excavator" saja. Tabel hasil di bagian bawah menunjukkan rincian per transaksi, seperti nomor PO, nama barang, jumlah, dan yang

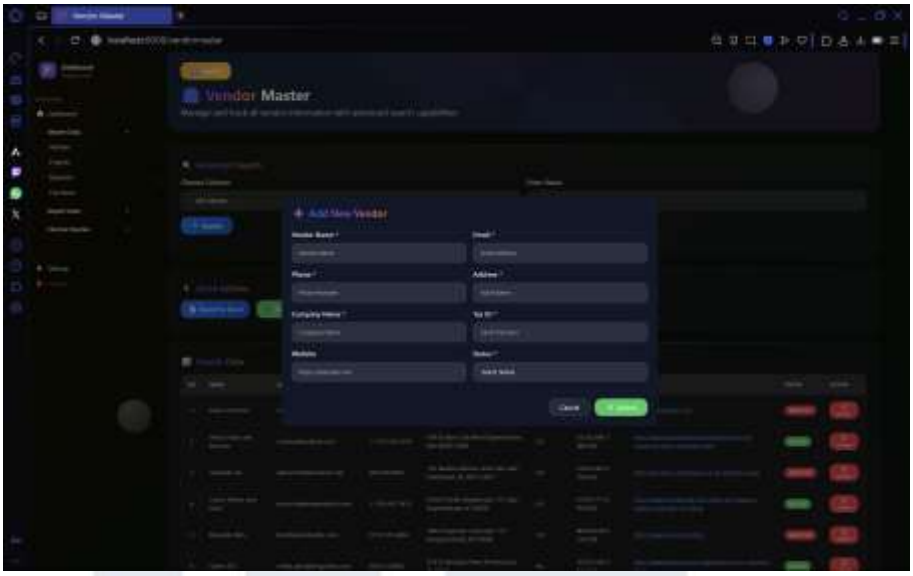
terpenting, *column* Total Profit, yang dihitung dari selisih (harga jual - harga beli) dikalikan jumlah barang.

Setelah membuat dashboard *analytics* sekarang tahap selanjutnya adalah membuat tabel master dimana pada *project* ini kita menggunakan 4 tabel utama yaitu vendor,project, request, dan purchase table. Tiap view master table akan memiliki fitur filtering sesuai berdasarkan *column* yang ada pada table, serta adanya fitur CRUD (Create, read, update & delete). Untuk masing-masing tabel data agar bisa diubah, dihilangkan, dan ditambah.



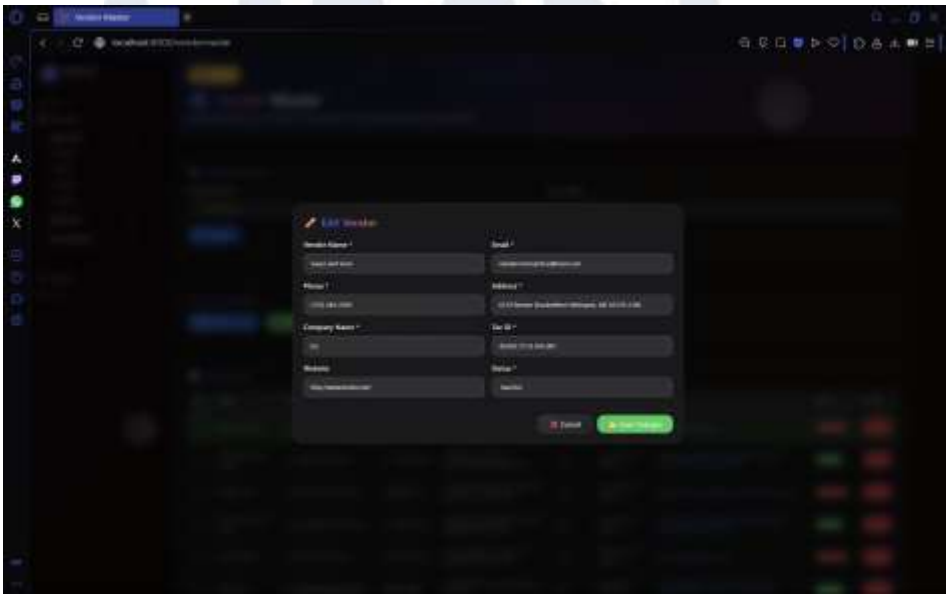
Gambar 3. 24 Vendor Master View

View ini adalah halaman administrasi inti yang berfungsi sebagai pusat data utama untuk semua vendor. Seperti pada gambar 3.24, halaman ini bukanlah laporan analitik, tetapi *interface* operasional untuk manajemen data vendor. Dalam *view* ini adalah untuk melakukan operasi CRUD (Create, Read, Update, Delete) secara penuh.



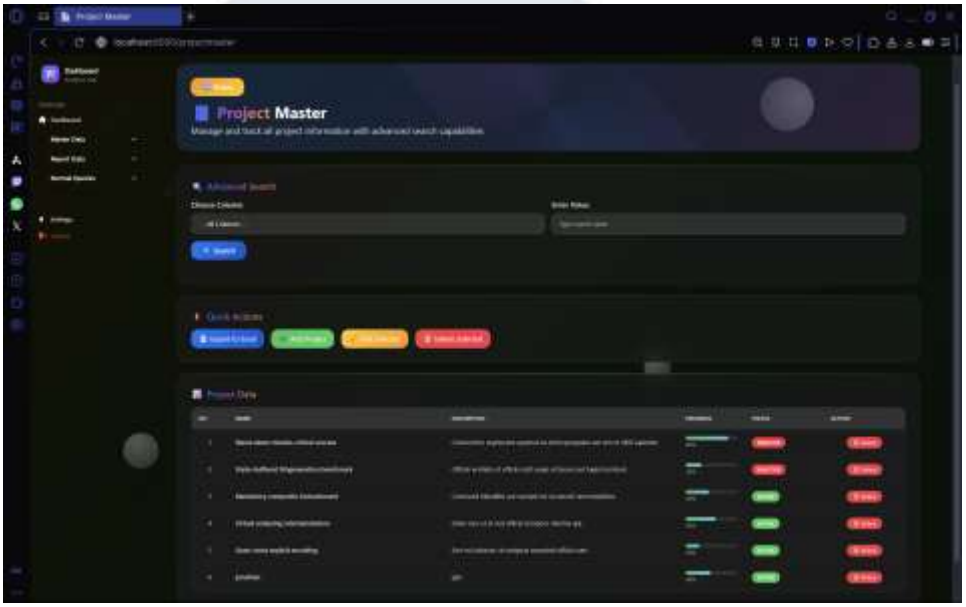
Gambar 3. 25 Create Vendor View

Pada gambar 3.25 administrator dapat menambahkan data vendor baru ke dalam sistem (misalnya, nama vendor, kode vendor, dan status) melalui *form* "Tambah Vendor". Halaman ini menampilkan tabel data lengkap dari semua vendor yang ada di *database* (vendors table). Tabel ini interaktif, dilengkapi dengan fitur pencarian (search) dan pagination (pembagian halaman) untuk mengelola data dalam jumlah besar.



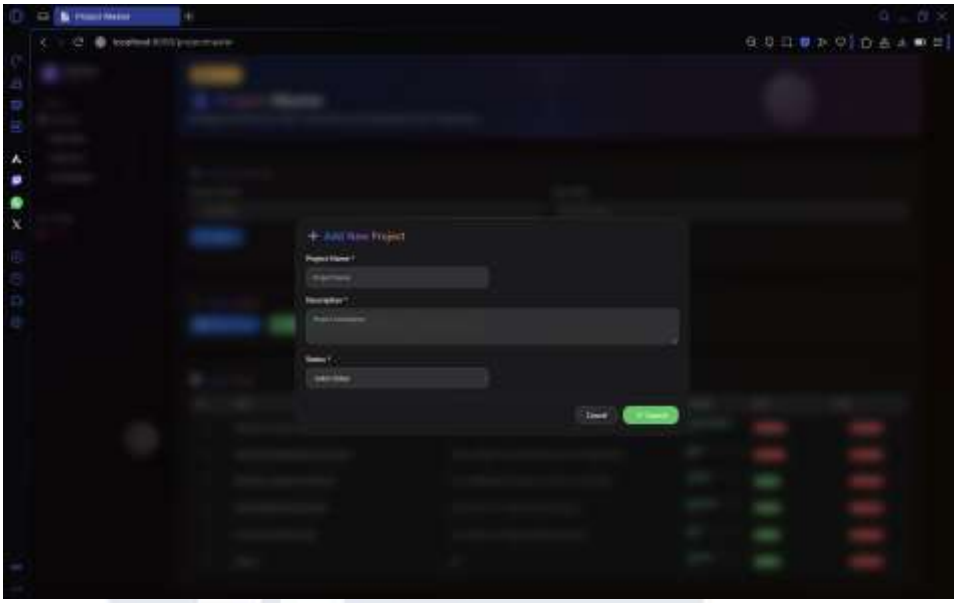
Gambar 3. 26 Update Vemdpr View

Seperti pada gambar 3.26, administrator dapat mengklik tombol "Edit" pada vendor yang ada untuk mengubah informasinya (misalnya, mengubah status vendor dari "Aktif" menjadi "Tidak Aktif").Lalu terakhir admin dapat menghapus data vendor dari *database* secara permanen. Selain CRUD *view* ini juga terdapat *export to excel* untuk membagikan data dalam bentuk table excel.



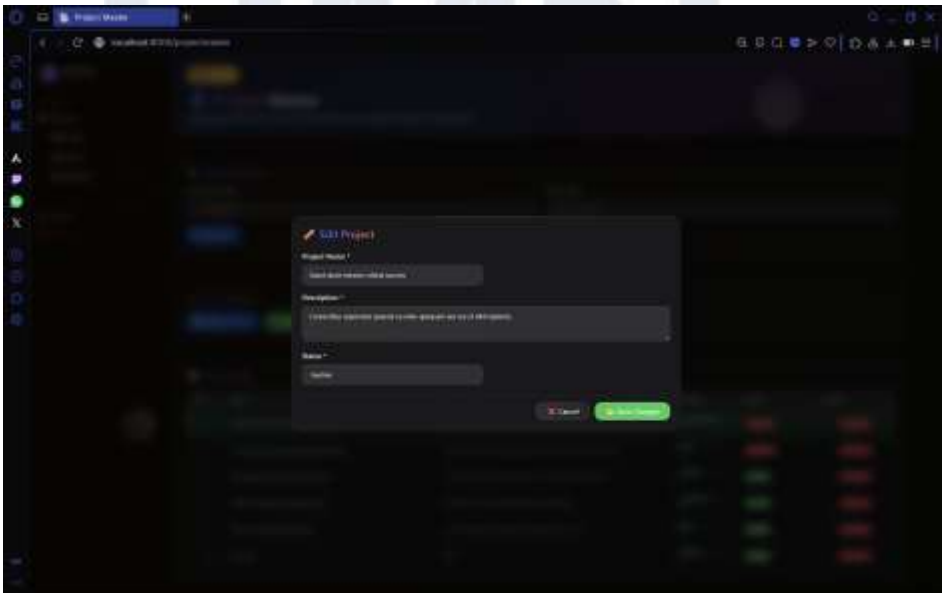
Gambar 3. 27 Project Master View

Seperti pada gambar 3.27, *view* ini adalah halaman administrasi inti yang berfungsi sebagai pusat data utama untuk semua *project*. Halaman ini bukanlah laporan analitik, melainkan *interface* operasional untuk manajemen data *project* Dalam *view* ini adalah untuk melakukan operasi CRUD (Create, Read, Update, Delete) secara penuh.



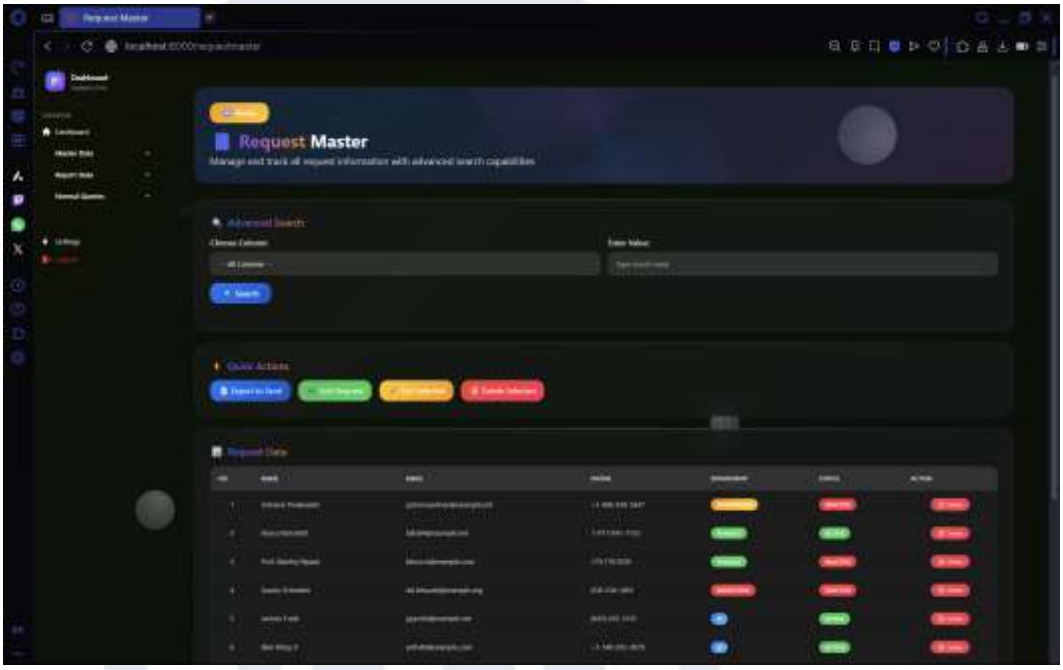
Gambar 3. 28 Create Project View

Pada gambar 3.28 administrator dapat menambahkan data *project* baru ke dalam sistem (misalnya, nama project description, dan status) melalui *form* "Tambah Project". Halaman ini menampilkan tabel data lengkap dari semua *project* yang ada di *database* (project table). Tabel ini interaktif, dilengkapi dengan fitur pencarian (search) dan pagination (pembagian halaman) untuk mengelola data dalam jumlah besar.



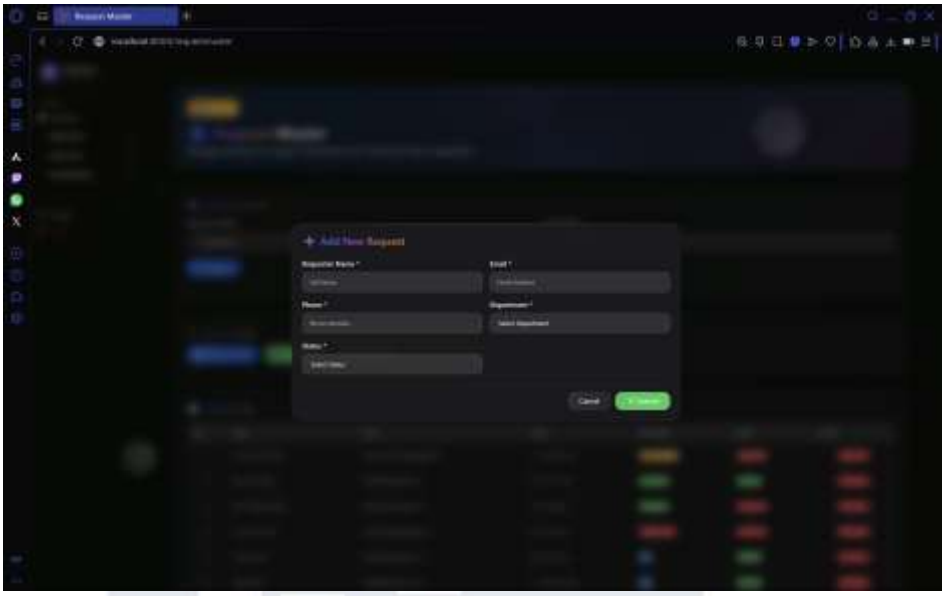
Gambar 3. 29 Update Project View

Seperti pada gambar 3.29, administrator dapat mengklik tombol "Edit" pada *project* yang ada untuk mengubah informasinya (misalnya, mengubah status *project* dari "Aktif" menjadi "Tidak Aktif").Lalu terakhir admin dapat menghapus data *project* dari *database* secara permanen. Selain CRUD *view* ini juga terdapat *export to excel* untuk membagikan data dalam bentuk table excel.



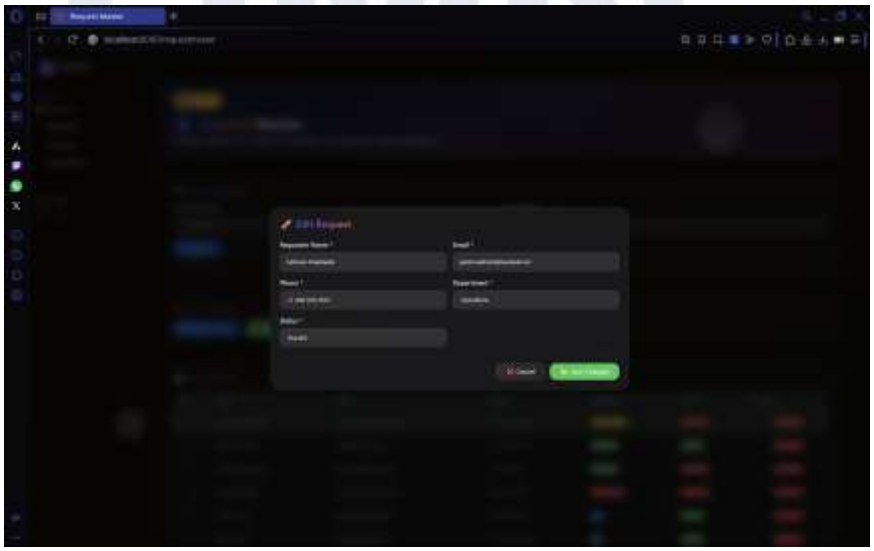
Gambar 3. 30 Request Master View

Seperti pada gambar 3.30, *view* ini adalah halaman administrasi inti yang berfungsi sebagai pusat data utama untuk semua *request*. Halaman ini bukanlah laporan analitik, tetapi *interface* operasional untuk manajemen data *request*. Dalam *view* ini adalah untuk melakukan operasi CRUD (Create, Read, Update, Delete) secara penuh.



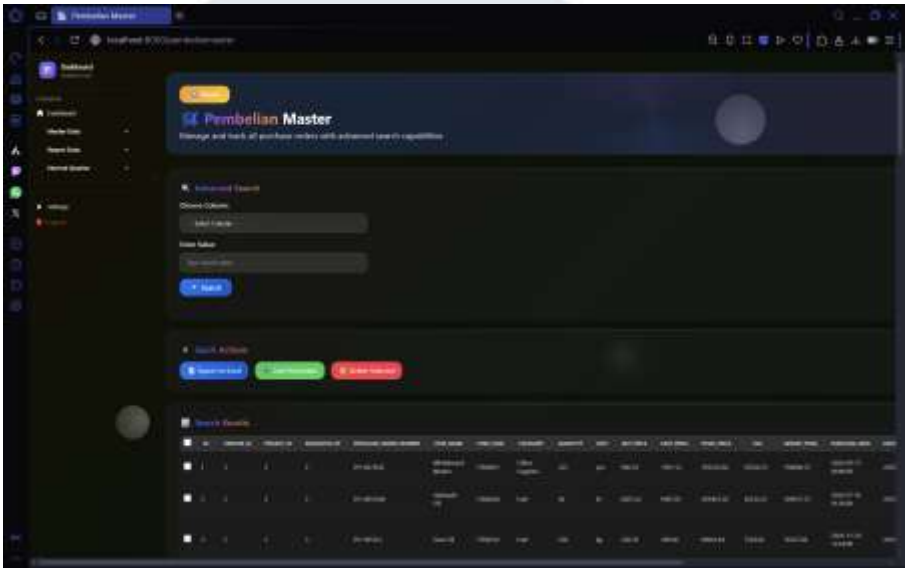
Gambar 3. 31 Create Request View

Pada gambar 3.31 administrator dapat menambahkan data *request* baru ke dalam sistem (misalnya, nama request, email, phone, department, dan status) melalui *form* "Tambah Request". Halaman ini menampilkan tabel data lengkap dari semua *request* yang ada di *database* (request table). Tabel ini interaktif, dilengkapi dengan fitur pencarian (search) dan pagination (pembagian halaman) untuk mengelola data dalam jumlah besar.



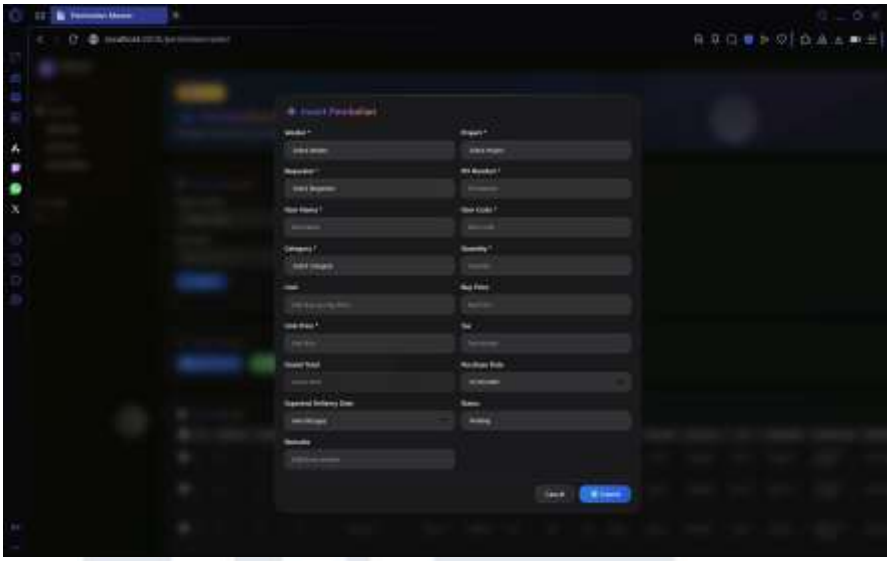
Gambar 3. 32 Request Master View

Seperti pada gambar 3.33, administrator dapat mengklik tombol "Edit" pada *request* yang ada untuk mengubah informasinya (misalnya, mengubah status request dari "Aktif" menjadi "Tidak Aktif").Lalu terakhir admin dapat menghapus data *project* dari *database* secara permanen. Selain CRUD *view* ini juga terdapat *export to excel* untuk membagikan data dalam bentuk table excel.



Gambar 3. 33 Purchase Master View

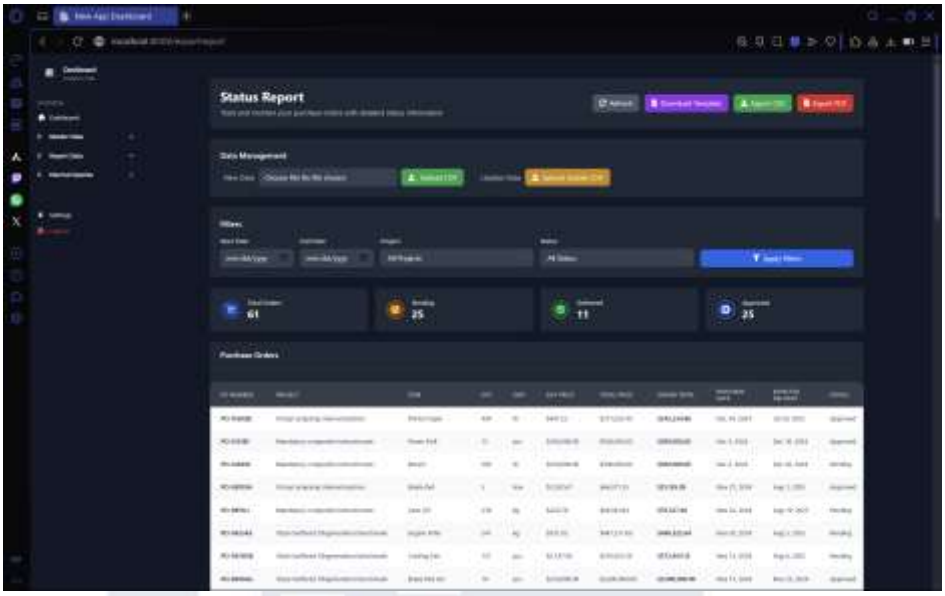
Seperti pada gambar 3.33, *view* ini adalah halaman administrasi inti yang berfungsi sebagai pusat data utama untuk semua pembelian. Halaman ini bukanlah laporan analitik, melainkan *interface* operasional untuk manajemen data pembelian Dalam *view* ini adalah untuk melakukan operasi CRUD (Create, Read, Update, Delete) secara penuh.



Gambar 3. 34 Create Purchase View

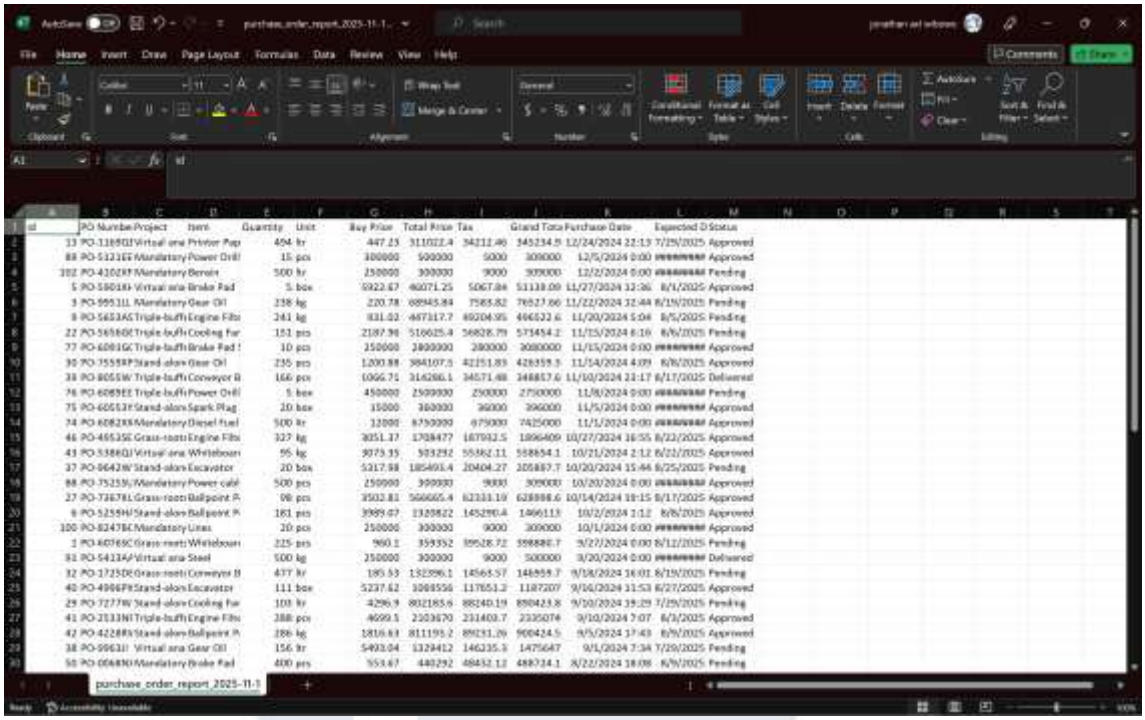
Pada gambar 3.34 administrator dapat menambahkan data pembelian baru ke dalam sistem (misalnya, vendor, project, requester, nomor PO, nama barang, item code, kategori, harga beli, harga jual, dan kuantitas, unit, buy price, unit price, tax, grand total, purchase date, expected delivery date, status dan remark) melalui form "Tambah Pembelian". Halaman ini menampilkan tabel data lengkap dari semua transaksi pembelian yang ada di database (pembelian table). Tabel ini interaktif, dilengkapi dengan fitur pencarian (search) dan pagination (pembagian halaman) untuk mengelola data dalam jumlah besar. Terakhir admin dapat menghapus data pembelian dari *database* secara permanen. Selain CRUD *view* ini juga terdapat *export to excel* untuk membagikan data dalam bentuk table excel.

Tahap selanjutnya membuat fitur *export* dan *import* data menggunakan file csv, lalu membuat fitur agar bisa *export* dengan output table excel / pdf seperti pada gambar 3.35.



Gambar 3. 35 Status Report View

View memuat tabel detail dari semua transaksi pembelian, memungkinkan administrator untuk melihat status pesanan (Pending, Delivered, Approved) berdasarkan filter yang diterapkan. Pada *View* ini terdapat *function mass import*, memasukkan ribuan data pembelian baru ke sistem sekaligus. Selain *import* data tetapi juga memperbarui data yang sudah ada (misalnya, mengubah status banyak PO dari "Pending" menjadi "Delivered") dengan mengunggah file CSV baru. Lalu pada *view* ini terdapat tombol untuk menghasilkan laporan resmi yang dapat dibagikan atau diarsipkan. Dimana ada *export CSV*, menghasilkan *spreadsheet* yang berisi data yang sedang ditampilkan di tabel dan *export PDF* membuat dokumen PDF Formal dari laporan yang telah difilter seperti pada gambar 3.36.

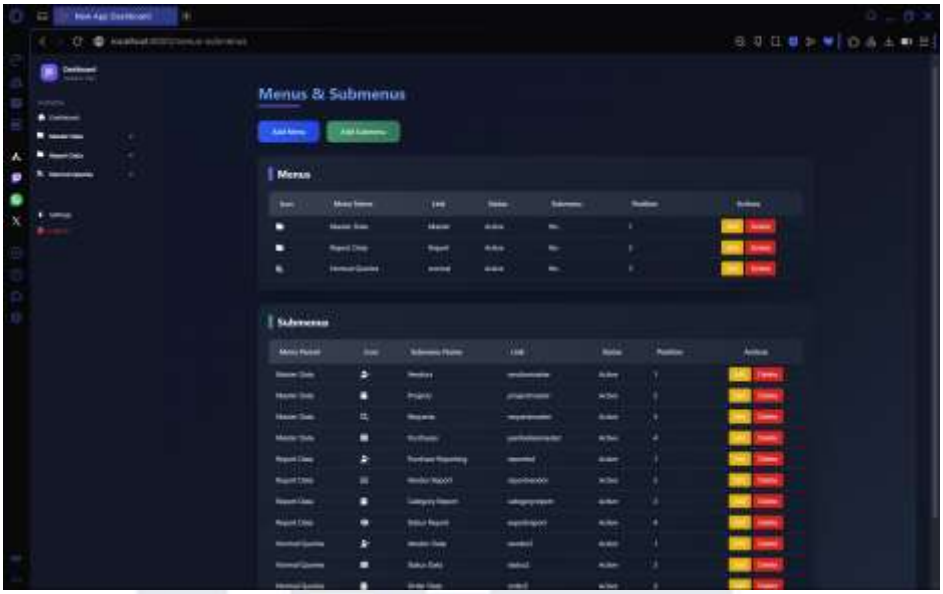


Gambar 3. 36 Export CSV

Tampilan frontend menyediakan filter berdasarkan Tanggal, Proyek, dan Status. Tombol "Apply Filters" yang kemudian memicu *function* JavaScript loadReport(), yang kemudian melakukan panggilan Ajax ($.ajax) ke endpoint /api/expected-delivery-date. Data yang dikembalikan kemudian langsung dimuat ke dalam tabel tanpa perlu memuat ulang seluruh halaman.

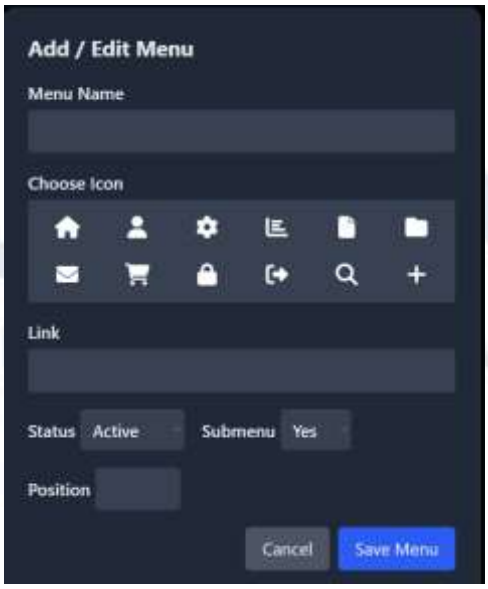
Gambar 3. 37 Export PDF

Tombol "Export PDF" berfungsi menjalankan function JavaScript yang mengambil semua nilai filter yang sedang digunakan, lalu mengarahkan browser ke route reports.purchase.pdf sambil menyertakan parameter filter tersebut di URL. Selanjutnya, controller function exportPurchaseReportPdf menerima parameter itu, mengambil data berdasarkan filter, lalu menampilkan hasilnya ke dalam view khusus PDF (reports.invoice-template) sebelum file PDF diunduh oleh pengguna seperti gambar 3.37. Terakhir dari *project* ini adalah Menus & submenus untuk membuat *navigation bar* yang fleksibel dimana *user* bisa hanya menambahkan tanpa memerlukan *hardcode.*



Gambar 3. 38 Menus & Submenus View

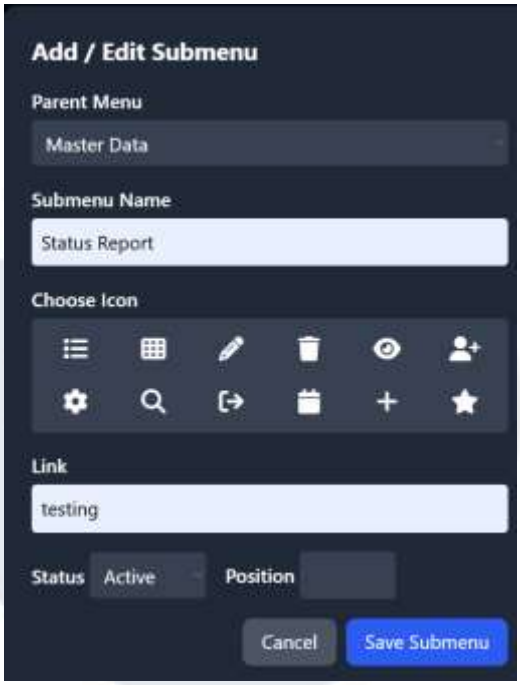
Pada gambar 3.38 *View* ini adalah pengaturan untuk mengatur *sidebar* yang ada pada project ini. Dimana administrator dapat menambahkan *dropdown* kategori baru ke *sidebar* (contoh: menambahkan kategori "Analytics" atau "Settings").



Gambar 3. 39 Add/Edit Menu

Seperti pada gambar 3.39, mengedit menu yang ada, seperti mengubah nama ("Reporting" menjadi "Laporan"), mengganti *icon*, atau mengatur ulang

urutan tampilannya misalnya, memindahkan "Master Data" ke atas "Reporting" dengan mengubah nilai position. Lalu juga dapat menghapus seluruh kategori menu dari *sidebar*.



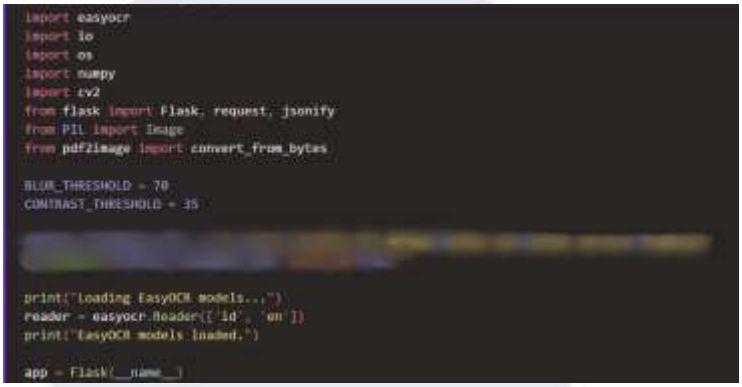
Gambar 3. 40 Add/Edit SubMenu

Seperti pada gambar 3.40, administrator dapat menambahkan tautan/halaman baru ke dalam sebuah *dropdown* menu utama misalnya, menambahkan halaman "Profit Report" ke dalam menu "Reporting". Selain itu *user* juga dapat melihat semua sub-menu yang ada di bawah setiap menu utama. Lalu sama seperti menu, submenu dapat diedit, seperti mengubah nama link, memperbarui URL (link) tujuannya, atau mengatur ulang posisinya di dalam *dropdown*.

3.3.1.3 OCR System Web Service

Pada proyek *OCR System Web Service* keperluan utama dari proyek ini adalah untuk memecahkan masalah *data entry* manual di perusahaan. PT. IMIP menerima banyak sekali dokumen fisik data diri yang di-scan seperti passport & KTP. Dimana di dalam format gambar atau PDF. Sistem yang dibangun ini berfungsi sebagai *backend* (layanan) terpisah yang dapat

menerima *input* file (gambar atau PDF), memprosesnya menggunakan *engine* Easy OCR, dan secara otomatis mengekstrak seluruh teks di dalamnya. Tujuannya adalah untuk mengotomatisasi proses *data entry*, mengurangi *human error*, dan membuat data dari dokumen-dokumen tersebut dapat dicari (*searchable*) dan diintegrasikan ke dalam sistem *dashboard* utama.



Gambar 3. 41 Import Libraries

Seperti pada gambar 3.41, pada bagian awal script ini, import semua library yang diperlukan. Telah dipilih engine EasyOCR karena library ini dikenal memiliki akurasi yang tinggi untuk berbagai bahasa, termasuk bahasa Indonesia. Lalu menggunakan OpenCV (cv2) untuk validasi kualitas gambar dan pdf2image untuk menangani file PDF. Baris reader = easyocr.Reader(['id', 'en']) adalah bagian krusial. Ini adalah perintah untuk memuat model machine learning EasyOCR ke dalam *memory server*. Secara spesifik untuk memuat model bahasa Indonesia ('id') dan Inggris ('en'), karena dokumen di perusahaan sering menggunakan kedua bahasa tersebut.

Gambar 3. 42 Image Clarity Validation Script

Pada gambar 3.42, setelah melakukan import library kemudian yang perlu dilakukan adalah *image clarity validation* seperti yang ada di gambar 3.38, tantangan terbesar OCR adalah gambar scan yang buram atau berkualitas rendah. Untuk itu, membuat fungsi validasi *validate_image_clarity* menggunakan library OpenCV (cv2). Tidak hanya memeriksa keburaman (blur), tapi juga *contrast*. Metode yang digunakan adalah *Laplacian Variance*, ini adalah teknik standar dalam *computer vision* untuk mengukur ketajaman gambar. Setelah beberapa kali uji coba, lalu *set threshold* (ambang batas) di angka 70. Gambar dengan skor di bawah ini akan ditolak. Lalu untuk meningkatkan akurasi diperlukan juga metrik baru menggunakan gray.std() (Standar Deviasi dari gambar *grayscale*). Ini adalah cara cepat untuk mengukur *contrast*. Dokumen yang di-*scan* terlalu terang atau 'pucat' akan memiliki skor *contrast* rendah. Kemudian di *set threshold* di 35.

```
#Data cleaning (penting biar json nya bisa di parse)
def make_json_serializable(data):
    if isinstance(data, list):
        return [make_json_serializable(item) for item in data]
    if isinstance(data, dict):
        return {key: make_json_serializable(value) for key, value in data.items()}
    if isinstance(data, numpy.int32):
        return int(data)
    if isinstance(data, numpy.ndarray):
        return data.tolist()
    return data
```

Gambar 3. 43 Data Cleaning Script

Pada proyek integrasi antara Python dan frontend berbasis Laravel ini, proses pertukaran data hasil pengolahan citra menghadapi tantangan pada tahap serialisasi JSON seperti yang ditunjukkan pada gambar 3.43. Hal ini disebabkan oleh keluaran dari library computer vision seperti NumPy dan EasyOCR yang menghasilkan tipe data non-standar, seperti numpy.int32 untuk nilai numerik dan numpy.ndarray untuk koordinat bounding box, yang tidak dapat langsung diubah ke format JSON. Untuk mengatasi permasalahan tersebut, seperti gambar 3.39, membuat sebuah *function* bernama *make_json_serializable*. *Function* ini berperan sebagai helper yang melakukan proses konversi secara rekursif terhadap seluruh struktur data hasil OCR. Melalui fungsi ini, tipe data khusus dari NumPy diubah menjadi tipe data Python standar (contohnya, numpy.int32 menjadi int, dan numpy.ndarray menjadi list). Dengan penerapan *function* ini, seluruh output dari service Python dapat dihasilkan dalam format JSON yang sepenuhnya valid, sehingga dapat dibaca dan di-parse oleh aplikasi Laravel tanpa menimbulkan kesalahan pada proses integrasi data.

Gambar 3. 44 PDF Processing Script

Kemudian tahap selanjutnya adalah membuat *function ocr_request* yang berperan sebagai endpoint utama dengan *route* /ocr, yang berfungsi sebagai jembatan antara aplikasi Laravel dan python service. Lalu mengoptimalkan fungsi ini agar dapat memproses file secara langsung dari memori menggunakan file_bytes = file.read(), tanpa perlu menyimpan file fisik ke disk terlebih dahulu. Pendekatan ini diterapkan untuk meningkatkan efisiensi dan mempercepat waktu respons API. Secara garis besar alur kerja *function* ini terdiri dari beberapa tahapan seperti pada gambar 3.44.

1. **Validasi Kualitas Gambar**, Apabila file yang diterima berupa gambar (bukan PDF), *system* akan langsung menjalankan *function* validate_image_clarity() untuk memeriksa tingkat ketajaman dan *contrast* gambar. Jika hasil validasi menunjukkan bahwa gambar buram atau memiliki *contrast* rendah, service akan mengembalikan pesan kesalahan kepada aplikasi Laravel sebagai bentuk early validation.
2. **Penanganan PDF,** Jika *file*-nya PDF, akan terjadi dua hal:
    - Membuat *Thumbnail***,** menggunakan library pdf2image untuk mengekstrak halaman pertama (first_page=1), menyimpannya sebagai *file* JPEG di *folder thumbnail* terpusat, dan menyimpan *path thumbnail* tersebut untuk dikirim kembali ke Laravel. Ini adalah fitur R&D agar aplikasi Laravel bisa menampilkan pratinjau dokumen.
    - Setelah itu mengonversi semua halaman PDF menjadi gambar di dalam *memory*, lalu melakukan *looping* untuk memproses setiap halaman dengan reader.readtext()

Gambar 3. 45 Image Processing Script

Pada gambar 3.45 apabila file yang diterima berupa gambar, proses yang dilakukan bersifat lebih sederhana. Sistem akan langsung mengirimkan *binary data* ke *function* reader.readtext() untuk dilakukan proses Optical Character Recognition (OCR) tanpa tahap konversi tambahan.

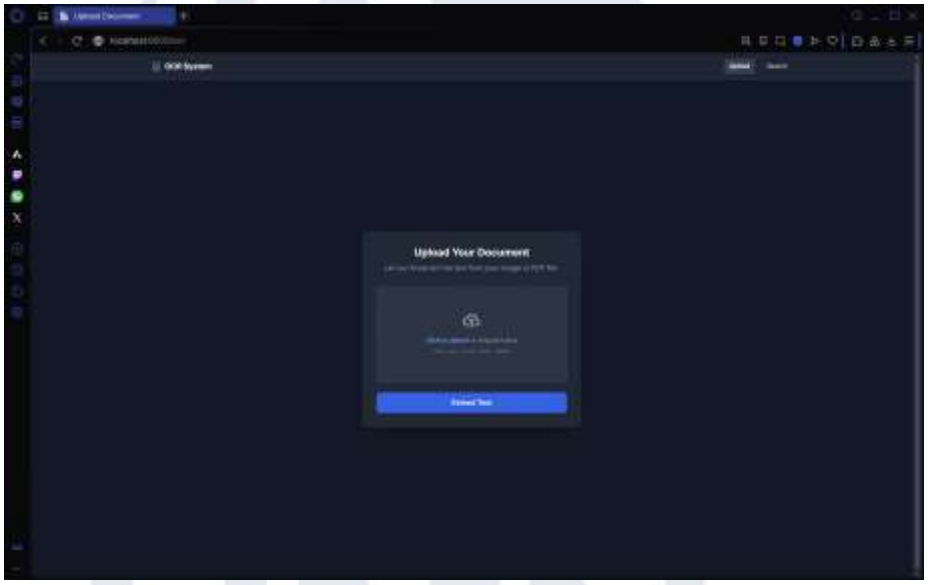
```
return jsonify({
    'text': full_text,
    'word_data': cleaned_word_data,
    'thumbnail_path': thumbnail_path
})
```

Gambar 3. 46 Output Script

Setelah data gambar sudah dibaca oleh EasyOCR , lalu tahap selanjutnya adalah membuat output dari hasil OCR, dimana text adalah hasil gabungan dari seluruh teks yang berhasil diekstraksi oleh EasyOCR. Seperti pada gambar 3.46 setiap potongan teks dipisahkan oleh karakter baris baru (\n) untuk memudahkan proses pencarian teks (*simple text search*) di dalam dashboard aplikasi Laravel. Lalu ada juga word_data dimana EasyOCR menghasilkan keluaran dengan tingkat detail tinggi (*detail=1*), mencakup teks serta koordinat bounding box (bbox) yang menunjukkan posisi teks pada gambar. Terakhir ada thumbnail_path dimana output ini adalah menyimpan *path* menuju file

thumbnail yang dihasilkan apabila berkas input berupa PDF. Data ini digunakan oleh frontend Laravel untuk menampilkan *thumbnail* dokumen secara visual di dashboard.

Setelah menyelesaikan bagian *script* python, mulai masuk kedalam pembuatan tampilan sederhana untuk testing apakah *script* yang sudah digunakan dapat diterapkan. Framework yang digunakan untuk website ini adalah laravel dan menggunakan MYSQL sebagai database nya.

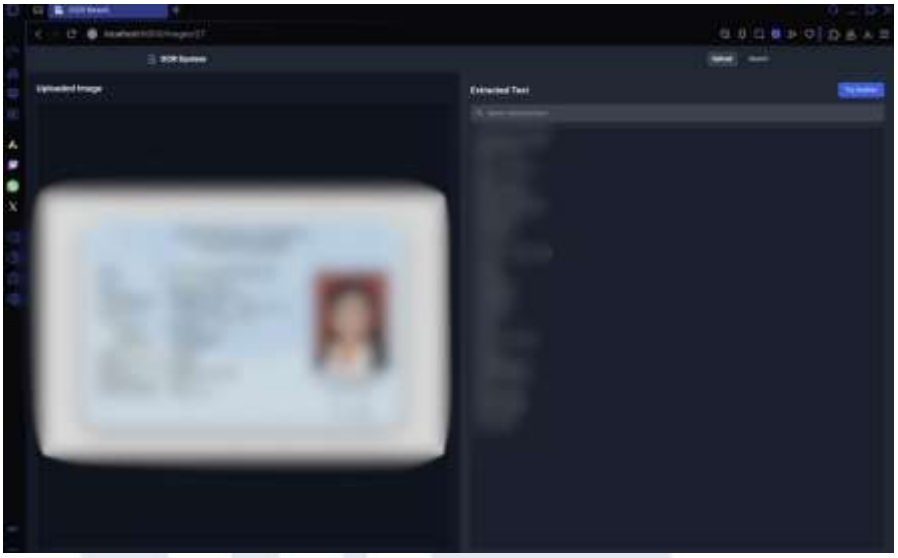


Gambar 3. 47 Input Photo View

Seperti pada gambar 3.47 tampilan *blade view* ini adalah tampilan pertama yang dilihat *user* pada saat menggunakan fitur OCR. Fungsi utama dari halaman ini sangat spesifik untuk *download* dokumen (PDF atau gambar) yang ingin mereka *scan*. *View* ini akan menampilkan sebuah *card box* untuk upload document seperti gambar / PDF. Lalu Ketika ada *user* yang upload maka akan dilempar ke controller (backend).

Gambar 3. 48 OCR Controller View

File OcrController.php pada gambar 3.48 merupakan komponen inti yang menangani seluruh logika backend pada sistem OCR. Controller ini bertanggung jawab untuk mengelola proses unggah file, menghubungkan aplikasi Laravel dengan service Python (OCR Engine), menyimpan hasil ekstraksi ke database, serta menyediakan fitur pencarian dan pratinjau hasil. Setelah *user* sudah *upload* file yang penting, terdapa fungsi bernama showUploadForm() yang bertanggung jawab untuk menampilkan *upload page* kepada pengguna. Laravel akan merender tampilan Blade bernama *upload-file.blade.php*, yang berisi form untuk mengunggah gambar atau PDF ke sistem OCR. Fungsi ini merupakan titik masuk awal dari interaksi pengguna dengan sistem.

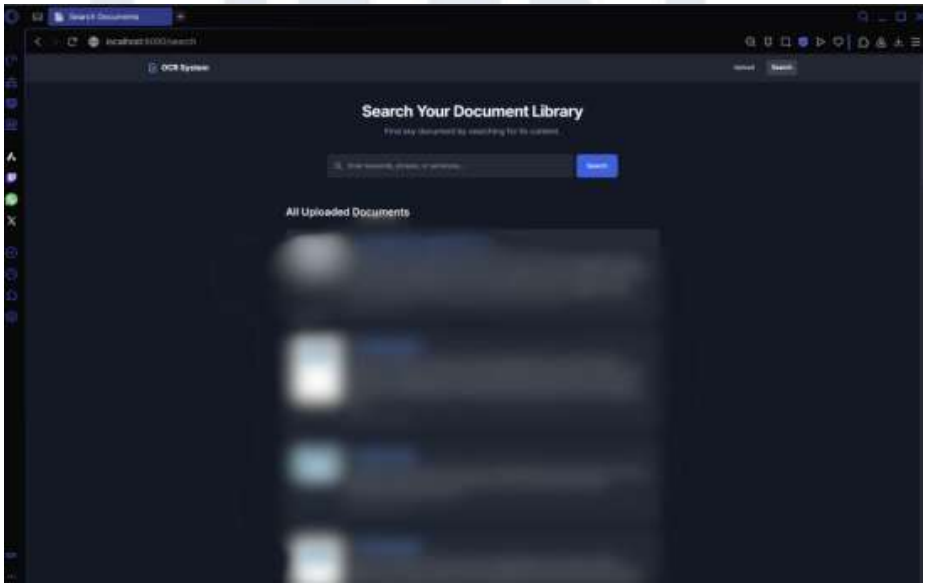


Gambar 3. 49 Image Viewers View

Tampilan pada gambar 3.49 ini adalah tampilan dari *route* /image/{filename} berfungsi bukan sebagai halaman yang diakses langsung oleh pengguna, melainkan sebagai endpoint backend yang berperan penting dalam menampilkan gambar pratinjau (thumbnail) secara aman. Tujuan utama dari halaman ini adalah untuk memberikan konfirmasi visual langsung kepada pengguna bahwa proses OCR telah berhasil. Halaman ini menyajikan perbandingan *side-by-side* antara gambar asli dan teks yang berhasil diekstraksi. Halaman ini akan muncul ketika *user* berhasil upload dokumen penting mereka. Selain itu halaman ini juga bisa diakses melalui *search view* ketika *user* mau lihat lebih *detail* dokumen atau foto yang diupload.

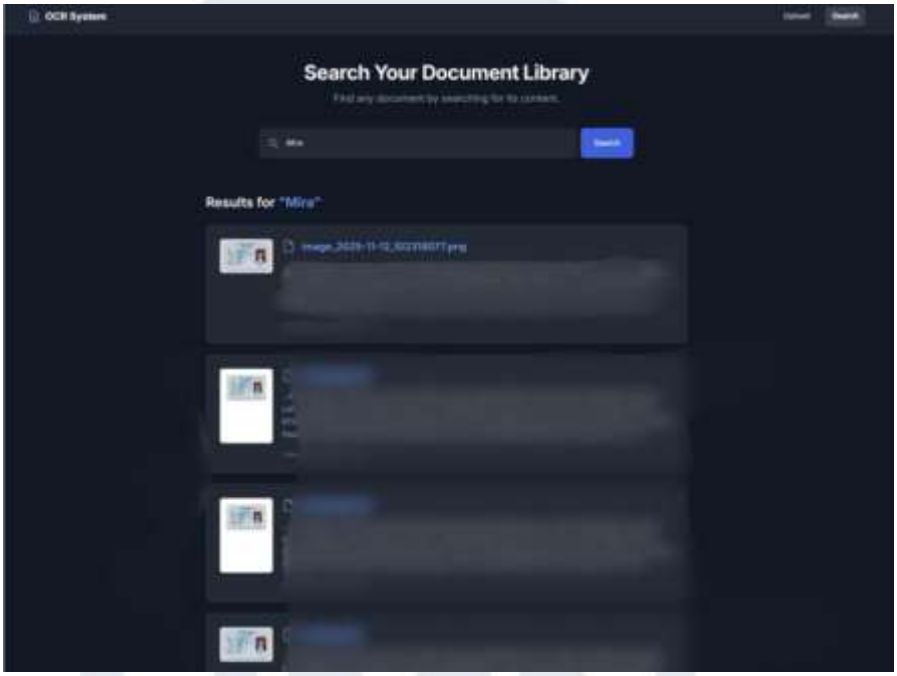
Gambar 3. 50 findOrFail Function Controller

Seperti gambar 3.50 ini adalah *function* yang digunakan untuk membuka detail *document.* Dengan memanfaatkan fitur Eloquent ORM Laravel untuk mengambil data dari database secara efisien dan aman. Fungsi findOrFail($id) secara otomatis mencari record dengan nilai kolom id yang sesuai dengan parameter yang diberikan. Jika data dengan ID tersebut tidak ada atau tidak ditemukan, maka laravel akan langsung menampilkan halaman error 404 (Not Found) tanpa perlu menulis logika pengecekan manual.



Gambar 3. 51 Dashboard Search View

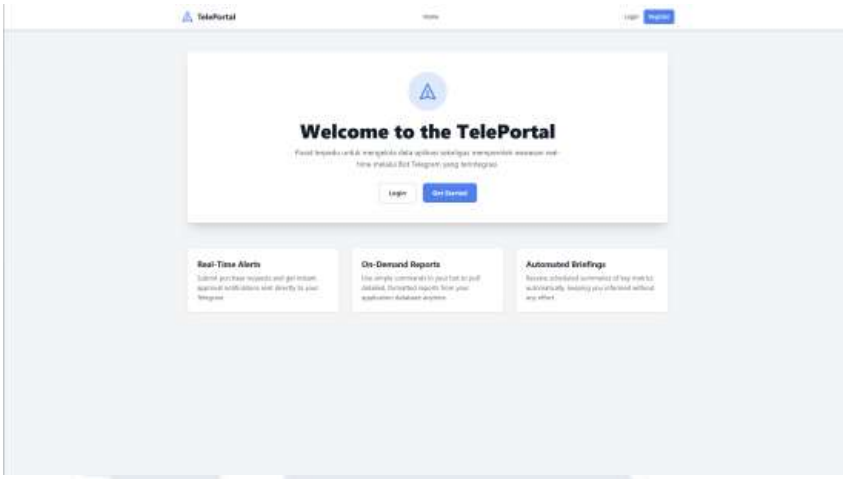
Halaman ini berperan sebagai *search engine* untuk seluruh dokumen yang telah diproses oleh OCR seperti gambar 3.51. Didalam *view* ini ada form pencarian dan disediakan kolom input bagi pengguna untuk mengetikkan kata kunci. Setelah itu sistem menampilkan daftar dokumen yang mengandung teks sesuai dengan kata kunci tersebut seperti yang ditunjukkan pada gambar 3.52.



Gambar 3. 52 Keyword Search View

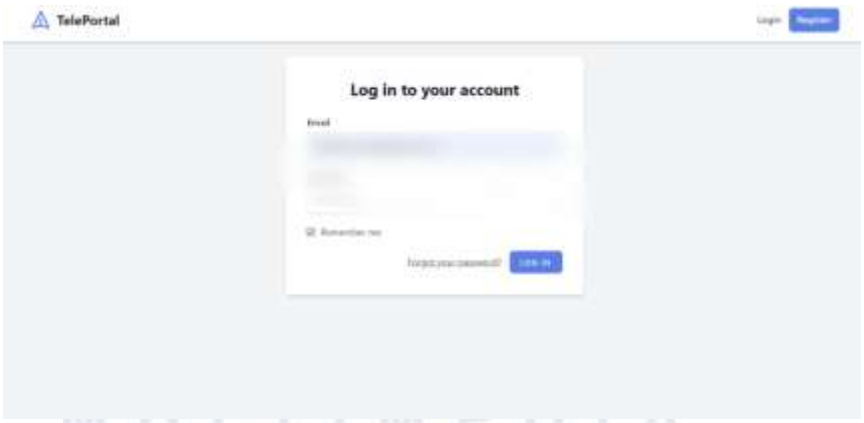
3.3.1.4 Automasi Persetujuan Pembelian via Chatbot Telegram

Project selanjutnya adalah membuat chatbot telegram yang digunakan untuk keperluan *supploy chain*, dimana bot ini akan diintegrasikan dengan sebuah web viewer (website) dimana dalam website ini pihak *user* dapat melakukan *request* pembelian / *request* lainnya dan notifikasi *request* akan muncul kedalam telegram chat manager. Dalam *project* ini terdapat dua pengerjaan yaitu pengerjaan *script* python chatbot untuk menyimpan logika chatbot, dan webservice menggunakan *framework* Laravel.



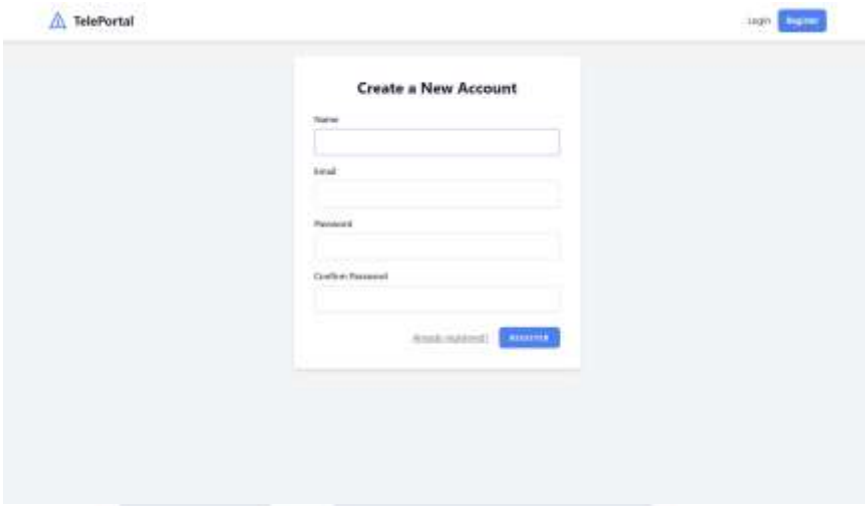
Gambar 3. 53 Home View

Seperti yang kita lihat pada gambar 3.53, ini adalah tampilan utama dari webservicenya, dimana terdapat navbar untuk registrasi dan penjelasan kecil berupa *card* untuk pengenalan dan informasi singkat mengenai apa yang bisa digunakan pada webservice ini. Nama dari *project* ini adalah Teleportal. Tampilan ini akan muncul ketika *user* mau masuk kedalam website sebelum registrasi.



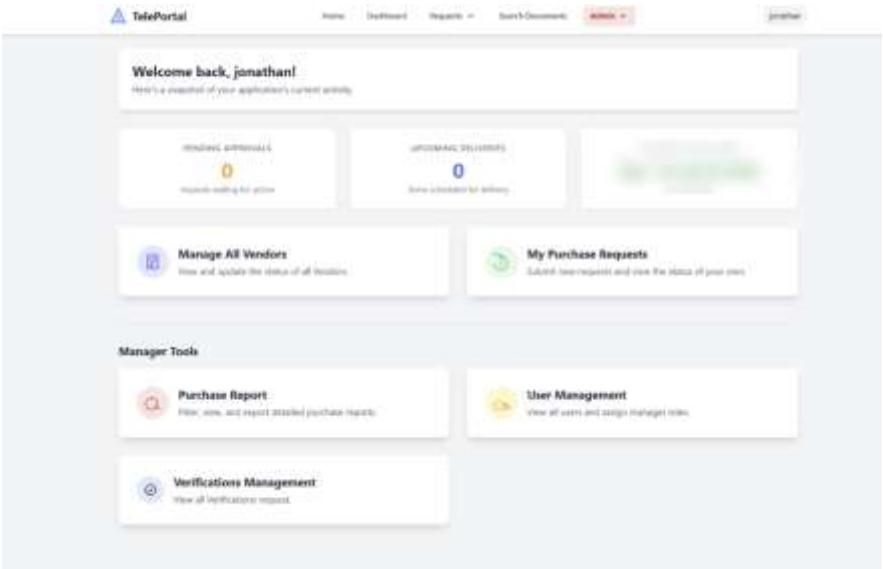
Gambar 3. 54 Log in View

Setelah tampilan home page tadi, sekarang tampilan login ini akan muncul ketika *user* mau masuk kedalam webservice ini seperti gambar 3.54. Memberikan email sebagai pengganti *username* dan password. Terdapat opsi remember me agar *token user* dapat tersimpan saat sudah *login.*



Gambar 3. 55 Register Gambar View

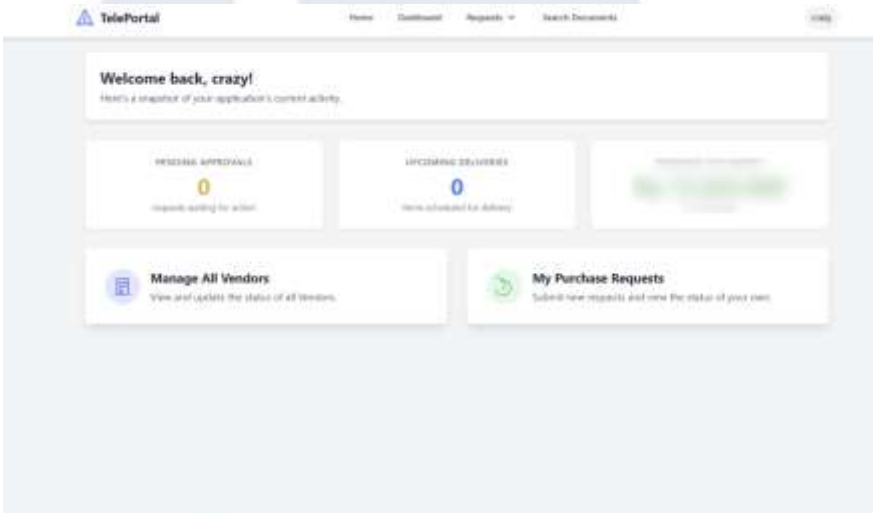
Jika *user* belum memiliki akun, maka bisa registrasi terlebih dahulu seperti pada gambar 3.55, ini adalah tampilan registrasi dimana *user* diminta untuk memberikan nama, email, password, dan konfirmasi password untuk membuat akun pada webservice ini. Masing-masing akun yang teregistrasi akan mendapat role *user* secara otomatis, dan hanya bisa diubah dengan *user* dengan role *manager*.



Gambar 3. 56 Dashboard View

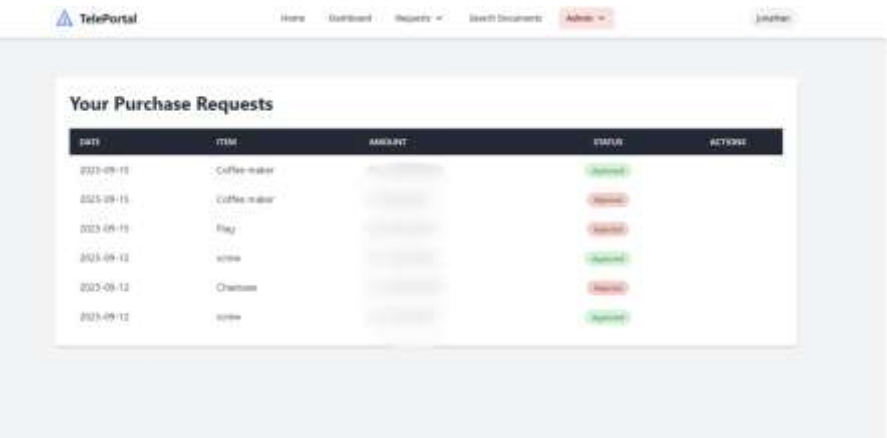
Tampilan pada gambar 3.56 adalah tampilan *dashboard* pada saat masuk kedalam webservice. Pada *dashboard* ini terdapat *navbar* yang

menunjukkan semua fitur yang ada pada webservice ini, tampilan *dashboard* ini adalah tampilan *user* dengan role *manager*. Pada role ini, *dashboard* nya berisi informasi seperti *pending approval,* yang berfungsi sebagai informasi tambahan bahwa ada *approval* yang belum dilihat, ada *upcoming deliveries*, dan *spending month* dimana semua card ini disetting untuk melihat informasi selama 1 bulan ini. Lalu sebagai role *manager* dapat melihat tampilan *manager tools* yang berisi shortcut untuk langsung kefitur manager.



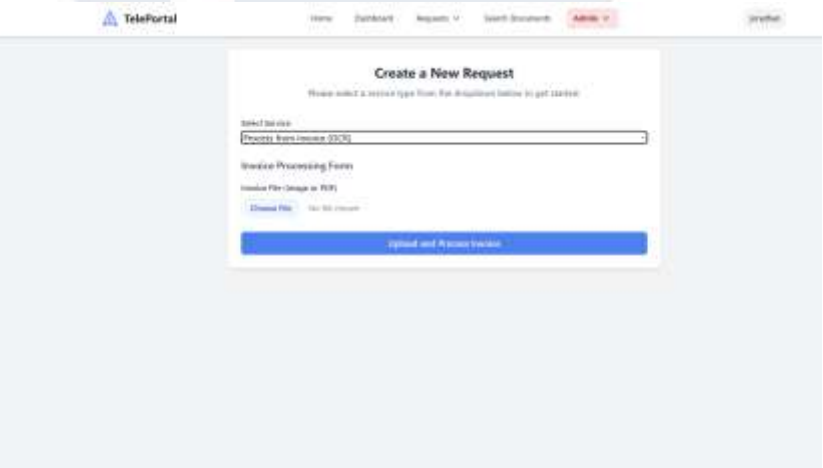
Gambar 3. 57 User Dashbaord View

Ini adalah tampilan dari role *user* dimana *manager tools* tidak tampil dan navbar tidak menunjukkan *admin* pada navbar. Seperti pada gambar 3.57 *user* hanya ditampilkan 3 card diatas dan juga *shortcut* untuk *manage vendor* dan *purchase request* yang memang kegunaan *webservice* ini adalah untuk itu.



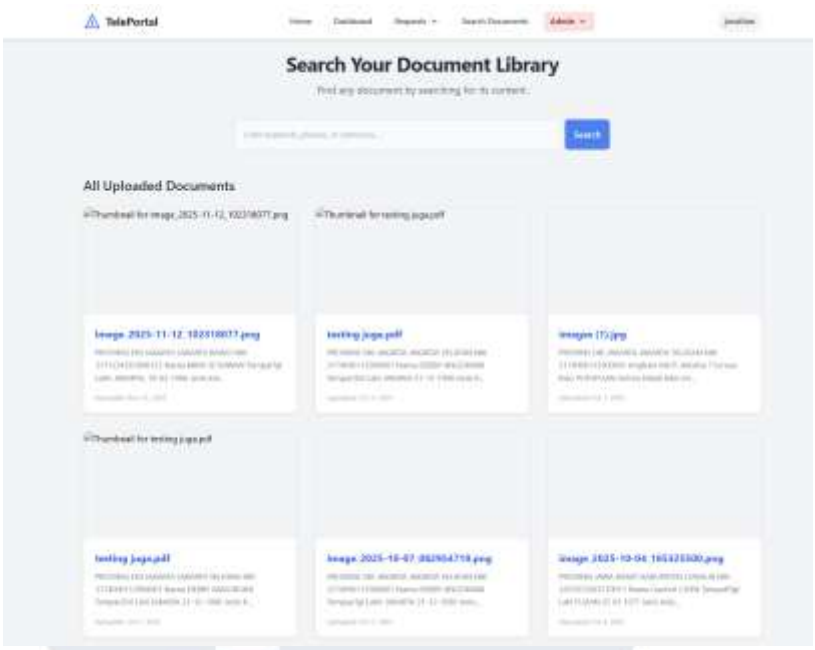
Gambar 3. 58 Purchase Request View

Tampilan pada gambar 3.58 adalah *view purchase request* dimana *user* dapat melihat barang-barang apa saja yang mereka *request* dengan harga berapa dan apakah status nya itu di *approve* atau di *reject*.



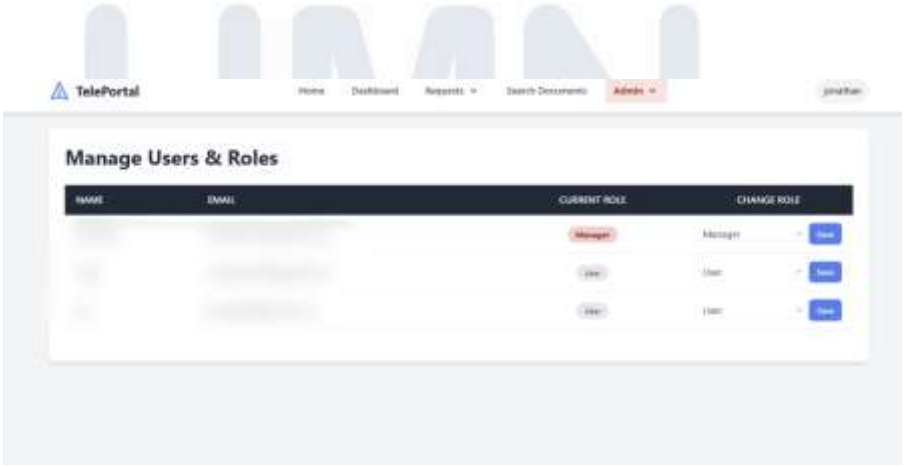
Gambar 3.58 OCR Service View

Tampilan berikutnya yang bisa dilihat pada gambar 3.58 adalah tampilan *request* OCR, dimana *project OCR* sebelumnya dimasukkan kedalam webservice ini kemudian menintegrasikan agar setiap ada inputan gambar / PDF akan terlebih dahulu disetujuin terlebih dahulu.



Gambar 3. 59 OCR Library View

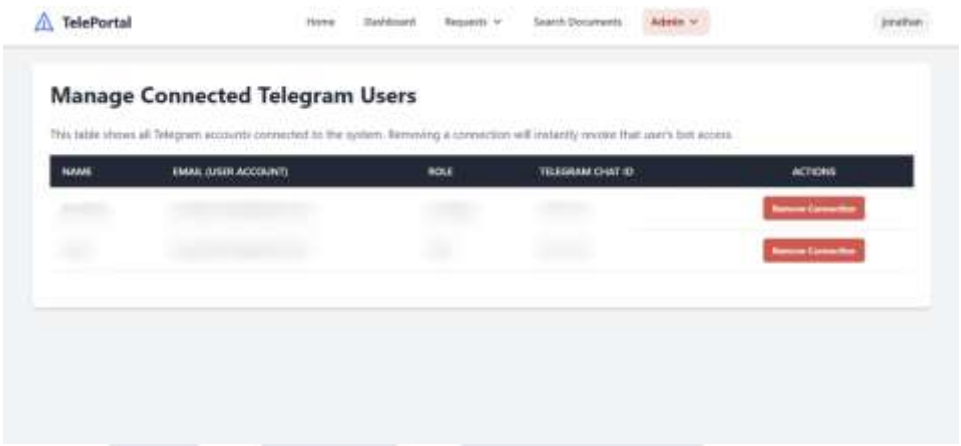
Tampilan berikutnya yang bisa dilihat pada gambar 3.59 adalah tampilan *library* dari OCR, dimana sekarang semua gambar yang sudah tersimpan /disetujui akan masuk kedalam *library* ini. Terdapat fitur search berdasarkan keyword dari isi *image* yang diinput.



Gambar 3. 60 Manage Users & Roles View

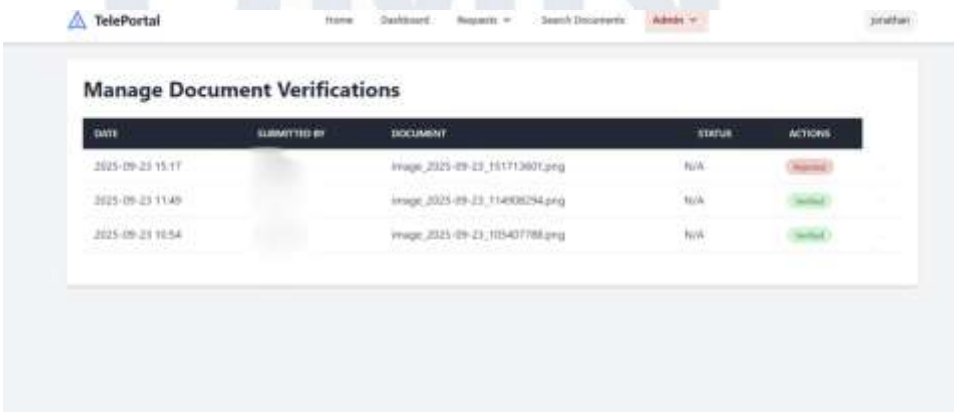
Sekarang tampilan 3.60 adalah isi dari *manager control panel*, dimana disini adalah tampilan untuk *manage* semua *user* yang di register pada webservice ini. Seorang *manager* dapat mengatur role-role dari

semua akun yang ada, bisa diubah menjadi *manager* juga atau menjadi *user*.



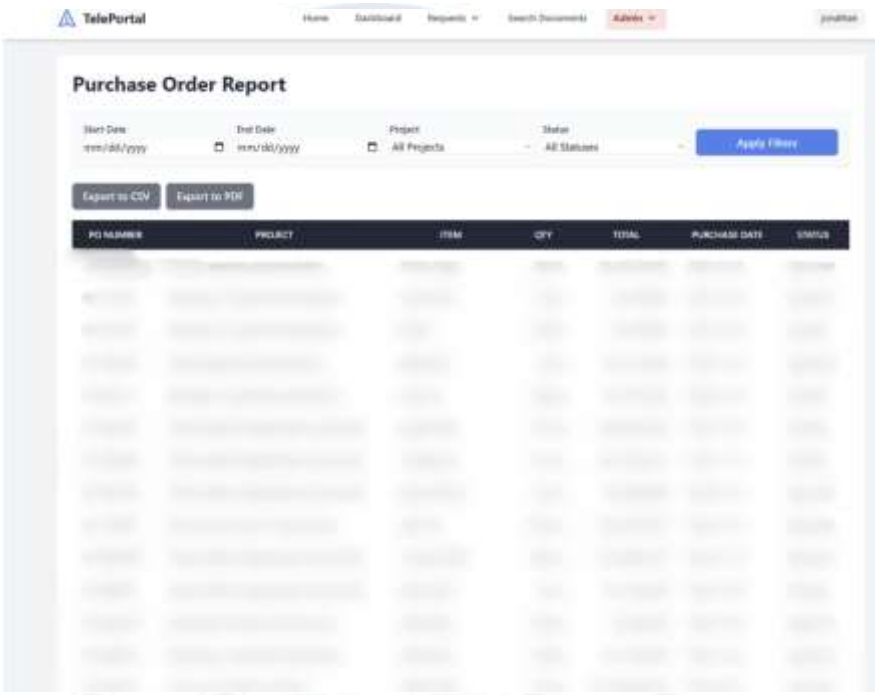
Gambar 3. 61 Manage Telegram View

Pada gambar 3.61 ini adalah fitur untuk menghapus *user* telegram yang tersambung pada webservice ini. Dimana semua *user* yang memakai webservice ini dan mau menggunakan telegram bot nya harus mendaftarkan terlebih dahulu pada telegram. Lalu semua yang sudah tersambung akan terlihat dari webservice *chat id* apa saja yang sudah tersambung dan seorang *manager* dapat menghapus koneksinya jika memang *user* tersebut sudah tidak memiliki akses bot atau sudah bukan lagi bagian dari lembaga perusahaan.



Gambar 3. 62 Manage Document View

Gambar 3.62 adalah tampilan dimana *manager* dapat melihat *history* atau *record* dari semua gambar yang masuk atau minta diterima. Tampilannya menunjukkan apakah direject atau diterima. Semua notifikasi ini akan masuk kedalam *chat id* telegram *manager* , setelah *manager* selesai *approval*, maka catatanya akan tersimpan pada *view* ini.



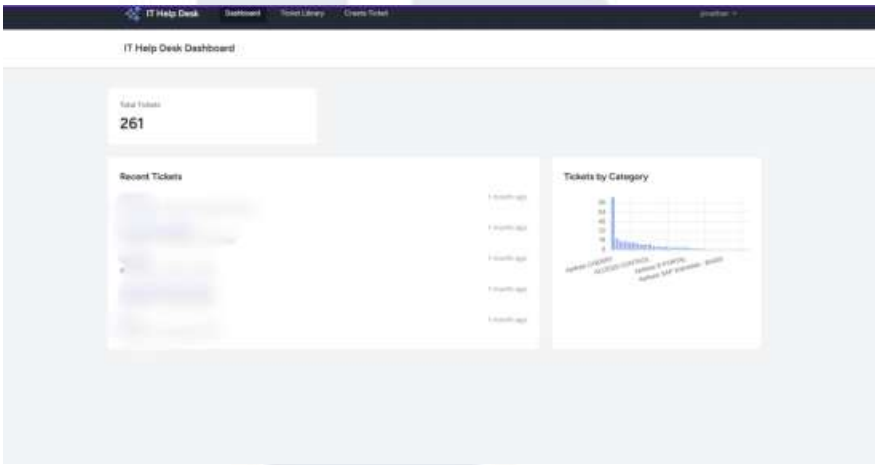
Gambar 3. 63 Purchase Order View

Pada gambar 3.63, ini adalah *purchase report* yang ada, dimana *manager* dapat melihat dan melakukan *filtering* sesuai dengan tanggal, *project* apa, dan status. Lalu *report* ini juga dapat diexport kedalam bentuk CSV atau bentuk PDF.

3.3.1.5 LLM Ticketing Sentence Similarity Microservice

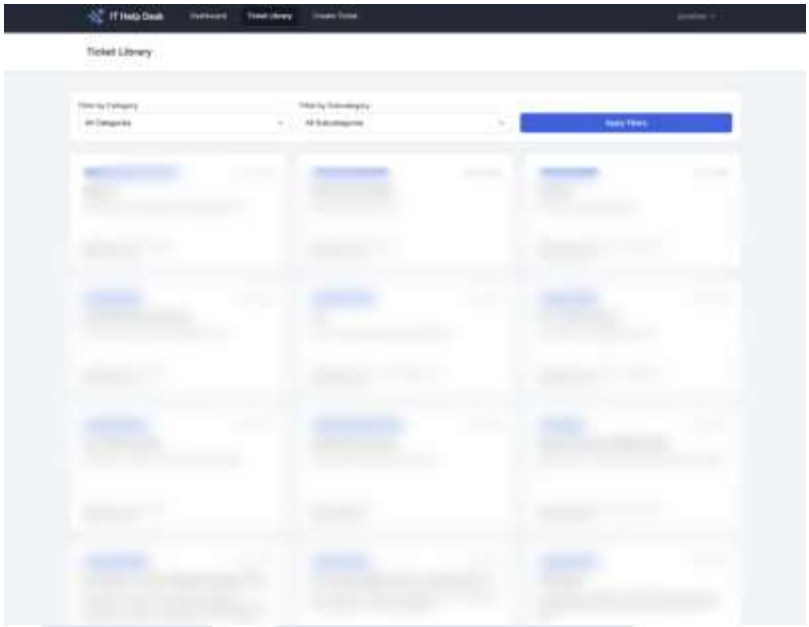
Project selanjutnya adalah membuat sebuah API AI yang digunakan untuk keperluan IT Helpdesk, dimana API ini dirancang untuk diintegrasikan dengan sebuah website portal ticketing internal. Dalam website ini, pihak *user* / karyawan IMIP dapat membuat tiket baru, dan notifikasi/rekomendasi tiket-tiket lama yang serupa akan muncul secara otomatis di dashboard mereka.

Dalam project ini, terdapat dua pengerjaan utama, yaitu pengerjaan script Python training_model.py untuk menyimpan logika fine-tuning model SentenceTransformer (menggunakan basis paraphrase-multilingual-mpnet-base-v2 dan data latih training_data.csv), dan sebuah webservice (API) menggunakan framework Flask (app.py) yang memuat model kustom (./it_helpdesk_model) untuk menyajikan hasil skor kemiripan.



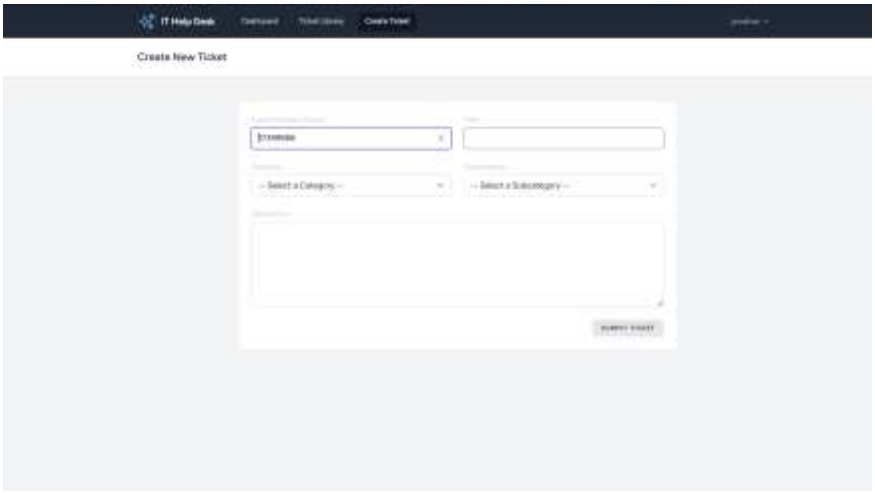
Gambar 3. 64 Dashboard LLM View

Pada tampilan *dashboard* akan dimunculkan ticket terakhir yang telah dibuat dan juga statistika simpel untuk melihat kira-kira dari banyaknya *ticket* yang dibuat oleh karyawan, aplikasi apa yang paling banyak menerima keluhan. Dari data sampel yang ada terdapat 261 tiket yang sudah dibuat.



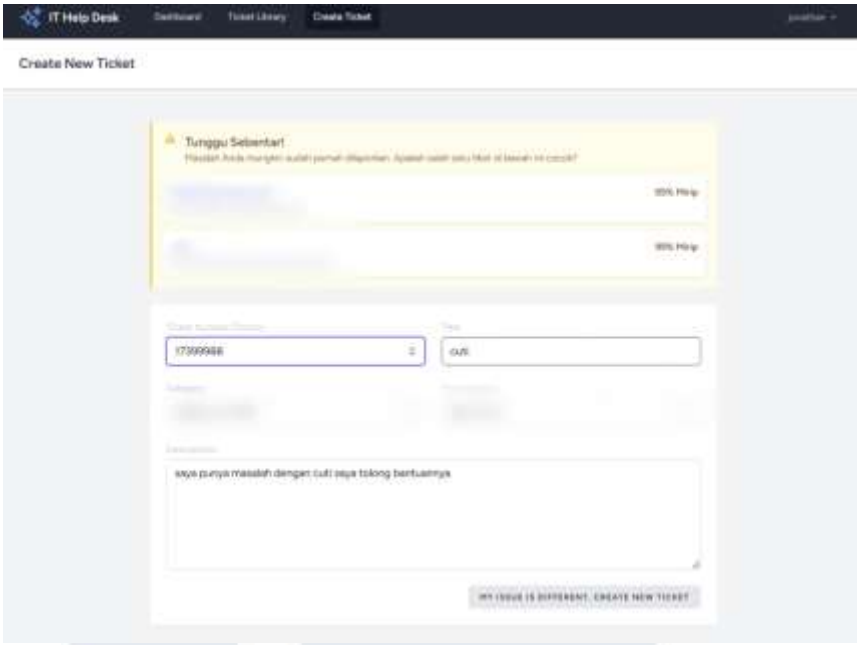
Gambar 3. 65 Ticket Library View

Tampilan pada gambar 3.64 adalah *library* sederhana dari semua tiket yang sudah dibuat. Library ini memiliki fitur *filtering* untuk bisa melihat tiket-tiket dari tiap aplikasi.



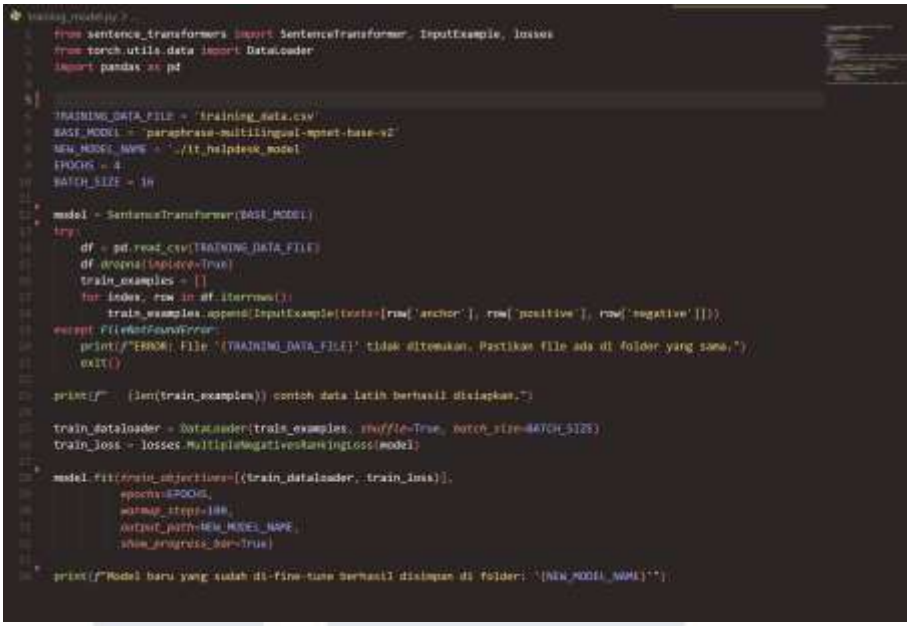
Gambar 3. 66 Ticket Creating View

Gambar 3.66 adalah tampilan pada saat *user* ingin membuat tiket, dimana *user* diminta mengisi judul lalu nama aplikasi nya dan juga category dari apa keluhan yang mau disampaikan.



Gambar 3. 67 Ticket LLM Result View

Setelah *user* sudah memberikan keluhannya, maka sebelum keluhannya diinput oleh sistem, LLM ini akan bekerja untuk mencari apakah ada kemiripan dengan *ticket* yang sudah pernah dibuat, dengan memberikan *scoring* seberapa mirip keluhan yang mau dibuat dengan keluhan yang sudah ada. Seperti pada gambar 3.67 bisa dilihat ada *ticket* lama yang memiliki kemiripan sebesar 99% dengan keluhan yang baru dibuat.



```
from sentence_transformers import SentenceTransformer, InputExample, losses
from torch.utils.data import DataLoader
import pandas as pd

TRAINING_DATA_FILE = 'training_data.csv'
BASE_MODEL = 'paraphrase-multilingual-mpnet-base-v2'
NEW_MODEL_NAME = './it_helpdesk_model'
EPOCHS = 4
BATCH_SIZE = 16

model = SentenceTransformer(BASE_MODEL)
try:
    df = pd.read_csv(TRAINING_DATA_FILE)
    df.dropna(inplace=True)
    train_examples = []
    for index, row in df.iterrows():
        train_examples.append(InputExample(texts=[row['anchor'], row['positive'], row['negative']]))
except FileNotFoundError:
    print(f"ERROR: File '{TRAINING_DATA_FILE}' tidak ditemukan. Pastikan file ada di folder yang sama.")
    exit()

print(f"   {len(train_examples)} contoh data latih berhasil disiapkan.")

train_dataloader = DataLoader(train_examples, shuffle=True, batch_size=BATCH_SIZE)
train_loss = losses.MultipleNegativesRankingLoss(model)

model.fit(train_objectives=[(train_dataloader, train_loss)],
          epochs=EPOCHS,
          warmup_steps=100,
          output_path=NEW_MODEL_NAME,
          show_progress_bar=True)

print(f"Model baru yang sudah di-fine-tune berhasil disimpan di folder: '{NEW_MODEL_NAME}'")
```

Gambar 3. 68 training_mode.py

Secara teknis seperti pada gambar 3.68, sebelum sistem dapat digunakan oleh *user*, ada proses persiapan yang dijalankan oleh *script* training_model.py. *script* ini bertanggung jawab untuk "mengajari" AI. Ia mengambil model *embedding* standar (paraphrase-multilingual-mpnet-base-v2), lalu melakukan *fine-tuning* menggunakan data latih training_data.csv. Data latih ini berisi tiga kolom (anchor, positive, negative) yang mengajari model membedakan mana kalimat yang mirip dan tidak mirip dalam konteks IT Helpdesk. Proses *fine-tuning* ini menggunakan losses.MultipleNegativesRankingLoss dan hasilnya disimpan sebagai *custom* model yang disimpan di folder ./it_helpdesk_model.

Gambar 3. 69 App.py

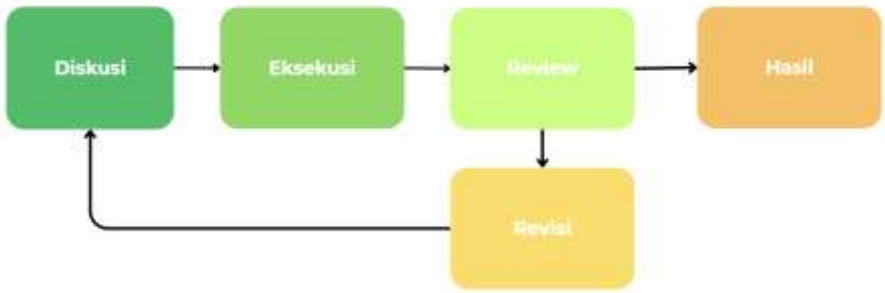
Setelah *custom* model itu siap, barulah API app.py dapat dijalankan. Seperti pada gambar 3.69, *Script* Flask ini pertama-tama memuat model kustom (./it_helpdesk_model) ke dalam memori saat server dimulai. Ketika *user* memberikan keluhan baru melalui *website*, *website* tersebut akan mengirimkan permintaan POST ke *endpoint* /find_similar pada API ini. Permintaan itu membawa data JSON berisi new_title, new_description, dan daftar old_tickets. *Script* app.py kemudian menggabungkan judul dan deskripsi baru, lalu menggunakan fungsi model.encode() untuk mengubah teks keluhan baru dan semua teks tiket lama menjadi vektor numerik (*embeddings*). Inti dari pekerjaan LLM ini ada di baris util.cos_sim(), di mana ia secara matematis menghitung skor *cosine similarity* antara vektor keluhan baru dengan semua vektor tiket lama.

Setelah LLM selesai bekerja dan memberikan *scoring*, *script* app.py tidak langsung menampilkan semua hasil. Ia menerapkan logika bisnis untuk memfilter tiket mana yang paling relevan. Pertama, ia hanya akan mengambil tiket-tiket yang skornya di atas RELEVANT_THRESHOLD (90%). Jika ada tiket lama yang memiliki kemiripan sebesar 99%, tiket itu

akan lolos filter pertama. Kemudian, *script* menerapkan filter kedua mengecek apakah ada tiket yang skornya di atas 98% (high_confidence_results). Jika tiket 99% itu ditemukan, API akan hanya menampilkan tiket-tiket dengan *confidence level* sangat tinggi (>98%) tersebut. Jika tidak ada yang di atas 98%, barulah API akan menampilkan semua tiket yang lolos 90%. Respon JSON akhir (similar_tickets) kemudian dikirim kembali ke *website* untuk ditampilkan kepada *user*.

3.3.1.6 Designing and Adjusting IMIP Apps

Pada *project* ini, perusahaan memiliki aplikasi mobile internal yang biasa digunakan oleh para karyawan. Didalam aplikasi ini terdapat banyak sekali informasi-informasi seputar perusahaan dan memiliki fitur *approval* dan juga permasalahan absensi dan nyambung dengan *cherry apps*. Karena *design* yang sudah tergolong tua, aplikasi ini memerlukan tampilan baru dan menggunakan *UI* yang lebih modern.



Gambar 3. 70 Alur Pengerjaan

Alur pengerjaan dari membuat *UI* ini dikerjakan dengan menggunakan figma, sesuai dengan gambar 3.70, hal pertama yang dilakukan adalah melakukan diskusi. Alur pengerjaan ini berisi referensi yang digunakan, gaya *design* yang mau digunakan, fitur-fitur apa yang mau dimasukkan, bentuk dari setiap *layout.* Alur pengerjaan selanjutnya adalah eksekusi dimana semua yang sudah didiskusikan akan dikerjakan dan dibuat secara langsung. Setelah itu ada *review* dimana setelah hasilnya sudah ada direview dulu oleh *supervisor* untuk dilihat apakah sudah

sesuai dengan standard perusahaan dan keinginan dari direksi. Jika ada revisi maka langkah diskusi akan dilakukan lagi untuk mencari solusi terbaru. Jika sudah tidak ada revisi maka pengerjaan dianggap selesai.

Gambar 3. 71 Tabel Inspirasi

Seperti pada gambar 3.71 ini adalah gambar-gambar aplikasi yang dijadikan inspirasi atau referensi dari pembaruan *UI* IMIP APPS. Semua akan mengikuti gaya *clean look* dan juga menggunakan font yang modern dan icon-icon populer.

Gambar 3. 72 Home Page

Gambar 3.72 adalah *design* awal yang diajukan untuk *landing page* aplikasi IMIP APPS. Tampilan ini kemudian diminta untuk direvisi karena layout yang masih kurang tepat dan *design* yang masih tergolong berantakan dan tidak memenuhi standard perusahaan.

Gambar 3. 73 Design setelah Revisi

Setelah melakukan revisi dan diskusi yang panjang, pada gambar 3.73 adalah draft akhir dari *design* IMIP APPS, dimana seluruh aspek yang dibutuhkan sudah terpenuhi, serta *layout* yang sudah teratur dan memenuhi kriteria konsep *design clean look.*

Gambar 3. 74 Hero Section News

Seperti yang ditampilkan pada gambar 3.74 ini adalah tampilan *news* ketika *user* click berita-berita dari *carousel* yang ada pada *hero section*. Tampilan ini mengikuti referensi dari aplikasi-aplikasi berita lainnya yang lebih mengutamakan isi konten dan memberikan gambar header sederhana dan isi dari keseluruhan tampilan ada isi dari beritanya.

Gambar 3. 75 News Sections

Saat *user* menekan salah satu berita yang ada pada bagian *news section* maka seperti gambar pada 3.75, *user* akan diperlihatkan tampilan seperti itu. Memiliki bentuk *design* yang sama dengan *news* pada *carousel*, tetapi isi dari konten ini hanya gambar dan isi teks beritanya. Judul sudah bisa dilihat *user* pada saat menekan beritanya.

Gambar 3. 76 Video Player

Gambar 3.76 adalah tampilan *videoplayer* yang ada pada IMIP APPS, dimana perusahaan memiliki banyak sekali SOP yang harus dijalankan atau dipahami oleh semua karyawan, sehingga perusahaan mengeluarkan aturan-aturan ini dalam bentuk video. Tampilan ini dibuat untuk lebih mempermudah *user* unutk mengetahui isi dari video yang mau ditonton, serta adanya rekomendasi video yang identik dengan video yang sedang ditonton.

3.3.2 Kendala yang Ditemukan

Selama proses magang berjalan, ada beberapa kendala yang dialami selama magang di perusahaan Indonesia Morowali Industrial Park:

1. Pengetahuan di bidang *Machine Learning* (ML) *Natural Language Processing* (NLP), dan menjadi seorang *fullstack developer* yang masih terbatas. Latar belakang yang lebih terfokus pada pengembangan model sederhana saja dan ilmu dasar pengembangan website, sementara

sebagai *IT Application Developer* ini membutuhkan pemahaman konsep dan ilmu *coding* yang kuat dan memiliki keahlihan dalam merancang suatu sistem dari *scratch*,

2. Keterbatasan panduan internal atau dokumentasi proyek yang merinci alur kerja *end-to-end*. Hal ini menjadi tantangan karena rata-rata proyek yang dikerjakan melibatkan dua teknologi yang berbeda *backend* AI (ditulis dengan Python) dan *frontend* web (ditulis dengan PHP Laravel) yang harus saling terhubung.

3.3.3 Solusi atas Kendala yang Ditemukan

Ditemukan solusi atas kendala yang ditemukan pada saat proses kerja magang . diperusahaan Indonesia Morowali Industrial Park adalah:

1. Melakukan pembelajaran mandiri secara proaktif untuk menjembatani kesenjangan *skill gap*. Dengan memfokuskan diri pada studi dokumentasi resmi dari semua teknologi yang dipakai. Selain dokumentasi adapun penggunaan panduan video *youtube* dan bantuan dengan *AI* untuk membantu memecahkan masalah saya.
2. Selama proses pelaksanaan magang, diterapkan metode *learning by doing* melalui penelusuran alur data secara menyeluruh. Melalui pendekatan ini, pemahaman diperoleh mengenai tahapan dan mekanisme pengerjaan sistem, mulai dari proses awal hingga implementasi, sehingga mendukung penguasaan teknis terhadap pekerjaan yang dilakukan.