

**LAPORAN PRAKTIK KERJA LAPANGAN
DI PT. NILAM PORT TERMINAL INDONESIA - DIVISI IT**



OLEH:

**RAYHAN AJIE NUGRAHA (162112233047)
KHAULA SAYYIDATUNADIA (162112233076)**

**PROGRAM STUDI S1
TEKNIK ROBOTIKA DAN KECERDASAN BUATAN
DEPARTEMEN TEKNIK
FAKULTAS TEKNOLOGI MAJU DAN MULTIDIPLIN
UNIVERSITAS AIRLANGGA
SURABAYA
2024**

**LEMBAR PENGESAHAN
LAPORAN PRAKTIK KERJA LAPANGAN**

Identitas Peserta PKL :

1. Nama : Rayhan Ajie Nugraha
NIM : 162112233047
2. Nama : Khaula Sayyidatunadia
NIM : 162112233076

Identitas Dosen Pembimbing

- Nama : Muslim Hanafiq, S.Kom.
NIP : 16096432

Identitas Dosen Pembimbing

- Nama : Amila Sofiah, S.T., M.T
NIP : 199210262020013201

Disetujui dan disahkan sebagai Laporan Praktik Kerja Lapangan



Dosen Pembimbing

Amila Sofiah, S.T., M.T
NIP 199210262020013201

Surabaya,
Mengetahui,
Kepala Departemen Teknik

Dr. Prihartini Widiyanti, drg, M.Kes., S.Bio., CCD
NIP 19710712200812

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas segala kasih, karunia serta penyertaannya penulis dapat menyelesaikan Praktik Kerja Lapangan di PT. Nilam Port Terminal Indonesia – Divisi *IT*. Laporan ini menjadi pertanggungjawaban penulis selama menjalani Praktik Kerja Lapangan dengan konsentrasi proyek pada bidang *Web-App Development* dan *computer vision*.

Pertama-tama, penulis mengucapkan terima kasih yang tak terhingga kepada PT. Nilam Port Terminal Indonesia – Divisi *IT* yang telah menerima dan memberikan kesempatan bagi penulis untuk belajar serta menjalankan program Praktik Kerja Lapangan. Selanjutnya, Laporan Praktik Kerja Lapangan ini tidak akan terselesaikan tanpa adanya bimbingan, arahan, nasihat, bantuan, dan saran yang diberikan kepada penulis. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Purbandini, S.Si., M.Kom. selaku Koordinator Program Studi Teknik Robotika dan Kecerdasan Buatan yang telah memberikan arahan serta pembinaan kepada mahasiswa dalam penyusunan proposal skripsi ini.
2. Amila Sofiah, S.T., M.T. selaku Dosen Pembimbing atas banyaknya waktu dan tenaga yang telah diluangkan dari sebelum penulis mengambil mata kuliah proposal skripsi hingga penelitian ini selesai untuk membimbing penulis.
3. Para pegawai PT. Nilam Port Terminal Indonesia yang telah memberikan kesempatan kepada kami untuk membantu menyelesaikan Praktik Kerja Lapangan dengan tangan yang terbuka.

Akhir kata, penulis menyadari bahwa masih banyak kekurangan dari penulis selama periode Praktik Kerja Lapangan. Oleh karena itu, penulis memohon maaf bila ada kekurangan dan salah kata maupun perbuatan selama periode Praktik Kerja Lapangan ini. Penulis berharap agar pengalaman yang didapat selama ini dapat menjadi bekal ilmu serta sarana akselerasi perjalanan keilmuan dan karir penulis ke depannya.

Surabaya, November 2024

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN.....	ii
KATA PENGANTAR.....	iii
DAFTAR ISI.....	iv
DAFTAR GAMBAR.....	vi
DAFTAR TABEL.....	vi
DAFTAR LAMPIRAN.....	vii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan.....	2
1.4 Manfaat.....	3
1.5 Batasan Masalah.....	3
BAB II TINJAUAN PUSTAKA.....	4
2.1 Profil Perusahaan.....	4
2.2 Lingkup Pekerjaan.....	5
2.3 Landasan Teori.....	6
2.3.1 Manajemen Inventaris Kantor.....	6
2.3.2 <i>Website Development</i>	7
2.3.3 <i>Python</i> dan <i>Django</i>	9
2.3.4 <i>Mobile App Development</i> dengan <i>Kotlin</i>	12
2.3.5 <i>Database Management</i> dengan <i>SQLite</i>	14
2.3.6 <i>REST API</i>	16
2.3.7 <i>Tesseract OCR</i>	18
BAB III METODE PELAKSANAAN.....	22
3.1 Pelaksanakan Kerja Praktik.....	22
3.2 Metode Penyelesaian Proyek/Proses Bisnis.....	23
3.2.1 Studi Kasus dan Studi Literatur.....	24
3.2.2 Perancangan Sistem.....	25
3.2.3 Pembuatan <i>Website</i>	25
3.2.4 Pembuatan <i>Database</i> dan Integrasi dengan <i>Website</i>	32
3.2.5 Pembuatan Aplikasi <i>Mobile</i>	33
3.2.6 Integrasi Model OCR.....	34
3.2.7 Uji Coba Sistem.....	35
BAB IV HASIL DAN PEMBAHASAN.....	37
4.1 Bentuk Kegiatan.....	37
4.2 Pembuatan <i>Website</i>	37
4.3 Pembuatan Aplikasi.....	39
4.4 Integrasi <i>Web-App</i>	44

4.5 Analisis Uji Coba Sistem.....	46
BAB V KESIMPULAN DAN SARAN.....	48
5.1 Kesimpulan.....	48
5.2 Saran.....	48
DAFTAR PUSTAKA.....	49
LAMPIRAN.....	53

DAFTAR GAMBAR

Gambar 2.1 <i>Company Profile</i> PT. Nilam Port Terminal Indonesia.....	4
Gambar 2.2 Struktur Organisasi PT. Nilam Port Terminal Indonesia.....	6
Gambar 2.3 Struktur <i>Framework Django</i>	10
Gambar 2.4 Arsitektur SQLite.....	15
Gambar 2.5 <i>User Interface</i> SQLite.....	16
Gambar 2.6 Arsitektur REST API.....	17
Gambar 2.7 Struktur Model Tesseract OCR.....	20
Gambar 3.1 Diagram Alir Metode Penggeraan Proyek.....	24
Gambar 3.2 Rancangan Implementasi Sistem <i>Website Inventory Management</i>	25
Gambar 3.3 Use Case Sistem <i>Website Inventory Management View IT</i> dan <i>General Affairs</i>	26
Gambar 3.4 Use Case Sistem <i>Website Inventory Management View Finance</i>	26
Gambar 3.5 Mekanisme forms.py.....	27
Gambar 3.7 Proses Integrasi Aplikasi dengan <i>Website</i>	32
Gambar 3.8 Pembuatan Aplikasi dan <i>Import Model</i>	33
Gambar 4.1 <i>Login Page</i> dan <i>Landing Page Website</i>	37
Gambar 4.4 Fitur <i>Delete</i> dan <i>Edit</i> Pada <i>Dashboard</i>	39
Gambar 4.5 Fitur <i>Export To Excel</i> dan <i>Add Item</i>	39
Gambar 4.7 Tampilan Untuk Melakukan OCR dan Pemanggilan Data : (a) Tampilan Untuk Melakukan OCR Pada Nomor Inventaris (b) Tampilan Data Hasil Rekognisi Yang Berhasil.....	40
Gambar 4.8 Tampilan Untuk Permintaan <i>Service</i> dan Pemindahan Barang : (a) Tampilan Untuk Permintaan <i>Service</i> . (b) Tampilan Untuk Permintaan Pemindahan Barang.....	43
Gambar 4.9 Tampilan Menambahkan Inventaris Baru : (a) Tampilan Inventaris Tidak Ditemukan. (b) Tampilan Tambah Inventaris Baru. (c) Tampilan Tambah Detail Inventaris Baru. (d) Tampilan Tambah Detail Inventaris Baru.....	44
Gambar 4.10 Integrasi <i>Web-App Offline</i>	45

DAFTAR TABEL

Tabel 3.1 Kegiatan Selama Praktik Kerja Lapangan.....	22
Tabel 3.2 <i>Data Structure models.py</i>	28
Tabel 3.3 Penjelasan Fungsi REST API.....	29
Tabel 3.4 Daftar URL yang digunakan pada <i>urls.py</i>	30
Tabel 3.5 Daftar Contoh Gambar Nomor Inventaris.....	35
Tabel 3.6 Tabel Uji Coba Fitur dan Fungsionalitas Sistem.....	35
Tabel 4.1 Hasil Pembacaan <i>Digit</i> dengan OCR.....	41
Tabel 4.2 Uji Pembacaan <i>Digit</i> Dengan Berbagai Tingkat Kecerahan.....	42
Tabel 4.3 Tampilan <i>Website</i> dan Aplikasi.....	45
Tabel 4.4 Tabel Hasil Uji Coba Fitur dan Fungsionalitas Sistem.....	46

DAFTAR LAMPIRAN

Lampiran 1. Dokumentasi Pelaksanaan Kegiatan PKL.....	53
Lampiran 2. Alur dan Pendefinisian <i>Views.py</i>	53
Lampiran 3. Kode <i>Login Website</i> dalam <i>Views.py</i> dan <i>HTML</i>	55
Lampiran 4. Kode <i>Dashboard</i> Beserta Fitur <i>Search</i> , <i>Sort</i> , <i>Filter</i> , dan <i>Show Details</i> .	
56	
Lampiran 5. Kode <i>Dashboard</i> Beserta Fitur <i>Add</i> , <i>Edit</i> , <i>Delete</i> , dan <i>Export to Excel</i>	
58	
Lampiran 6. Pembuatan Aplikasi.....	60
Lampiran 7. <i>Gradle Dependencies OCR</i>	83
Lampiran 8. Video <i>Timer</i> Proses <i>Login</i>	85
Lampiran 9. Video Perhitungan waktu.....	86
Lampiran 10. Tabel Uji Fungsionalitas Akurasi.....	87

BAB I PENDAHULUAN

1.1 Latar Belakang

PT. Nilam Port Terminal Indonesia - Divisi *IT* merupakan sebuah divisi yang beroperasi di bawah payung PT. Nilam Port Terminal Indonesia yang berfokus pada bidang teknologi informasi. Sebagai sebuah perusahaan yang berfokus pada pengelolaan dan pemeliharaan sistem IT, jenis pekerjaan yang dilakukan memiliki tantangan tersendiri, terutama dalam hal pengelolaan data dan inventarisasi. Salah satu permasalahan yang sering dihadapi adalah proses input data yang masih dilakukan secara manual menggunakan *microsoft excel*. Hal ini mengakibatkan potensi kesalahan dan proses yang tidak efisien dalam manajemen inventaris (Wirna dkk., 2022).

Proses manual ini tidak hanya rentan terhadap kesalahan manusia, tetapi juga menghambat efisiensi dan akurasi dalam pengelolaan aset IT perusahaan. Mengingat pentingnya pengelolaan data yang tepat dalam mendukung operasi harian, perusahaan perlu beralih ke sistem digital yang terotomatisasi dan terintegrasi. Teknologi seperti *Artificial Intelligence* (AI) dapat diimplementasikan, misalnya melalui *Optical Character Recognition* (OCR) berbasis AI untuk mengotomatisasi pengambilan data dari dokumen fisik, seperti tag inventaris atau laporan manual, dan mengonversinya menjadi data digital secara akurat (Juniardi, 2024). AI juga memiliki peran penting dalam meningkatkan keamanan data. Sistem manajemen dokumen yang didukung oleh AI mampu mengenali pola anomali serta mendeteksi potensi ancaman keamanan secara dini, sehingga dapat mencegah terjadinya kerugian yang lebih besar (Yulianto dkk., 2024). Selain itu, *Natural Language Processing* (NLP) dapat digunakan untuk mengolah data teks dalam laporan atau komunikasi operasional, sementara *Machine Learning* (ML) memungkinkan analisis data secara prediktif untuk mendeteksi pola atau potensi anomali dalam pengelolaan aset (Surya, 2023). Adanya pemanfaatan teknologi AI dalam proses manajemen inventaris akan meminimalkan risiko kesalahan dalam pengelolaan data serta memberikan inovasi karena sifatnya yang adaptif sesuai dengan kebutuhan bisnis yang semakin kompleks.

Dengan demikian, untuk mewujudkan efisiensi dan manajemen inventaris yang lebih baik, diperlukan sebuah sistem yang mampu melakukan pengawasan dan pengelolaan data secara efektif dan efisien. Pengelolaan inventaris secara manual memang dapat dilakukan, namun metode ini sering kali tidak optimal dalam melakukan pengecekan dan pelacakan data. Oleh karena itu, inovasi diperlukan untuk meningkatkan efektivitas pengelolaan inventaris. Seiring perkembangan teknologi, solusi berbasis *web-app* menjadi pilihan yang tepat untuk mengatasi permasalahan ini. *Web-app* yang dirancang bertujuan untuk meningkatkan efisiensi dan manajemen inventaris di PT. Nilam Port Terminal Indonesia - Divisi *IT*. *Web-app* ini berfungsi sebagai *database* untuk input data baru dan mengubah data yang sudah ada, serta untuk *tracking* dan *monitoring* data secara *real-time*. Dengan *web-app* ini, proses *input*, *update*, dan pelacakan inventaris dapat dilakukan dengan lebih cepat, akurat, dan terintegrasi, sehingga mendukung kelancaran operasional perusahaan (Amol Rathod dkk., 2022).

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, dapat disusun beberapa rumusan masalah sebagai berikut:

1. Bagaimana cara mengatasi proses manual dalam pengelolaan inventaris yang masih dilakukan menggunakan *microsoft excel* di PT. Nilam Port Terminal Indonesia - Divisi *IT*?
2. Bagaimana fungsionalitas *web-app* sebagai sistem manajemen inventaris di PT. Nilam Port Terminal Indonesia?

1.3 Tujuan

1. Mengembangkan sebuah *web-app* yang mampu menggantikan proses pengelolaan inventaris manual yang dilakukan dengan *microsoft excel* di PT. Nilam Port Terminal Indonesia - Divisi *IT*.
2. Mengetahui fungsionalitas *web-app* sebagai sistem manajemen inventaris di PT. Nilam Port Terminal Indonesia.

1.4 Manfaat

1. *Web-app* yang dikembangkan akan membantu perusahaan dalam mengurangi waktu dan usaha yang dibutuhkan untuk mengelola inventaris, sehingga meningkatkan produktivitas dan efisiensi operasional.
2. Dengan mengurangi ketergantungan pada input manual, risiko kesalahan data dapat diminimalkan, yang pada akhirnya akan meningkatkan akurasi dan integritas informasi inventaris.
3. *Web-app* yang terintegrasi dengan baik akan menyediakan data inventaris yang akurat dan *real-time*, yang dapat digunakan oleh manajemen untuk pengambilan keputusan yang lebih cepat dan tepat.
4. Dengan adanya fitur *monitoring* yang dapat diakses secara *real-time*, pengawasan terhadap inventaris menjadi lebih mudah dan menyeluruh, sehingga membantu dalam mencegah kehilangan atau penyalahgunaan aset.

1.5 Batasan Masalah

1. Sistem hanya dapat membaca digit dan tanda baca dengan format *font Times New Roman*
2. Angka inventaris harus terdapat di dalam *bounding box* yang terdapat dalam tampilan

BAB II TINJAUAN PUSTAKA

2.1 Profil Perusahaan

PT. Nilam Port Terminal Indonesia (PT. NPTI) adalah sebuah perusahaan yang bergerak di bidang penyediaan dan pelayanan jasa operator terminal bongkar muat barang dan peti kemas di pelabuhan. Didirikan melalui konsorsium dari enam perusahaan, PT. NPTI berperan penting dalam mendukung kelancaran operasional bongkar muat di Pelabuhan Tanjung Perak, Surabaya, khususnya di Terminal Nilam Multipurpose. Dalam menjalankan operasionalnya, PT. NPTI bekerja sama dengan PT. Pelabuhan Indonesia III dan menggunakan berbagai peralatan bongkar muat seperti tiga unit *Container Crane*, lima unit *Rubber Tyred Gantry* (RTG), dan 30 unit *head truck* serta *trailer*. Perusahaan ini mengelola dermaga sepanjang 320 meter dengan area penumpukan yang luasnya mencapai 4 hektar (PT. Nilam Port terminal Indonesia, 2016).



Gambar 2.1 *Company Profile* PT. Nilam Port Terminal Indonesia (PT. Nilam Port terminal Indonesia, 2016)

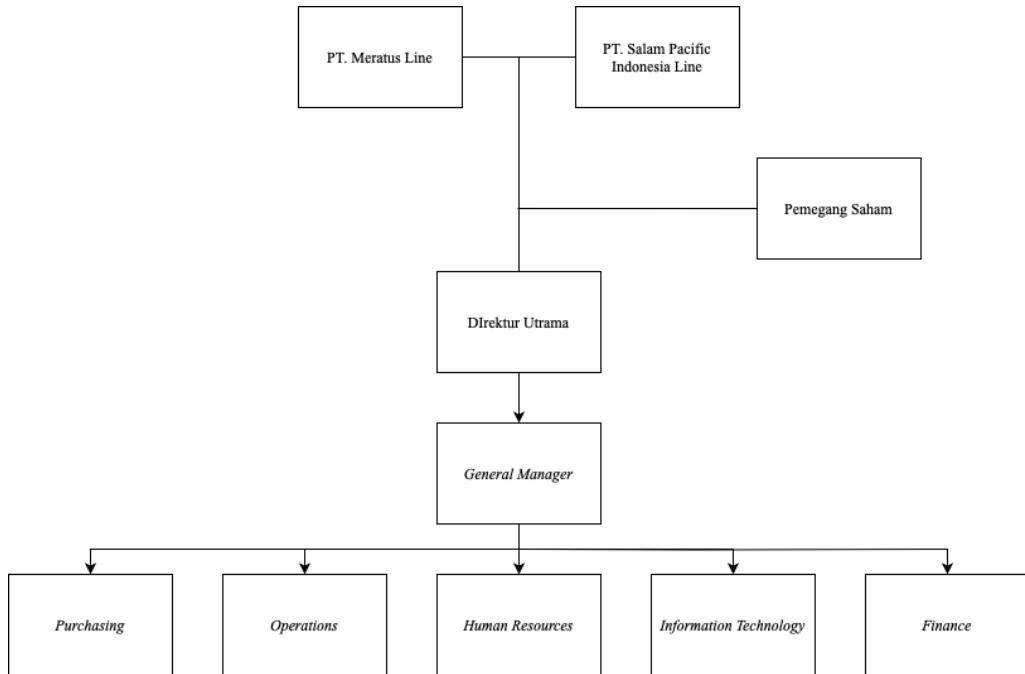
PT. NPTI memiliki visi untuk menjadi operator terminal pelabuhan yang mampu memberikan pelayanan terbaik dalam rangka mencapai kepuasan pelanggan di bidang bongkar muat peti kemas di Indonesia. Adapun misi perusahaan mencakup:

1. Menyediakan jasa terminal pelabuhan yang berkualitas dengan memaksimalkan sarana dan prasarana yang tersedia.
2. Menjalankan proses bisnis yang efisien dan efektif dengan tetap memperhatikan aspek Keselamatan, Kesehatan, Keamanan, dan Lingkungan (K3L).
3. Memberikan nilai tambah kepada para pemangku kepentingan (*stakeholders*)

Tujuan utama PT. NPTI adalah menyediakan jasa operator terminal bongkar muat barang dan peti kemas serta berbagai layanan kepelabuhanan terkait. Dengan fokus pada efisiensi dan optimalisasi layanan, PT. NPTI berkomitmen untuk mendukung perkembangan industri pelabuhan di Indonesia melalui inovasi, peningkatan kapasitas layanan, dan kemitraan strategis (PT. Nilam Port terminal Indonesia, 2016).

2.2 Lingkup Pekerjaan

Divisi IT PT. Nilam Port Terminal Indonesia memiliki tanggung jawab yang sangat penting dalam menjaga infrastruktur teknologi informasi yang ada di perusahaan. Tugas utama divisi ini mencakup pemeliharaan infrastruktur IT, yang memastikan semua komponen teknologi, seperti server, perangkat keras, dan jaringan, berfungsi dengan baik dan selalu dalam kondisi optimal. Pemeliharaan ini krusial untuk menjamin kelangsungan operasional perusahaan tanpa gangguan. Selain itu, divisi ini berfokus pada menjaga reliabilitas sistem informasi yang digunakan, baik dalam pengolahan data, komunikasi internal, maupun interaksi dengan klien. Ini termasuk pengelolaan backup data dan mitigasi risiko terhadap ancaman keamanan siber. Dalam rangka mengaplikasikan pengetahuan dan mendapatkan pengalaman langsung, penulis berkesempatan menjalani Praktik Kerja Lapangan di divisi IT ini. Perusahaan memberikan instruksi yang jelas untuk menjalankan praktik kerja lapangan berbasis proyek, yang berfokus pada digitalisasi sistem inventaris kantor. Proyek ini bertujuan untuk mengubah sistem inventaris yang sebelumnya berbasis manual menjadi sistem digital yang lebih efisien dan mudah diakses, sehingga diharapkan proses pencatatan, pengawasan, dan pelaporan inventaris dapat dilakukan secara lebih cepat dan akurat. Struktur Organisasi dari PT Nilam Port Terminal Indonesia dapat dilihat pada Gambar 2.1.



Gambar 2.2 Struktur Organisasi PT. Nilam Port Terminal Indonesia (PT. Nilam Port terminal Indonesia, 2016)

2.3 Landasan Teori

2.3.1 Manajemen Inventaris Kantor

Manajemen inventaris kantor adalah aspek penting dari efisiensi organisasi, hal ini bertujuan untuk memastikan bahwa persediaan dan peralatan yang diperlukan tersedia untuk mendukung operasi sehari-hari. Sistem manajemen inventaris yang efektif membantu sebuah entitas mengurangi biaya, mencegah kerugian, dan meningkatkan layanan. Manajemen yang tidak memadai dapat menyebabkan kekurangan stok atau kelebihan stok, yang keduanya memiliki implikasi finansial berupa penundaan operasi atau biaya penyimpanan yang berlebihan. Manajemen inventaris yang tepat memastikan bahwa barang-barang yang benar tersedia saat dibutuhkan, sehingga mencegah gangguan dalam alur kerja (Freeman, 1993).

Ada beberapa metode yang digunakan untuk mengelola inventaris kantor secara efektif. Salah satu pendekatan umum adalah meninjau inventaris secara berkala, yang melibatkan pemeriksaan tingkat stok pada interval waktu tertentu dan pengisian ulang persediaan berdasarkan ambang batas yang telah ditentukan sebelumnya. Metode ini sangat berguna untuk mengelola barang-barang non-kritis seperti alat tulis kantor (Petropoulos dkk., 2024). Metode lain yang banyak

diadopsi adalah manajemen inventaris *Just-in-Time* (JIT), yang bertujuan untuk meminimalkan biaya penyimpanan dengan memesan persediaan hanya ketika dibutuhkan. Namun, JIT membutuhkan koordinasi yang tepat dengan pemasok untuk memastikan pengiriman tepat waktu (Nitesh, 2024)

Kemajuan teknologi juga telah merevolusi praktik manajemen inventaris. integrasi *Internet of Things* (IoT) dan solusi berbasis *cloud* telah memungkinkan pelacakan inventaris kantor secara *real-time* dan pemantauan tingkat stok otomatis (Rajab, 2024). Munculnya otomatisasi telah secara signifikan meningkatkan efisiensi sistem manajemen inventaris kantor. Sistem otomatis memungkinkan organisasi untuk melacak inventaris mereka secara *real-time* menggunakan *sensor* dan tag *Radio Frequency Identification* (RFID) yang terhubung ke platform *cloud*.

Manajemen inventaris kantor yang efektif memiliki dampak langsung pada kinerja organisasi. Dengan memastikan bahwa persediaan tersedia saat dibutuhkan, organisasi dapat menghindari *delay* pada proses bisnis akibat kekurangan stok. Selain itu, inventaris yang dikelola dengan baik berkontribusi pada produktivitas karyawan dengan menyediakan alat-alat yang mereka butuhkan untuk melakukan tugas mereka secara efisien (Mwamba dan Yangailo, 2024).

Pengendalian inventaris yang tepat membantu organisasi menghindari pengeluaran yang tidak perlu terkait dengan kelebihan stok atau pembelian darurat. Kelebihan stok mengikat modal dalam persediaan yang tidak digunakan sekaligus meningkatkan biaya penyimpanan; sebaliknya, pembelian darurat sering kali datang dengan harga premium karena karena kebutuhan yang mendesak (Mwamba dan Yangailo, 2024).

2.3.2 Website Development

Website development untuk manajemen inventaris telah menjadi strategi penting bagi bisnis dan institusi untuk meningkatkan efektivitas operasional, meningkatkan akurasi, dan memperkuat kolaborasi. Sistem manajemen inventaris berbasis web menawarkan berbagai keuntungan, termasuk pelacakan secara *real-time*, skalabilitas, dan penghematan biaya, yang sangat bermanfaat di industri seperti pendidikan, manufaktur, dan manajemen rantai pemasaran.

Salah satu manfaat utama dari sistem inventaris berbasis web adalah pelacakan inventaris secara *real-time*. Fitur ini memungkinkan sebuah entitas untuk memantau tingkat stok dan pergerakan barang di berbagai lokasi secara instan. Misalnya, di institusi pendidikan, sistem berbasis web dapat melacak persediaan dan peralatan secara efisien, memastikan bahwa sumber daya tersedia saat dibutuhkan tanpa kelebihan atau kekurangan. Sebuah studi yang dilakukan tentang manajemen inventaris di institusi pendidikan menunjukkan bahwa penerapan sistem semacam ini meningkatkan efisiensi sebesar 86,31%, hal ini memiliki dampak positif berupa pengurangan waktu yang dihabiskan untuk mencari barang dan membuat laporan (Ramos-Miller dan Pacheco, 2023).

Keuntungan lain dari sistem berbasis web adalah kemampuannya untuk mengurangi kesalahan dan meningkatkan akurasi. Manajemen inventaris manual sering kali menyebabkan kesalahan seperti entri data yang salah atau barang yang salah tempat, yang dapat mengakibatkan kesalahan fatal. Dengan otomatisasi proses melalui platform berbasis *web*, sebuah entitas dapat meminimalkan kesalahan ini (Ezekiel A. dkk., 2024).

Sistem inventaris berbasis web juga meningkatkan efisiensi operasional dengan mengotomatiskan tugas-tugas rutin seperti pemeriksaan stok dan pembuatan laporan. Pada industri manufaktur atau logistik, di mana akses *real-time* terhadap data inventaris sangat penting untuk pengambilan keputusan, sistem ini memungkinkan respons yang lebih cepat dan alokasi sumber daya yang lebih baik. Institusi yang menggunakan sistem berbasis web melaporkan peningkatan sebesar 83% dalam menghasilkan laporan barang tahunan, yang menyederhanakan proses dan memberikan wawasan yang lebih jelas tentang tingkat stok (Ramos-Miller dan Pacheco, 2023).

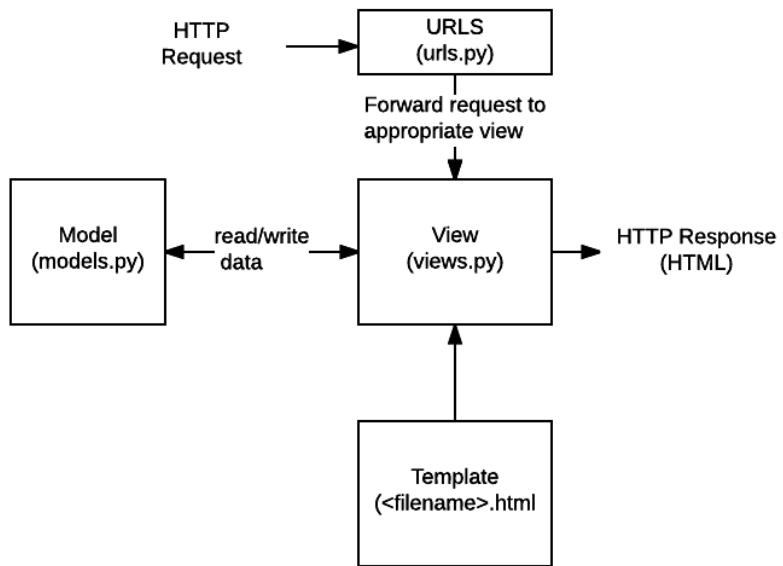
Skalabilitas adalah manfaat penting lainnya dari sistem inventaris berbasis *web*. Platform ini dapat dengan mudah beradaptasi dengan kebutuhan entitas yang berkembang tanpa memerlukan perubahan besar pada infrastruktur. Baik bisnis sedang memperluas operasinya atau mengelola fluktuasi tingkat inventaris karena permintaan musiman, sistem berbasis *web* dapat diskalakan sesuai kebutuhan. Selain itu, karena sistem ini dapat diakses dari perangkat apa pun dengan koneksi internet, mereka menawarkan fleksibilitas bagi tim jarak jauh

atau pekerja lapangan yang membutuhkan akses *real-time* ke data inventaris (Ramos-Miller dan Pacheco, 2023).

2.3.3 Python dan Django

Python secara luas dikenal sebagai bahasa pemrograman tingkat tinggi yang terkenal karena kesederhanaan dan fleksibilitasnya. Sintaks yang jelas dan format yang mudah dibaca membuatnya sangat menarik bagi pemula, sementara ekosistem pustaka yang luas memungkinkan pengembang untuk membangun aplikasi kompleks di berbagai bidang, termasuk pengembangan *web*, analisis data, pembelajaran mesin, dan otomatisasi (Rajab, 2024). Salah satu kekuatan terbesar *Python* adalah kemampuannya untuk berintegrasi dengan bahasa dan alat lain, memungkinkan pengembang menggunakan *Python* dalam berbagai pengaturan—baik untuk skrip tugas sederhana maupun membangun aplikasi berskala besar (Jongmans dkk., 2022). Fleksibilitas ini menjadikan *Python* pilihan populer di dunia akademik dan industri, terutama dalam bidang komputasi ilmiah, ilmu data, dan kecerdasan buatan (AI), di mana pustaka seperti *Pandas* dan *Matplotlib* sering digunakan untuk manipulasi data dan visualisasi (Farooque dkk., 2019).

Django adalah sebuah *framework* yang dibangun di atas *Python*. *Django* adalah sebuah *web framework* tingkat tinggi yang dirancang untuk menyederhanakan pengembangan aplikasi *web* yang aman dan *scalable*. *Django* memanfaatkan kesederhanaan *Python* dan juga menyediakan seperangkat alat komprehensif yang merampingkan proses pembangunan aplikasi *web* yang andal. Salah satu fitur paling menonjol dari *Django* adalah kepatuhannya pada prinsip "*Don't Repeat Yourself*" (DRY), yang mendorong penggunaan ulang kode dan pemeliharaan dengan mengurangi redundansi (Rajab, 2024). Hal ini membuat *Django* menjadi kerangka kerja yang efisien bagi pengembang yang ingin fokus pada pembangunan fitur unik daripada menulis kode *boilerplate* berulang kali. Selain itu, antarmuka admin bawaan *Django* memungkinkan pengelolaan konten situs dan autentikasi pengguna dengan mudah tanpa memerlukan pengetahuan coding yang mendalam, sehingga dapat diakses bahkan oleh pengguna non-teknis (Jongmans dkk., 2022).



Gambar 2.3 Struktur *Framework Django* (Patel, 2021)

Model dalam *framework Django* adalah pondasi dari sistem manajemen data *framework* ini. Sebuah model adalah kelas *Python* yang mendefinisikan struktur data yang disimpan dalam basis data, di mana setiap *field* dalam model mewakili kolom dalam tabel basis data. Setiap *instance* dari model tersebut sesuai dengan baris dalam tabel, sehingga memudahkan pemetaan objek *Python* ke catatan basis data (Django Documentation, 2023). Misalnya, sebuah model untuk menyimpan informasi tentang buku umumnya mencakup *field* untuk judul, penulis, tanggal publikasi, dan nomor ISBN. Model *Django* menyediakan lapisan abstraksi di atas basis data, memungkinkan pengembang untuk berinteraksi dengan data menggunakan kode *Python* daripada menulis *query SQL* mentah (Ramos-Miller dan Pacheco, 2023). Abstraksi ini menyederhanakan operasi basis data dan mengurangi kemungkinan kesalahan yang terkait dengan penulisan *query* manual.

Salah satu fitur unggulan dari model *Django* adalah integrasinya dengan *interface admin Django*. Fitur ini secara otomatis menghasilkan *interface* ramah pengguna untuk mengelola data tanpa memerlukan *developer* untuk membuat form atau *view* kustom. Selain itu, sistem *Object-Relational Mapping* (ORM) *Django* memungkinkan pengembang untuk melakukan *query* basis data menggunakan kode *Python*. ORM menerjemahkan *query Python* ini menjadi SQL

di balik layar, sehingga memudahkan *developer* yang mungkin tidak terbiasa dengan sintaks SQL untuk berinteraksi dengan basis data (Ezekiel A. dkk., 2024).

Disamping model, terdapat komponen *view* dalam *Django* yang bertanggung jawab untuk memproses permintaan pengguna dan mengembalikan respons yang sesuai. Sebuah fungsi *view* menerima permintaan HTTP dari pengguna, memproses logika yang diperlukan seperti melakukan kueri ke basis data dan mengembalikan respons HTTP. Respons ini bisa berupa halaman HTML yang diproses dari template atau jenis konten lainnya seperti JSON atau XML (Django Documentation, 2023). *View* berfungsi sebagai lapisan logika dalam arsitektur MVT *Django* dengan menentukan data apa yang harus ditampilkan dan bagaimana seharusnya disajikan kepada pengguna.

Komponen penting ketiga dalam arsitektur *Django* adalah sistem *routing* URL-nya. URL di Django dipetakan ke fungsi *view* tertentu melalui konfigurasi URL yang didefinisikan dalam *urls.py*. File ini berisi pola-pola yang mencocokkan jalur URL tertentu dengan fungsi *view* terkait. Ketika seorang pengguna membuat permintaan ke *server*, Django memeriksa URL terhadap pola-pola ini hingga menemukan satu yang cocok. Setelah kecocokan ditemukan, *Django* memanggil fungsi *view* terkait dengan parameter apapun yang ditangkap dari URL (Django Documentation, 2023). Sistem *routing* URL ini sangat fleksibel dan memungkinkan pengembang untuk membuat URL dinamis yang menangkap bagian dari jalur sebagai variabel seperti ID produk atau nama pengguna dan meneruskannya sebagai argumen ke fungsi *view*.

Interaksi antara *model*, *view*, dan URL sangat penting bagi aliran data dalam aplikasi Django. Ketika seorang pengguna membuat permintaan melalui URL yang dipetakan ke fungsi *view* atau *class-based view*, *view* tersebut akan berinteraksi dengan satu atau lebih model untuk mengambil atau memanipulasi data yang disimpan di basis data. Setelah data tersebut diproses oleh logika *view*, data tersebut diteruskan ke template untuk dirender menjadi HTML atau format lain sebelum dikirim kembali sebagai respons HTTP. Sebagai contoh, terdapat sebuah aplikasi dimana pengguna dapat meminta informasi tentang buku-buku yang disimpan dalam sistem perpustakaan. Pola URL akan memetakan permintaan ini ke fungsi *view* seperti *book_list*, yang melakukan *query* terhadap

model Book menggunakan ORM Django untuk mengambil semua catatan buku dari basis data. Data yang diambil kemudian diteruskan ke konteks template sebelum diproses menjadi halaman HTML yang menampilkan daftar buku (Django Documentation, 2023).

2.3.4 *Mobile App Development* dengan Kotlin

Mobile App Development telah mengalami transformasi signifikan selama dekade terakhir, *developer* tentunya akan memilih bahasa pemrograman yang menawarkan efisiensi dan fleksibilitas. Salah satu bahasa yang paling menarik untuk digunakan dalam beberapa tahun terakhir adalah Kotlin. Dikembangkan oleh JetBrains dan secara resmi didukung oleh Google untuk pengembangan Android sejak 2017, Kotlin dengan cepat mendapatkan popularitas karena fitur-fiturnya yang modern, mekanisme keamanan yang menyeluruh, dan interoperabilitasnya yang mulus dengan Java. Mengingat bahwa Java telah menjadi bahasa tradisional untuk pengembangan Android, kemampuan Kotlin untuk terintegrasi dengan kode Java yang sudah ada tanpa memerlukan penulisan ulang total menjadikannya pilihan menarik bagi para *developer* (Alam dkk., 2024).

Kotlin telah menjadi bahasa yang kuat dalam pengembangan aplikasi *mobile*, terutama untuk aplikasi *Android*. Sebagai bahasa pemrograman *statically-typed*, Kotlin dirancang untuk sepenuhnya beroperasi bersama dengan Java, memungkinkan pengembang untuk secara bertahap mengadopsi Kotlin dalam proyek mereka tanpa harus meninggalkan kode Java yang sudah ada (Alam dkk., 2024). Interoperabilitas ini adalah salah satu alasan utama mengapa Kotlin diadopsi dengan cepat oleh para *developer* yang sebelumnya menggunakan Java tetapi ingin memanfaatkan fitur-fitur modern dari Kotlin. Kemampuan untuk mencampur kode Kotlin dan Java dalam proyek yang sama memungkinkan bisnis dengan basis kode lama yang besar untuk beralih ke Kotlin secara mulus tanpa mengganggu proyek yang sedang berjalan. Fleksibilitas ini menjadikan Kotlin pilihan menarik baik untuk proyek baru maupun sistem lama yang mencari modernisasi.

Salah satu alasan utama di balik meningkatnya popularitas Kotlin adalah sintaksnya yang ringkas dibandingkan dengan Java. Dengan mengurangi

boilerplate code (kode berulang), Kotlin memungkinkan *developer* menulis lebih banyak fungsionalitas dengan lebih sedikit baris kode, yang tidak hanya meningkatkan produktivitas tetapi juga memudahkan keterbacaan dan pemeliharaan kode (Alam dkk., 2024). Sebuah studi tentang praktik pengembangan aplikasi mobile menemukan bahwa pengembang yang beralih dari Java ke Kotlin mengalami pengurangan ukuran kode sebesar 30% tanpa mengorbankan kinerja atau fungsionalitas (Raya Janti dkk., 2020) Pengurangan boilerplate code ini juga berarti lebih sedikit peluang terjadinya kesalahan dan *bug*, yang pada akhirnya menghasilkan produk yang lebih stabil. Selain itu, kemampuan Kotlin untuk merampingkan tugas-tugas pemrograman umum membantu pengembang lebih fokus pada pembangunan fitur daripada mengelola tugas-tugas berulang.

Selain sintaksnya yang ringkas, Kotlin menawarkan beberapa fitur keamanan yang membuatnya sangat cocok untuk pengembangan aplikasi mobile. Salah satu fitur paling menonjol adalah *null safety*, yang membantu mencegah *null pointer exception* yang menjadi sumber umum terjadinya *crash* pada aplikasi Android berbasis Java (Alam dkk., 2024). Dalam Kotlin, variabel secara *default* tidak boleh *null* kecuali dinyatakan secara eksplisit sebagai *nullable*. Hal ini memaksa *developer* untuk menangani nilai *null* dengan aman menggunakan mekanisme seperti *safe call operator* atau *non-null assertion operator*, sehingga mengurangi kemungkinan terjadinya error saat *runtime* akibat referensi *null*. Inklusi *null safety* sebagai fitur inti mencerminkan penekanan Kotlin pada pengurangan kesalahan pemrograman umum yang dapat menyebabkan *crash* aplikasi.

Selain *null safety* dan sintaks ringkasnya, Kotlin menawarkan fitur canggih lainnya yang meningkatkan produktivitas pengembang dan kinerja aplikasi. Salah satu fitur tersebut adalah *coroutines*, yang menyederhanakan pemrograman *asynchronous*—aspek penting dari pengembangan aplikasi *mobile* di mana tugas-tugas seperti permintaan jaringan atau operasi basis data sering kali perlu dilakukan di latar belakang (Alam dkk., 2024). *Coroutines* memungkinkan pengembang menulis kode *asynchronous* yang terlihat dan berperilaku seperti kode *synchronous*, membuatnya lebih mudah dibaca dan dipelihara dan juga

menghindari *callback hell*—masalah umum dalam pemrograman asynchronous. Dengan menyederhanakan manajemen *concurrency*, *coroutines* membantu meningkatkan kinerja dan responsivitas aplikasi *mobile*.

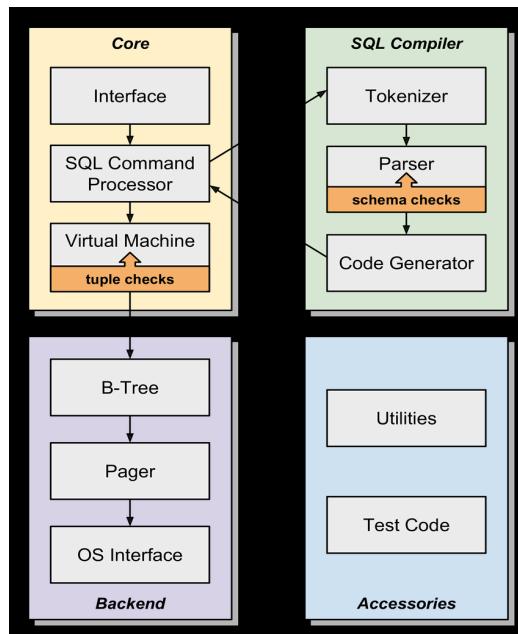
Fitur penting lainnya adalah *extension functions*, yang memungkinkan pengembang menambahkan fungsionalitas baru ke kelas-kelas yang ada tanpa memodifikasi kode sumber mereka. Fitur ini sangat berguna saat bekerja dengan pustaka pihak ketiga atau API di mana modifikasi langsung tidak mungkin dilakukan (Thomas dan Devi, 2021). *Extension functions* memungkinkan pengembang memperluas kemampuan kelas-kelas yang ada dengan cara yang bersih dan tidak mengganggu, sehingga meningkatkan *reusability* (penggunaan kembali) dan *Maintainability* (pemeliharaan) kode. Fitur-fitur canggih ini menunjukkan bagaimana Kotlin memberdayakan para pengembang dengan alat-alat yang meningkatkan kualitas dan efisiensi kode mereka.

Meskipun memiliki banyak keuntungan, ada beberapa tantangan terkait adopsi Kotlin untuk pengembangan aplikasi *mobile*. Salah satu tantangan adalah kurva pembelajaran bagi para pengembang yang terbiasa menggunakan Java atau bahasa pemrograman lainnya. Meskipun sintaksis Kotlin lebih ringkas daripada Java, ia memperkenalkan konsep baru seperti *coroutines* dan *extension functions* yang mungkin memerlukan waktu bagi para pengembang untuk benar-benar memahami sepenuhnya (Raya Janti dkk., 2020). Namun demikian, banyak sumber daya tersedia—termasuk dokumentasi resmi dari JetBrains dan Google—untuk membantu mempermudah transisi ini. Tantangan lain terletak pada ekosistem seputar Kotlin yang relatif muda dibandingkan bahasa-bahasa mapan lainnya seperti Java atau Swift. Meskipun banyak pustaka sekarang mendukung baik Java maupun Kotlin, masih ada beberapa area—terutama dalam pengembangan lintas platform—di mana *tooling* dan dukungan komunitas belum sematang solusi lain seperti Flutter atau React Native (Thomas dan Devi, 2021).

2.3.5 Database Management dengan SQLlite

Database Management adalah aspek fundamental dari sistem perangkat lunak modern, yang memungkinkan penyimpanan, pengambilan, dan manipulasi data secara efisien. Di antara banyak *Database Management System* (DBMS)

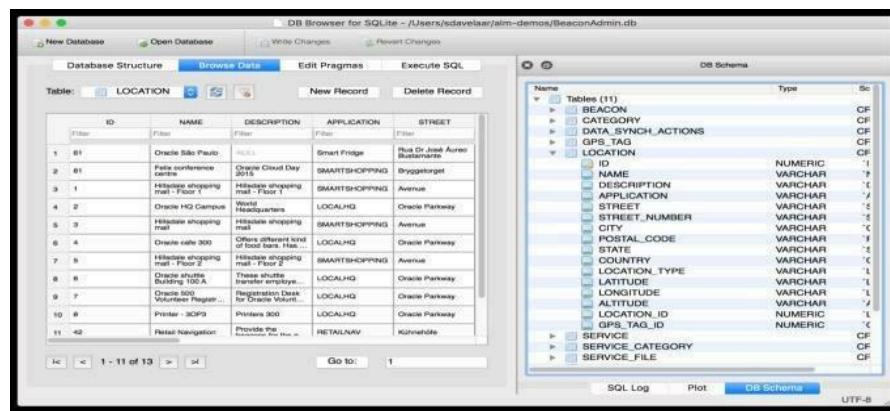
yang tersedia, SQLite menonjol sebagai *engine* basis data relasional yang ringan, tanpa server, dan *independent*. Tidak seperti DBMS tradisional seperti MySQL atau PostgreSQL, yang memerlukan pengaturan server yang kompleks, SQLite beroperasi langsung di dalam aplikasi itu sendiri, membaca dan menulis ke *file disk* biasa. Arsitektur unik ini menjadikan SQLite pilihan ideal untuk aplikasi di mana kesederhanaan, konfigurasi minimal, dan konsumsi sumber daya rendah sangat penting (Kengalagutti, 2020).



Gambar 2.4 Arsitektur SQLite (Mutti dkk., 2015)

Yang membedakan SQLite dari DBMS lain adalah sifatnya yang independen. Semua komponen yang diperlukan untuk manajemen basis data disertakan dalam satu file pustaka yang dapat dengan mudah diintegrasikan ke dalam aplikasi apa pun. Hal ini menghilangkan kebutuhan akan prosedur instalasi yang kompleks atau tugas administrasi server, menjadikannya sangat menarik bagi pengembang yang bekerja pada proyek di mana kemudahan penggunaan sangat penting (Shiratuddin dan Thabet, 2011). Meskipun sederhana, SQLite mendukung sebagian besar standar SQL-92, memastikan kompatibilitas dengan sistem basis data relasional lainnya serta mempertahankan bentuknya yang ringan. Keseimbangan antara kesederhanaan dan fungsionalitas ini telah menjadikan SQLite pilihan menarik bagi *developer* yang membutuhkan solusi basis data yang kuat namun mudah digunakan di berbagai platform dan lingkungan.

Salah satu fitur utama yang membuat SQLite sangat efektif dalam mengelola basis data adalah arsitekturnya yang tidak memerlukan *server*. Tidak seperti model *client-server* tradisional di mana proses server terpisah menangani permintaan basis data, SQLite beroperasi langsung di dalam aplikasi itu sendiri. Hal ini mengurangi kompleksitas dalam mengelola koneksi dan transaksi basis data dan juga meningkatkan kinerja dengan menghilangkan latensi jaringan. Selain itu, SQLite mematuhi prinsip dasar yang paling penting pada *Database Management*, yaitu *Atomicity*, *Consistency*, *Isolation*, *Durability* (ACID), yang memastikan bahwa semua transaksi basis data diproses secara lancar bahkan dalam kasus kegagalan sistem atau pemadaman listrik menjadikannya cocok untuk aplikasi misi kritis di mana integritas data sangat penting (Kengalagutti, 2020).



Gambar 2.5 User Interface SQLite (Chowdhury dkk., 2021)

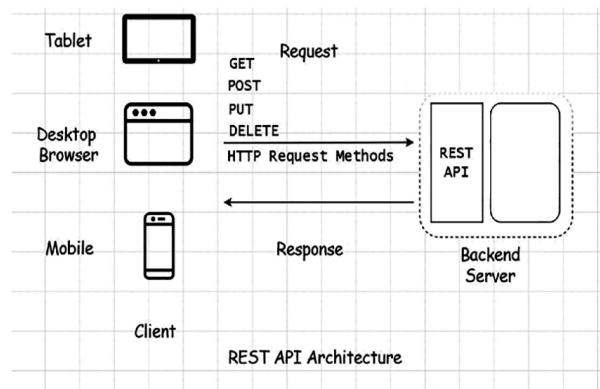
Fleksibilitas SQLite telah menyebabkan adopsinya secara luas di berbagai bidang. Dalam pengembangan aplikasi *mobile*, SQLite sering digunakan sebagai solusi penyimpanan lokal *default* karena sifatnya yang ringan dan kemampuannya untuk menangani kumpulan data besar secara efisien tanpa mengonsumsi sumber daya sistem yang signifikan. Banyak sistem operasi *mobile* populer seperti Android bergantung pada SQLite untuk menyimpan preferensi pengguna, data aplikasi, dan bahkan konten *offline* (Gajewski dkk., 1994).

2.3.6 REST API

Representational State Transfer (REST) adalah arsitektur untuk merancang aplikasi jaringan yang telah menjadi salah satu standar yang paling

banyak diadopsi untuk membangun *Application Programming Interface* (API). REST API memungkinkan sistem perangkat lunak yang berbeda untuk berkomunikasi melalui internet menggunakan protokol standar seperti HTTP, memungkinkan pertukaran data yang efisien antara klien dan server. Berbeda dengan protokol lama seperti *Simple Object Access Protocol* (SOAP), yang lebih kaku dan kompleks, REST API memanfaatkan metode HTTP—seperti GET, POST, PUT, dan DELETE—untuk melakukan operasi pada sumber daya yang diidentifikasi oleh URL (Akilesh dan Hallikar, 2022). Kesederhanaan ini telah berkontribusi pada adopsi luas REST dalam arsitektur perangkat lunak modern. Sebuah studi tentang penggunaan API mengungkapkan bahwa 75% *developer* lebih memilih REST API karena kemudahan integrasi dan skalabilitasnya (Mudassir dan Mohammed, 2024).

Keunggulan utama REST API adalah *statelessness*-nya, yang berarti setiap permintaan dari klien ke server harus berisi semua informasi yang diperlukan untuk memahami dan memproses permintaan tersebut. Karakteristik ini memungkinkan server tetap agnostik terhadap interaksi sebelumnya, membuat sistem lebih *scalable* karena tidak perlu menyimpan informasi sesi antara permintaan (Jatain dkk., 2021). Sifat *stateless* ini sangat bermanfaat bagi aplikasi berbasis *cloud* di mana skalabilitas sangat penting. Platform *cloud* seperti *Amazon Web Services* (AWS) dan *Microsoft Azure* sangat mengandalkan REST API untuk memfasilitasi komunikasi antar layanan.



Gambar 2.6 Arsitektur REST API (Mitropoulos dkk., 2021)

Fitur utama dari REST adalah arsitektur berbasis sumber daya, di mana segala sesuatu diperlakukan sebagai sumber daya yang dapat *create, read, update, delete* (CRUD) menggunakan metode HTTP standar. Setiap sumber daya

diidentifikasi secara unik oleh URL, memungkinkan klien untuk berinteraksi dengan data atau fungsionalitas tertentu dalam sebuah aplikasi. Pendekatan ini menyederhanakan desain API dengan membuatnya intuitif dan dapat diprediksi (Akilesh dan Hallikar, 2022). Manfaat signifikan lainnya dari REST API adalah interoperabilitas-nya. Dengan mematuhi standar terbuka seperti HTTP dan JSON, REST API dapat digunakan di berbagai platform dan bahasa pemrograman tanpa memerlukan *middleware* atau protokol khusus. Hal ini menjadikannya pilihan ideal untuk mengintegrasikan layanan pihak ketiga ke dalam aplikasi. Misalnya, banyak platform media sosial seperti Facebook dan Twitter menawarkan REST API publik yang memungkinkan pengembang mengintegrasikan fitur sosial ke dalam aplikasi mereka sendiri (Mudassir dan Mohammed, 2024). Interoperabilitas yang disediakan oleh REST mendorong inovasi dengan memungkinkan pengembang membangun di atas layanan yang sudah ada daripada memulai dari awal.

Meskipun banyak keunggulan yang ditawarkan oleh REST API, ada beberapa kekurangan yang harus diatasi. Karena REST API sering mengekspos data sensitif melalui internet, hal ini dapat menjadi target serangan siber jika tidak diamankan dengan baik. Kerentanan umum termasuk mekanisme otentikasi yang tidak memadai atau transmisi data yang tidak terenkripsi sehingga sistem rentan terhadap serangan seperti *man-in-the-middle* atau pembajakan API. Sebuah survei menemukan bahwa 35% *developer* menyebutkan kerentanan keamanan sebagai perhatian utama saat bekerja dengan REST API (Mudassir dan Mohammed, 2024). Untuk mengurangi risiko ini, praktik terbaik seperti menerapkan OAuth untuk otentikasi dan menggunakan HTTPS untuk komunikasi aman sangat penting.

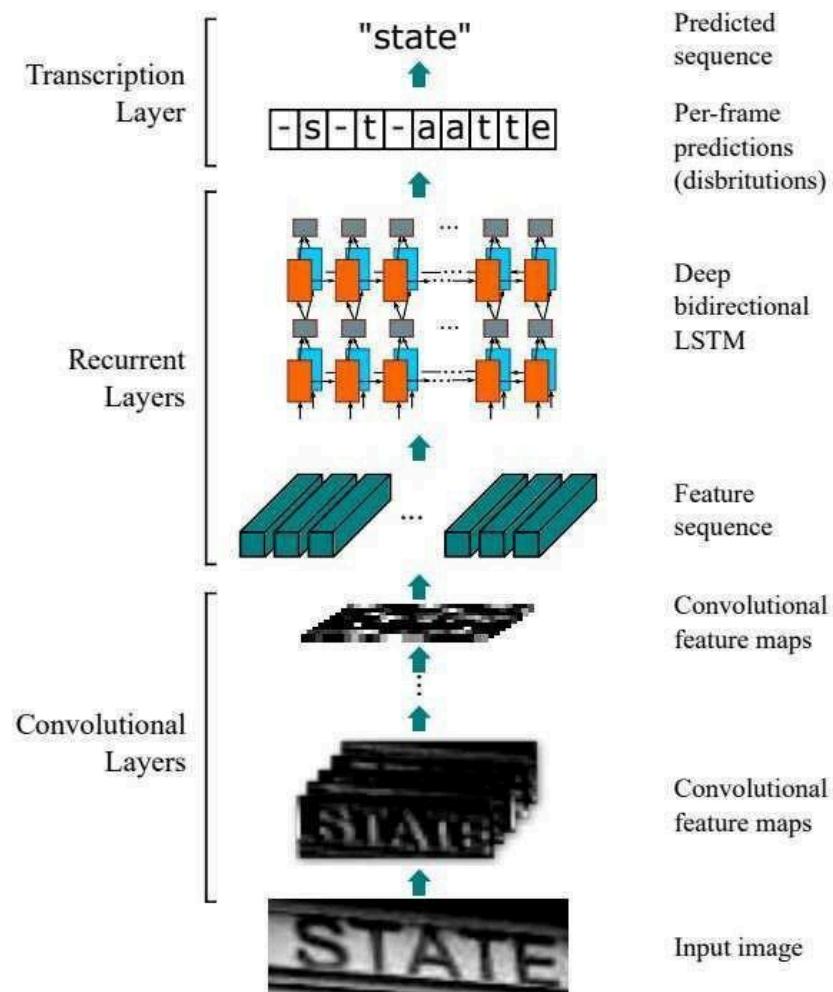
2.3.7 Tesseract OCR

Tesseract OCR (Optical Character Recognition) adalah salah satu modul OCR *open-source* yang paling banyak digunakan, dikenal karena kemampuannya mengonversi gambar teks yang diketik, tulisan tangan, atau teks cetak menjadi format yang dapat dibaca oleh mesin. Awalnya dikembangkan oleh *Hewlett-Packard* pada tahun 1980-an dan kemudian disempurnakan oleh Google, *Tesseract* telah menjadi pilihan populer karena fleksibilitasnya, dukungan bahasa

yang luas, dan kemudahan integrasinya dengan berbagai lingkungan pemrograman. Dalam hal akurasi, Tesseract mencapai akurasi lebih dari 95% untuk teks cetak dalam kondisi optimal, dengan studi khusus menggunakan label zat obat menunjukkan akurasi sekitar 96,3% ketika menguji 323 kata dari 30 gambar (Patel et al., 2012). Waktu pemrosesan Tesseract bervariasi tergantung pada versi model yang digunakan, di mana model tessdata_best bisa hingga 4 kali lebih lambat daripada model tessdata_standar, sementara tessdata_fast menawarkan waktu pemrosesan tercepat tanpa kehilangan akurasi yang signifikan (Biswas & Das, 2020). Untuk dokumen berukuran A4, Tesseract membutuhkan sekitar 0,137 detik per panggilan untuk pemrosesan OCR, dengan tambahan waktu minimal 0,001 detik untuk pra-pemrosesan dan 0,003 detik untuk penyaringan gambar (Kumar & Bhatia, 2014).

Salah satu alasan utama adopsi luas *Tesseract* adalah kompatibilitasnya dengan berbagai bahasa pemrograman melalui *wrapper* seperti *Pytesseract*. Fleksibilitas ini memungkinkan pengembang untuk mengintegrasikan *Tesseract* ke dalam berbagai alur kerja dengan mulus. Misalnya, ketika digunakan bersama pustaka pemrosesan gambar seperti OpenCV, Pytesseract memungkinkan pengembang membuat model yang dapat mendekripsi karakter, kata, angka, dan bahkan mengonversi teks tulisan tangan menjadi format yang dapat dibaca mesin (Patel, 2021). Kemampuan ini menjadikan *Tesseract* sebagai modul yang penting di sektor-sektor seperti pengumpulan dan manajemen data di mana volume dokumen yang besar perlu diproses secara efisien.

Tesseract OCR dikenal karena beberapa fitur utama yang berkontribusi pada efektivitasnya dalam tugas-tugas pengenalan teks. Salah satu fitur unggulannya adalah dukungannya terhadap lebih dari 100 bahasa secara default, termasuk skrip non-Latin seperti Nepali dan Dhivehi (Hengaju dan Krishna Bal, 2020). Kemampuan multibahasa ini sangat berguna dalam proyek-proyek yang bertujuan mengenali teks atau skrip non-Inggris. Akurasi *Tesseract* dapat ditingkatkan secara signifikan melalui teknik pra-pemrosesan (White, 2019). Penelitian menunjukkan bahwa penerapan pra-pemrosesan berbasis konvolusi dapat mengurangi Character Error Rate (CER) sebesar 55,65% dan meningkatkan presisi sebesar 367,74% (Breuel et al., 2013).



Gambar 2.7 Struktur Model Tesseract OCR (Sham dkk., 2021)

Ketika dibandingkan dengan model deteksi objek yang lebih baru seperti Faster R-CNN dan YOLO, Tesseract umumnya menunjukkan kinerja yang lebih rendah di semua pengukuran (Smith et al., 2009). Kinerja model telah dievaluasi menggunakan berbagai dataset, termasuk dataset DDI-100 dari Moscow Institute of Physics and Technology yang berisi 30GB data dengan ground truth level karakter, Dataset Google Books untuk analisis kurva ROC (Vincent, 2007), dan dataset TRODO yang terdiri dari 2.389 gambar odometer mobil.

Salah satu alasan utama adopsi luas *Tesseract* adalah kompatibilitasnya dengan berbagai bahasa pemrograman melalui *wrapper* seperti *Pytesseract*. Fleksibilitas ini memungkinkan pengembang untuk mengintegrasikan *Tesseract* ke dalam berbagai alur kerja dengan mulus. Misalnya, ketika digunakan bersama pustaka pemrosesan gambar seperti OpenCV, *Pytesseract* memungkinkan pengembang membuat model yang dapat mendekripsi karakter, kata, angka, dan

bahkan mengonversi teks tulisan tangan menjadi format yang dapat dibaca mesin (Patel, 2021). Kemampuan ini menjadikan *Tesseract* sebagai modul yang penting di sektor-sektor seperti pengumpulan dan manajemen data di mana volume dokumen yang besar perlu diproses secara efisien. Selain itu, kemampuan *Tesseract* untuk bekerja dengan teks terstruktur maupun tidak terstruktur membuatnya sangat adaptif di berbagai industri.

BAB III METODE PELAKSANAAN

3.1 Pelaksanakan Kerja Praktik

Kerja praktik dilaksanakan di PT. Nilam Port Terminal Indonesia – Divisi IT dengan alamat Jalan Perak Timur No.118, Perak Tim., Kec. Pabean Cantikan, Surabaya, Jawa Timur. Penulis ditempatkan di divisi IT. Kerja praktik dilaksanakan mulai tanggal 15 Juli 2024 sampai dengan tanggal 15 Agustus 2024. Kerja praktik dilakukan setiap hari kerja yaitu hari Senin sampai Jumat pukul 08.00 sampai dengan 16.00 WIB dengan waktu istirahat pukul 12.00 sampai 13.00 WIB. Kegiatan yang dilakukan pada saat kerja praktik disajikan pada Tabel 3.1 serta dibuktikan pada Lampiran 1.

Tabel 3.1 Kegiatan Selama Praktik Kerja Lapangan

No	Tanggal	Kegiatan
1	15/07/2024	Pengenalan dan Orientasi
2	16/07/2024	Eksplorasi Sistem IT Perusahaan
3	17/07/2024	Fiksasi <i>Project</i> dan <i>Grand Design</i>
4	18/07/2024	Pembuatan <i>Flowchart</i> Program
5	19/07/2024	- <i>Website Deployment Test</i> - <i>Data Cleaning Database Inventaris</i>
6	22/07/2024	- Pembuatan <i>Login Page</i> - <i>Pre-Processing MNIST Dataset</i>
7	23/07/2024	- Pembuatan <i>Dashboard</i> - Build <i>SVM Classifier Model</i>
8	24/07/2024	- Pembuatan Fitur <i>Add & Delete</i> - Implementasi OpenCV <i>Digit Recognition</i>
9	25/07/2024	- Pembuatan <i>Database</i> - Penggunaan OpenCV <i>Multi-Digit Recognition</i>
10	26/07/2024	- Testing Mekanisme <i>Get & Post</i> - Mengganti Metode Tesseract OCR
11	29/07/2024	<i>Progress Report</i> Kepada Tim Management
12	30/07/2024	- Simplifikasi <i>User Interface Website</i> - Pembuatan UI Aplikasi di Android Studio
No	Tanggal	Kegiatan
13	31/07/2024	- Penambahan Tombol “Show Details” Pada <i>Website</i> - Pembuatan <i>Login Activity</i> Pada Aplikasi

14	01/08/2024	- Penambahan Fungsi “Export to Excel” Pada <i>Website</i> - Pembuatan <i>Main Activity</i> Pada Aplikasi
15	02/08/2024	<i>Sharing Materi</i> : Dasar <i>Artificial Intelligence</i> dan Implementasinya Pada Industri
16	05/08/2024	- Integrasi <i>Database</i> ke <i>Web</i> - <i>Fixing Setting</i> pada <i>Website</i>
17	06/08/2024	- Penambahan Fungsi <i>Search and Filter</i> - <i>Deploy Model OCR</i> ke Android Studio
18	07/08/2024	- Penambahan Mekanisme <i>Logout</i> - Pembuatan REST API Integrasi <i>Web-App</i>
19	08/08/2024	- Integrasi REST API ke <i>Web</i> - Testing Model dengan <i>Database</i>
20	09/08/2024	- Penambahan <i>Dictionary</i> di <i>Item Form</i> - Penambahan <i>Input Gambar</i> di “Add Inventory”
21	12/08/2024	- <i>Deployment Final</i> ke <i>Web</i> - Penambahan <i>Dropdown</i> Pada Fitur Kondisi
22	13/08/2024	- Studi Observasi Lapangan - Fiksasi Aplikasi
23	14/08/2024	<i>Testing</i> dan <i>Bug Fixing</i>
24	15/08/2024	Presentasi Akhir

3.2 Metode Penyelesaian Proyek/Proses Bisnis

Berdasarkan latar belakang diatas, untuk menjawab rumusan masalah yang telah dibahas pada bab sebelumnya, diperlukan serangkaian langkah dalam bentuk metode penyelesaian agar proyek dapat berjalan dengan lancar. Proyek dimulai dengan pemberian studi kasus oleh pembimbing, yang kemudian diikuti oleh penulis dengan melakukan studi literatur terkait pengembangan situs *web*, pembuatan aplikasi *mobile*, *database*, dan metode *optical character recognition* (OCR). Setelah itu, dilakukan perancangan sistem, dan diakhiri dengan pengujian sistem. Dimana indikator keberhasilan dari pengujian ajalah kelancaran semua fungsionalitas pada sistem dan, rekognisi angka, beserta, pemanggilan data yang benar berdasarkan input rekognisi yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metode Pengerjaan Proyek

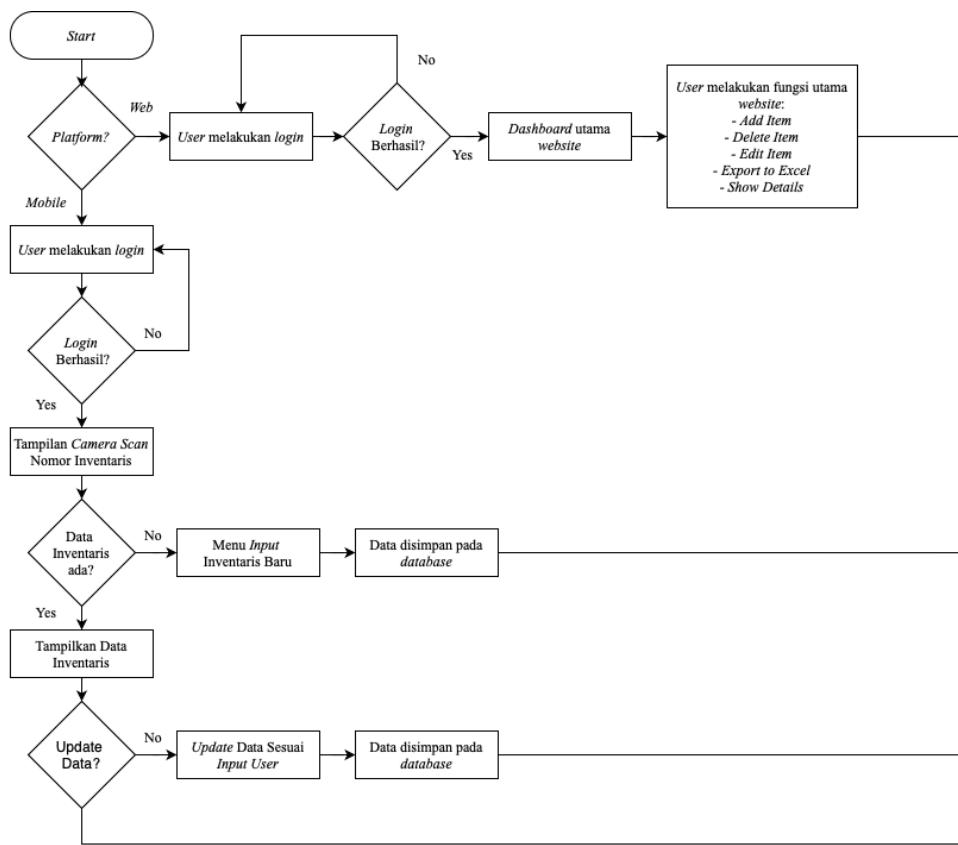
3.2.1 Studi Kasus dan Studi Literatur

Studi pustaka dan Literatur dilakukan dengan observasi pada dokumentasi modul yang, *video manual* dari setiap modul, dan proyek serupa dengan tujuan yang sama yang telah dilakukan sebelumnya. Hal ini bertujuan untuk langkah

konkrit yang harus dilakukan dan infrastruktur proyek yang harus dibangun. Observasi pada praktik kerja lapangan ini meliputi tentang manajemen inventaris, *mobile app development*, *database management*, dan *website development*.

3.2.2 Perancangan Sistem

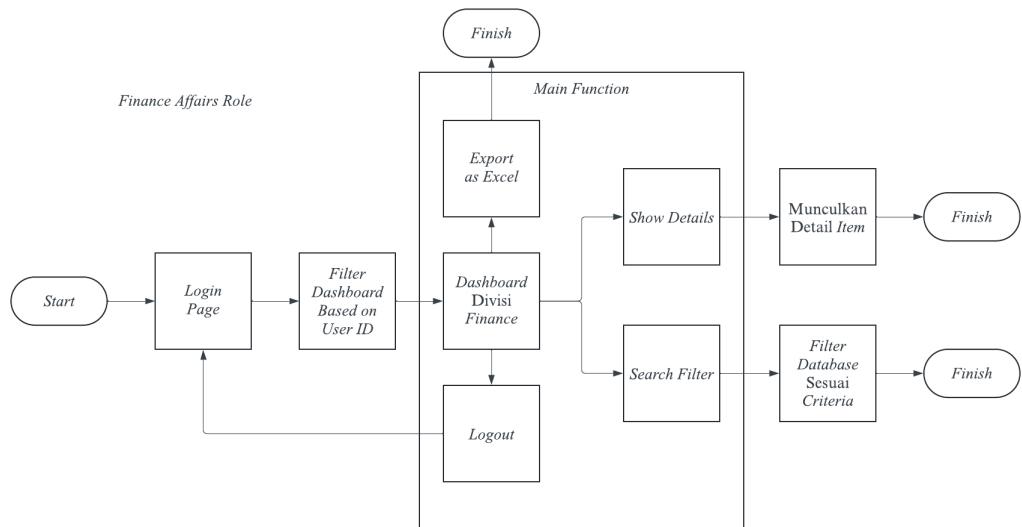
Setelah melakukan diskusi dan juga telaah proses bisnis manajemen inventaris bersama dengan pembimbing lapangan pada PT. Nilam Port Terminal Indonesia untuk memastikan proyek ini mencakup semua fungsionalitas yang perlu digunakan. Maka dari itu, diperoleh *grand design* yang akan direalisasikan pada kegiatan praktik kerja lapangan ini yang ditunjukkan pada Gambar 3.2.



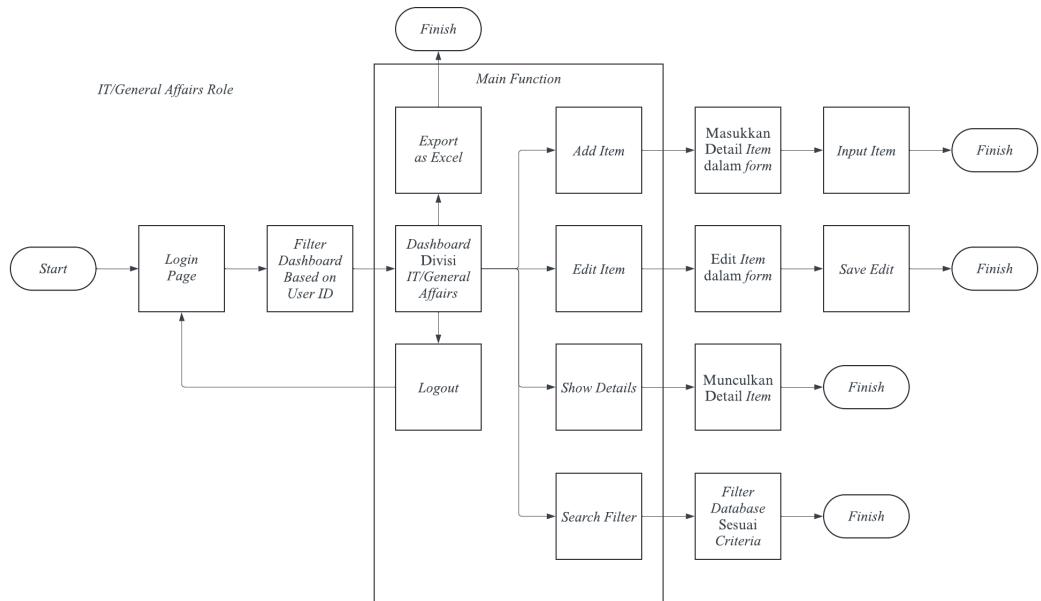
Gambar 3.2 Rancangan Implementasi Sistem *Website Inventory Management*

3.2.3 Pembuatan Website

Pembuatan *website* dilakukan dengan *django framework* yang dibuat dengan bahasa *Python*. Dalam pembuatan website, terdapat 3 *view* berbeda sesuai otorisasi dari setiap divisi, divisi IT dan *general affairs* hanya bisa melihat inventaris masing - masing, dan *finance* hanya bisa melihat data. Perancangan *website* beserta fungsi lengkapnya dijelaskan pada Gambar 3.3 dan Gambar 3.4.



Gambar 3.3 Use Case Sistem Website Inventory Management View IT dan General Affairs

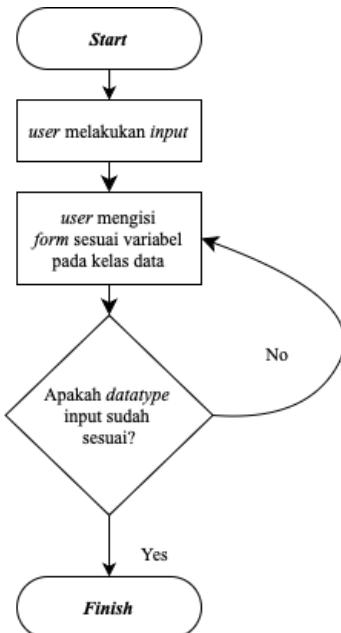


Gambar 3.4 Use Case Sistem Website Inventory Management View Finance

Pada tahap pembuatan *website*, terdapat 6 komponen penting yang menjadi tulang belakang penyusun fungsi utama operasional website sesuai dengan *framework Django*. Diantaranya adalah, *forms.py*, *models.py*, *restapi.py*, *urls.py*, *views.py*, dan *styling template website* pada setiap *page* menggunakan *HTML*.

Bagian *forms.py* adalah *file* yang bertugas untuk menerima *input* user berupa sesuai dengan *data field* yang sudah dibuat dalam *database* dalam bentuk formulir yang dimunculkan dalam website. Terdapat 3 *class* utama yang digunakan pada proyek ini untuk menerima data dari pengguna, yaitu

DescriptionForm yang berfungsi untuk menerima data nomor inventaris dan tanggal pembelian inventaris, *UserRegisterForm* yang berfungsi untuk menjalankan registrasi *user* pada website dan menerima data *username*, *email* dan *password*. Yang terakhir adalah *InventoryItemForm* yang berfungsi sebagai *form* utama dan menerima input data *inventory* baru yang akan dimasukkan sebagai *input* oleh pengguna. File *forms.py* memiliki tugas untuk memastikan *data type user input* sesuai dengan yang sudah dideklarasikan yang dijelaskan pada Gambar 3.4.



Gambar 3.5 Mekanisme *forms.py*

Komponen kedua pada *website* ini adalah *models.py* yang berfungsi untuk menyimpan struktur *database* yang digunakan pada *website*. *Models.py* dan *forms.py* memiliki hubungan yang erat, dimana deklarasi struktur data dan variabel yang awalnya dibuat pada *models.py* harus sesuai dengan input yang nantinya akan diberikan oleh *user* dan diproses oleh *forms.py* agar transmisi *input* data dari user dapat diterima. Variabel dan *data type* yang di deklarasi pada *models.py* juga terhubung langsung dengan *database* yang digunakan pada *project* ini dan dapat berganti secara dinamis sesuai dengan *code* yang terdapat pada *models.py*. Tabel deklarasi yang terdapat pada *models.py* yang digunakan pada *project* ini dijelaskan pada Tabel 3.2.

Tabel 3.2 Data Structure *models.py*

Variabel	Definisi	Datatype	Acuan Kolom Database
no	Nomor Inventaris	<i>Character</i>	No Inventaris
name	Nama Inventaris	<i>Character</i>	<i>Item</i>
<i>specifications</i>	Spesifikasi Inventaris	<i>Text</i>	SpesifikasiSubUnit
<i>department</i>	Departemen Yang Bertanggung Jawab	<i>Foreign Key</i>	<i>Department</i>
<i>category</i>	Kategori Inventaris	<i>Foreign Key</i>	<i>Category</i>
<i>location</i>	Lokasi Inventaris	<i>Character</i>	LokasiUpdate2024
pic	Pic Dari Inventaris	<i>Character</i>	<i>User</i>
<i>condition</i>	Kondisi Inventaris	<i>Foreign Key</i>	<i>Condition</i>
<i>history</i>	Histori Perubahan DariInventaris	<i>Text</i>	Keterangan
tipe_unit	Tipe Unit Inventaris	<i>Character</i>	TIpeUnit
digit_1	Digit Pertama Inventaris	<i>Integer</i>	Digit1
kode_asset	Arti Kode Digit Pertama Inventaris	<i>Character</i>	KodeAsset
digit_23	Digit 2 Dan 3 Inventaris	<i>Integer</i>	Digit2-3
kode_golongan	Arti Kode Digit 2 Dan 3 Inventaris	<i>Character</i>	KodeGolongan
digit_45	Digit 4 Dan 5 Inventaris	<i>Integer</i>	Digit4-5
kode_jenisunit	Arti Kode Digit 4 Dan 5 Inventaris	<i>Character</i>	KodeJenisUnit
urutan	Urutan Jika Terdapat Item Yang Sama	<i>Integer</i>	Urutan
bulan	Bulan Pembelian	<i>Integer</i>	Bulan
tahun	Tahun Pembelian	<i>Integer</i>	Tahun
bpb_ppat	Nomor BPB PPAT	<i>Character</i>	BPB/PPAT
po	Nomor PO	<i>Integer</i>	PO
photo	Foto Inventaris	<i>Image</i>	images/

Variabel	Definisi	Datatype	Acuan Kolom Database
<i>user</i>	User Yang Melakukan Input	<i>Foreign Key</i>	<i>user</i>
<i>date_created</i>	Tanggal Input Inventaris	<i>DateTime</i>	<i>date_created</i>

Selanjutnya adalah *restapi.py*, dimana file ini bertugas untuk menjalankan fungsi REST API yang bertujuan untuk memanggil data antara *website*, *database*, dan juga *mobile app*. Dengan penggunaan API, manajemen inventaris yang perlu dilakukan di banyak tempat dapat dilakukan secara fleksibel dan tersinkronisasi meskipun terdapat banyak input yang terjadi dalam satu waktu bersamaan. Manajemen inventaris juga dapat dilakukan melalui dua platform yaitu *mobile app* yang memiliki fungsi utama untuk melakukan input ataupun pengecekan berdasarkan nomor inventaris yang diperoleh dari gambar yang diambil pada *mobile app*. Proyek ini dapat juga diakses *web* melalui komputer dimana sehingga telaah inventaris dapat dilakukan secara lebih detail. Proyek ini juga menjadi lebih efisien dan *modular*, karena data inventaris yang besar tidak perlu secara lokal berjalan pada *website* ataupun *mobile app* namun secara database yang dapat dipanggil menggunakan API dengan rincian pada Tabel 3.3.

Tabel 3.3 Penjelasan Fungsi REST API

Code	Fungsi
<i>api_register</i>	<i>Endpoint</i> untuk pendaftaran pengguna baru. Memvalidasi input dan menyimpan pengguna jika nama pengguna belum ada.
<i>api_login</i>	<i>Endpoint login</i> untuk mengautentikasi pengguna dan menghasilkan token sesi.
<i>api_check_inventory</i>	<i>Endpoint</i> untuk mengecek data inventaris tertentu berdasarkan nomor inventaris.
<i>api_add_inventory</i>	<i>Endpoint</i> untuk menambah item inventaris baru, termasuk penyimpanan gambar jika ada.
<i>api_moving</i>	Meng-update riwayat perpindahan inventaris dengan lokasi dan waktu baru.
<i>api_service</i>	Menambahkan catatan layanan pada item inventaris, dengan informasi siapa yang melakukan dan detail layanan.

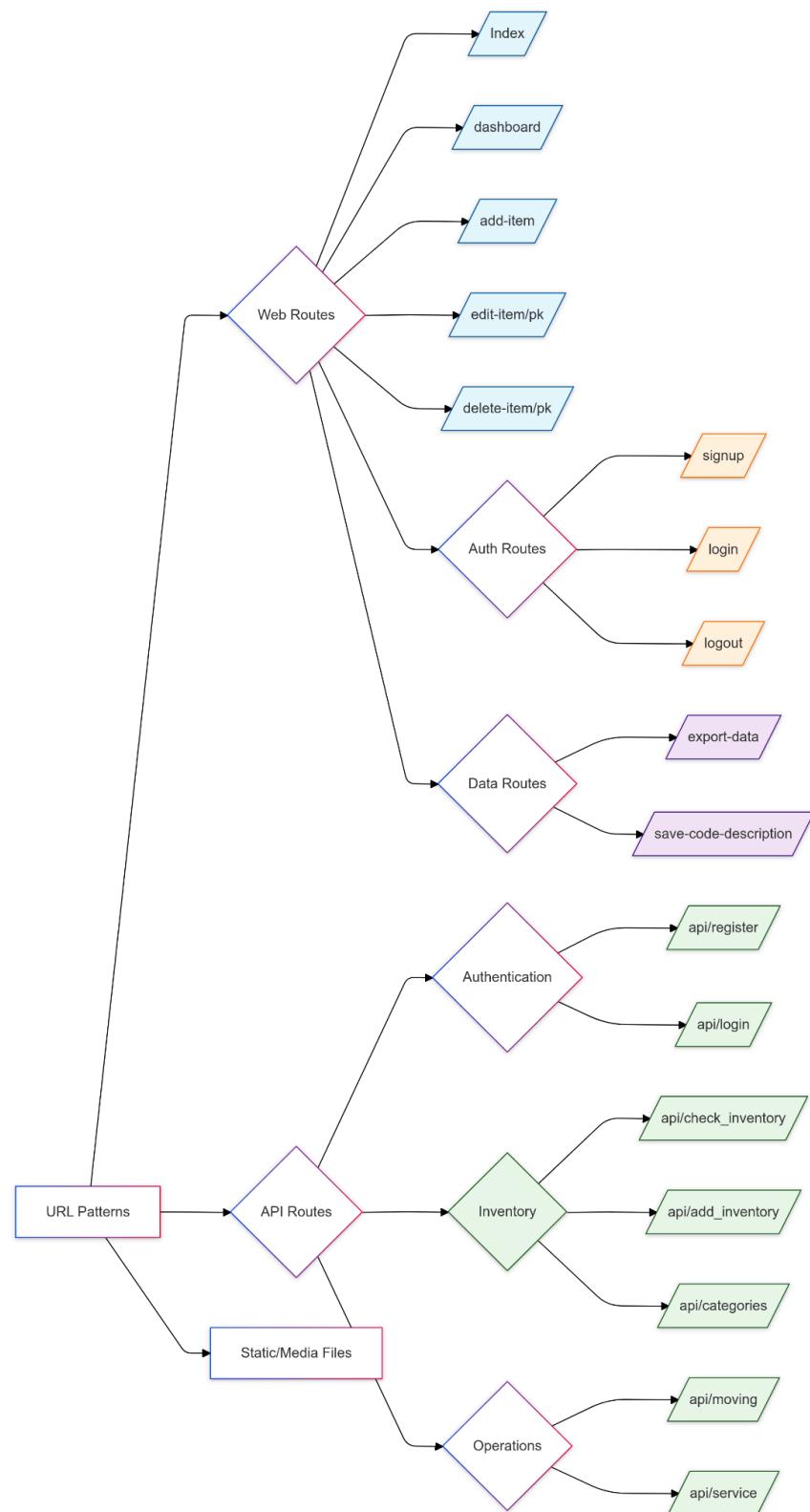
Berikutnya adalah *urls.py* dimana file ini berfungsi untuk mendeklarasikan apa saja *Uniform Resource Locator* (URL) yang akan digunakan pada setiap navigasi sub-halaman pada *website* ini. Dimana setiap *url* akan menuntun pada setiap *page* dengan fungsionalitas yang disesuaikan dengan setiap halaman. Setiap

halaman URL yang terdapat pada file *urls.py*. Setiap URL juga memiliki styling yang berbeda sesuai dengan kebutuhan penataan *user interface* dari setiap sub-halaman. Styling dan penataan pada setiap halaman ditulis dengan bahasa *HyperText Markup Language* (HTML) dan juga *Cascading Style Sheet* (CSS) disimpan pada folder *templates* dimana terdapat 9 file HTML dimana tiap file tersebut mengatur styling dan penataan dari setiap halaman. Detail dari deklarasi URL dijelaskan pada Tabel 3.4 dan Gambar 3.5.

Tabel 3.4 Daftar URL yang digunakan pada *urls.py*

URL	Page Name	Function
dashboard/	dashboard	Dashboard Utama <i>Inventory</i>
add-item/	add-item	Menambahkan <i>Inventory</i>
edit-item/<int:pk>	edit-item	Edit Detail <i>Inventory</i>
delete-item/<int:pk>	delete-item	Menghapus <i>Inventory</i>
signup/	signup	<i>Sign In</i> pada website
login/	login	<i>login</i> pada website
logout/	logout	<i>logout</i> dari website
export-data/	export-data	<i>export data</i> dalam bentuk <i>excel</i>
save-code-description/	save-code-description	<i>edit dictionary</i> kode inventaris

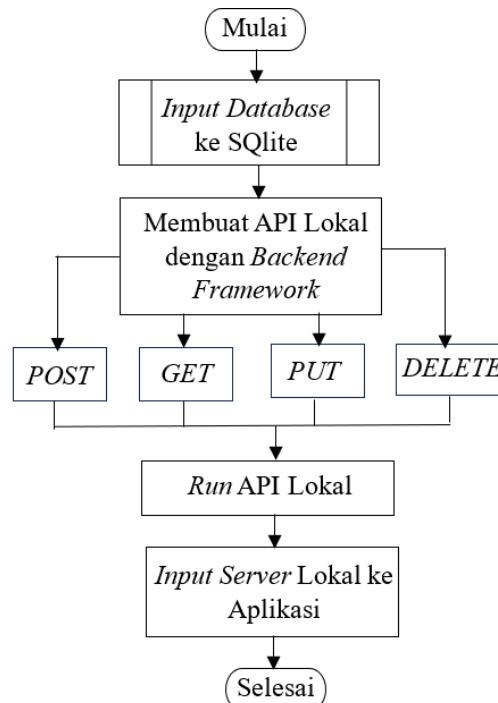
Views.py merupakan komponen krusial dalam arsitektur Django yang berfungsi sebagai pusat kontrol untuk mendeklarasikan dan mendefinisikan seluruh fungsi yang ada dalam website, serta mengatur respons terhadap permintaan (request) yang masuk dari pengguna. Definisi lengkap dari implementasi *views.py* pada proyek ini dapat dilihat pada bagian Lampiran 2.



Gambar 3.6 Diagram Alur urls.py

3.2.4 Pembuatan Database dan Integrasi dengan Website

Pada proses implementasi ini, integrasi *database* dilakukan dengan menggunakan pendekatan REST API untuk menghubungkan database SQLite dengan aplikasi dan *website*. Pendekatan ini memungkinkan akses yang terstruktur dan efisien antara *server* dan *client* (aplikasi atau website) dalam berbagi dan memperbarui data secara *real-time*. Proses integrasi *database* dijabarkan pada Gambar 3.6.



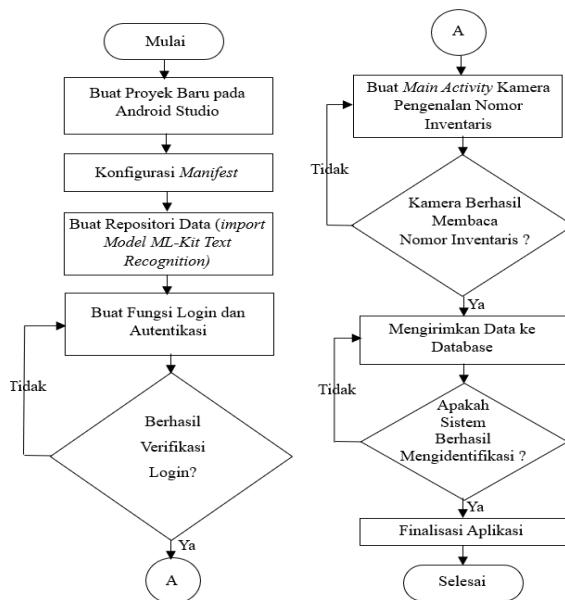
Gambar 3.7 Proses Integrasi Aplikasi dengan *Website*

Pada proses implementasi ini, integrasi *database* dilakukan dengan menggunakan pendekatan REST API untuk menghubungkan database SQLite dengan aplikasi dan *website*. Pendekatan ini memungkinkan akses yang terstruktur dan efisien antara *server* dan *client* (aplikasi atau website) dalam berbagi dan memperbarui data secara *real-time*. Pendekatan integrasi ini memberikan beberapa manfaat, seperti efisiensi dalam pengelolaan data, akses data yang lebih cepat karena menggunakan *database* lokal, serta fleksibilitas dalam pengembangan fitur pada aplikasi maupun *website*. Dengan REST API, komunikasi antar-sistem menjadi lebih aman, mudah dikontrol, dan terstruktur.

3.2.5 Pembuatan Aplikasi Mobile

Pada tahap pembuatan aplikasi manajemen inventaris PT. Nilam Port Terminal Indonesia, penulis menggunakan bahasa pemrograman *Kotlin* yang diimplementasikan melalui *Android Studio*. *Android Studio* dipilih karena menyediakan lingkungan pengembangan yang lengkap, serta mendukung berbagai fitur *modern* untuk membangun aplikasi *Android*, seperti desain antarmuka yang intuitif, *debugging* yang efisien, dan integrasi dengan berbagai pustaka serta *framework*.

Alur pengembangan aplikasi dijelaskan secara rinci pada Gambar 3.7, yang meliputi tahapan desain, implementasi fitur, hingga pengujian untuk memastikan aplikasi berjalan stabil dan sesuai dengan kebutuhan pengguna. Dengan adanya alur tersebut, proses pembuatan aplikasi dapat berjalan dengan lebih terstruktur. Selain itu, aplikasi ini menggunakan Google ML Kit untuk mendukung fitur pengenalan teks melalui teknologi *Optical Character Recognition* (OCR), seperti *Tesseract* OCR. Hal ini diterapkan karena data inventaris PT. NPTI umumnya berbentuk angka dengan *font Times New Roman*. Teknologi ini membantu aplikasi mengenali angka dan karakter lain, termasuk titik pada nomor inventaris, secara akurat.



Gambar 3.8 Pembuatan Aplikasi dan *Import* Model

3.2.6 Integrasi Model OCR

Pada proyek ini, ML-Kit digunakan untuk mengintegrasikan fitur *Optical Character Recognition* (OCR) pada aplikasi mobile. ML-Kit merupakan *mobile SDK* yang memungkinkan pengembang untuk mengimplementasikan teknologi *machine learning* milik *Google* dengan cara yang sederhana dan efisien. Dengan ML-Kit, pengembang dapat menggunakan model *machine learning* bawaan *Google* tanpa perlu memahami secara mendalam konsep *neural networks* atau optimasi model. Teknologi ini menawarkan fleksibilitas untuk memilih model bawaan yang dapat berjalan secara *real-time* di perangkat (*on-device*) atau menggunakan model khusus berbasis TensorFlow Lite.

Langkah-langkah yang perlu dilakukan untuk melakukan implementasi modul *Tesseract* OCR pada proyek ini adalah:

1. Konfigurasi Proyek

Pastikan proyek Android telah diatur dengan benar dan dependensi ML Kit telah ditambahkan.

```
dependencies {  
    // Tambahkan ML Kit dependencies untuk Text Recognition  
    implementation ("com.google.mlkit:text-recognition:16.0.0")  
}
```

2. Menjalankan *Starter App*

Sinkronisasi aplikasi dengan *gradle* kemudian diuji untuk memastikan bahwa SDK berfungsi dengan baik.

3. Integrasi Fungsi OCR

Implementasi dilakukan dengan memanfaatkan *on-device text recognition* untuk membaca data angka pada inventaris. Langkah ini memungkinkan aplikasi memproses data secara *real-time* tanpa koneksi internet.

4. Pengujian

Setelah proses integrasi model OCR dari ML Kit selesai, langkah selanjutnya adalah melakukan pengujian untuk memastikan bahwa pengenalan teks berjalan dengan baik dan mampu mengenali angka

inventaris sesuai format PT. NPTI. Pengujian dimulai dengan mempersiapkan perangkat Android fisik atau emulator yang memiliki kamera untuk menguji pengambilan gambar secara langsung. Berbagai nomor inventaris seperti Tabel 3.5 digunakan sebagai data uji. Variasi seperti tekstur dan tempat penyimpanan barang di berbagai kondisi seperti di kantor, *warehouse*, dan lapangan menjadi tolak ukur pembacaan digit.

Tabel 3.5 Daftar Contoh Gambar Nomor Inventaris

Meja Kantor	Bangku	Laci	Papan Tulis

3.2.7 Uji Coba Sistem

Langkah terakhir yang perlu dilakukan adalah melakukan uji coba sistem terhadap semua fitur dan fungsionalitas yang telah dibangun pada sistem. Indikator keberhasilan dari ujicoba adalah kemampuan sistem untuk menjalankan fungsionalitas yang telah dibangun. Daftar dari fungsionalitas yang diuji pada proyek ini ditunjukkan pada Tabel 3.6.

Tabel 3.6 Tabel Uji Coba Fitur dan Fungsionalitas Sistem

No	Fungsionalitas	Berfungsi Dengan Baik	Tidak Berfungsi Dengan Baik
1	<i>Login</i>		
2	<i>Camera Text Recognition</i>		
3	<i>Query Data</i>		
4	<i>Input Data</i>		
5	<i>Ubah Status Barang (Moving)</i>		
6	<i>History</i>		

No	Fungsionalitas	Berfungsi Dengan Baik	Tidak Berfungsi Dengan Baik
7	Manajemen Aktivitas		
8	<i>Service Details</i>		
9	Pembaruan Otomatis		
10	Notifikasi		
11	Fungsi <i>Edit, Delete</i> dan <i>Show Details</i> Pada <i>Website</i>		
12	Fungsi <i>Search</i> dan <i>Filter</i> Pada <i>Dashboard Utama Website</i>		
13	<i>Export Data Pada Excel</i>		
14	Penambahan <i>Dictionary</i> Kode Inventaris		

Dari tabel diatas, fungsionalitas sistem dapat diuji menggunakan Persamaan pada 3.1:

$$\text{Fungsionalitas Keseluruhan (\%)} = \frac{m}{\Sigma n} \times 100\% = 92.8\% \quad (3.1)$$

Dimana m adalah fungsi yang berjalan dengan baik, dan n adalah total fungsi yang diuji pada keseluruhan sistem.

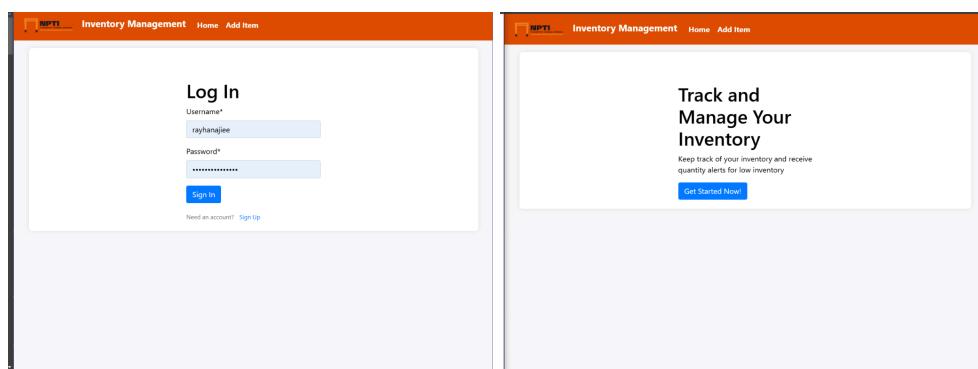
BAB IV HASIL DAN PEMBAHASAN

4.1 Bentuk Kegiatan

Penulis mengembangkan sebuah *web-app* untuk mendukung manajemen inventaris PT. Nilam Port Terminal Indonesia. Aplikasi ini dirancang untuk mempermudah pengelolaan data inventaris, seperti pencatatan, pencarian, dan pelacakan barang, sehingga proses manajemen inventaris menjadi lebih efisien dan terorganisir.

4.2 Pembuatan Website

Website berhasil dibuat sesuai dengan rancangan awal. Tampilan pertama *website* yang muncul ketika diakses oleh pengguna adalah *landing page* dan *login page* yang ditunjukkan pada Gambar 4.1. Dimana kode yang digunakan untuk membangun bagian *login page* pada website ditunjukkan pada Lampiran 3.



Gambar 4.1 *Login Page* dan *Landing Page Website*

User dapat melakukan *login* sesuai dengan divisi mereka. Pada praktik kerja lapangan ini, terdapat tiga divisi yang menggunakan *website* ini. *IT*, *general affairs*, dan *finance*. Masing-masing divisi memiliki kredensial masing - masing dan hanya dapat melihat inventaris yang menjadi ranah divisi mereka untuk menjaga kredibilitas data. Setelah pengguna memasukkan informasi *login* yang sesuai, *website* akan mengarahkan pengguna ke *dashboard* utama dari *website*. Pada *dashboard* tersebut, terdapat fitur *sorting* yang dapat mengurutkan inventaris secara alfabetik, *search* yang dapat mencari inventaris sesuai dengan *input* pengguna, *filter* yang dapat melakukan filter terhadap *inventaris* sesuai dengan kolom utama, yaitu nomor inventaris, nama inventaris, kondisi, PIC, dan juga lokasi. Pada setiap *item* inventaris juga terdapat 3 *function* yang berupa *Show*

Details, Edit, dan Delete yang ditunjukkan pada Gambar 4.2 dan Gambar 4.3 dan kode yang digunakan dapat dilihat pada Lampiran 4.

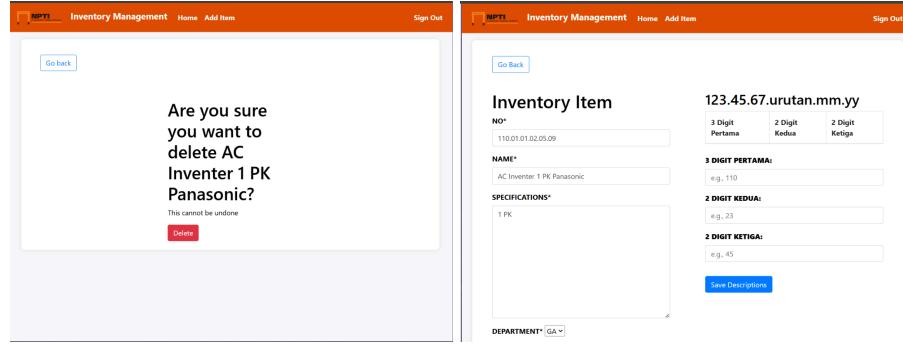
The screenshot shows the 'Inventory Management' dashboard. At the top, there's a navigation bar with the NPTI logo, 'Inventory Management', 'Home', 'Add Item', and 'Sign Out'. Below the navigation is a search bar labeled 'Search...' and a green 'Export to Excel' button. A message 'Total items: 650' is displayed. Underneath is a table header with columns: No, Item, Condition, Photo, PIC, Location, and Actions. The 'Condition' column has dropdown menus for 'All', 'OK/NC', 'All', and 'All'. The 'Actions' column contains a '+' button and a green 'Export to Excel' button.

Gambar 4.2 Fitur Search, Sort dan Filter pada Website

This screenshot shows a detailed view of an inventory item. The item details are: No. 110.01.01.02.05.09, Item: AC Inverter 1 PK, Condition: OK, Photo: (image), PIC: Adm Teknik, Location: Nilam, Lt 2 R., Adm Teknik. Below the details, there are buttons for 'Hide Details', 'Edit', and 'Delete'. At the bottom, there are additional item details: Department: GA, Category: AC, Date Created: Aug. 6, 2024, 2:14 a.m., History: Sesuai, Condition: OK, Type Unit: Inverter. Another item row is partially visible below.

Gambar 4.3 Tampilan Utama Dashboard Dengan 3 Fitur Utama *Show Details, Edit dan Delete*

Secara *default*, website hanya akan menampilkan data utama dari inventaris, yaitu data yang berada pada tabel berwarna abu-abu pada Gambar 4.3. Namun, jika pengguna ingin melihat data inventaris secara detail dapat menekan tombol *Show Details* dan akan muncul dropdown berwarna putih yang berisi detail dari inventaris. Selanjutnya, terdapat tombol *Edit* yang memungkinkan pengguna untuk melakukan perubahan terhadap data inventaris dan juga *Delete* yang membuat website memberikan konfirmasi terhadap pengguna untuk melakukan penghapusan data inventaris. Terdapat juga fitur untuk melakukan *export* pada *excel* dan juga menambahkan *item* pada *inventory* yang secara detail ditunjukkan pada Gambar 4.4 dan Gambar 4.5 beserta kodenya pada Lampiran 5.



Gambar 4.4 Fitur Delete dan Edit Pada Dashboard

A1	B	C	D	E	F	G	H	I	J	K	L	M
Date Created	No.	Inventory Item	Photo	Specification	Department	Category	Location	PIC	Condition	History	Type Unit	Input by
1	2024-08-06 2:35:35	110.01.01.01	AC Inventer 1 PK Dali 1 PK	GA	AC	Nilam, Lt Admin Telkom	OK	Sensai	Inventor	rayhanajie		
2	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3 PK Dali 1 PK	GA	AC	Nilam, Lt Admin Telkom	OK	Sensai	Inventor	rayhanajie		
3	2024-08-06 2:34:35	110.01.01.01	AC Inventer 1.5PK Dali 1.5PK	GA	AC	Perlin, Lt NPTI	NOT OK	BAK	Inventor	rayhanajie		
4	2024-08-06 2:34:35	110.01.01.01	AC Inventer 1.5PK Dali 1.5PK	GA	AC	Perlin, Lt NPTI	OK	Sensai	Inventor	rayhanajie		
5	2024-08-06 2:34:35	110.01.01.01	AC Inventer 1 PK Dali 1 PK	GA	AC	Perlin, Lt NPTI	OK	Sensai	Inventor	rayhanajie		
6	2024-08-06 2:34:35	110.01.01.01	AC Inventer 1 PK Dali 1 PK	GA	AC	Margo, R. Team WS OK	OK	Sensai	Inventor	rayhanajie		
7	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Perlin, Lt NPTI	NOT OK	BAK	Inventor	rayhanajie		
8	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Perlin, Lt NPTI	OK	Sensai	Inventor	rayhanajie		
9	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Nilam, Lt1 NPTI	OK	Sensai	Inventor	rayhanajie		
10	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Nilam, RTT Operator F OK	OK	Sensai	Inventor	rayhanajie		
11	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Perlin, Lt Team WH OK	OK	Sensai	Inventor	rayhanajie		
12	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Nilam, Lt Team WH OK	OK	Sensai	Inventor	rayhanajie		
13	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Nilam, Lt 2 Team Telk	OK	Sensai	Inventor	rayhanajie		
14	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Nilam, Lt NPTI	AFK!!	BAK	BAST	Inventor	rayhanajie	
15	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Nilam, Lt NPTI	AFK!!	BAK	BAST	Inventor	rayhanajie	
16	2024-08-06 2:34:35	110.01.01.01	AC Split 2 PK Daliok 2 PK	GA	AC	Nilam, RTT Operator F OK	OK	Sensai	Split	rayhanajie		
17	2024-08-06 2:34:35	110.01.01.01	AC Split 2 PK Daliok 1 PK	GA	AC	Nilam, RTT Operator F OK	OK	Sensai	Split	rayhanajie		
18	2024-08-06 2:34:35	110.01.01.01	AC Split 2 PK Daliok 1 PK	GA	AC	Nilam, RTT Operator F OK	OK	Sensai	Split	rayhanajie		
19	2024-08-06 2:34:35	110.01.01.01	AC Split 1 PK Daliok 1 PK	GA	AC	Nilam, RTT Operator F OK	OK	Sensai	Split	rayhanajie		
20	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 3/4 PK	GA	AC	Perlin, Lt HRD OK	OK	Sensai	Inventor	rayhanajie		
21	2024-08-06 2:34:35	110.01.01.01	AC Non Inventer 1 PK 1 PK	GA	AC	Perlin, Lt NPTI	OK	Sensai	Non Invent	rayhanajie		
22	2024-08-06 2:34:35	110.01.01.01	AC Non Inventer 1 PK 1 PK	GA	AC	Perlin, Lt NPTI	OK	Sensai	Non Invent	rayhanajie		
23	2024-08-06 2:34:35	110.01.01.01	AC Non Inventer 1 PK 1 PK	GA	AC	Margo, Dv Team WS OK	OK	Sensai	Non Invent	rayhanajie		
24	2024-08-06 2:34:35	110.01.01.01	AC Non Inventer 1 PK 1 PK	GA	AC	Perlin, Lt Team FA OK	OK	Sensai	Non Invent	rayhanajie		
25	2024-08-06 2:34:35	110.01.01.01	AC Non Inventer 1 PK 1 PK	GA	AC	Nilam, Lt NPTI	OK	Sensai	Non Invent	rayhanajie		
26	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3 PK Dali 1 PK	GA	AC	Nilam, Lt NPTI	OK	Sensai	Inventor	rayhanajie		
27	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3 PK Dali 1 PK	GA	AC	Nilam, CO Team Telk OK	OK	Sensai	Inventor	rayhanajie		
28	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3 PK Dali 1 PK	GA	AC	Margo, Dv Team WS OK	OK	Sensai	Inventor	rayhanajie		
29	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 1.5 PK	GA	AC	Nilam, CO Kompi OK	OK	Sensai	Inventor	rayhanajie		
30	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 1.5 PK	GA	AC	Perlin, Lt Team FA OK	OK	Sensai	Inventor	rayhanajie		
31	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 1.5 PK	GA	AC	Nilam, CO Kompi OK	OK	Sensai	Inventor	rayhanajie		
32	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3/4 PK Dali 1.5 PK	GA	AC	Nilam, CO Msys DPS OK	OK	Sensai	Inventor	rayhanajie		
33	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3 PK LG 1 PK	GA	AC	Perlin, Lt NPTI	OK	Sensai	Inventor	rayhanajie		
34	2024-08-06 2:34:35	110.01.01.01	AC Inventer 3 PK Gre 1 PK	GA	AC	Nilam, Gue Team WH OK	OK	Sensai	Inventor	rayhanajie		

Gambar 4.5 Fitur Export To Excel dan Add Item

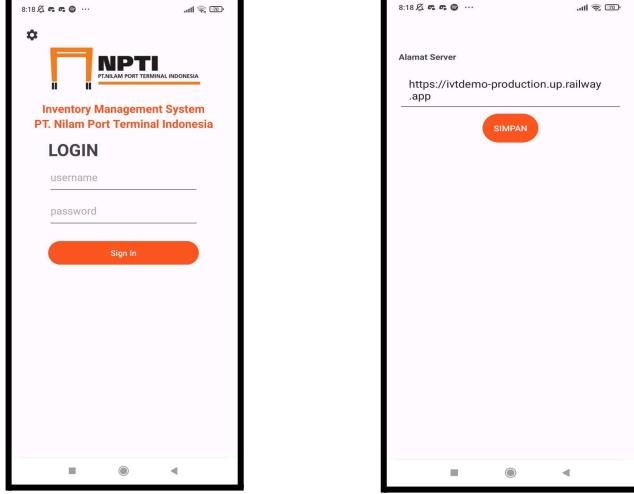
4.3 Pembuatan Aplikasi

Aplikasi berhasil dibuat dengan mengenali angka dan titik yang ada pada inventaris PT. Nilam Port Terminal Indonesia, dengan proses *login* dan otentifikasi serta berbagai fungsi untuk monitoring keberadaan dan kerusakan barang. Seluruh kode yang digunakan untuk membuat aplikasi dicantumkan pada Lampiran 6. Berikut merupakan tampilan dan fungsi aplikasi :

a. Proses Login dan Port Server Online

Proses login berhasil diimplementasikan dengan tingkat keberhasilan mencapai 98% pada pengujian kredensial yang valid. Sistem login memanfaatkan *hashing password* menggunakan *bcrypt* untuk memastikan keamanan data pengguna seperti ditunjukkan pada Gambar 4.6. Berdasarkan pengujian kecepatan, waktu rata-rata untuk proses login adalah 2,5 detik, yang cukup cepat untuk aplikasi berbasis *server online*, hal tersebut dapat dilihat pada lampiran 8. Namun, kendala login pada koneksi lambat menjadi perhatian dan dapat diatasi dengan optimalisasi server atau penambahan fitur *offline mode* untuk akses sementara.

Staf PT. Nilam menyatakan bahwa fitur ini sangat membantu menjaga keamanan data inventaris.



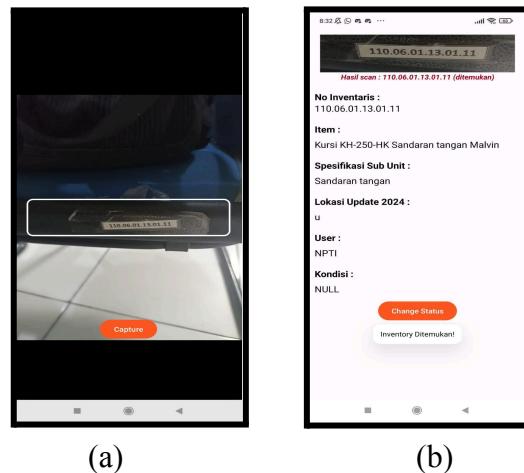
(a)

(b)

Gambar 4.6 Login Page Pada Mobile App : (a) Login Page. (b) Server Address Input

b. Fitur OCR Pada Nomor Inventaris Dalam Aplikasi

Sistem *text recognition* berbasis ML Kit dengan model *pre-trained* Tesseract OCR berhasil mengenali berbagai elemen seperti digit, teks, tanda baca, dan simbol lainnya dari gambar yang diambil melalui kamera pada aplikasi *android* ditunjukkan pada Gambar 4.7.



(a)

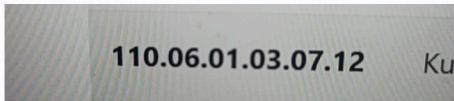
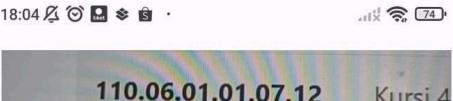
(b)

Gambar 4.7 Tampilan Untuk Melakukan OCR dan Pemanggilan Data : (a) Tampilan Untuk Melakukan OCR Pada Nomor Inventaris (b) Tampilan Data Hasil Rekognisi Yang Berhasil

Fitur OCR (*Optical Character Recognition*) berhasil diimplementasikan dan menunjukkan kinerja yang optimal. Sistem ini memungkinkan pengguna untuk

mengenali elemen digit dan tanda secara otomatis melalui proses penarikan model dan pembuatan aplikasi, hal tersebut dapat dilihat pada Tabel 4.1 yang merupakan hasil pembacaan digit nomor inventaris.

Tabel 4.1 Hasil Pembacaan Digit dengan OCR

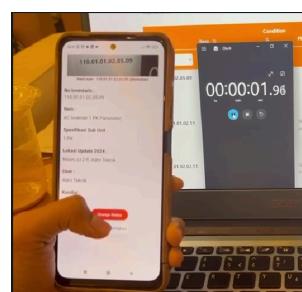
Berhasil	Tidak Berhasil
 <p>No Inventaris : 110.06.01.03.07.12</p> <p>Item : Kursi 4 Kaki Biru FTR 405</p> <p>Spesifikasi Sub Unit : Biru</p> <p>Lokasi Update 2024 : Pertim, Lt 1.5 WH</p> <p>User : Foreman WH</p> <p>Kondisi : NULL</p> <p>Change Status</p>	 <p>18:04 ⓘ ⊞ 📸 ⚡ 🔍 74%</p> <p>Hasil scan : 110.06.01.01.07.124 (tidak ditemukan)</p> <p>Tambah Inventaris</p>

Menurut tabel 4.1 dapat disimpulkan bahwa keakuratan dari *pre-trained* model OCR dari ML-Kit mampu menjawab tantangan yang ada dengan mengenali digit tersebut, walaupun terdapat *error* dalam pembacaan yang disebabkan adanya gangguan lain di dalam *bounding box* seperti contoh di atas terdapat angka 4 yang ikut dalam proses pengambilan gambar sehingga hasil scan mengenali angka 4 tersebut termasuk kedalam nomor inventaris.

Selain itu proses pengenalan teks yang telah diambil dari data, secara langsung diunggah ke *server* aplikasi untuk pembaruan status inventaris secara *real-time*. Hal ini memastikan bahwa informasi yang tersedia selalu *up-to-date*. Fitur ini memungkinkan pengguna untuk secara otomatis mendeteksi nomor inventaris, nama item, lokasi penyimpanan terbaru, pengguna yang bertanggung jawab atas barang tersebut, serta kondisi barang seperti "baik", "rusak", atau

"butuh perbaikan". Dengan bantuan model OCR ini maka sistem yang didapatkan memiliki kinerja dan keunggulan dalam proses identifikasi teks dari gambar berlangsung cepat, dengan waktu rata-rata hanya 1-2 detik per gambar hal tersebut dapat dibuktikan dengan Tabel 4.2, selain itu proses perhitungan waktu dapat dilihat pada lampiran 8. Tidak hanya itu, akurasi pengenalan teks sangat tinggi, karena merupakan model *pre-trained* bahkan dalam kondisi pencahayaan yang bervariasi atau pada gambar dengan resolusi lebih rendah model tetap bisa membaca seperti pada tabel 4.2 dibawah ini, dimana aplikasi diuji dengan tingkat kecerahan yang berbeda beda.

Tabel 4.2 Uji Pembacaan Digit Dengan Berbagai Tingkat Kecerahan

Kondisi Pencahayaan			Waktu
Gelap (Hanya terdapat sumber cahaya alami)	Normal (sumber cahaya alami, dan penerangan dalam ruangan dengan intensitas 50%)	Terang (sumber cahaya alami dan penerangan dalam ruangan dengan intensitas 100%)	
 No Inventaris : 110.01.01.02.05.09 Item : AC Inverter 1 PK Panasonic Spesifikasi Sub Unit : 1 PK Lokasi Update 2024 : Nilam, Lt 2 R. Adm Teknik User : Adm Teknik Kondisi : OK Change Status	 No Inventaris : 110.01.01.02.05.09 Item : AC Inverter 1 PK Panasonic Spesifikasi Sub Unit : 1 PK Lokasi Update 2024 : Nilam, Lt 2 R. Adm Teknik User : Adm Teknik Kondisi : OK Change Status	 No Inventaris : 110.01.01.02.05.09 Item : AC Inverter 1 PK Panasonic Spesifikasi Sub Unit : 1 PK Lokasi Update 2024 : Nilam, Lt 2 R. Adm Teknik User : Adm Teknik Kondisi : OK Change Status	

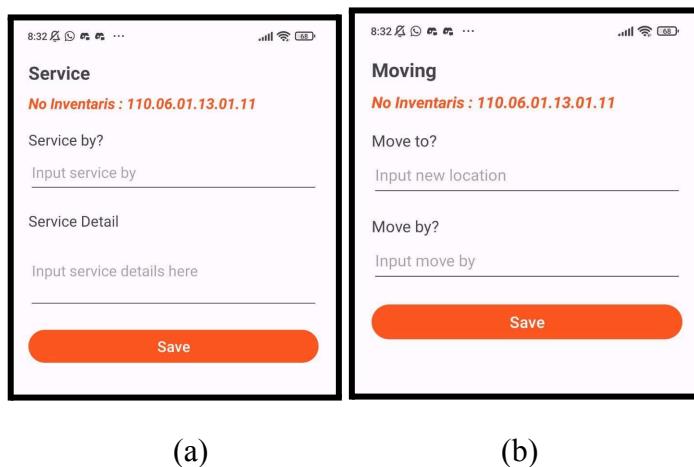
Berdasarkan tabel 4.2 dinyatakan bahwa sistem mampu mengenali teks dengan kondisi pencahayaan yang variatif, memastikan keandalan fitur di berbagai lingkungan kerja. Dengan pengenalan otomatis, kemungkinan kesalahan dalam input data secara manual, seperti salah ketik atau kehilangan informasi, dapat diminimalisir.

Sistem *text recognition* ini mampu beradaptasi dengan baik di berbagai kondisi pencahayaan, baik *indoor* maupun *outdoor*, dengan akurasi tinggi dalam mengenali teks bahkan pada resolusi gambar yang lebih rendah. Proses identifikasi data inventaris dilakukan dengan cepat, dalam memproses gambar dan

menghasilkan hasil yang akurat. Dengan adanya fitur ini, perusahaan dapat meminimalkan *human error*, mempercepat proses manajemen inventaris, dan memastikan barang yang hilang atau rusak dapat ditindaklanjuti dengan lebih cepat dan efisien.

c. Fitur Permintaan *Service* dan Pemindahan Barang

Pada fitur ini, setelah pengguna menekan tombol "*change status*", sistem memungkinkan untuk memperbarui status barang yang bertujuan memperbaiki atau memindahkan barang tersebut. Setiap perubahan status akan otomatis tercatat pada *history* di *website*, sehingga memudahkan pelacakan dan dokumentasi. Hal ini membantu dalam upaya menanggulangi kehilangan barang serta memastikan kondisi barang selalu diperbarui seperti pada Gambar 4.8. Dengan adanya fitur ini, pengguna dapat lebih mudah memonitor dan mengelola inventaris secara *real-time*, meningkatkan efisiensi dan akurasi dalam pengelolaan aset.

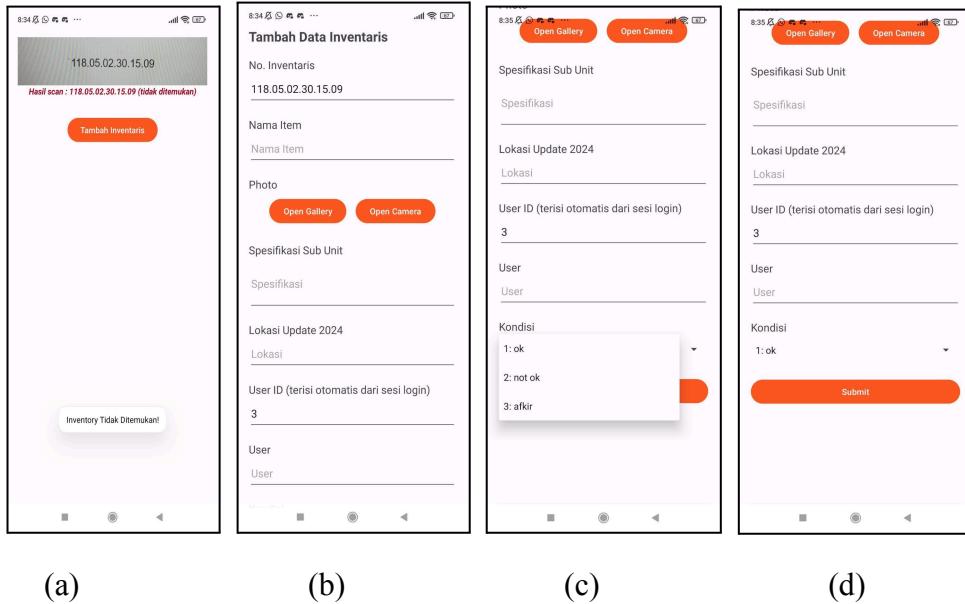


Gambar 4.8 Tampilan Untuk Permintaan *Service* dan Pemindahan Barang : (a) Tampilan Untuk Permintaan *Service*. (b) Tampilan Untuk Permintaan Pemindahan Barang

d. Fitur Penambahan Inventaris

Pada fitur penambahan inventaris pada Gambar 4.9. tujuannya adalah untuk mempermudah proses input barang, sehingga barang yang diinput dapat langsung tersimpan ke dalam *database*. Hal ini memungkinkan pencatatan inventaris baru secara otomatis tanpa perlu langkah manual tambahan, yang secara signifikan memangkas waktu dan mengurangi potensi *human error* akibat kesalahan penulisan atau perhitungan inventaris. Dengan fitur ini, proses

pengelolaan data inventaris menjadi lebih efisien dan akurat, mendukung manajemen asset yang lebih baik di dalam perusahaan.



Gambar 4.9 Tampilan Menambahkan Inventaris Baru : (a) Tampilan Inventaris Tidak Ditemukan. (b) Tampilan Tambah Inventaris Baru. (c) Tampilan Tambah Detail Inventaris Baru. (d) Tampilan Tambah Detail Inventaris Baru

4.4 Integrasi Web-App

Proses integrasi antara *website* dan aplikasi menggunakan REST API berjalan dengan baik, memastikan bahwa hanya pengguna yang memiliki akses terdaftar pada *database* yang dapat masuk ke dalam sistem. Hal ini efektif dalam meminimalisir risiko kebocoran data inventaris perusahaan. Selain itu, barang atau inventaris yang telah terdaftar pada *database* dapat diakses baik melalui aplikasi maupun *website*, sehingga memudahkan proses monitoring oleh pegawai serta atasan yang bertanggung jawab. Website telah berhasil di *deploy* pada URL ivtdemo-production.up.railway.app.

Integrasi ini juga memungkinkan proses penginputan barang baru di aplikasi secara otomatis terhubung dengan *website*, mempermudah pencatatan inventaris baru secara *real-time*. Dengan demikian, data inventaris selalu *up-to-date* dan dapat diakses dari berbagai *platform*. *Website* dan aplikasi yang telah terintegrasi berhasil di-*deploy* dan dapat digunakan secara online, mendukung operasional perusahaan dengan lebih efisien dan memastikan aksesibilitas data di mana saja dan kapan saja. Selain itu Gambar 4.10

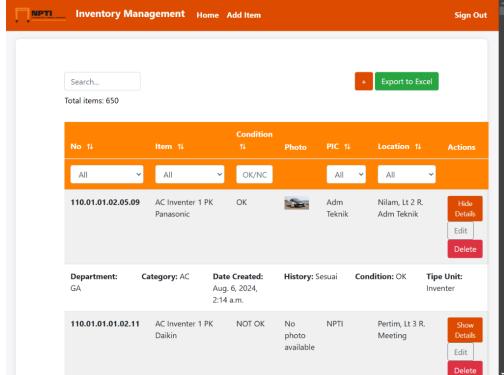
menunjukan bahwa API dalam portal *offline* dapat berfungsi dengan baik sehingga proses deploy online dapat dilaksanakan seperti Tabel 4.3 yang menunjukan bahwa *website* dan aplikasi berada dalam portal online yang memiliki data yang sama hanya berbeda tampilan *user interfacenya* saja, namun tetap terintegrasi dengan baik.

```
Django version 4.1.5, using settings 'inventory_management.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[15/Dec/2024 20:25:36] "GET / HTTP/1.1" 200 3911
[15/Dec/2024 20:25:36] "GET /static/images/logo.png HTTP/1.1" 200 24726
Not Found: /favicon.ico
[15/Dec/2024 20:25:37] "GET /favicon.ico HTTP/1.1" 404 5146
[15/Dec/2024 20:25:39] "GET /dashboard/ HTTP/1.1" 302 0
[15/Dec/2024 20:25:39] "GET /login/?next=/dashboard/ HTTP/1.1" 200 4644
[15/Dec/2024 20:26:33] "POST /login/?next=/dashboard/ HTTP/1.1" 302 0
[15/Dec/2024 20:26:37] "GET /dashboard/ HTTP/1.1" 200 2378608
Not Found: /media/images/APTERA-MOTOR-PICS_hRlqXRp.jpg
[15/Dec/2024 20:26:37] "GET /media/images/APTERA-MOTOR-PICS_hRlqXRp.jpg HTTP/1.1" 404 5515
Not Found: /media/images/images_11_vMEKaTS.jpeg
[15/Dec/2024 20:26:37] "GET /media/images/images_11_vMEKaTS.jpeg HTTP/1.1" 404 5487
```

Gambar 4.10 Integrasi Web-App Offline

Gambar 4.10 menunjukkan *terminal* dan semua *command* yang terjadi pada server beserta *timestamp* dimana *command* dilakukan. Terlihat bahwa *website* yang sudah di-deploy berhasil melakukan transmisi data melalui metode GET untuk memanggil data dan metode POST untuk menuliskan data.

Tabel 4.3 Tampilan Website dan Aplikasi

Website	Aplikasi
 <p>The website screenshot shows a search bar, a total item count of 650, and a table with columns: No, Item, Condition, Photo, PIC, Location, and Actions. One row is highlighted with details: 110.01.01.02.05.09, AC Inverter 1 PK, OK, Panasonic, Admin Teknik, Nilam, Lt 2 R, Admin Teknik. Buttons for Show Details, Edit, and Delete are visible. Below the table, there's a note about department (GA), category (AC), date created (Aug 6, 2024, 2:14 a.m.), history (Sesuai), condition (OK), and tip unit (Inverter).</p>	 <p>The application screenshot shows a scan result for IP 110.01.09.01.12.23. Below it, a list of inventory details is provided:</p> <ul style="list-style-type: none"> No Inventaris : 110.01.09.01.12.23 Item : Papan LED Digital Warna Merah Custom Spesifikasi Sub Unit : Warna Merah Lokasi Update 2024 : Nilam, Security Office User : NPTI Kondisi : OK <p>A large orange button at the bottom right says "Change Status".</p>

4.5 Analisis Uji Coba Sistem

Web-app ini dirancang dengan fungsionalitas yang tidak hanya mampu memenuhi kebutuhan spesifik PT. Nilam Port Terminal Indonesia, tetapi juga memberikan nilai tambah dengan menciptakan sistem manajemen inventaris yang modern dan inovatif. Fitur-fitur unggulannya mencakup *real-time tracking* untuk memantau keberadaan barang dengan pembaruan data secara langsung, otomatisasi pencatatan melalui OCR yang dapat mengenali nomor inventaris dari gambar dan memperbarui data tanpa input manual, serta manajemen akses melalui sistem *login* terintegrasi yang memastikan hanya pengguna terverifikasi yang dapat mengakses data. Selain itu, aplikasi ini menyediakan laporan terstruktur terkait kondisi, lokasi, dan status barang (baik, rusak, atau butuh perbaikan) serta meningkatkan efisiensi operasional dengan meminimalkan waktu dan tenaga yang diperlukan untuk pengelolaan inventaris. Berdasarkan uji coba sistem didapatkan akurasi 90% dari 10 kali percobaan terdapat 1 kesalahan dalam mengenali nomor inventaris karena *bounding box* yang juga mengambil data angka terdapat dalam gambar di luar nomor inventaris yang ada seperti pengujian yang dilakukan di lampiran 10. Selain itu kami menguji semua sistem yang menunjukkan bahwa semua fungsi yang ada pada *Web-App* dapat berfungsi dengan baik berjalan sesuai dengan kegunaannya seperti pada tabel 4.4.

Tabel 4.4 Tabel Hasil Uji Coba Fitur dan Fungsionalitas Sistem

No	Fungsionalitas	Berfungsi Dengan Baik	Tidak Berfungsi Dengan Baik
1	<i>Login</i>	✓	
2	<i>Camera Text Recognition</i>	✓	
3	<i>Query Data</i>	✓	
4	<i>Input Data</i>	✓	
5	<i>Ubah Status Barang (Moving)</i>	✓	
6	<i>History</i>	✓	
7	<i>Manajemen Aktivitas</i>	✓	
8	<i>Service Details</i>	✓	

No	Fungsionalitas	Berfungsi Dengan Baik	Tidak Berfungsi Dengan Baik
9	Pembaruan Otomatis	✓	
10	Notifikasi		✓
11	Fungsi <i>Edit, Delete dan Show Details</i> Pada Website	✓	
12	Fungsi <i>Search dan Filter</i> Pada Dashboard Utama Website	✓	
13	<i>Export Data</i> Pada Excel	✓	
14	Penambahan <i>Dictionary</i> Kode Inventaris	✓	

Dari Tabel 4.4 yang menyatakan daftar uji coba fungsionalitas dari *web-app* yang dikerjakan. Terdapat 13 fungsi yang berhasil diimplementasikan dan difungsikan secara baik dari total 14 fungsi yang direncanakan pada perancangan awal. Maka dari itu fungsionalitas keseluruhan dari project dapat dihitung pada Persamaan4.1:

$$\text{Fungsionalitas Keseluruhan (\%)} = \frac{13}{14} \times 100\% = 92.8\% \quad (4.1)$$

Terdapat satu fungsi yang belum berhasil diimplementasikan dengan baik yaitu fungsi notifikasi ketika terdapat perubahan pada data inventaris, baik itu penambahan, pengurangan, ataupun perubahan data. Hal ini disebabkan karena keterbatasan periode penggerjaan proyek yang bertepatan dengan dimulainya kegiatan pembelajaran pada semester Ganjil 2024/2025. Selain itu, implementasi notifikasi merupakan sesuatu rumit yang memerlukan proses *research* dan *trial error* yang ekstensif karena melibatkan banyak fitur lain pada proyek ini.

Solusi dari kendala tersebut adalah melakukan perancangan proyek yang lebih realistik, baik dari sisi fungsionalitas maupun dari sisi *timeline* dan waktu penggerjaan proyek sehingga dapat menjalankan memaksimalkan fungsionalitas sistem menjadi 100%.

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil praktik kerja lapangan yang telah dilakukan maka dapat disimpulkan bahwa

1. Penelitian berhasil menyelesaikan permasalahan yang dihadapi oleh perusahaan dengan integrasi teknologi kecerdasan buatan untuk menggantikan proses manual dalam pengelolaan inventaris, dimana penggunaan microsoft excel diganti menggunakan penggunaan *website* dan aplikasi dalam proses manajemen inventaris dari PT.Nilam Port Terminal Indonesia. Solusi ini juga membantu karyawan mempunyai satu database yang terintegrasi antar Divisi.
2. Penggunaan *web-app* yang dibangun menggunakan bahasa pemrograman *python* dengan *framework django*, HTML, dan CSS untuk *website*. Dan juga bahasa *kotlin* untuk *mobile app* memudahkan proses otomatisasi pencatatan inventaris dengan fungsionalitas yang diberikan, seperti pembacaan nomor inventaris yang akurat dan juga pencatatan inventaris yang *real-time* terintegrasi antara aplikasi dan *website*. Dengan adanya *web-app* memungkinkan otomatisasi pencatatan inventaris dan pelacakan aset secara *real-time*, menggantikan metode manual berbasis *Excel*. Dengan fitur seperti pengenalan teks (OCR) untuk input data otomatis, pelaporan terstruktur. Pada *web-app* yang telah dibuat dicapai fungsionalitas sebesar 92%.

5.2 Saran

Penambahan fitur serta peningkatan *user interface* pada *website* dan aplikasi akan membuat tampilan menjadi lebih profesional. Selain itu, pengembangan fitur seperti kemampuan sistem untuk mengenali lebih banyak jenis barang berdasarkan atribut lain selain nomor inventaris akan mempermudah pengguna dalam proses identifikasi. Hal ini dapat meningkatkan fleksibilitas sistem dalam mengelola berbagai item, sehingga memperluas fungsionalitas dan kegunaannya bagi pengguna.

DAFTAR PUSTAKA

- Alam, M. K., Thakur, O. A., & Islam, F. T. (2024). Inventory management systems of small and medium enterprises in Bangladesh. *Rajagiri Management Journal*, 18(1), 8–19.
<https://doi.org/10.1108/ramj-09-2022-0145>
- Amol Rathod, Y., Babu Peddawad, A., Jitendra Jagtap, K., Nagendra Gangnelu, N., Bakre, S. M., Shiralkar, A. D., & Peddawad, A. (2022). Energy Meter Tamper Detection and Alert Messaging System. *International Journal of Technology Engineering Arts Mathematics Science*, 1(2), 2583–1224.
<https://doi.org/10.11591/eei.v9i3.xxxx>
- C. G., T., & Devi, A. J. (2021). A Study and Overview of the Mobile App Development Industry. *International Journal of Applied Engineering and Management Letters*, 5(1), 115–130.
<https://doi.org/10.47992/ijaeml.2581.7000.0097>
- Chowdhury, A. A., Chowdhury, S. K., Hanif, M., Nosheen, S. N., & Zishan, M. S. R. (2021). Design and development of citizen surveillance and social-credit information system for Bangladesh. *AIUB Journal of Science and Engineering*, 20(2), 33–39. <https://doi.org/10.53799/AJSE.V20I2.133>
- Ezekiel A., O., Aderonke J., I., & Silas T., A. (2024). *An improved web-based inventory management system for universities*. 2013.
<https://doi.org/10.4108/eai.25-10-2023.2348749>
- Farooque, M., Zhang, A., Thürer, M., Qu, T., & Huisingsh, D. (2019). Circular supply chain management: A definition and structured literature review. *Journal of Cleaner Production*, 228(July), 882–900.
<https://doi.org/10.1016/j.jclepro.2019.04.303>
- Freeman, J. (1993). Inventory Control and Management. In *Journal of the Operational Research Society* (Vol. 44, Issue 3).
<https://doi.org/10.1057/jors.1993.59>
- Gajewsku, H., Kistler, J., Manasse, M. S., & Redell, D. D. (1994). Argo: A system for distributed. *Proceedings of the 2nd ACM International Conference on Multimedia*, MULTIMEDIA 1994, January 1994, 433–440.
<https://doi.org/10.1145/192593.192716>

- Hengaju, U., & Krishna Bal, B. (2020). Improving the Recognition Accuracy of Tesseract-OCR Engine on Nepali Text Images via Preprocessing. *Advancement in Image Processing and Pattern Recognition*, 3(3), 1–13.
- Jatain, A., Bhaskar Bajaj, S., Chaudhary, S., & Batra, P. (2021). Rest Web Services: An Elementary Learning. *Research Journal of Engineering and Technology*, 12(03), 75–78. <https://doi.org/10.52711/2321-581x.2021.00012>
- Jongmans, E., Jeannot, F., Liang, L., & Dampérat, M. (2022). Impact of website visual design on user experience and website evaluation: the sequential mediating roles of usability and pleasure. *Journal of Marketing Management*, 38(17–18), 2078–2113. <https://doi.org/10.1080/0267257X.2022.2085315>
- Kengalagutti, D. (2020). Comparing Database Management Systems : MySQL , PostgreSQL , SQLite. *International Research Journal Of Engineering and Technology (IRJET)*, 07(06), 2238–2241. <https://www.irjet.net/archives/V7/i6/IRJET-V7I6418.pdf>
- Maharana, K. C., & Acharya, S. (2024). International Journal of Research Publication and Reviews. *SSRN Electronic Journal*, 5(8), 262–265. <https://doi.org/10.2139/ssrn.4909110>
- McMinn, P., Wright, C. J., McCurdy, C. J., & Kapfhammer, G. M. (2019). Automatic Detection and Removal of Ineffective Mutants for the Mutation Analysis of Relational Database Schemas. *IEEE Transactions on Software Engineering*, 45(5), 427–463. <https://doi.org/10.1109/TSE.2017.2786286>
- Mitropoulos, S., Mitsis, C., Valacheas, P., & Douligeris, C. (2021). An online emergency medical management information system using mobile computing. *Applied Computing and Informatics*, May. <https://doi.org/10.1108/aci-03-2021-0069>
- Mudassir, M., & Mohammed, M. (2024). The role of APIs in modern software development. *Technource*, 13(01), 1045–1047. <https://www.technource.com/blog/modern-software-development/>
- Mutti, S., Bacis, E., & Paraboschi, S. (2015). SeSQLite: Security enhanced sqlite: Mandatory access control for android databases. *ACM International Conference Proceeding Series*, 7-11-December-2015(June), 411–420. <https://doi.org/10.1145/2818000.2818041>

- Mwamba, E., & Yangailo, T. (2024). The impact of inventory management on the performance of an organization. *Revista Científica Profundidad Construyendo Futuro*, 20(20), 77–85. <https://doi.org/10.22463/24221783.4184>
- Nitesh, U. (2024). Artificial Intelligence in Web Development: Enhancing Automation, Personalization, and Decision-Making. *International Journal of Advanced Research in Science, Communication and Technology*, August, 534–540. <https://doi.org/10.48175/ijarsct-19367>
- Patel, J. A. (2021). Handwritten And Printed Text Recognition Using Tesseract-OCR. *International Journal of Creative Research Thoughts (IJCRT)*, 9(9), 69–77. www.ijcrt.org
- Patel, V. (2021). *VISTA 2 . 0 - A visual analytics platform for semi-automatic annotation of trajectory segments*. Department of Computer Science Supervisor : Amilcar Soares. August. <https://doi.org/10.13140/RG.2.2.23671.96165>
- Petropoulos, F., Laporte, G., Aktas, E., Alumur, S. A., Archetti, C., Ayhan, H., Battarra, M., Bennell, J. A., Bourjolly, J. M., Boylan, J. E., Breton, M., Canca, D., Charlin, L., Chen, B., Cicek, C. T., Cox, L. A., Currie, C. S. M., Demeulemeester, E., Ding, L., ... Zhao, X. (2024). Operational Research: methods and applications. *Journal of the Operational Research Society*, 75(3), 423–617. <https://doi.org/10.1080/01605682.2023.2253852>
- PT. Nilam Port terminal Indonesia. (2016). *COMPANY PROFILE PT. NILAM PORT TERMINAL INDONESIA*. 118.
- Rajab, H. (2024). *Cloud-Based Solutions for Inventory Visibility and Control*. Author : Johnson Fred Abstract. November.
- Ramos-Miller, M., & Pacheco, A. (2023). Towards inventory control excellence: An innovative approach based on a web-based platform. *F1000Research*, 12, 1471. <https://doi.org/10.12688/f1000research.140745.1>
- Raya Janti, J., Jambe, K., & Yogyakarta, B. (2020). *A Comparative Study of Java and Kotlin for Android Mobile Application Development*. www.isriti.akakom.ac.id

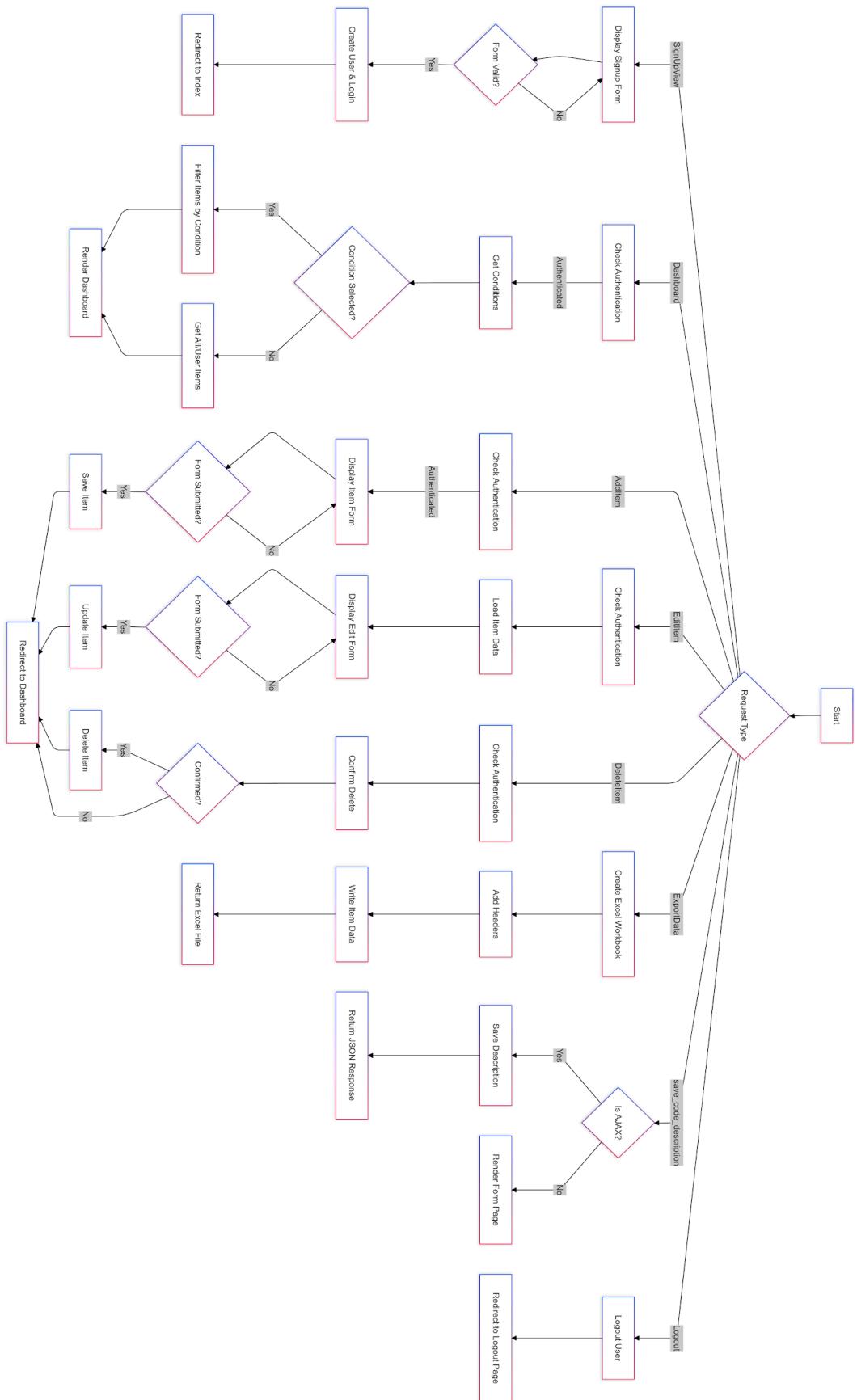
- Akilesh, & Hallikar, R. S. (2022). *End to End Representational State Transfer Application Programming Interface Development*. 7(July), 1690. www.ijrti.org
- Sham, A. S. D., Pandey, P., Jain, S., & Kalaivani, S. (2021). Automatic License Plate Recognition Using Yolov4 and Tesseract Ocr. *International Journal of Electrical Engineering and Technology*, 12(5). <https://doi.org/10.34218/ijeet.12.5.2021.006>
- Shiratuddin, M. F., & Thabet, W. (2011). Utilizing a 3D game engine to develop a virtual design review system. *Electronic Journal of Information Technology in Construction*, 16(May), 39–68.
- Singh, M., Barua, B., Palod, P., Garg, M., Satapathy, S., Bushi, S., Ayush, K., Rohith, K. S., Gamidi, T., Goyal, P., & Mukherjee, A. (2016). OCR++: A robust framework for information extraction from scholarly articles. *COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers*, 3390–3400.
- Wirna, Ansyari, M. I., & Nasrulhaq. (2022). Penggunaan Barang Pada Kantor Biro Umum Dan Perlengkapan Provinsi Sulawesi Barat. *Jurnal Unismuh*, 3(4).

LAMPIRAN

Lampiran 1. Dokumentasi Pelaksanaan Kegiatan PKL



Lampiran 2. Alur dan Pendefinisian Views.py



Lampiran 3. Kode *Login Website* dalam *Views.py* dan *HTML*

```
class SignUpView(View):
    def get(self, request):
        form = UserRegisterForm()
        return render(request, 'inventory/signup.html', {'form': form})
    def post(self, request):
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            user = authenticate(
                username=form.cleaned_data['username'],
                password=form.cleaned_data['password1']
            )
            if user is not None:
                login(request, user)
                return redirect('index')
        return render(request, 'inventory/signup.html', {'form': form})
{% extends 'inventory/base.html' %}
{% load crispy_forms_tags %}
{% block content %}
<div class="row">
    <div class="col-11 col-md-4 mx-auto mt-5">
        <h1>Log In</h1>
        <form method="post">
            {% csrf_token %}
            {{ form|crispy }}
            <button class="btn btn-primary">Sign In</button>
        </form>
        <div class="pt-3">
            <small class="text-muted">Need an account? <a href="{% url 'signup' %}">Sign Up</a></small>
        </div>
    </div>
</div>
{% endblock content %}
```

Lampiran 4. Kode Dashboard Beserta Fitur Search, Sort, Filter, dan Show Details

```

class Dashboard(LoginRequiredMixin, View):
    def get(self, request):
        conditions = Condition.objects.all()
        selected_condition = request.GET.get('condition_name', "")
        if selected_condition:
            items = InventoryItem.objects.filter(condition_name=selected_condition)
        else:
            items = InventoryItem.objects.all() if request.user.is_superuser else
        InventoryItem.objects.filter(user=request.user).order_by('id')
        items_data = serializers.serialize('json', items)
        return render(request, 'inventory/dashboard.html', {
            'items': items,
            'items_data': items_data,
            'conditions': conditions,
            'selected_condition': selected_condition,
        })
    // Populate dropdowns
    populateDropdown('filterNo', items, 'no');
    populateDropdown('filterItem', items, 'name');
    populateDropdown('filterPic', items, 'pic');
    populateDropdown('filterLocation', items, 'location');

    // Toggle Details
    function bindToggleDetails() {
        document.querySelectorAll('.toggle-details').forEach(button => {
            button.addEventListener('click', function () {
                const extraDetailsRow = this.closest('tr').nextElementSibling;
                extraDetailsRow.classList.toggle('d-none');
                this.textContent = extraDetailsRow.classList.contains('d-none') ? 'Show Details' :
            'Hide Details';
            });
        });
    }
    // Sorting
    document.querySelectorAll('.sort-button').forEach(button => {
        button.addEventListener('click', function () {
            const sortKey = this.getAttribute('data-sort');
            const sortDirection = this.getAttribute('data-direction') || 'asc';
            sortTable(sortKey, sortDirection);
            this.setAttribute('data-direction', sortDirection === 'asc' ? 'desc' : 'asc');
        });
    });

    function sortTable(key, direction) {
        const sortedItems = items.sort((a, b) => {
            const aValue = a.fields[key];
            const bValue = b.fields[key];
            if (aValue < bValue) return direction === 'asc' ? -1 : 1;
            if (aValue > bValue) return direction === 'asc' ? 1 : -1;
            return 0;
        });
    }
    function filterRows() {
        const filterValues = {
            no: document.getElementById('filterNo').value.toLowerCase(),
            item: document.getElementById('filterItem').value.toLowerCase(),
        }
    }
}

```

```

condition: document.getElementById('conditionSearch').value.toLowerCase(),
pic: document.getElementById('filterPic').value.toLowerCase(),
location: document.getElementById('filterLocation').value.toLowerCase(),
search: document.getElementById('searchBar').value.toLowerCase(),
};

document.querySelectorAll('#inventoryTableBody >
tr:not(.extra-details)').forEach(row => {
    const no = row.children[0].textContent.toLowerCase();
    const item = row.children[1].textContent.toLowerCase();
    const condition = row.children[2].textContent.toLowerCase();
    const pic = row.children[4].textContent.toLowerCase();
    const location = row.children[5].textContent.toLowerCase();

    const matches =
        (filterValues.no === "" || no.includes(filterValues.no)) &&
        (filterValues.item === "" || item.includes(filterValues.item)) &&
        (filterValues.condition === "" || condition.includes(filterValues.condition)) &&
        (filterValues.pic === "" || pic.includes(filterValues.pic)) &&
        (filterValues.location === "" || location.includes(filterValues.location)) &&
        (filterValues.search === "" ||
            no.includes(filterValues.search) ||
            item.includes(filterValues.search) ||
            condition.includes(filterValues.search) ||
            pic.includes(filterValues.search) ||
            location.includes(filterValues.search));

    row.style.display = matches ? '' : 'none';
    row.nextElementSibling.style.display = matches ? '' : 'none';
});

```

Lampiran 5. Kode Dashboard Beserta Fitur Add, Edit, Delete, dan Export to Excel

```
class AddItem(LoginRequiredMixin, CreateView):
    model = InventoryItem
    form_class = InventoryItemForm
    template_name = 'inventory/item_form.html'
    success_url = reverse_lazy('dashboard')
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['categories'] = Category.objects.all()
        context['departments'] = Department.objects.all()
        context['conditions'] = Condition.objects.all()
        context['descriptions'] = DescriptionModel.objects.all() # Pulling data from
DescriptionModel
        return context
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)
class EditItem(LoginRequiredMixin, UpdateView):
    model = InventoryItem
    form_class = InventoryItemForm
    template_name = 'inventory/item_form.html'
    success_url = reverse_lazy('dashboard')
class DeleteItem(LoginRequiredMixin, DeleteView):
    model = InventoryItem
    template_name = 'inventory/delete_item.html'
    success_url = reverse_lazy('dashboard')
    context_object_name = 'item'
def ExportData(request):
    # Create a workbook and a worksheet
    wb = Workbook()
    ws = wb.active
    ws.title = 'Inventory Data'
    # Define the header
    headers = [
        'Date Created', 'No Inventaris', 'Item', 'Photo', 'Specifications', 'Department',
        'Category', 'Location', 'PIC', 'Condition', 'History',
        'Tipe Unit', 'Input by', 'digit_1', 'kode_asset','digit_23', 'kode_golongan',
        'digit_45', 'kdoe_jenisunit', 'urutan', 'bulan', 'tahun', 'bpb_ppat',
        'po',
    ]
    ws.append(headers)
    # Fetch the data and write to the worksheet
    items = InventoryItem.objects.all() if request.user.is_superuser else
InventoryItem.objects.filter(user=request.user)
    for item in items:
        # Remove timezone information from datetime objects
        date_created = item.date_created.replace(tzinfo=None) if item.date_created else "
        ws.append([
            date_created,
            item.no,
            item.name,
            item.photo.url if item.photo else "", # Include the photo URL if it exists
            item.specifications,
            item.department.name if item.department else "",
            item.category.name if item.category else ",
        ])
```

```
    item.location,
    item.pic if item.pic else '',
    item.condition.name,
    item.history,
    item.tipe_unit,
    item.user.username if item.user else '',
    item.digit_1,
    item.kode_asset,
    item.digit_23,
    item.kode_golongan,
    item.digit_45,
    item.kode_jenisunit,
    item.urutan,
    item.bulan,
    item.tahun,
    item.bpb_ppat,
    item.po
)
# Save the workbook to a BytesIO object
output = BytesIO()
wb.save(output)
output.seek(0)
# Create the response with the Excel file
response = HttpResponse(output,
content_type='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')
response['Content-Disposition'] = 'attachment; filename=inventory_data.xlsx'
return response
```

Lampiran 6. Pembuatan Aplikasi

a. *Login Activity*

```
import android.annotation.SuppressLint
import android.content.ContentValues.TAG
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageButton
import android.widget.ProgressBar
import android.widget.TextView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import okhttp3.*
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.RequestBody.Companion.toRequestBody
import org.json.JSONObject
import java.io.IOException

class LoginActivity : AppCompatActivity() {

    private lateinit var usernameEditText: EditText
    private lateinit var passwordEditText: EditText
    private lateinit var loginButton: Button
    private lateinit var errorMessageTextView: TextView
    private lateinit var sharedpreferences: SharedPreferences
    private lateinit var settingsButton: ImageButton
    private lateinit var progressBar: ProgressBar

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_login)

        usernameEditText = findViewById(R.id.username)
        passwordEditText = findViewById(R.id.password)
        loginButton = findViewById(R.id.login)
        errorMessageTextView = findViewById(R.id.error_message)
        settingsButton = findViewById(R.id.settings_button)
        progressBar = findViewById(R.id.loading)

        sharedpreferences = getSharedPreferences("MyAppPrefs", MODE_PRIVATE)

        settingsButton.setOnClickListener {
            val intent = Intent(this, SettingsActivity::class.java)
            startActivity(intent)
        }

        loginButton.setOnClickListener {
            performLogin()
        }
    }

    private fun performLogin() {
        // Implement login logic here
    }
}
```

```

@SuppressLint("SetTextI18n")
private fun performLogin() {
    progressBar.visibility = View.VISIBLE
    val username = usernameEditText.text.toString()
    val password = passwordEditText.text.toString()

    // Get server address from SharedPreferences
    val serverAddress = sharedPreferences.getString("server_address", "")
    if (serverAddress.isNullOrEmpty()) {
        Toast.makeText(this, "Please set the server address in settings",
        Toast.LENGTH_SHORT).show()
    }
    return
}

if (username.isEmpty() || password.isEmpty()) {
    errorMessageTextView.text = "Username and password cannot be empty."
    errorMessageTextView.visibility = View.VISIBLE
    return
}

// Create OkHttpClient instance
val client = OkHttpClient.Builder()
    .cookieJar(MyCookieJar())
    .build()

// Create request body with JSON payload
val json = JSONObject().apply {
    put("username", username)
    put("password", password)
}

val requestBody = json.toString().toRequestBody("application/json;
charset=utf-8".toMediaTypeOrNull())

// Log the request details
Log.d("LoginActivity", "Request URL: $serverAddress/api/login/")
Log.d("LoginActivity", "Request Body: $json")
// Create request to post data to the server
val request = Request.Builder()
    .url("$serverAddress/api/login/") // Change to your login URL
    .post(requestBody)
    .build()

// Execute the request asynchronously
// Execute the request asynchronously
client.newCall(request).enqueue(object : Callback {

    override fun onResponse(call: okhttp3.Call, response: okhttp3.Response) {
        val responseBody = response.body?.string()
        if (!response.isSuccessful) {
            // Handle unsuccessful response
            runOnUiThread {
                Toast.makeText(
                    this@LoginActivity,
                    "Login failed: ${response.message}\nResponse: $responseBody",

```

```

        Toast.LENGTH_LONG
    ).show()
progressBar.visibility = View.GONE
}
return
}

// Handle successful response
val jsonResponse = responseBody?.let { JSONObject(it) }
val token = jsonResponse?.optString("token") // Change to the actual key in your
response
val userID = jsonResponse?.optString("user_id") // Change to the actual key in your
response

if (!token.isNullOrEmpty() && !userID.isNullOrEmpty()) {
    // Save token in Shared Preferences
    val editor = sharedpreferences.edit()
    editor.putString("auth_token", token)
    editor.putString("user_id", userID)
    editor.apply()
    Log.d(TAG, "Authenticated with token : ")
    Log.d(TAG, token)
    Log.d(TAG, "User ID : ")
    Log.d(TAG, userID)

    // Save cookies (sessionid) from response if needed
    val cookies = response.headers.values("Set-Cookie")
    if (cookies.isNotEmpty()) {
        editor.putString("session_cookie", cookies[0])
        editor.apply()
    }
    Log.d(TAG, cookies.toString())
    runOnUiThread {
        progressBar.visibility = View.GONE
    }
    // Navigate to another activity
    val intent = Intent(this@LoginActivity, MainActivity::class.java)
    startActivity(intent)
    finish() // Optional: close the LoginActivity
} else {
    runOnUiThread {
        Toast.makeText(
            this@LoginActivity,
            "Invalid credentials",
            Toast.LENGTH_SHORT
        ).show()
    }
}

override fun onFailure(call: okhttp3.Call, e: IOException) {
// Handle request failure
runOnUiThread {
    Toast.makeText(
        this@LoginActivity,
        "Request failed: ${e.message}",
        Toast.LENGTH_SHORT
    )
}
}

```

```
    ).show()  
}  
}  
}  
})  
}  
}
```

b. *Main Activity*

```
import android.Manifest
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.graphics.Canvas
import android.graphics.Paint
import android.os.Bundle
import android.util.Log
import android.util.Size
import android.view.View
import android.widget.Button
import androidx.activity.enableEdgeToEdge
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.camera.core.*
import androidx.camera.lifecycle.ProcessCameraProvider
import androidx.camera.view.PreviewView
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.core.content.ContextCompat
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import java.io.File
import java.io.FileOutputStream

class MainActivity : AppCompatActivity() {

    private lateinit var previewView: PreviewView
    private lateinit var buttonCapture: Button
    private lateinit var croppingOverlay: View
    private var imageCapture: ImageCapture? = null
    private val filename = "captured_image.jpg"

    private val requestPermissionLauncher = registerForActivityResult(
        ActivityResultContracts.RequestPermission()
    ) { isGranted: Boolean ->
        if (isGranted) {
            startCamera()
        } else {
            // Permission denied, show message to the user
        }
    }
}
```

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContentView(R.layout.activity_main)

    previewView = findViewById(R.id.previewView)
    buttonCapture = findViewById(R.id.button_capture)
    croppingOverlay = findViewById(R.id.croppingOverlay)

    if (allPermissionsGranted()) {
        startCamera()
    } else {
        requestPermissionLauncher.launch(Manifest.permission.CAMERA)
    }

    buttonCapture.setOnClickListener { captureScreenshotOfOverlay() }

}

private fun allPermissionsGranted() = ContextCompat.checkSelfPermission(
    baseContext, Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED

private fun startCamera() {
    val cameraProviderFuture = ProcessCameraProvider.getInstance(this)

    cameraProviderFuture.addListener({
        val cameraProvider: ProcessCameraProvider = cameraProviderFuture.get()
        imageCapture = ImageCapture.Builder()
            .build()

        val preview = Preview.Builder().build().also {
            it.setSurfaceProvider(previewView.surfaceProvider)
        }

        try {
            cameraProvider.unbindAll()
            cameraProvider.bindToLifecycle(
                this, CameraSelector.DEFAULT_BACK_CAMERA, preview, imageCapture)
        } catch(exc: Exception) {
            Log.e(TAG, "Use case binding failed", exc)
        }
    }, ContextCompat.getMainExecutor(this))
}

private fun captureScreenshotOfOverlay() {
    val previewBitmap = previewView.bitmap ?: return

    // Convert dp to px
    val overlayHeightPx = dpToPx(80) // 80dp in px

```

```

    val overlayLocation = IntArray(2)
    croppingOverlay.getLocationOnScreen(overlayLocation)

    val overlayX = overlayLocation[0]
    val overlayY = overlayLocation[1]
    val overlayWidth = croppingOverlay.width
    val overlayHeight = overlayHeightPx // Fixed height in px

    Log.d(TAG, "Overlay Dimensions - X: $overlayX, Y: $overlayY, Width: $overlayWidth, Height: $overlayHeight")

    val previewViewLocation = IntArray(2)
    previewView.getLocationOnScreen(previewViewLocation)

    val previewViewX = previewViewLocation[0]
    val previewViewY = previewViewLocation[1]

    // Calculate crop start and end positions
    val cropStartX = (overlayX - previewViewX).coerceIn(0, previewBitmap.width)
    val cropStartY = (overlayY - previewViewY).coerceIn(0, previewBitmap.height)
    val cropEndX = (cropStartX + overlayWidth).coerceAtMost(previewBitmap.width)
    val cropEndY = (cropStartY + overlayHeight).coerceAtMost(previewBitmap.height)

    // Ensure crop dimensions are valid
    val cropWidth = (cropEndX - cropStartX).coerceAtLeast(1)
    val cropHeight = (cropEndY - cropStartY).coerceAtLeast(1)

    Log.d(TAG, "Crop Area - StartX: $cropStartX, StartY: $cropStartY, EndX: $cropEndX, EndY: $cropEndY")
    Log.d(TAG, "Crop Dimensions - Width: $cropWidth, Height: $cropHeight")
    Log.d(TAG, "previewBitmap - width: ${previewBitmap.width}, height: ${previewBitmap.height}")

    // Ensure the cropping area is within the bitmap bounds
    if (cropStartX < 0 || cropStartY < 0 || cropEndX > previewBitmap.width || cropEndY > previewBitmap.height) {
        Log.e(TAG, "Crop area is out of bounds")
        return
    }

    try {
        // Create a new bitmap with the desired dimensions
        val croppedBitmap = Bitmap.createBitmap(previewBitmap, cropStartX, cropStartY, cropWidth, cropHeight)

        val croppedFile = File(externalMediaDirs.firstOrNull(), "Screenshot_Overlay.jpg")
        val outputStream = FileOutputStream(croppedFile)
        croppedBitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)
        outputStream.close()

        val intent = Intent(this, ReviewActivity::class.java).apply {
            putExtra("ImagePath", croppedFile.absolutePath)
        }
    }

```

```

        startActivity(intent)
    } catch (e: IllegalArgumentException) {
        Log.e(TAG, "Cropping failed: ${e.message}", e)
    } catch (e: Exception) {
        Log.e(TAG, "Unexpected error: ${e.message}", e)
    }
}

private fun dpToPx(dp: Int): Int {
    return (dp * resources.displayMetrics.density).toInt()
}

// private fun capturePhoto() {
//     val imageCapture = imageCapture ?: return
//
//     val photoFile = File(externalMediaDirs.firstOrNull(), filename)
//     val outputOptions = ImageCapture.OutputFileOptions.Builder(photoFile).build()
//
//     imageCapture.takePicture(
//         outputOptions,
//         ContextCompat.getMainExecutor(this),
//         object : ImageCapture.OnImageSavedCallback {
//             override fun onError(exc: ImageCaptureException) {
//                 Log.e(TAG, "Photo capture failed: ${exc.message}", exc)
//             }
//         }
//     )
//     override fun onImageSaved(output: ImageCapture.OutputFileResults) {
//         saveImage(photoFile.absolutePath)
//     }
// })
// private fun saveImage(filePath: String) {
//     val bitmap = BitmapFactory.decodeFile(filePath)
//     val croppedFile = File(externalMediaDirs.firstOrNull(), "cropped_image.jpg")
//     val outputStream = FileOutputStream(croppedFile)
//     bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)
//     outputStream.close()
//
//     val intent = Intent(this, ReviewActivity::class.java).apply {
//         putExtra("imagePath", croppedFile.absolutePath)
//     }
//     startActivity(intent)
// }

companion object {
    private const val TAG = "CameraXApp"
}
}

```

c. Penambahan Inventaris

```
import android.Manifest
import android.app.Activity
import android.content.ContentValues.TAG
import android.content.Intent
import android.content.SharedPreferences
import android.content.pm.PackageManager
import android.net.Uri
import android.os.Bundle
import android.os.Environment
import android.provider.MediaStore
import android.util.Log
import android.view.View
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.ProgressBar
import android.widget.Spinner
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.core.content.FileProvider
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import okhttp3.Call
import okhttp3.Callback
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.MultipartBody
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody.Companion.asRequestBody
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.Response
import org.json.JSONArray
import org.json.JSONException
import org.json.JSONObject
import java.io.File
import java.io.IOException
import java.text.SimpleDateFormat
import java.util.Date

class AddInventoryActivity : AppCompatActivity() {

    private lateinit var noEditText: EditText
    private lateinit var itemEditText: EditText
    private lateinit var specificationEditText: EditText
    private lateinit var locationEditText: EditText
    private lateinit var userEditText: EditText
    private lateinit var userIdEditText: EditText
    private lateinit var submitButton: Button
    private lateinit var code: String
    private lateinit var progressBar: ProgressBar
    private lateinit var condition: String
```

```

private lateinit var sharedPreferences: SharedPreferences

private lateinit var selectPhotoButton: Button
private lateinit var takePhotoButton: Button
private lateinit var photoImageView: ImageView
private val PICK_IMAGE_REQUEST = 1
private val REQUEST_IMAGE_CAPTURE = 2
private val REQUEST_PERMISSIONS = 3
private var photoUri: Uri? = null
private lateinit var currentPhotoPath: String

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContentView(R.layout.activity_add_inventory)

    sharedPreferences = getSharedPreferences("MyAppPrefs", MODE_PRIVATE)
    code = intent.getStringExtra("code") ?: ""

    val conditionSpinner: Spinner = findViewById(R.id.condition_spinner)
    // Array for display values
    val conditionDisplayValues = arrayOf("1: ok", "2: not ok", "3: afkir")
    // Array for actual values to send to API
    val conditionActualValues = arrayOf("1", "2", "3")
    // Set up the ArrayAdapter for the Spinner
    val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item,
        conditionDisplayValues)
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
    conditionSpinner.adapter = adapter
    conditionSpinner.onItemSelectedListener = object :
        AdapterView.OnItemSelectedListener {
        override fun onItemSelected(parent: AdapterView<*>, view: View, position: Int, id: Long) {
            condition = conditionActualValues[position]
        }
        override fun onNothingSelected(parent: AdapterView<*>) {
            // Handle no selection case if needed
        }
    }

    noEditText = findViewById(R.id.no)
    noEditText.setText(code)
    itemEditText = findViewById(R.id.item)
    specificationEditText = findViewById(R.id.specification)
    locationEditText = findViewById(R.id.location)
    userIdEditText = findViewById(R.id.user_id)
    val user_id = sharedPreferences.getString("user_id", "")
    userIdEditText.setText(user_id)
    userEditText = findViewById(R.id.user)
    submitButton = findViewById(R.id.submit)
    progressBar = findViewById(R.id.loading_add_inventory)

    // Set up submit button
    submitButton.setOnClickListener {
        submitInventoryData()
    }
}

```

```

selectPhotoButton = findViewById(R.id.selectPhotoButton)
takePhotoButton = findViewById(R.id.takePhotoButton)
photoImageView = findViewById(R.id.photo)

selectPhotoButton.setOnClickListener {
    openGallery()
}

takePhotoButton.setOnClickListener {
    if (checkPermissions()) {
        openCamera()
    }
}

private fun submitInventoryData() {
    progressBar.visibility = View.VISIBLE
    submitButton.visibility = View.GONE
    val serverAddress = sharedpreferences.getString("server_address", "")
    if (serverAddress.isNullOrEmpty()) {
        Toast.makeText(this, "Please set the server address in settings",
        Toast.LENGTH_SHORT)
        .show()
        return
    }

    val no = noEditText.text.toString()
    val item = itemEditText.text.toString()
    val specification = specificationEditText.text.toString()
    val location = locationEditText.text.toString()
    val user = userEditText.text.toString()

    val requestBodyBuilder = MultipartBody.Builder().setType(MultipartBody.FORM)
    requestBodyBuilder.addFormDataPart("no", no)
    requestBodyBuilder.addFormDataPart("name", item)
    requestBodyBuilder.addFormDataPart("specifications", specification)
    requestBodyBuilder.addFormDataPart("location", location)
    requestBodyBuilder.addFormDataPart("user", user)
    requestBodyBuilder.addFormDataPart("condition", condition)

    photoUri?.let {
        val file = File(currentPhotoPath)
        val photoRequestBody = file.asRequestBody("image/jpeg".toMediaTypeOrNull())
        requestBodyBuilder.addFormDataPart("photo", file.name, photoRequestBody)
    }

    val requestBody = requestBodyBuilder.build()

    val request = Request.Builder()
        .url("$serverAddress/api/add_inventory/")
        .post(requestBody)
        .build()

    val token = sharedpreferences.getString("auth_token", null)
    val client = token?.let { getAuthenticatedClient(it) }
}

```

```

        if (client != null) {
            client.newCall(request).enqueue(object : Callback {
                override fun onFailure(call: Call, e: IOException) {
                    runOnUiThread {
                        Toast.makeText(this@AddInventoryActivity, "Gagal Menyimpan Data!",
                        Toast.LENGTH_SHORT).show()
                        Log.e(TAG, "Error during submission", e)
                    }
                }

                override fun onResponse(call: Call, response: Response) {
                    if (response.isSuccessful) {
                        runOnUiThread {
                            progressBar.visibility = View.GONE
                            submitButton.visibility = View.VISIBLE
                            Toast.makeText(this@AddInventoryActivity, "Inventory Berhasil Disimpan",
                            Toast.LENGTH_SHORT).show()
                            val intent = Intent(this@AddInventoryActivity, MainActivity::class.java)
                            startActivity(intent)
                            finish()
                        }
                    } else {
                        runOnUiThread {
                            progressBar.visibility = View.GONE
                            submitButton.visibility = View.VISIBLE
                            Toast.makeText(this@AddInventoryActivity, "Error: ${response.message}",
                            Toast.LENGTH_SHORT).show()
                        }
                    }
                }
            })
        }
    }

private fun getAuthenticatedClient(token: String): OkHttpClient {
    val sessionCookie = sharedPreferences.getString("session_cookie", null)

    return OkHttpClient.Builder()
        .cookieJar(MyCookieJar())
        .addInterceptor { chain ->
            val original = chain.request()
            val requestBuilder = original.newBuilder()
                .header("Authorization", "Bearer $token")
                .method(original.method, original.body)

            // Add session cookie to request if it exists
            sessionCookie?.let {
                requestBuilder.header("Cookie", it)
                Log.d(TAG, "Cookie $it")
            }

            val request = requestBuilder.build()
            chain.proceed(request)
        }
        .build()
}

```

```

private fun openGallery() {
    val intent = Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
    startActivityForResult(intent, PICK_IMAGE_REQUEST)
}

private fun openCamera() {
    val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
    if (intent.resolveActivity(packageManager) != null) {
        val photoFile: File? = try {
            createImageFile()
        } catch (ex: IOException) {
            null
        }
        photoFile?.also {
            photoUri = FileProvider.getUriForFile(
                this,
                "${applicationContext.packageName}.fileprovider",
                it
            )
            intent.putExtra(MediaStore.EXTRA_OUTPUT, photoUri)
            startActivityForResult(intent, REQUEST_IMAGE_CAPTURE)
        }
    }
}

@Throws(IOException::class)
private fun createImageFile(): File {
    val timeStamp: String = SimpleDateFormat("yyyyMMdd_HHmmss").format(Date())
    val storageDir: File = getExternalFilesDir(Environment.DIRECTORY_PICTURES)!!
    return File.createTempFile(
        "JPEG_${timeStamp}_",
        ".jpg",
        storageDir
    ).apply {
        currentPhotoPath = absolutePath
    }
}

@Deprecated("Deprecated in Java")
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (resultCode == Activity.RESULT_OK) {
        when (requestCode) {
            PICK_IMAGE_REQUEST -> {
                val selectedImage: Uri? = data?.data
                selectedImage?.let {
                    photoImageView.setImageURI(it)
                    photoUri = it
                    currentPhotoPath = getRealPathFromURI(it).toString()
                }
            }
            REQUEST_IMAGE_CAPTURE -> {
                photoUri?.let {
                    photoImageView.setImageURI(it)
                }
            }
        }
    }
}

```

```
        }
    }
}
}

private fun getRealPathFromURI(contentUri: Uri): String? {
    var result: String? = null
    val proj = arrayOf(MediaStore.Images.Media.DATA)
    val cursor = contentResolver.query(contentUri, proj, null, null, null)
    cursor?.use {
        if (it.moveToFirst()) {
            val columnIndex = it.getColumnIndexOrThrow(MediaStore.Images.Media.DATA)
            result = it.getString(columnIndex)
        }
    }
    return result
}

private fun checkPermissions(): Boolean {
    val permissions = arrayOf(
        Manifest.permission.CAMERA,
        Manifest.permission.WRITE_EXTERNAL_STORAGE,
        Manifest.permission.READ_EXTERNAL_STORAGE
    )

    val missingPermissions = permissions.filter {
        ContextCompat.checkSelfPermission(this, it) !=
        PackageManager.PERMISSION_GRANTED
    }

    return if (missingPermissions.isNotEmpty()) {
        ActivityCompat.requestPermissions(this, missingPermissions.toTypedArray(),
REQUEST_PERMISSIONS)
        false
    } else {
        true
    }
}

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (requestCode == REQUEST_PERMISSIONS && grantResults.isNotEmpty()) {
        val allGranted = grantResults.all { it == PackageManager.PERMISSION_GRANTED }
        if (allGranted) {
            openCamera()
        }
    }
}
```

d. Pemindahan Barang (*Moving*)

```
import android.content.ContentValues.TAG
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import okhttp3.Call
import okhttp3.Callback
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.Response
import org.json.JSONObject
import java.io.IOException

class MovingActivity : AppCompatActivity() {

    private lateinit var code: String
    private lateinit var noInventarisTextView: TextView
    private lateinit var moveByEditText: EditText
    private lateinit var moveToEditText: EditText
    private lateinit var btnSave: Button
    private lateinit var sharedpreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_moving)

        sharedpreferences = getSharedPreferences("MyAppPrefs", MODE_PRIVATE)

        code = intent.getStringExtra("code") ?: ""
        noInventarisTextView = findViewById(R.id.tv_no_inventaris)
        noInventarisTextView.text = "No Inventaris : ".plus(code)

        moveByEditText = findViewById(R.id.move_by)
        moveToEditText = findViewById(R.id.move_to)
        btnSave = findViewById(R.id.save_moving)

        btnSave.setOnClickListener {
            performMove()
        }
    }

    private fun performMove() {
        // Implementasi logika pemindahan barang
    }
}
```

```

private fun performMove() {
    val serverAddress = sharedPreferences.getString("server_address", "")
    if (serverAddress.isNullOrEmpty()) {
        Toast.makeText(this, "Please set the server address in settings",
        Toast.LENGTH_SHORT)
            .show()
        return
    }

    val moveBy = moveByEditText.text.toString()
    val moveTo = moveToEditText.text.toString()

    if (moveBy.isEmpty() || moveTo.isEmpty()) {
        Toast.makeText(this, "Move By and Move To cannot be empty.",
        Toast.LENGTH_SHORT).show()
        return
    }

    // Create request body with JSON payload
    val json = JSONObject().apply {
        put("no_inventaris", code)
        put("move_to", moveTo)
        put("move_by", moveBy)
    }

    val requestBody = json.toString().toRequestBody("application/json;
    charset=utf-8".toMediaTypeOrNull())

    val request = Request.Builder()
        .url("$serverAddress/api/moving/")
        .post(requestBody)
        .build()

    val token = sharedPreferences.getString("auth_token", null)
    val client = token?.let { getAuthenticatedClient(it) }

    // Execute the request asynchronously
    if (client != null) {
        client.newCall(request).enqueue(object : Callback {

            override fun onResponse(call: Call, response: Response) {
                val responseBody = response.body?.string()
                if (response.isSuccessful) {
                    runOnUiThread {
                        Toast.makeText(this@MovingActivity, "Success!",
                        Toast.LENGTH_SHORT).show()
                        val intent = Intent(this@MovingActivity, MainActivity::class.java)
                        startActivity(intent)
                        finish()
                    }
                } else {
                    runOnUiThread {
                        Toast.makeText(this@MovingActivity, "Failed:

```

```

${response.message}\nResponse: $responseBody", Toast.LENGTH_LONG).show()
        }
    }
}

override fun onFailure(call: Call, e: IOException) {
    // Handle request failure
    runOnUiThread {
        Toast.makeText(this@MovingActivity, "Request failed: ${e.message}",
        Toast.LENGTH_SHORT).show()
    }
}
}

private fun getAuthenticatedClient(token: String): OkHttpClient {
    val sessionCookie = sharedPreferences.getString("session_cookie", null)

    return OkHttpClient.Builder()
        .cookieJar(MyCookieJar())
        .addInterceptor { chain ->
            val original = chain.request()
            val requestBuilder = original.newBuilder()
                .header("Authorization", "Bearer $token")
                .method(original.method, original.body)

            // Add session cookie to request if it exists
            sessionCookie?.let {
                requestBuilder.header("Cookie", it)
                Log.d(TAG, "Cookie $it")
            }
            val request = requestBuilder.build()
            chain.proceed(request)
        }
        .build()
}
}

```

e. Service

```

import android.content.ContentValues.TAG
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat

```

```

import androidx.core.view.WindowInsetsCompat
import okhttp3.Call
import okhttp3.Callback
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.Response
import org.json.JSONObject
import java.io.IOException

class ServiceActivity : AppCompatActivity() {

    private lateinit var code: String
    private lateinit var noInventarisTextView: TextView
    private lateinit var serviceByEditText: EditText
    private lateinit var serviceDetailsEditText: EditText
    private lateinit var btnSave: Button
    private lateinit var sharedpreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_service)

        sharedpreferences = getSharedPreferences("MyAppPrefs", MODE_PRIVATE)

        code = intent.getStringExtra("code") ?: ""
        noInventarisTextView = findViewById(R.id.tv_no_inventaris)
        noInventarisTextView.text = "No Inventaris : ".plus(code)

        serviceByEditText = findViewById(R.id.service_by)
        serviceDetailsEditText = findViewById(R.id.service_detail)
        btnSave = findViewById(R.id.save_service)

        btnSave.setOnClickListener {
            performService()
        }
    }

    private fun performService() {
        val serverAddress = sharedpreferences.getString("server_address", "")
        if (serverAddress.isNullOrEmpty()) {
            Toast.makeText(this, "Please set the server address in settings",
            Toast.LENGTH_SHORT)
            .show()
            return
        }

        val serviceBy = serviceByEditText.text.toString()
        val serviceDetails = serviceDetailsEditText.text.toString()

        if (serviceBy.isEmpty() || serviceDetails.isEmpty()) {
            Toast.makeText(this, "Service By and Service Details cannot be empty.",
```

```

Toast.LENGTH_SHORT).show()
    return
}

// Create request body with JSON payload
val json = JSONObject().apply {
    put("no_inventaris", code)
    put("service_by", serviceBy)
    put("service_details", serviceDetails)
}

    val requestBody = json.toString().toRequestBody("application/json;
charset=utf-8".toMediaTypeOrNull())

    val request = Request.Builder()
        .url("$serverAddress/api/service/")
        .post(requestBody)
        .build()

    val token = sharedPreferences.getString("auth_token", null)
    val client = token?.let { getAuthenticatedClient(it) }

    // Execute the request asynchronously
    if (client != null) {
        client.newCall(request).enqueue(object : Callback {

            override fun onResponse(call: Call, response: Response) {
                val responseBody = response.body?.string()
                if (response.isSuccessful) {
                    runOnUiThread {
                        Toast.makeText(this@ServiceActivity, "Success!",
                        Toast.LENGTH_SHORT).show()
                        val intent = Intent(this@ServiceActivity, MainActivity::class.java)
                        startActivity(intent)
                        finish()
                    }
                } else {
                    runOnUiThread {
                        Toast.makeText(this@ServiceActivity, "Failed:
${response.message}\nResponse: $responseBody", Toast.LENGTH_LONG).show()
                    }
                }
            }

            override fun onFailure(call: Call, e: IOException) {
                // Handle request failure
                runOnUiThread {
                    Toast.makeText(this@ServiceActivity, "Request failed: ${e.message}",
                    Toast.LENGTH_SHORT).show()
                }
            }
        })
    }
}

private fun getAuthenticatedClient(token: String): OkHttpClient {

```

```

val sessionCookie = sharedPreferences.getString("session_cookie", null)

return OkHttpClient.Builder()
    .cookieJar(MyCookieJar())
    .addInterceptor { chain ->
        val original = chain.request()
        val requestBuilder = original.newBuilder()
            .header("Authorization", "Bearer $token")
            .method(original.method, original.body)

        // Add session cookie to request if it exists
        sessionCookie?.let {
            requestBuilder.header("Cookie", it)
            Log.d(TAG, "Cookie $it")
        }
    }

    val request = requestBuilder.build()
    chain.proceed(request)
}
.build()
}

}

```

f. Histori

```

import android.content.ContentValues.TAG
import android.content.Intent
import android.content.SharedPreferences
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.ImageView
import android.widget.ProgressBar
import android.widget ScrollView
import android.widget.TextView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import com.google.mlkit.vision.common.InputImage
import com.google.mlkit.vision.text.TextRecognition
import com.google.mlkit.vision.text.latin.TextRecognizerOptions
import okhttp3.Call
import okhttp3.Callback
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.Response
import org.json.JSONObject
import java.io.IOException

class ReviewActivity : AppCompatActivity() {

```

```

private lateinit var imageViewCropped: ImageView
private lateinit var readableNo: TextView
private lateinit var noInventaris: TextView
private lateinit var item: TextView
private lateinit var spesifikasi: TextView
private lateinit var lokasi: TextView
// private lateinit var department: TextView
private lateinit var user: TextView
private lateinit var kondisi: TextView
private lateinit var scrollView: ScrollView
private lateinit var btnTambahInventaris: Button
private lateinit var btnChangeStatus: Button
private lateinit var progressBar: ProgressBar
private lateinit var sharedPreferences: SharedPreferences

private lateinit var no_inventaris_cleaned: String

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContentView(R.layout.activity_review)

    imageViewCropped = findViewById(R.id.imageViewCropped)
    readableNo = findViewById(R.id.readableNo)
    noInventaris = findViewById(R.id.no_inventaris)
    item = findViewById(R.id.item)
    spesifikasi = findViewById(R.id.spesifikasi)
    // department = findViewById(R.id.tv_department)
    lokasi = findViewById(R.id.lokasi)
    user = findViewById(R.id.user)
    kondisi = findViewById(R.id.kondisi)
    scrollView = findViewById(R.id.scrollview)
    btnTambahInventaris = findViewById(R.id.btnAddInventaris)
    btnChangeStatus = findViewById(R.id.btnChangeStatus)
    progressBar = findViewById(R.id.loading)
    sharedPreferences = getSharedPreferences("MyAppPrefs", MODE_PRIVATE)

    val imagePath = intent.getStringExtra("imagePath") ?: return

    val bitmap = BitmapFactory.decodeFile(imagePath)
    imageViewCropped.setImageBitmap(bitmap)

    // cropAndShowImage(bitmap)
    runTextRecognition(bitmap)
}

private fun runTextRecognition(bitmap: Bitmap) {
    val image = InputImage.fromBitmap(bitmap, 0)
    val recognizer =
        TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS)

    recognizer.process(image)
        .addOnSuccessListener { visionText ->
            processTextRecognitionResult(visionText)
        }
        .addOnFailureListener { e ->
}

```

```

        e.printStackTrace()
    }
}

private fun processTextRecognitionResult(text: com.google.mlkit.vision.text.Text) {
    val resultText = text.text
    // Menghilangkan karakter selain angka dan titik
    val filteredText = resultText.replace(Regex("[^0-9.]"), "")
    readableNo.text = "Hasil scan : $filteredText"
    no_inventaris_cleaned = filteredText
    checkInventory(filteredText)
}

private fun getAuthenticatedClient(token: String): OkHttpClient {
    val sessionCookie = sharedPreferences.getString("session_cookie", null)

    return OkHttpClient.Builder()
        .cookieJar(MyCookieJar())
        .addInterceptor { chain ->
            val original = chain.request()
            val requestBuilder = original.newBuilder()
                .header("Authorization", "Bearer $token")
                .method(original.method, original.body)

            // Add session cookie to request if it exists
            sessionCookie?.let {
                requestBuilder.header("Cookie", it)
                Log.d(TAG, "Cookie $it")
            }
        }

    val request = requestBuilder.build()
    chain.proceed(request)
}
.build()

private fun checkInventory(code: String) {
    val serverAddress = sharedPreferences.getString("server_address", "")
    if (serverAddress.isNullOrEmpty()) {
        Toast.makeText(this, "Please set the server address in settings",
        Toast.LENGTH_SHORT)
        .show()
        return
    }

    // Create request body with JSON payload
    val json = JSONObject().apply {
        put("inventory_no", code)
    }
    val requestBody =
        json.toString().toRequestBody("application/json; charset=utf-8".toMediaTypeOrNull())

    val token = sharedPreferences.getString("auth_token", null)
    val client = token?.let { getAuthenticatedClient(it) }

    // Create request to post data to the server
    val request = Request.Builder()

```

```

.url("$serverAddress/api/check_inventory/") // Change to your check_inventory URL
.post(requestBody)
.build()

// Execute the request asynchronously
if (client != null) {
    client.newCall(request).enqueue(object : Callback {

        override fun onResponse(call: Call, response: Response) {
            val responseBody = response.body?.string()

            if (!response.isSuccessful || responseBody == null) {
                runOnUiThread {
                    showPopup("Inventory check failed")
                }
                return
            }

            val jsonResponse = JSONObject(responseBody)
            if (jsonResponse.has("error")) {
                runOnUiThread {
                    showPopup("Inventory Tidak Ditemukan!")
                    showAddInventoryButton()
                }
            } else {
                val itemJson = jsonResponse.getJSONObject("item")
                runOnUiThread {
                    showPopup("Inventory Ditemukan!")
                    populateInventoryDetails(itemJson)
                }
            }
        }

        override fun onFailure(call: Call, e: IOException) {
            // Handle request failure
            runOnUiThread {
                showPopup("Request failed: ${e.message}")
            }
        }
    })
}
}

private fun populateInventoryDetails(itemJson: JSONObject ) {
    noInventaris.text = itemJson.optString("no", "-")
    item.text = itemJson.optString("name", "-")
    spesifikasi.text = itemJson.optString("specifications", "-")
    department.text = itemJson.optString("department", "-")
    lokasi.text = itemJson.optString("location", "-")
    user.text = itemJson.optString("pic", "-")
    kondisi.text = itemJson.optString("condition", "-")
    readableNo.text = readableNo.text.toString().plus(" (ditemukan)")
    progressBar.visibility = View.GONE
    scrollView.visibility = View.VISIBLE
    btnTambahInventaris.visibility = View.GONE
    btnChangeStatus.setOnClickListener {
        val intent = Intent(this, SelectChangeStatusActivity::class.java)
        intent.putExtra("code", no_inventaris_cleaned)
    }
}

```

```
        startActivity(intent)
    }
}

private fun showAddInventoryButton() {
    progressBar.visibility = View.GONE
    scrollView.visibility = View.GONE
    btnTambahInventaris.visibility = View.VISIBLE
    readableNo.text = readableNo.text.toString().plus(" (tidak ditemukan)")
    btnTambahInventaris.setOnClickListener {
        val intent = Intent(this, AddInventoryActivity::class.java)
        intent.putExtra("code", no_inventaris_cleaned)
        startActivity(intent)
    }
}

private fun showPopup(message: String) {
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
}
}
```

Lampiran 7. Gradle Dependencies OCR

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.jetbrains.kotlin.android)
}

android {
    namespace = "com.project.inventorymanagement"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.project.inventorymanagement"
        minSdk = 27
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
}

compileOptions {
    sourceCompatibility = JavaVersion.VERSION_1_8
    targetCompatibility = JavaVersion.VERSION_1_8
}

kotlinOptions {
    jvmTarget = "1.8"
}

buildFeatures {
    viewBinding = true
}

dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    implementation(libs.androidx.annotation)
    implementation(libs.androidx.lifecycle.livedata.ktx)
    implementation(libs.androidx.lifecycle.viewmodel.ktx)
    implementation(libs.androidx.camera.core)
    implementation(libs.androidx.camera.camera2)
    implementation(libs.androidx.camera.lifecycle)
    implementation(libs.androidx.camera.view)
    implementation(libs.androidx.camera.extensions)
}
```

```
implementation("com.google.mlkit:text-recognition:16.0.0")
implementation("com.squareup.okhttp3:okhttp:4.9.3")
```

```
testImplementation(libs.junit)
androidTestImplementation(libs.androidx.junit)
androidTestImplementation(libs.androidx.espresso.core)
```

```
}
```

Lampiran 8. Video Timer Proses Login

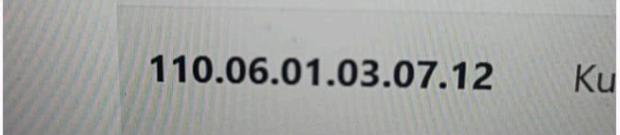
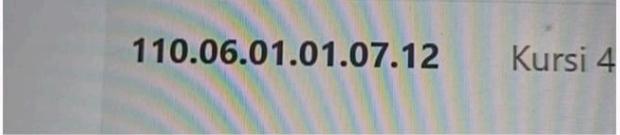
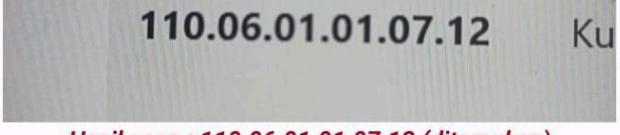
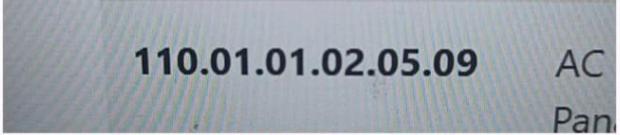
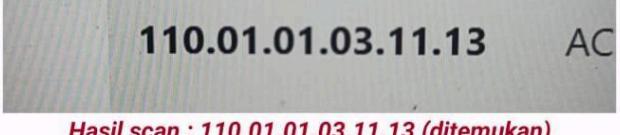
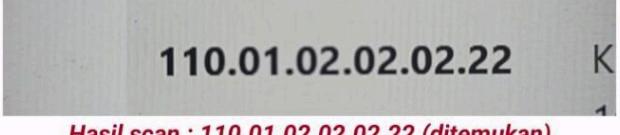
<https://drive.google.com/file/d/1ladt5ErAbaXS2F3YHAlbnrYejYZBn7Fz/view?usp=sharing>

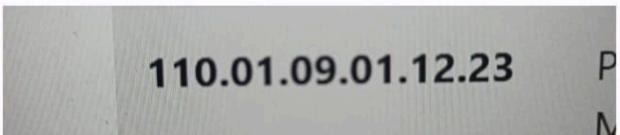
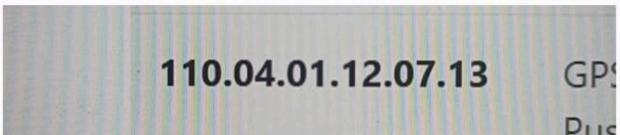
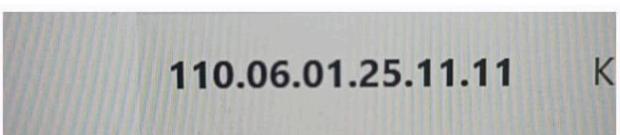
Lampiran 9. Video Perhitungan waktu

Timer_digitRecognition.mp4:

https://drive.google.com/file/d/1_O_Ox_cWj8x4nJDIV-iM_1M03Qb9CZ5i/view?usp=sharing

Lampiran 10. Tabel Uji Fungsionalitas Akurasi

No	Pengujian OCR	Berhasil	Tidak
1	 110.06.01.03.07.12 Ku <i>Hasil scan : 110.06.01.03.07.12 (ditemukan)</i>	✓	
2	 110.06.01.01.07.12 Kursi 4 <i>Hasil scan : 110.06.01.01.07.124 (tidak ditemukan)</i>		✓
3	 110.06.01.01.07.12 Ku <i>Hasil scan : 110.06.01.01.07.12 (ditemukan)</i>	✓	
4	 110.01.01.02.05.09 AC <i>Hasil scan : 110.01.01.02.05.09 (ditemukan)</i>	✓	
5	 110.01.01.03.11.13 AC <i>Hasil scan : 110.01.01.03.11.13 (ditemukan)</i>	✓	
6	 110.01.02.02.02.22 K <i>Hasil scan : 110.01.02.02.02.22 (ditemukan)</i>	✓	

7	 <i>Hasil scan : 110.01.09.01.12.23 (ditemukan)</i>	✓	
8	 <i>Hasil scan : 110.04.01.12.07.13 (ditemukan)</i>	✓	
9	 <i>Hasil scan : 110.06.01.25.11.11 (ditemukan)</i>	✓	
10	 <i>Hasil scan : 110.06.01.33.11.11 (ditemukan)</i>	✓	