BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama melaksanakan magang di PT Satya Solusindo Indonesia, penulis ditempatkan pada posisi *intern* AI *engineer* di bawah supervisi oleh Charles H. Langkoh sebagai *lead software engineer*. Dalam perannya, penulis bertanggung jawab membangun seluruh sistem pada produk *storage availability system*, mulai dari perancangan *frontend, backend,* dan pelatihan model AI. Pada perancangan *frontend,* penulis bertugas untuk merancang UI (*User Interface*) dan UX (*User Experience*) untuk *website*. Pada bagian *backend,* penulis bertanggung jawab mengintegrasikan basis data (*database)* dan *object detection* ke dalam sistem *website*. Sedangkan pada tahap pelatihan model, penulis memiliki tugas untuk mengumpulkan dataset dan melatih model hingga memperoleh tingkat akurasi yang diinginkan.

Alur kerja tugas magang pada divisi ini, dimulai dari perancangan *template website* menggunakan *software* Figma. Proses ini mencangkup perancangan *template* untuk *login page, home page, camera page,* dan *model page*. Setelah proses perancangan *template* selesai, dilakukan sesi presentasi kepada CEO (*Chief Executive Officer*) Satsindo, yaitu Henry Susanto. Pada tahap ini, *template* dan alur kerja akan dievaluasi untuk menghasilkan *feedback.* Setelah dilakukan revisi berdasarkan *feedback* dan disetujui oleh CEO, pekerjaan dilanjutkan ke tahap pembuatan *mockup* produk. Tahap ini diawali dengan penentuan program, *framework,* dan *library* yang akan digunakan. Untuk pembuatan template *website* (*frontend*), digunakan tiga bahasa pemrograman : HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*), JavaScript. Pada pengembangan *backend,* perlu ditentukan *framework* yang akan digunakan. *Framework* merupakan kumpulan kode, fungsi, dan *library* yang digunakan untuk mempermudah pengembangan *website*. Untuk produk ini, penulis memilih untuk menggunakan *framework* Flask. Hal tersebut didasari pada kelebihan Flask yang berbasis bahasa

pemrograman Python dan fleksibel terhadap *library* lainnya, seperti `Ultralytics` dan `OpenCV`. Pada proyek ini, sebanyak 11 *library* digunakan meliputi : `Flask`, `base64`, `openCV`, `Numpy`, `time`, `request`, `threading`, `os`, `mysql.connector`, `dotenv`, dan `Ultralytics`. Sebelum pengembangan dimulai pemilihan bahasa pemrograman, *framework*, dan *library* terlebih dahulu perlu disetujui oleh *lead software engineering*. Pembuatan *website* juga diikuti dengan pelatihan model AI. Untuk pelatihan model, langkah pertama adalah mengumpulkan dataset berupa gambar dari objek yang ingin dideteksi, yaitu kardus. Dataset kemudian diberi label (*labeling*) menggunakan *software roboflow* sebelum digunakan melatih model AI. Model yang dihasilkan akan diuji terlebih dahulu, untuk mengukur nilai F1 *score*. Hal tersebut bertujuan untuk mengetahui kualitas akurasi dari model yang akan digunakan. Minimum nilai F1 *score* akan ditentukan oleh *lead software engineering.* Sehingga jika nilai F1 *score* tidak memenuhi, maka akan dilakukan pengumpulan dan pelatihan dataset kembali. Setelah model AI memenuhi kriteria, model tersebut akan diintegrasikan dengan *website* yang telah dibuat. Setelah produk selesai pada tahap *mockup*, dilakukan presentasi kepada supervisi dan CEO untuk memastikan kualitas dari produk. Jika dinyatakan selesai, produk akan didokumentasikan dalam bentuk video dan diteruskan ke divisi *sales* untuk dipasarkan.

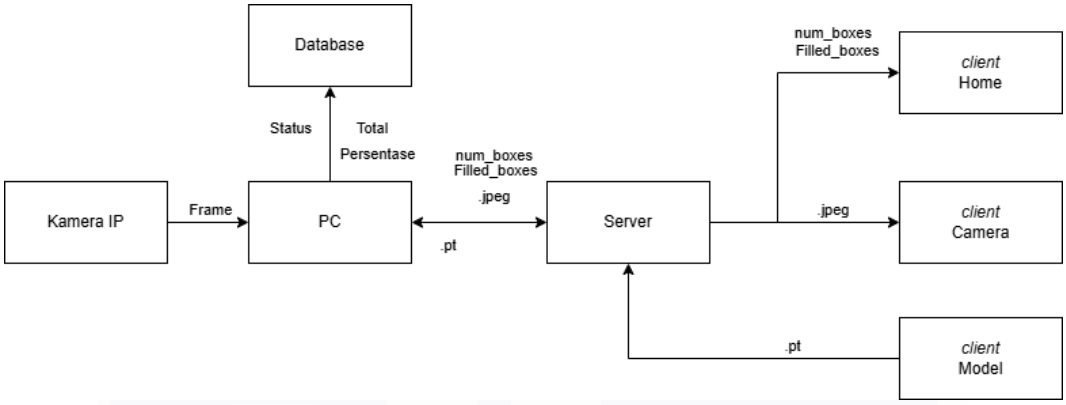
3.2 Tugas dan Uraian Kerja Magang

Selama melaksanakan magang, penulis mempelajari banyak hal baru, khususnya terkait otomasi industri dan teknologi AI. Pada awal masa magang, penulis diberikan pilihan proyek yang nantinya dikerjakan sebagai bagian dari proyek magang. langkah pertama setelah memilih proyek adalah merancang proyek tersebut. Pada tahap ini, penulis terlebih dahulu mempresentasikan rancangannya kepada CEO dan supervisi untuk mendapatkan evaluasi terkait alur kerja proyek. Penulis menerima beberapa masukan, salah satunya adalah menjadikan kamera IP sebagai CCTV (*Closed-Circuit Television*) tambahan agar operator dapat melakukan verifikasi secara *real time*. Setelah itu, penulis melanjutkan membuat tampilan *website* yang diintegrasikan dengan *object*

detection. Selain itu, penulis diajarkan cara membuat *state transition diagram* dan diberikan tugas untuk membuat *state diagram* untuk subsistem tertentu untuk PT HB. Penulis juga mempelajari penerapan teknologi AI, khususnya dalam industri. Misalnya, mendeteksi produk cacat (*reject*) dengan menggunakan *object classification*. sehingga output yang dihasilkan nantinya akan berupa *Good* (Produk tidak mengalami kerusakan) dan *Bad* (Produk mengalami kerusakan). Penulis juga diberi tugas untuk menguji tiga *library text recognition*, yaitu : `EasyOCR`, `Pytesseract`, dan `PaddleOCR`. Dari hasil pengujian diperoleh bahwa, `PaddleOCR` memiliki akurasi yang jauh lebih baik dibandingkan dengan `EasyOCR` dan `Pytesseract`. Penulis juga ditugaskan untuk belajar mengenai konsep NLP (*Natural Language Processing*), yang nantinya akan diterapkan pada proyek ChatBot dan *Multi-Model* AI for *Video Analysis.*

Selama proses magang, penulis mengerjakan *daily tasks* sebagai bagian dari jam kerja, yang meliputi proyek *Storage Availability System* dan *Multi-Model* AI for *Video Analysis.* Penulis memilih proyek *storage availability system* untuk dijadikan sebagai proyek utama dalam laporan magang. Dalam proyek ini, penulis bertanggung jawab dalam membangun keseluruhan sistem, termasuk pengembangan *frontend, backend,* dan pelatihan model AI. Proyek ini berfokus pada penerapan teknologi AI untuk memantau ketersediaan ruang penyimpanan secara *real time* melalui sebuah *website*. Sistem ini dirancang memungkinkan pengguna untuk mengakses data ruang penyimpanan dengan mudah dan efisien, yang dapat meningkatkan produktivitas dalam pengolahan ruang penyimpanan.

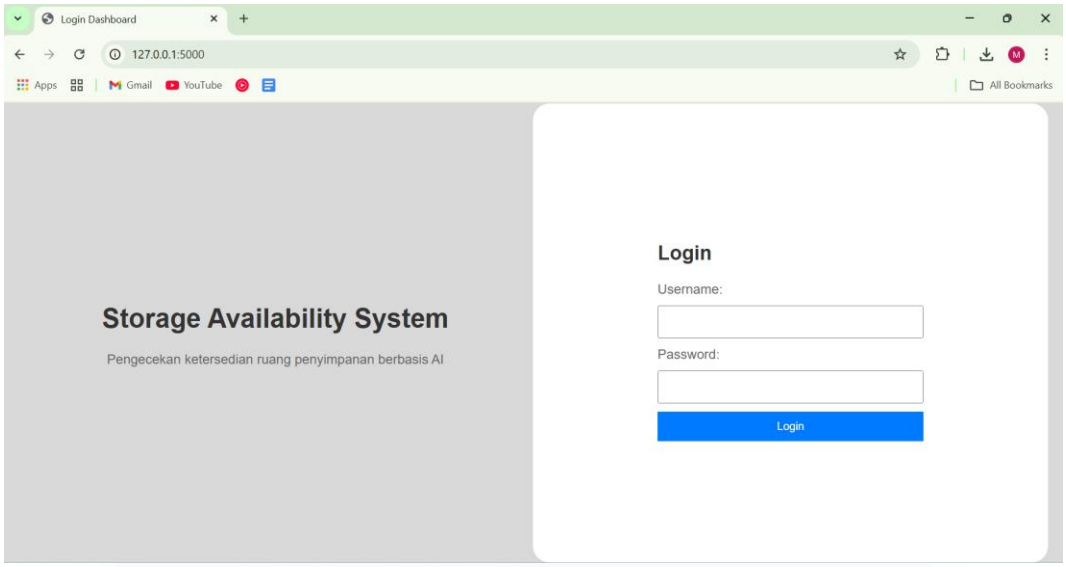
Alur sistem pada proyek *storage availability system* dapat dilihat pada DFD berikut.



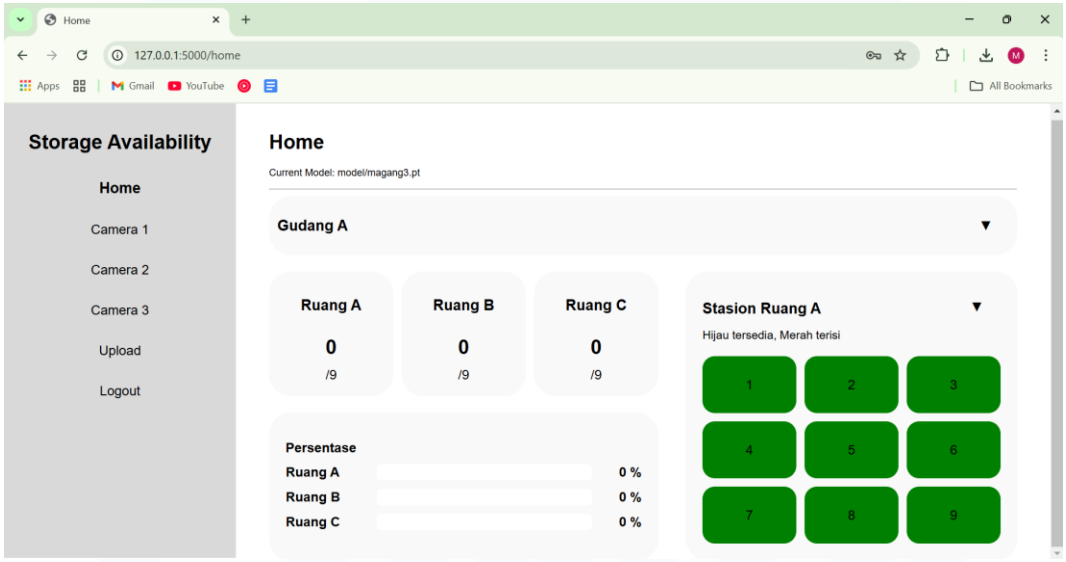
Gambar 3.1. DFD Storage Availability System

Pada Gambar-3.1, ditampilkan alur kerja sistem melalui DFD. Sistem ini melibat PC yang terhubung dengan kamera IP, di mana kamera IP akan mengirimkan data berupa *frame* menggunakan metode RTSP (*Real Time Streaming Protocol*). *Frame* yang diterima akan diolah oleh program *object detection* untuk menghasilkan output berupa *Num_boxes* (jumlah deteksi objek) dan *Filled_boxes* (posisi objek terhadap *grid*). Output ini akan dikirimkan ke server menggunakan metode HTTP POST dan GET. PC juga akan mengirimkan data berupa *image* dalam format .jpeg ke server menggunakan metode HTTP Multipart. Data yang telah dikirimkan ke server kemudian akan diteruskan ke *client* untuk diolah dan divisualisasikan melalui *website*. Server juga akan menerima data dari *client* (model) berupa model AI dalam format .pt yang akan dikirimkan ke PC untuk diintegrasikan ke dalam program *object detection.* Data-data yang telah dikirimkan ke server akan dikirimkan ke *client* untuk nantinya diolah dan divisualisasikan dalam *website*. Server akan menerima data dari *client* model berupa model AI dalam format .pt yang akan dikirimkan ke PC untuk nantinya diolah dalam program *objek detection*. Selain itu, PC juga terhubung dengan *database* untuk mengirimkan data seperti *date (*YYYY-MM-DD HH:MM:SS*)* status (persentase ketersediaan ruang penyimpanan), dan total (jumlah ruang penyimpanan yang terisi).

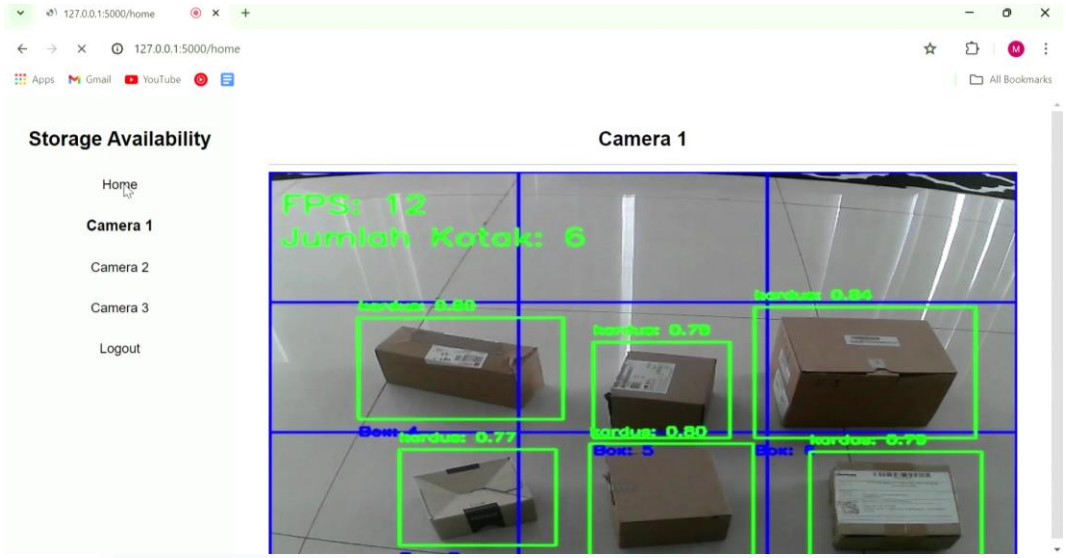
Pada bagian *website* terbagi menjadi 4 halaman yang meliputi : *login page, home page, camera page,* dan *model page* seperti pada Gambar-3.1 dibawah ini.


Login Dashboard
127.0.0.1:5000
Apps
Gmail
YouTube
All Bookmarks
Storage Availability System
Pengecekan ketersedian ruang penyimpanan berbasis AI
Login
Username:
Password:
Login

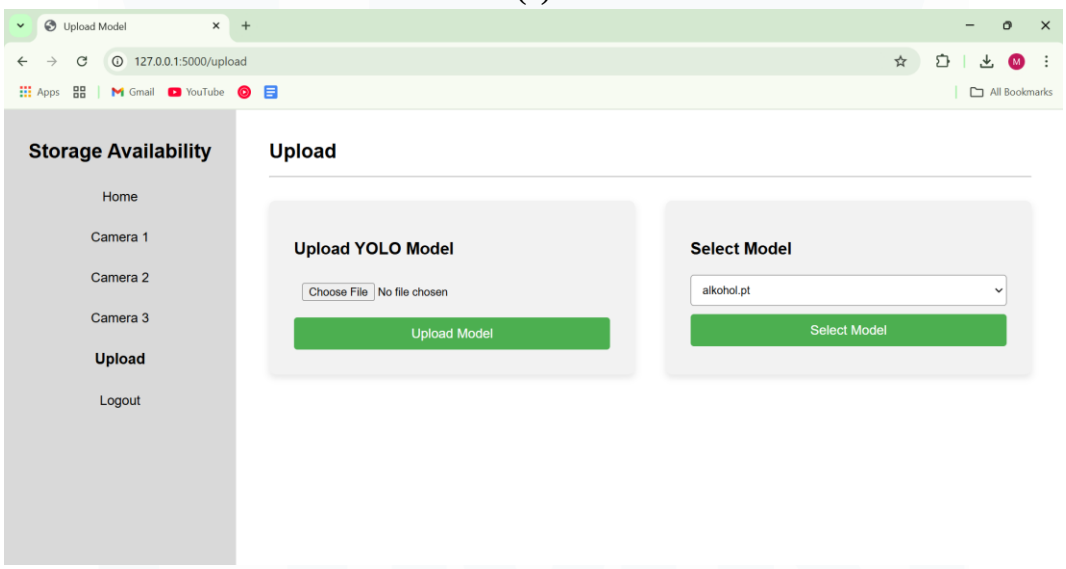
(a)


Home
127.0.0.1:5000/home
Apps
Gmail
YouTube
All Bookmarks
Storage Availability
Home
Camera 1
Camera 2
Camera 3
Upload
Logout
Home
Current Model: model/magang3.pt
Gudang A
Ruang A
0
/9
Ruang B
0
/9
Ruang C
0
/9
Stasion Ruang A
Hijau tersedia, Merah terisi
1
2
3
4
5
6
7
8
9
Persentase
Ruang A
0 %
Ruang B
0 %
Ruang C
0 %

(b)



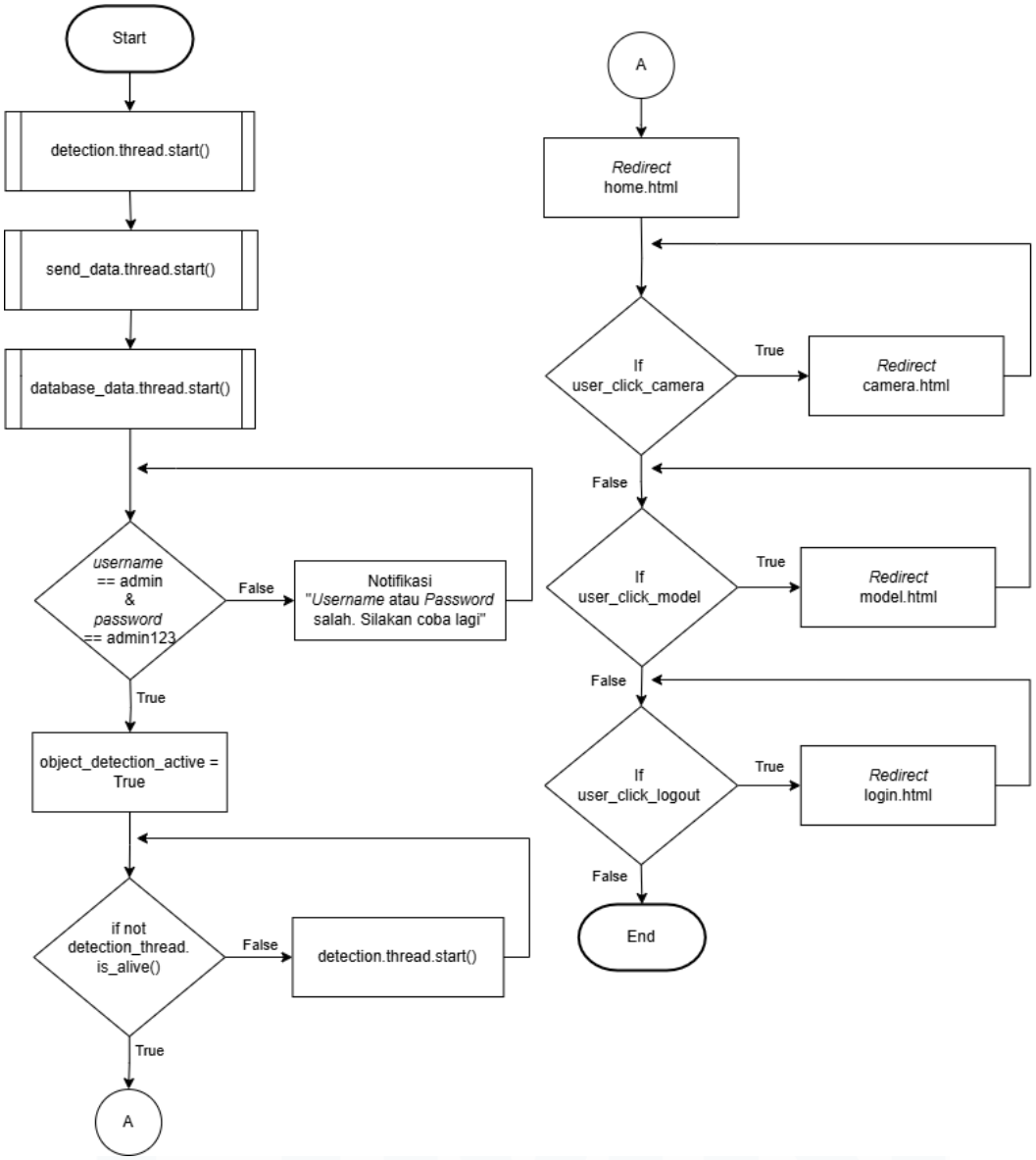
(c)



(d)

Gambar 3.2. *Login Page* (a) *Home Page* (b) *Camera Page* (c) *Upload Page* (d)

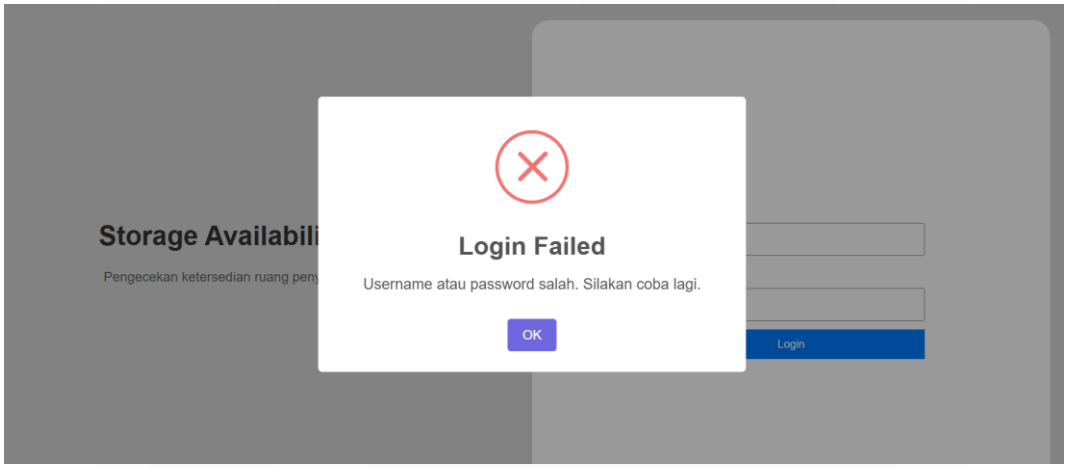
Halaman *website* yang ditampilkan pada Gambar-3.2, memiliki peran penting dalam mendukung implementasi *storage availability system.* Setiap fitur pada halaman tersebut, dirancang untuk mempermudah pengguna dalam memantau kondisi ruang penyimpanan secara efektif. Alur kerja program *storage availability system* dapat dijelaskan melalui *flowchart* yang memetakan setiap proses utama.



Gambar 3.3 *Flowchart Storage Availability System*

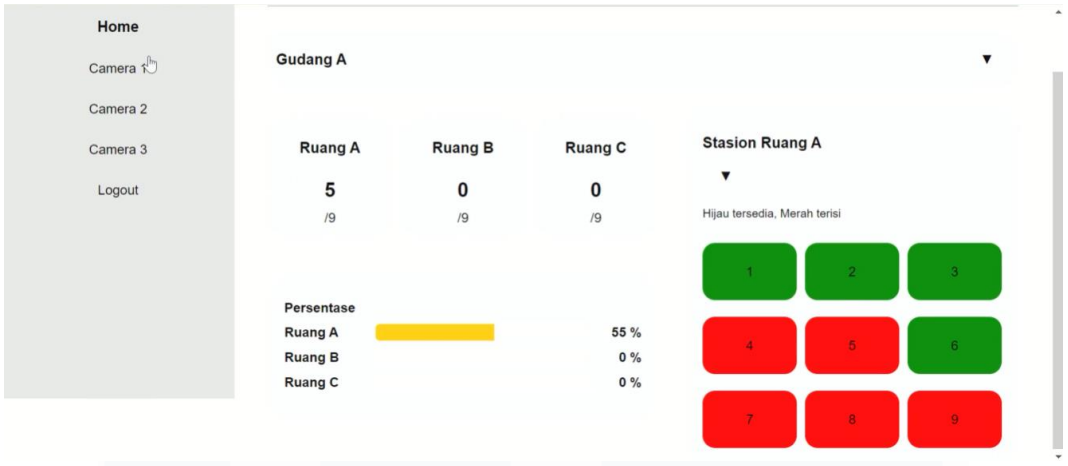
Pada Gambar-3.3, diperlihatkan bahwa *function detection, send_data,* dan *database* dijalankan terlebih dahulu sebelum pengguna memasukan *username* dan *password. Function* pada *flowchart* diimplementasikan sebagai *thread.* Artinya, *function* tersebut dijalankan secara paralel tanpa perlu saling menunggu. Tujuan utama dari implementasi *threading* adalah memastikan proses pemantauan dan pengiriman data tetap berlangsung secara terus-menerus, bahkan ketika pengguna belum atau tidak sedang aktif menggunakan *website*. Pada *login page,* pengguna diminta untuk memasukan *username* dan *password* yang telah terdaftar sebelumnya. Setelah

berhasil *login*, pengguna akan ke *home page*. Namun, apabila pengguna salah memasukan *username* atau *password*, sistem akan menampilkan notifikasi sebagai peringatan.

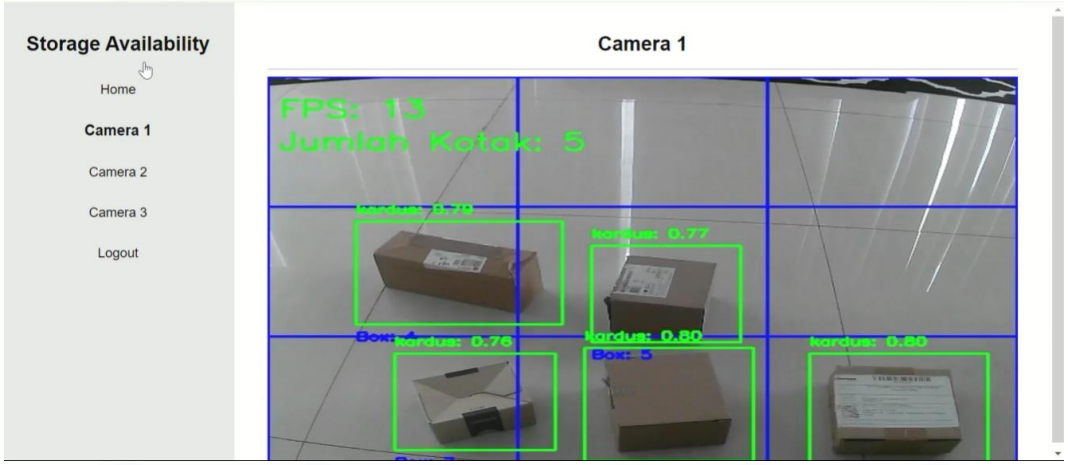


Gambar 3.4. Notifikasi Peringatan Pada Login Page

Pada Gambar-3.4 menunjukan notifikasi peringatan yang ditampilkan ketika pengguna salah memasukan *username* atau *password.* Notifikasi dibuat dengan menggunakan *SweetAlert.* Pada *home page* seperti Gambar-3.2 bagian (a), pengguna dapat menavigasi halaman *website* menggunakan *text* yang terdapat pada bagian *sidebar home page.* Selain itu, pengguna juga dapat memonitoring setiap gudang atau ruang penyimpanan menggunakan fitur *dropdown button*. Di sisi kanan, terdapat *container* untuk bagian *station*, yang terdiri dari beberapa *box* berwarna hijau. Warna hijau menunjukan *station* pada ruang penyimpanan sedang kosong. *Box* tersebut dapat juga berubah warna menjadi merah ketika pada stasiun tersebut sedang terisi barang (kardus).



(a)

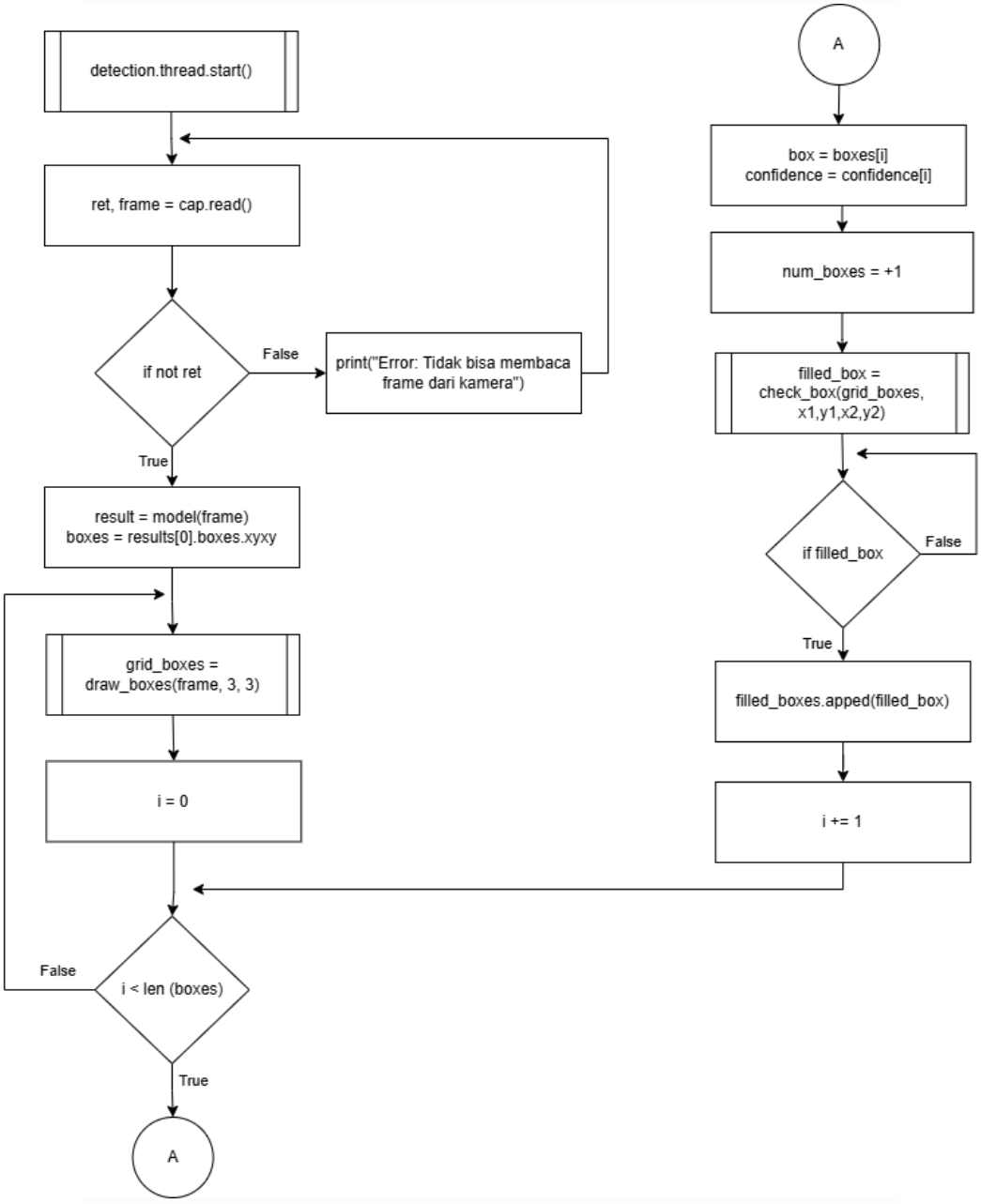


(b)

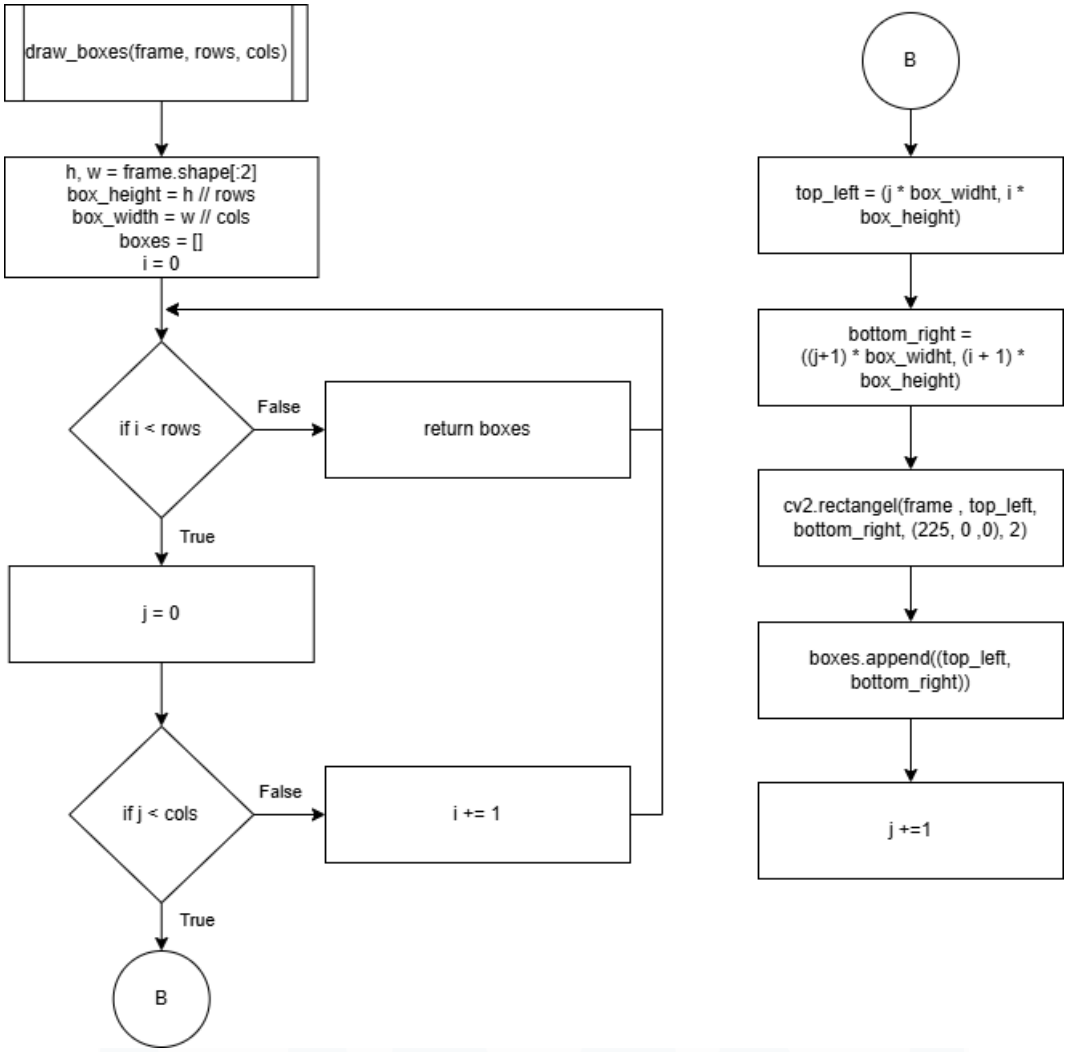
Gambar 3.5. Tampilan *Home Page* Ketika Program Pendeteksi Objek (a)Tampilan *Home Page* Ketika Program Pendeteksi Objek (b)

Pada Gambar-3.5 bagian (a), terlihat bahwa *box* mengalami perubahan warna ketika *frame* mendeteksi adanya kardus pada *grid* tertentu. setiap *grid* pada *frame* mewakili *box*, sehingga warna atau status dari *box* dipengaruhi oleh data yang dikirimkan program *object detection* ke *client* menggunakan API (*Application Programming Interface*). Pada bagian Gambar-3.3 bagian (b), terlihat *frame* pada kamera IP ditampilkan secara *real time*. Proses tersebut dapat dilihat pada Gambar-3.1, dimana PC akan mengirimkan data berupa gambar dalam format .jpeg ke server, yang diteruskan ke *client Camera*. Pengiriman *frame* dilakukan menggunakan metode HTTP Multipart, yang lebih efisien untuk mengirim data

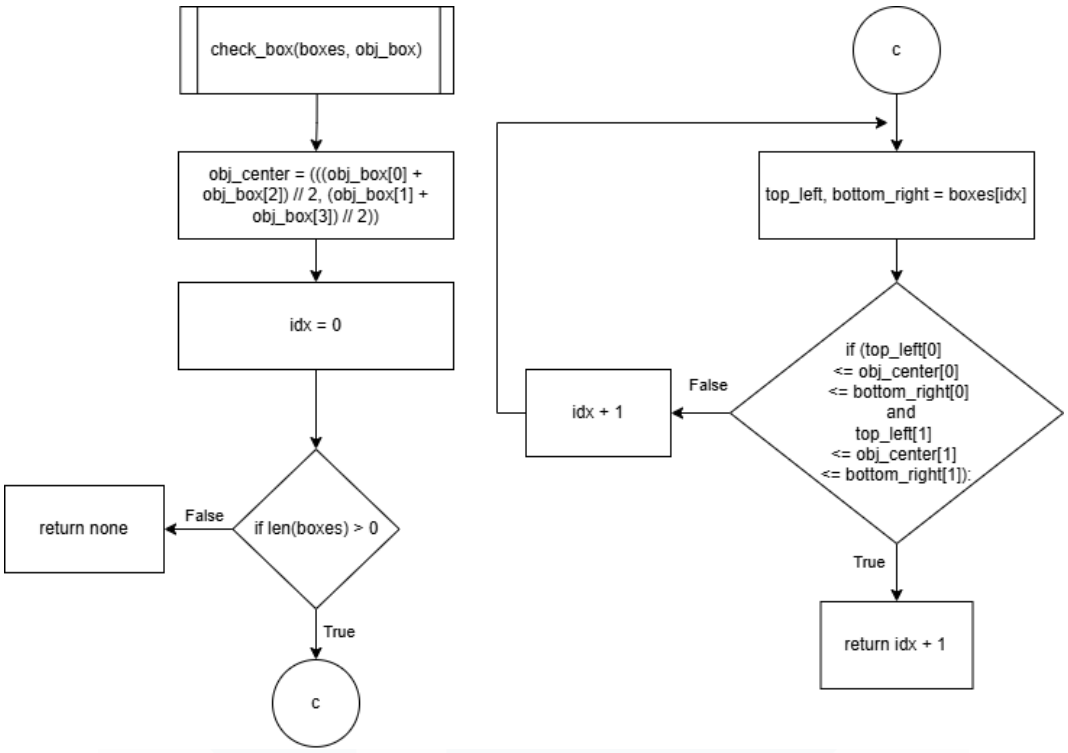
dalam format .jpeg. Hal ini dikarenakan HTTP Multipart mengirim data dalam format biner, sehingga lebih optimal untuk transfer file gambar. Berikut tampilan *flowchart* program untuk proses pendeteksian objek.



(a)


draw_boxes(frame, rows, cols)
h, w = frame.shape[:2]
box_height = h // rows
box_width = w // cols
boxes = []
i = 0
if i < rows
False
return boxes
True
j = 0
if j < cols
False
i += 1
True
B
B
top_left = (j * box_widht, i * box_height)
bottom_right = ((j+1) * box_widht, (i + 1) * box_height)
cv2.rectangel(frame , top_left, bottom_right, (225, 0 ,0), 2)
boxes.append((top_left, bottom_right))
j +=1

(b)

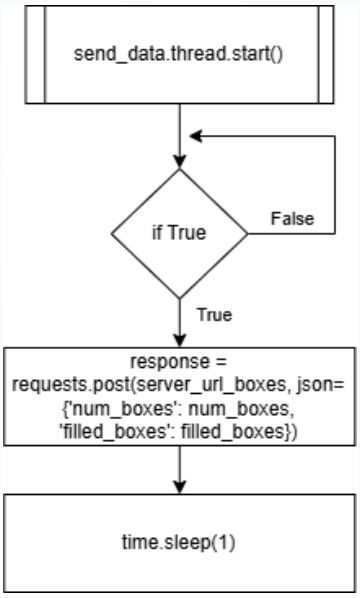


(c)

Gambar 3.6 *Flowchart Function detection* (a) *Flowchart Function draw_boxes* (b) *Flowchart Function check_box (c)*

pada Gambar-3.6 bagian a, terlihat alur kerja program untuk proses pendeteksian. Langkah pertama adalah mendeteksi *frame* dari kamera IP menggunakan model AI yang telah dilatih. Setelah *frame* diperoleh, program juga akan memvisualisasikan *grid* berdasarkan jumlah kolom dan baris yang diinginkan menggunakan *function draw_boxes* sebagaimana terlihat pada Gambar-3.5 bagian b. *Function* ini memungkinkan pengguna untuk mengatur jumlah *columns* dan *rows* dengan mudah pada *grid.* Penyesuaian *grid* dilakukan berdasarkan ukuran *frame*, sehingga proporsi *grid* akan selalu sesuai. Selanjutnya, program akan melakukan *looping* berdasarkan jumlah hasil deteksi objek. Dalam proses ini, dua variabel akan dihitung, yaitu : *num_boxes* dan *filled_boxes*. Variabel *num_boxes* merepresentasikan jumlah objek yang terdeteksi sedangkan variabel *filled_boxes* merepresentasikan *grid* indeks yang terisi. Kedua variabel tersebut adalah variabel global sehingga dapat diakses penuh oleh *function lainnya.* Nantinya kedua variabel tersebut, yang akan dikirimkan ke server oleh *function send_data.thread.start().*

Pengecekan *grid* mana saja yang telah terisi akan dilakukan oleh *function check_boxes()* seperti pada Gambar-3.4 bagian c. *function* akan memeriksa titik tengah dari *bounding box* objek berada pada *range* (batas koordinat sudut kiri-atas dan kanan-bawah) dari salah satu *grid*. Jika titik tengah tersebut berada pada *range*, maka *function* akan mengembalikan nilai indeks dari *grid* tersebut (dimulai dari 1). Akan tetapi, jika tidak *function* akan mengembalikan nilai *none*. Proses tersebut akan dilakukan hingga semua *grid* selesai diperiksa.

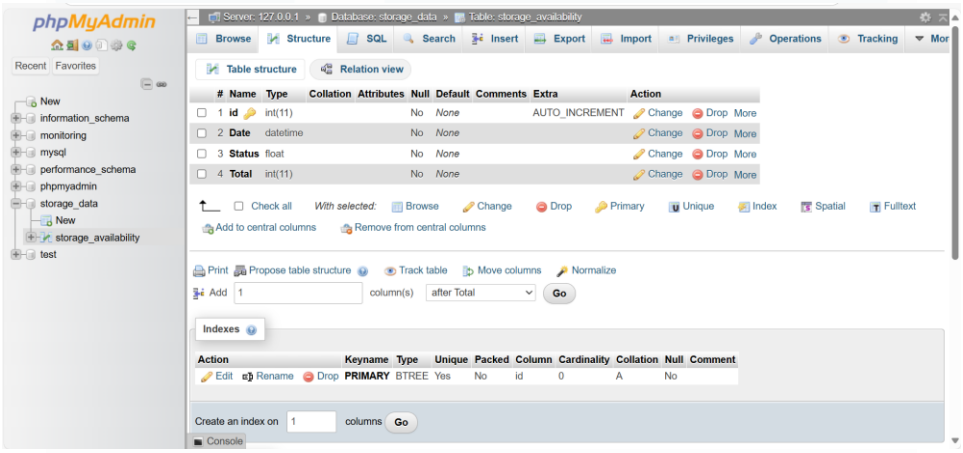


Gambar 3.7 *Flowchart Function Send_data*

Proses pengiriman data dari variabel *num_boxes* dan *filled_boxes* akan dilakukan oleh *function send_data* seperti pada Gambar-3.7. Data tersebut nantinya akan divisualisasikan melalui *website* seperti pada Gambar 3.3 bagian a. Pengiriman data dapat berlangsung karena kedua variabel tersebut merupakan *global* variabel sehingga dapat diakses penuh oleh *function* lain. Pada Gambar-3.5, dapat dilihat proses pengiriman akan menggunakan metode HTTP (HyperText Transfer Protocol) POST yang akan diterima oleh server menggunakan metode HTTP GET. Proses pengiriman tersebut akan dilakukan selama 1 detik sekali.

Pada *model page*, seperti yang ditunjukkan Gambar-3.2 bagian (d), pengguna dapat mengganti model AI yang akan digunakan. Fitur ini dirancang untuk meningkatkan fleksibilitas proyek, Sehingga mungkinkan pengguna untuk

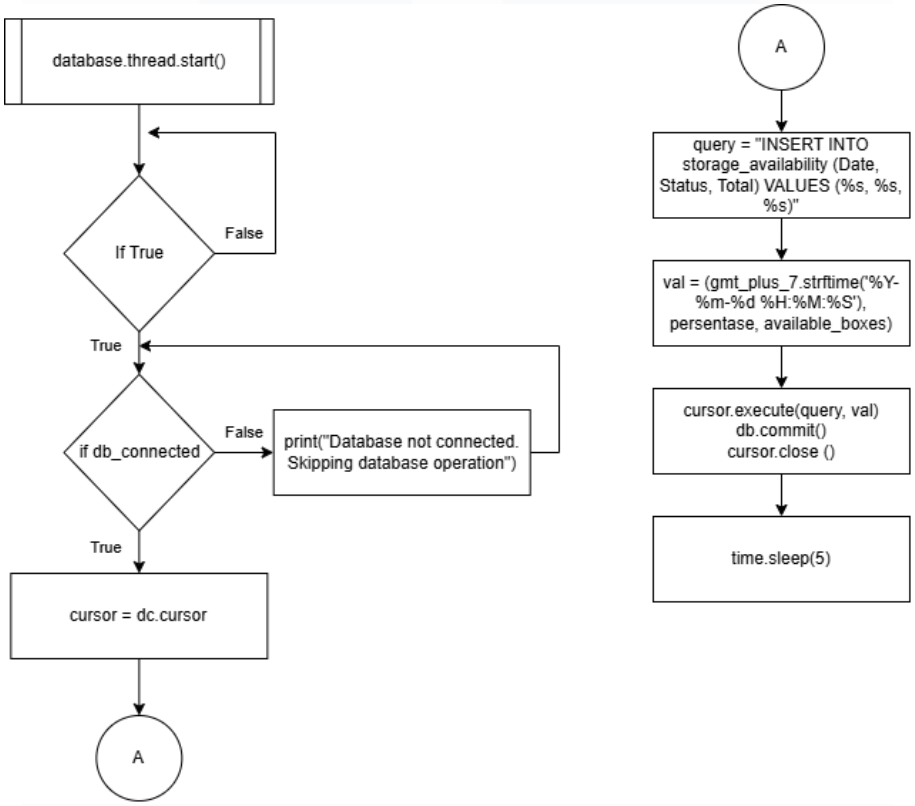
mengganti objek yang ingin dideteksi. Selain itu, pengguna juga dapat meningkatkan atau mengganti model AI dengan mudah tanpa harus menggantinya melalui program (*backend*). Proyek *Storage Availability System* juga akan diintegrasikan dengan *database* MariaDB untuk melakukan penyimpanan data secara terstruktur. Akses *database* dapat menggunakan aplikasi XAMPP, yang menyediakan antarmuka berbasis web melalui PHPMyAdmin. Melalui web tersebut, pengguna untuk dapat membuat, membaca, memperbaharui, dan menghapus data pada *database*, sehingga dapat mempermudah pengelolaan dan monitoring data pada sistem.



Gambar. 3.8. *Database Storage Availability System*

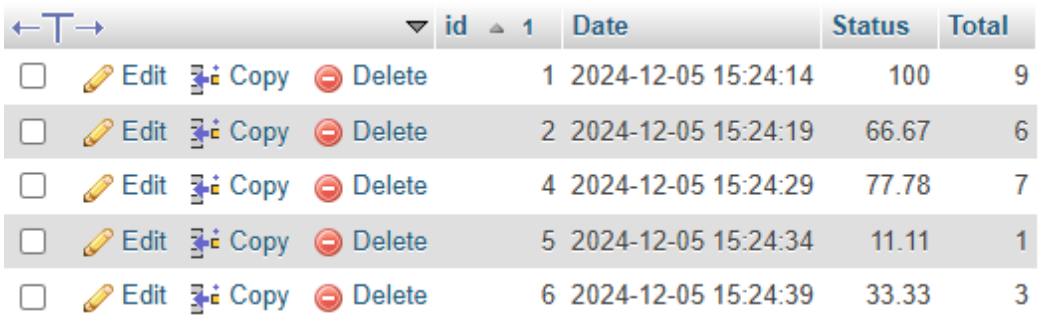
Pada Gambar-3.8, dapat dilihat bahwa terdapat 4 kolom pada tabel. Kolom pertama merupakan kolom *id,* yang merupakan *primary key*. Data tersebut digunakan untuk mengidentifikasi setiap baris secara unik, sehingga nilai pada kolom *id* tidak boleh duplikat. Ketika sistem memasukkan data ke *database*, Nilai tersebut akan secara otomatis bertambah (*AUTO_INCREMENT*) setiap data baru masuk. kolom kedua merupakan kolom *Date,* dengan tipe data *datetime.* Kolom ini menerima data dalam format 'YYYY-MM-DD HH:MM:SS' merupakan tahun (YYYY), bulan (MM), hari (DD), jam (HH), menit (MM), dan detik (SS). Data pada kolom ini, akan merekam waktu setiap kali sistem memasukan data ke *database.* Kolom ketiga merupakan Status, dengan tipe data *float.* kolom ini menyimpan data berupa persentase ruang penyimpanan yang terisi. Nilai pada kolom ini, dapat di monitoring secara *real time* melalui bagian *container* persentase pada *home page*.

Kolom keempat adalah kolom Total, dengan tipe data *integer*. Kolom ini menyimpan jumlah total bilik atau *station* yang terisi. Berikut alur kerja program untuk pengiriman data ke *database* dalam bentuk *flowchart.*



Gambar 3.9. *Flowchart Function Database*

Pada Gambar-3.9 terlihat proses pengiriman data *Date*, *Status*, dan *Total* ke dalam *database*. Langkah pertama adalah menghubungkan program ke database melalui informasi *host, user, password,* dan *database.* Data tersebut disimpan kedalam bentuk .env (*environment variable*) untuk meningkatkan keamanan. Setelah proses pengkoneksian berhasil, program akan memasukkan data berdasarkan *global variabel num_boxes* dan *filled_boxes* yang nilainya dipengaruhi oleh *function detection.* Proses pengiriman data akan dilakukan secara berulang dengan interval waktu 5 detik sekali, yang dapat disesuaikan dengan kebutuhan pengguna nantinya. Data ini juga dapat di monitoring secara *real time* melalui bagian *container station* pada *home page.*

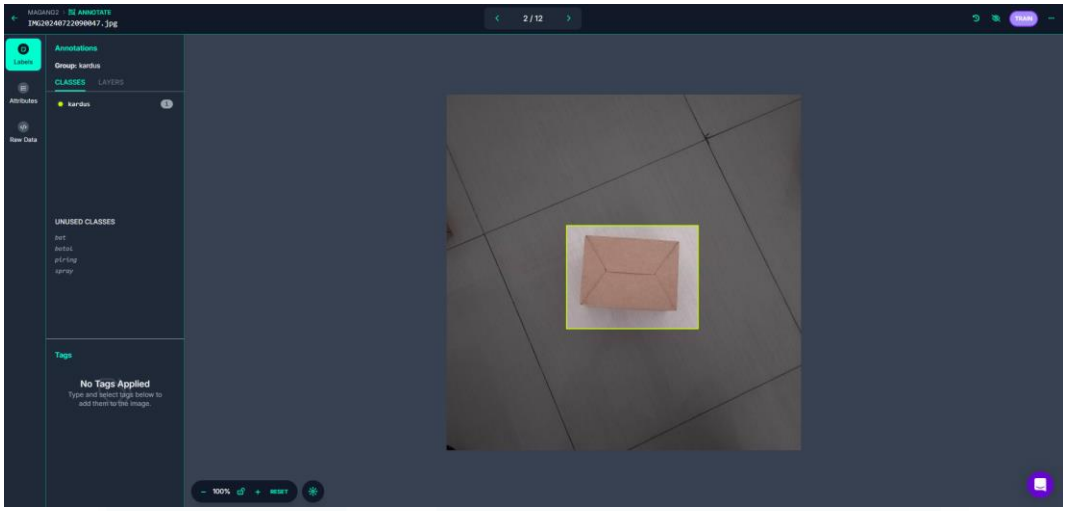


		id	Date	Status	Total
☐	Edit Copy Delete	1	2024-12-05 15:24:14	100	9
☐	Edit Copy Delete	2	2024-12-05 15:24:19	66.67	6
☐	Edit Copy Delete	4	2024-12-05 15:24:29	77.78	7
☐	Edit Copy Delete	5	2024-12-05 15:24:34	11.11	1
☐	Edit Copy Delete	6	2024-12-05 15:24:39	33.33	3

Gambar 3.10. Tabel *Storage Availability*

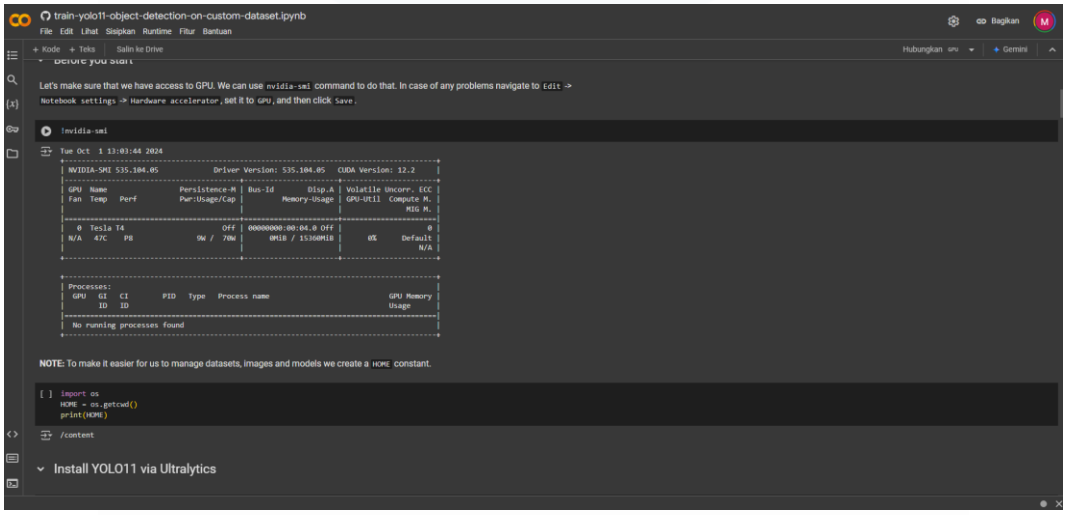
Pada Gambar-3.10, terlihat tabel pada *database* yang telah terisi dengan data oleh sistem. Setiap baris pada tabel memiliki *id* yang berbeda, yang menunjukan bahwa setiap data yang diinputkan memiliki identifikasi unik. Selain itu, data terlihat diperbaharui setiap 5 detik sekali, yang berfungsi untuk mengupdate status presentasi dan total bilik (*stasiun*) yang terisi. Pembaharuan berkala ini, memungkinkan sistem untuk mencatat perubahan secara *real time* dan menjadi informasi tetap tersimpan dengan baik.

Akurasi model AI menjadi hal yang krusial dalam proyek *Storage Availability System*. Maka dari itu, perlu dilakukan pengujian terhadap hasil pelatihan model AI menggunakan algoritma YOLOv11. Sebelum melakukan pelatihan, dataset yang telah dikumpulkan perlu diberi label agar program dapat mempelajari karakteristik dari objek yang dideteksi. Proses *labelling* akan dilakukan menggunakan *software* Roboflow, seperti yang ditunjukan pada Gambar-3.11.



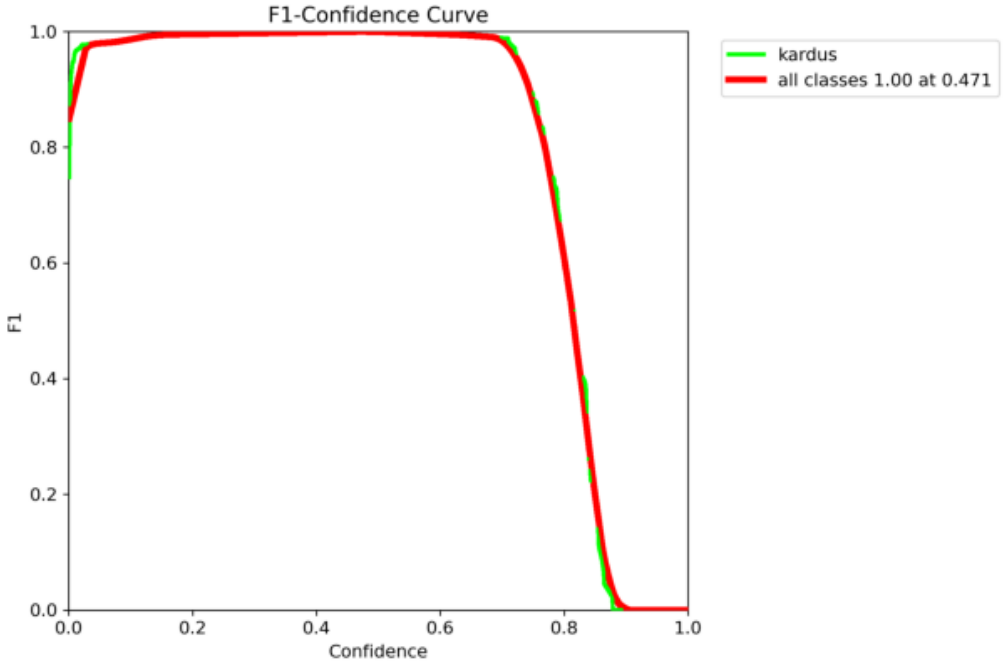
Gambar. 3.11 *Software Roboflow*

Setelah proses *labelling* selesai, dataset perlu dibagi menjadi tiga kategori, yaitu : *train*, *valid*, dan *test.* Pembagian ini bertujuan untuk mencegah terjadinya *overfitting* yang dapat mengakibatkan model kesulitan memahami pola. Konfigurasi pembagian dataset sebagai berikut : *train* 85 %, *valid* 10 %, dan test 5 %. Setelah dataset dibagi, proses pelatihan model akan dilakukan menggunakan *software* Google collab, seperti yang ditunjukan pada Gambar-3.12.



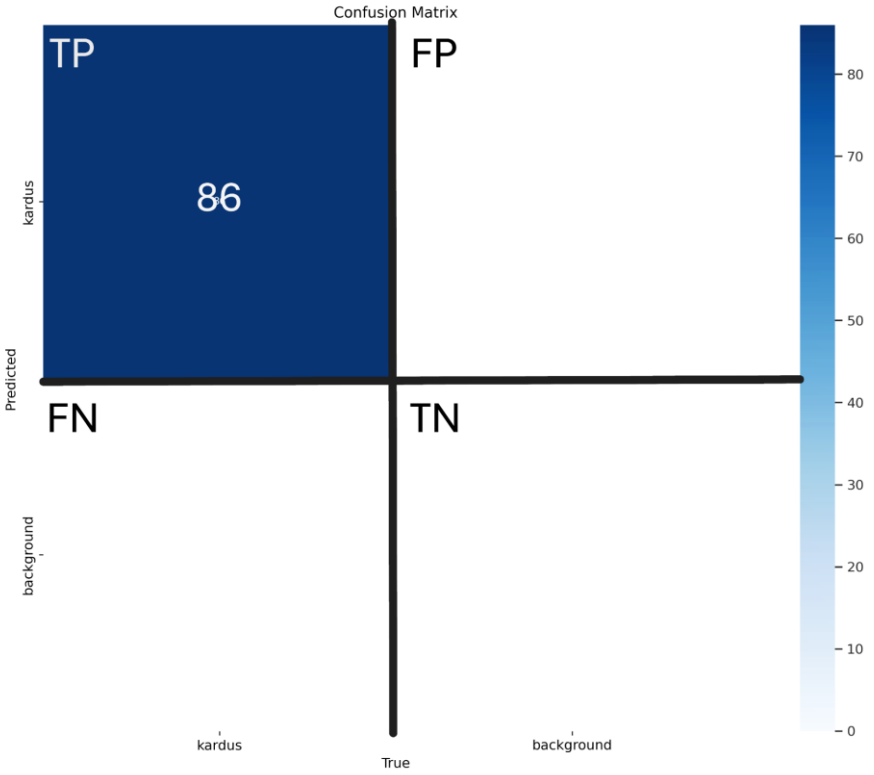
gambar 3.12. *Software* Google Collab

Pengujian model AI menggunakan *software* Google Collab bertujuan untuk memanfaatkan fitur GPU T4 untuk melatih model AI. Mengingat pelatihan memerlukan kemampuan komputasi yang tinggi, sehingga GPU sangat diperlukan. Model dilatih menggunakan konfigurasi *image size* 640 X 640 *pixels*, *epochs* 20, dan *batch size* 16 dengan total dataset 454 gambar.



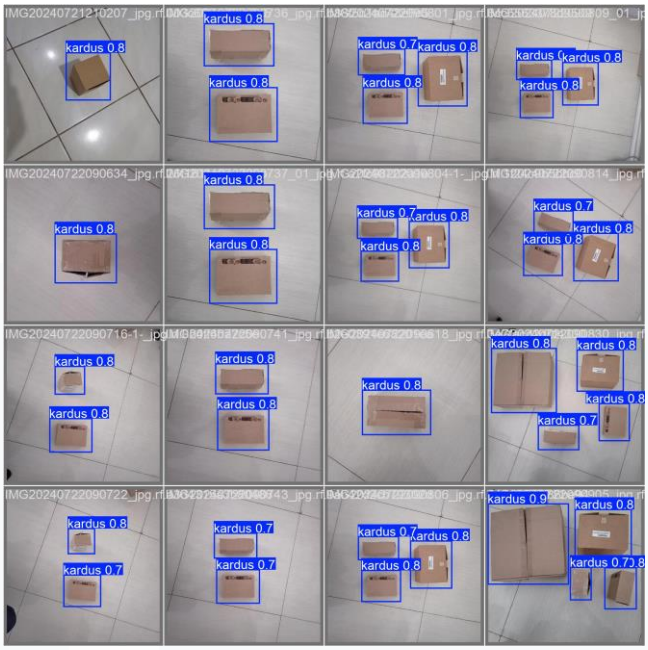
Gambar 3.13. F1-*Confidence Curve*

Pengujian akurasi model AI dilakukan dengan melihat parameter F1 *score*, seperti pada Gambar-3.13. Pada gambar tersebut, terlihat nilai tertinggi F1 *score* mencapai nilai 1 pada *confidence level* 0,471. Nilai F1 *score* yang tinggi menunjukan akurasi model yang sangat baik dalam mendeteksi objek. Meskipun nilai *confidence level* tergolong rendah, namun pada Gambar-3.13 menunjukan nilai yang F1 *score* yang stabil pada berbagai nilai *confidence level*. Namun, pada *confidence level* > 0.8 mengalami penurunan yang cukup signifikan, yang menyebabkan model sulit mencapai nilai *confidence level* tersebut selama pendeteksian. Hal ini dibuktikan pada percobaan *label* VS *predict* yang hanya memperoleh 5 dari 31 percobaan yang memiliki *confidence level* > 0.8. Pengujian tentu perlu melihat berbagai matrix yang dapat mengevaluasi kualitas dari model AI, salah satunya adalah *confusion matrix*.

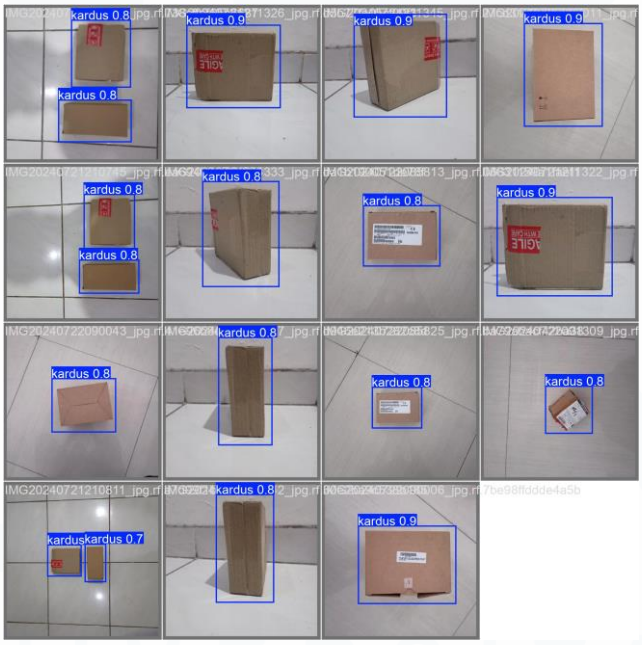


Gambar 3.14. *Confusion Matrix*

Pengujian pada Gambar-3.14 berfokus untuk melihat nilai *True Positive* (bagian kiri atas), *False Negatif* (bagian kanan atas), *False Positive* (bagian kiri bawah), *True Negatif* (bagian kanan bawah). Nilai - nilai tersebut sangat berpengaruh dalam menentukan nilai F1 *score.* Pada pengujian tersebut, diperoleh nilai *True Positive* bernilai 86 sementara nilai lainnya adalah 0. Hasil ini diperoleh dari pengujian menggunakan 5% dari total dataset. Total pengujian dilakukan dengan 24 sampel menggunakan metode *label vs predict.* Dalam metode ini, model membandingkan hasil deteksi dengan label yang telah ditentukan sebelum proses pelatihan.



(a)

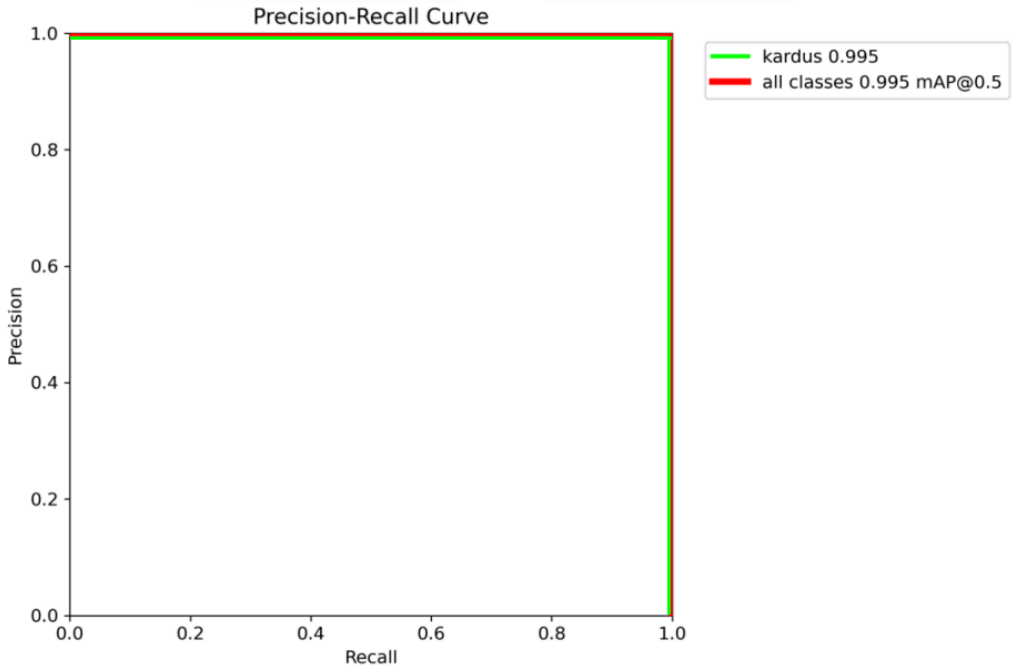


(b)

Gambar 3.15. Pengujian *Label VS Predict*

Pada pengujian yang dilakukan, seperti yang ditunjukan pada Gambar-3.15, model berhasil mengidentifikasi objek tanpa adanya kesalahan identifikasi. Dari hasil

pengujian, tercatat bahwa model berhasil memprediksi seluruh objek pada sampel dengan total 86 objek. Selain itu, kualitas dari model AI juga dapat dilihat melalui kesimbangan antara *precision* dan *recall.*



Gambar 3.16. *Precision-Recall Curve*

Pada Gambar-3.16, terlihat bahwa nilai *precision* dan *recall* saling tegak lurus, yang menandakan tidak terjadi *trade-off* antara *precision* dan *recall.* Hal ini berarti model dapat memprediksi hasil dengan akurat (*precision*) sekaligus memprediksi hampir semua objek yang relevan (*recall*). *Precision* yang tinggi dapat dilihat dari nilai *False Positif* yang minim pada *confusion matrix*, yang menunjukan hampir setiap prediksi *class* "kardus" dideteksi dengan akurat. Sementara itu, *recall* yang tinggi dapat dilihat dari nilai *false negatif* yang minim pada *confusion matrix*, yang menandakan semua objek kardus pada data dapat dideteksi oleh model. Hal ini juga didukung dengan nilai mAP (*mean Average Precision*) yang mencapai 0.995 pada IoU (*Intersection over Union*) ≥ 0.5

3.3 Kendala yang Ditemukan

Selama pengerjaan proyek *storage availability system*, terdapat beberapa permasalahan yang dialami oleh penulis :

1. Monitoring kamera IP tidak dapat berjalan secara *real time*. Pada *camera page*, berfungsi untuk menampilkan hasil kamera IP, serupa seperti fungsi CCTV pada umumnya. Akan tetapi, terdapat delay yang cukup signifikan dalam tampilannya. Hal tersebut disebabkan oleh penggunaan metode GET dan POST yang merupakan sistem komunikasi *default* dari Flask.
2. Model AI yang digunakan sulit untuk mencapai F1 *score* ± 0.8. Hal tersebut disebabkan karena terdapat nilai FN (*False Positive*) yang terdeteksi selama pengujian. *False positive* menunjukan bahwa model mengidentifikasi objek sebagai *class* yang salah, yang berkontribusi dalam penurunan F1 *score.*
3. FPS (*Frame Per Second*) yang dihasilkan pada proses pendeteksian objek terbatas di 10 FPS. Hal ini disebabkan karena program berjalan menggunakan CPU tanpa menggunakan GPU.

3.4 Solusi atas Kendala yang Ditemukan

Masalah yang ditemukan selama pengerjaan proyek *storage availability system* dan solusi yang ditawarkan oleh penulis sebagai berikut :

1. Monitoring kamera IP tidak dapat berjalan secara *real time*. Penulis menawarkan solusi dengan mengubah proses pengiriman *frame* yang sebelumnya menggunakan *GET* dan *POST* menjadi pendekatan video stream menggunakan multipart/x-mixed-replace. Penggunaan metode tersebut dapat mengurangi latensi dibandingkan menggunakan *GET* dan *POST*. Hal tersebut disebabkan karena metode *GET* dan *POST* bekerja sesuai dengan permintaan, sehingga tidak cocok untuk *streaming* video.
2. Model AI yang digunakan sulit mencapai F1 *Score* ± 0.8. Hal tersebut disebabkan karena banyak *background* yang terdeteksi selama proses pengujian. Hal tersebut disebabkan penggunaan algoritma YOLOv9 yang

memiliki akurasi yang lebih rendah dibandingkan dengan versi terbaru, yaitu YOLOv11. Hal tersebut disebabkan YOLOv11 telah mengintegrasikan ViT (Vision Transformer), yang mampu memahami pola objek dengan baik, sehingga dapat mengidentifikasi objek dalam berbagai ukuran dengan baik. Sebaliknya, YOLOv9 masih menggunakan CSP (*Cross Stage Partial*) yang dirancang untuk dapat mengurangi redudansi informasi dan meningkatkan efisiensi,sehingga dapat meningkatkan FPS. Namun, kurang optimal dalam hal akurasi deteksi objek.

3. FPS (*Frame Per Second*) yang dihasilkan pada proses pendeteksian objek terbatas di 10 FPS. Hal ini disebabkan oleh penggunaan YOLOv11 sebagai algoritma *object detection* yang berfokus pada peningkatan akurasi. Penurunan kecepatan ini terjadi karena pembaharuan versi YOLO. Maka dari itu, perangkat yang digunakan perlu memiliki GPU, khususnya GPU NVIDIA. Agar dapat menginstal CUDA (*Compute Unified Device Architecture*) versi 12.4 sesuai dengan spesifikasi *library* Pytorch.