BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam melakukan program magang di SATSINDO, jabatan magang yang diduduki adalah *Automation Engineer*. Proyek utama magang dibantu oleh *supervisor* dan *lead engineer* yang lebih berpengalaman karena sudah pernah membuat projek terkait RFID dan *website*. *Supervisor* magang dan *lead engineer* juga akan mengetes hasil proyek dan akan memantau kinerja selama proses pengerjaan.

Tugas posisi magang selain mengerjakan proyek utama yang telah diberikan adalah membantu para junior dan *lead engineer* dalam proyek yang akan dilakukan dan mengajar ekstrakurikuler di sekolah-sekolah. Dalam membantu para *engineer* lain, tugas yang diberikan dapat berupa membuat mengetes alat dan membuat program yang ingin digunakan untuk sebuah proyek (sensor, PLC, *valve*, kamera, dan lainnya). Selain itu, tugas juga dapat berupa visitasi ke lapangan untuk proyek yang sedang dikerjakan.

3.2 Tugas dan Uraian Kerja Magang

Selama proses magang, proyek yang dikerjakan adalah pengunaan RTLS berbasis RFID yang digunakan untuk manajemen aset. Proses pengerjaan dibagi menjadi 4 bagian, yaitu riset proyek, pembuatan *front-end website*, pembuatan *back-end* dan integrasi dengan sensor maupun *front-end*, scenario pemakaian, dan hasil pengujian.

3.2.1 Riset Proyek

Sebelum mengerjakan proyek, diperlukan sebuah riset terlebih dahulu untuk mengerti pentingnya RTLS dalam automasi industri dan bagaimana implementasinya di dalam keadaan industri nyata. Seperti pada [5] yang menjelaskan RTLS digunakan pada bengkel manufaktur dapat meningkatkan efisiensi pada operasional dan membangun proses manufactur yang cerdas. Pada [6], RTLS menggunakan RFID telah diimplementasikan untuk mengelola inventaris pada industri busana. Melalui riset yang telah dilakukan, dapat disimpulkan penggunaan RTLS memberikan keuntungan yang signifikan pada berbagai sektor industri. RTLS yang dirancang untuk proyek ini telah disesuaikan dengan kebutuhan industri sehingga menjadi lebih efektif.

3.2.2 Pembuatan *Front-end*

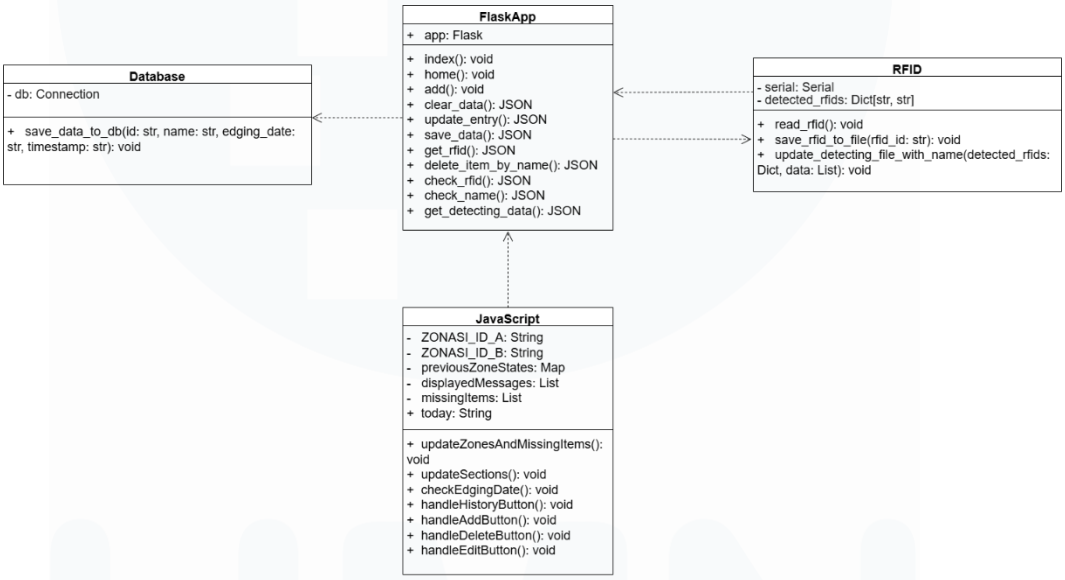
Pembuatan sistem ini menggunakan empat bahasa pemrograman, yaitu HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript (JS), dan Python. HTML, CSS, dan JS khusus digunakan untuk pembuatan *front-end* sedangkan Python khusus digunakan pada *back-end*. HTML digunakan untuk membuat tampilan pada *browser* yang terlihat terstruktur. CSS digunakan untuk menangani tampila, nuansa, posisi masing-masing konten, skema warna, dan responsivitas halaman *website*. JS digunakan untuk membuat konten yang ada pada *website* menjadi dinamis dan interaktif.

Sistem ini dibuat dengan menggunakan bantuan *front-end framework*, yaitu Bootstrap. Bootstrap fitur-fitur pada *library*-nya yang bisa digunakan agar *website* menjadi lebih tertata. Selain menggunakan Bootstrap, sistem juga menggunakan Boxicon untuk menampilkan ikon pada halaman *login* agar *website* menjadi lebih menarik. Terakhir, *library* JS yang digunakan untuk membuat *website* menjadi lebih interaktif adalah SweetAlert2. SweetAlert2 mengubah *alert* standar yang disediakan oleh *browser* menjadi *alert* yang berupa *popup* yang responsif. Pengunaan ketiga *library* eksternal tersebut berguna untuk membuat sistem menjadi lebih tertata, responsif, dan menarik untuk dilihat.

Terdapat 3 halaman HTML yang ada pada sistem, halaman pertama adalah index.html yang menampilkan halaman *login*, home.html yang menampilkan halaman utama pada sistem, dan add.html yang digunakan untuk menambahkan barang baru pada sistem.

3.2.3 Pembuatan dan Integrasi *Back-end*

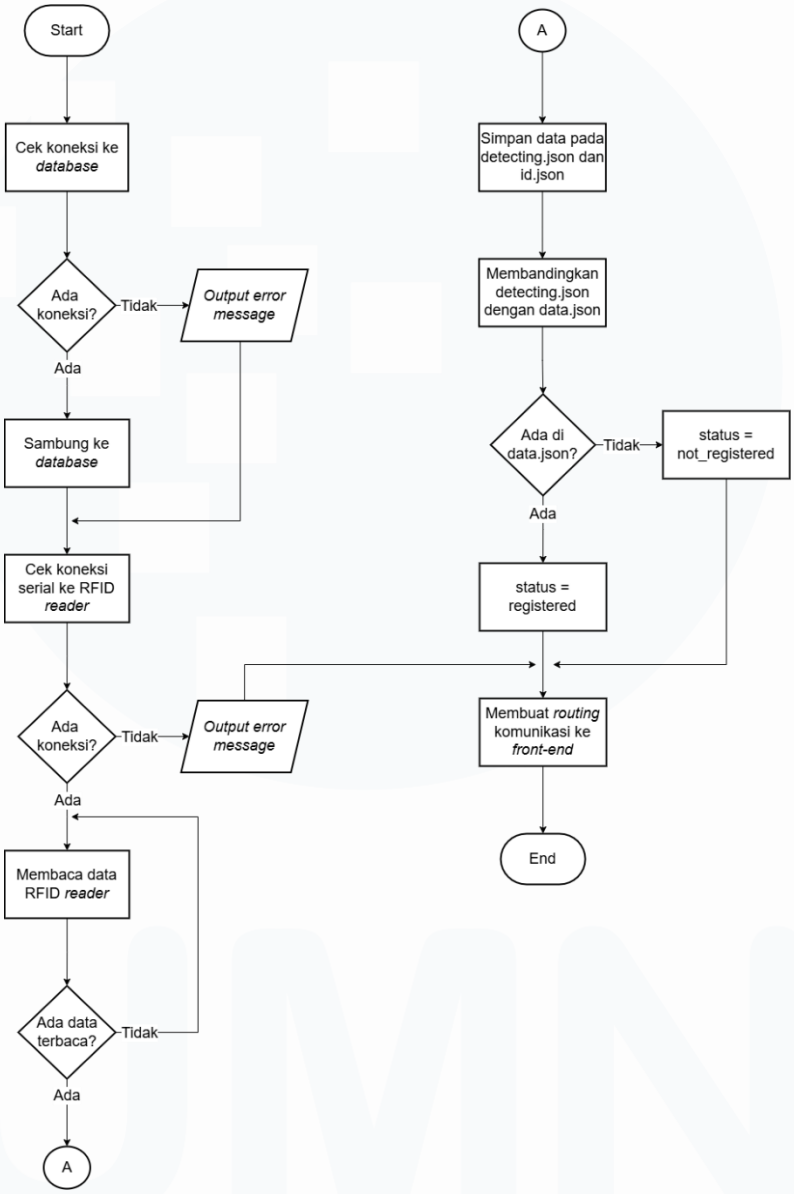
Framework yang dipilih untuk membangun sistem ini adalah Flask dengan mengguanakan bahasa pemrograman Python karena *framework* ini cocok untuk prototipe skala kecil ke menengah dan mudah untuk diimplementasikan oleh pemula. Sistem dapat direpresentasikan dalam *Unified Modeling Language* (UML).



Gambar 3.1 UML Sistem

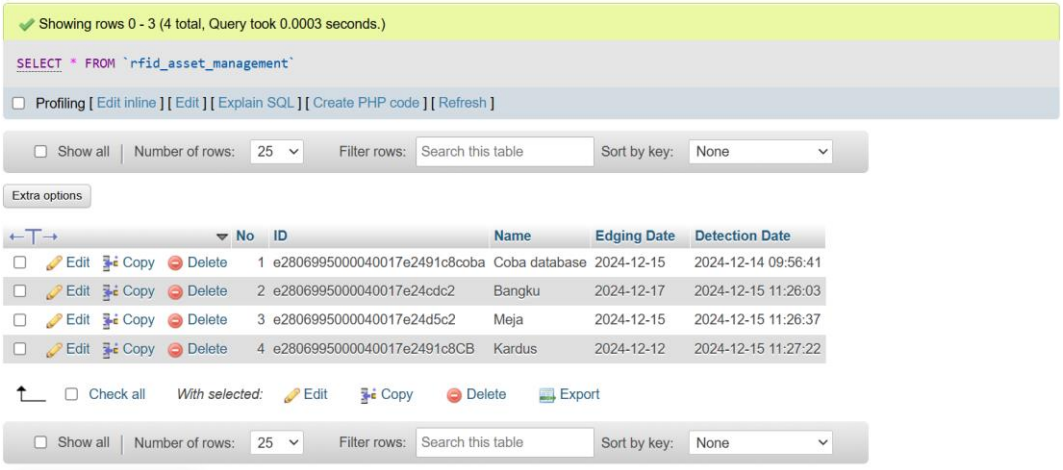
Gambar 3.1 menjelaskan bahwa *class* utama pada sistem merupakan FlaskApp, yang bertanggung jawab untuk membuat *file* JSON dan *routing* untuk menunjukkan tampilan *website*. FlaskApp memiliki hubungan *dependency* terhadap *class* lain, yaitu RFID dan Database. FlaskApp bergantung pada RFID untuk mendeteksi dan memproses data, sedangkan RFID bergantung pada FlaskApp untuk menampilkan datanya pada *website*. *Class* terakhir merupakan JavaScript yang bergantung pada FlaskApp untuk menampilkan datanya pada *website*.

Alur kerja sistem *back-end* dapat diamati melalui *flowchart* untuk memvisualisasikan urutan langkah-langkah dalam sistem.



Gambar 3.2 *Flowchart Back-end*

Program dimulai dengan pengecekan koneksi ke *database* yang menggunakan MySQL, seperti yang ditunjukkan pada Gambar 3.2. *Database* yang digunakan merupakan *database* lokal yang menggunakan MySQL dengan program XAMPP seperti yang bisa dilihat pada Gambar 3.3. *Database* akan menyimpan seluruh data yang dimasukkan oleh pengguna pada sistem.

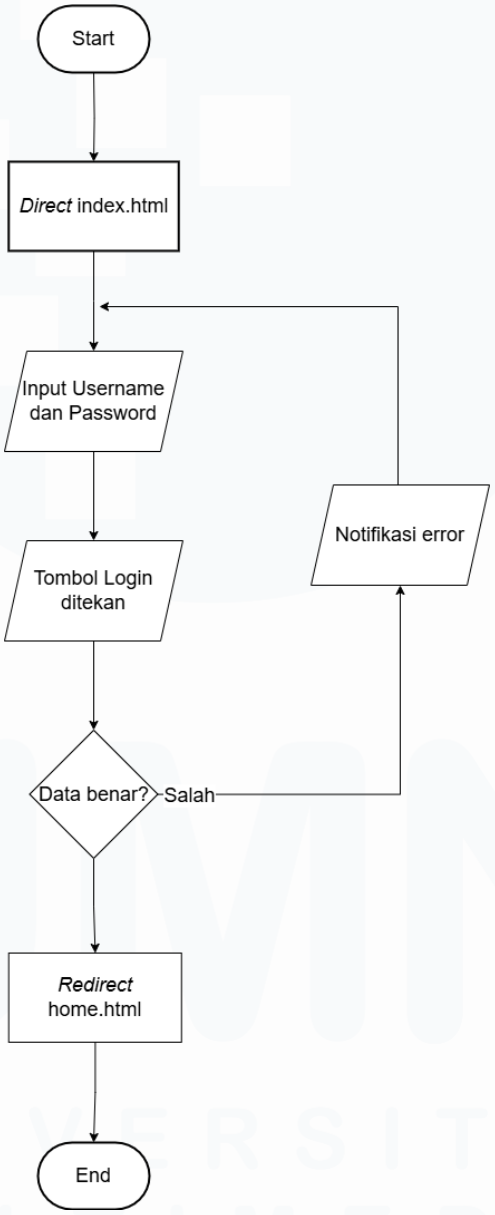


Gambar 3.3 Tampilan *Database* Sistem

Setelah pengecekan koneksi ke *database*, program juga akan mengecek koneksi ke RFID *reader* yang digunakan. Sensor ini menggunakan komunikasi serial yang langsung dihubungkan dengan *port* USB A yang ada di komputer. Jika kedua koneksi gagal terhubung, program akan menampilkan *alert* untuk masing-masing masalah. Program akan dilanjutkan dengan membaca data yang dibaca oleh RFID *reader*, data berupa heksadesimal yang merepresentasikan nilai 23 bit. Seluruh *tag* yang terbaca akan dimasukkan ke *file* JSON bernama detecting.json.

File detecting.json akan dibandingkan dengan *file* data.json. ID yang sudah terdaftar akan masuk ke dalam data.json, jika detecting.json mendeteksi ID yang sudah terdaftar maka status akan menjadi ‘registered’ sedangkan pendeteksian *tag* yang tidak terdaftar akan mengubah status ke ‘not_registered’. Setelah itu, tahap terakhir pada program *back-end* adalah membuat *routing* untuk memastikan komunikasi antara *front-end* dengan *back-end*.

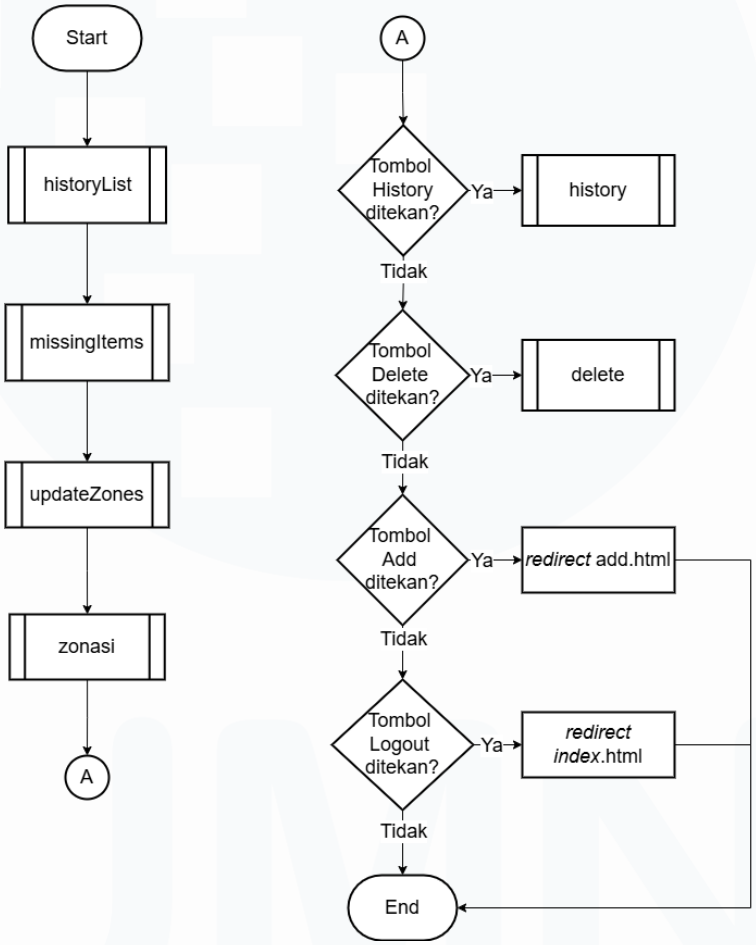
Routing merupakan salah satu fitur terpenting pada *framework* Flask, *routing* digunakan sebagai komunikasi antara *front-end* dengan *back-end*, sehingga memungkinkan agar sistem memiliki banyak fitur. Saat program dimulai, sistem akan membuat *route* untuk setiap bagian pada sistem dan yang pertama ditampilkan adalah tampilan index.html atau halaman *login*.



Gambar 3.4 *Flowchart* Halaman Login

Gambar 3.4 menjelaskan program dimulai dengan menampilkan halaman index.html atau halaman *login*. Pada halaman *login*, pengguna harus memasukkan *username* dan *password* yang benar agar dapat lanjut ke halaman berikutnya.

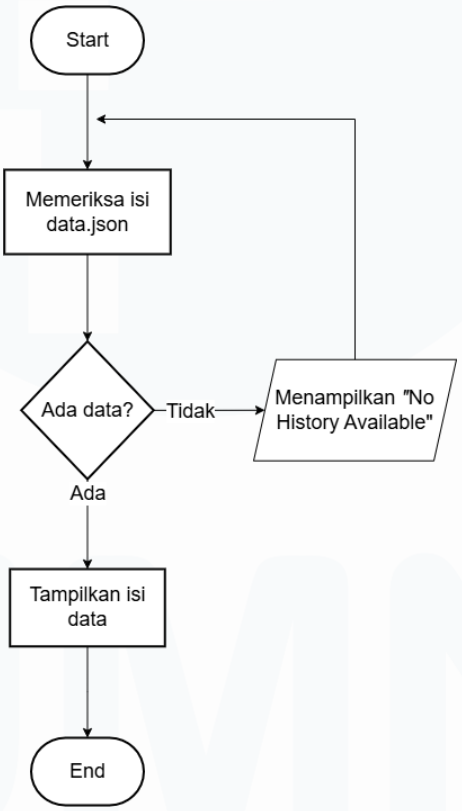
Setelah *login* berhasil sistem akan lanjut ke halaman selanjutnya, yaitu home.html. Halaman ini merupakan halaman utama dalam sistem.



Gambar 3.5 *Flowchart* Halaman Home

Halaman ini memiliki beberapa fitur, seperti yang ditunjukkan pada Gambar 3.5, yaitu: menampilkan seluruh barang yang sudah terdaftar, memberikan peringatan jika ada barang yang hilang, menampilkan perpindahan barang, menampilkan zonasi barang, menampilkan seluruh barang dalam bentuk tabel untuk dapat diedit. Selain itu, pengguna juga dapat menghapus barang, melanjutkan ke halaman berikutnya, dan kembali ke halaman *login*.

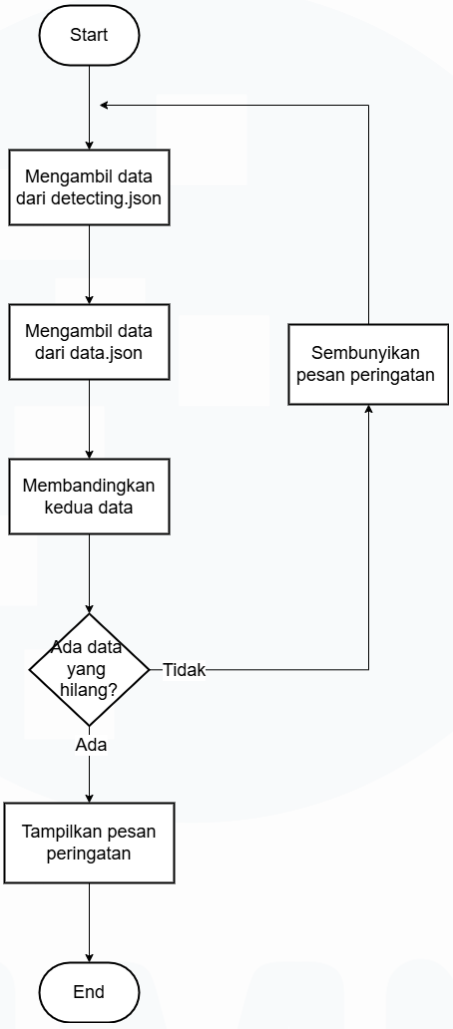
Salah satu fitur yang terdapat di Gambar 3.5 merupakan menampilkan seluruh data dari barang yang sudah didaftarkan.



Gambar 3.6 *Flowchart* Subproses historyList

Gambar 3.6 menunjukkan subproses dari fungsi historyList, yang berfungsi menampilkan data barang. Program akan memeriksa *file* data.json untuk memeriksa ada isinya atau tidak. Program akan menampilkan pesan “No History Available” jika tidak ada data, jika ada maka data akan ditampilkan pada halaman *home*.

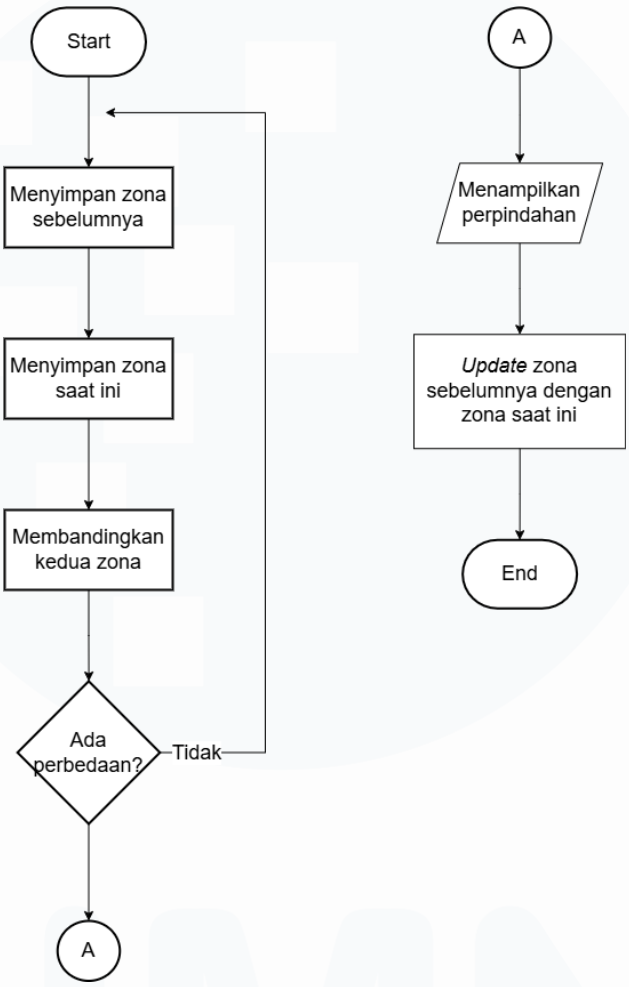
Fitur selanjutnya merupakan menampilkan barang hilang pada halaman *home*.



Gambar 3.7 *Flowchart* Subproses missingItems

Subproses selanjutnya merupakan missingItems, seperti yang terlihat pada Gambar 3.7, yang digunakan untuk menampilkan barang hilang. Program akan mengambil data dari detecting.json dan data.json untuk dibandingkan. Program akan menampilkan pesan peringatan barang hilang jika ada data yang hilang dan tidak akan menampilkan pesan peringatan jika tidak ada data yang hilang.

Fitur selanjutnya merupakan fitur menampilkan perpindahan barang pada halaman *home*.



Gambar 3.8 *Flowchart* Subproses updateZones

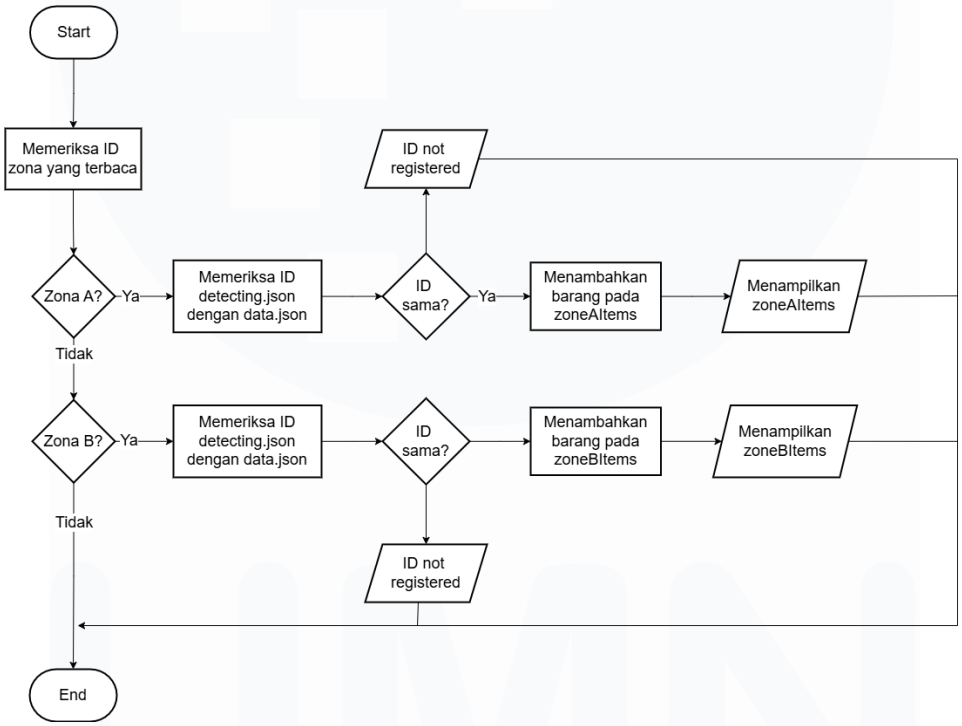
Pada Gambar 3.8, ditampilkan subproses dari fungsi updateZones yang bertujuan untuk menampilkan perpindahan barang. Program akan menyimpan zona sebelumnya dan zona saat ini untuk dibandingkan. Jika ada perbedaan maka program akan menampilkannya di halaman *home* dan memperbarui zona sebelumnya dengan zona saat ini untuk siklus pendeteksian selanjutnya.

Selanjutnya merupakan fitur untuk menentukan zona di mana barang terdeteksi oleh RFID *reader*.

```
1. ZONASI_ID_A = "e2806995000050017e2499c2"
2. ZONASI_ID_B = "e2806995000050017e24d1c2"
```

Gambar 3.9 ID *Tag* yang Digunakan Sebagai Penanda Zona

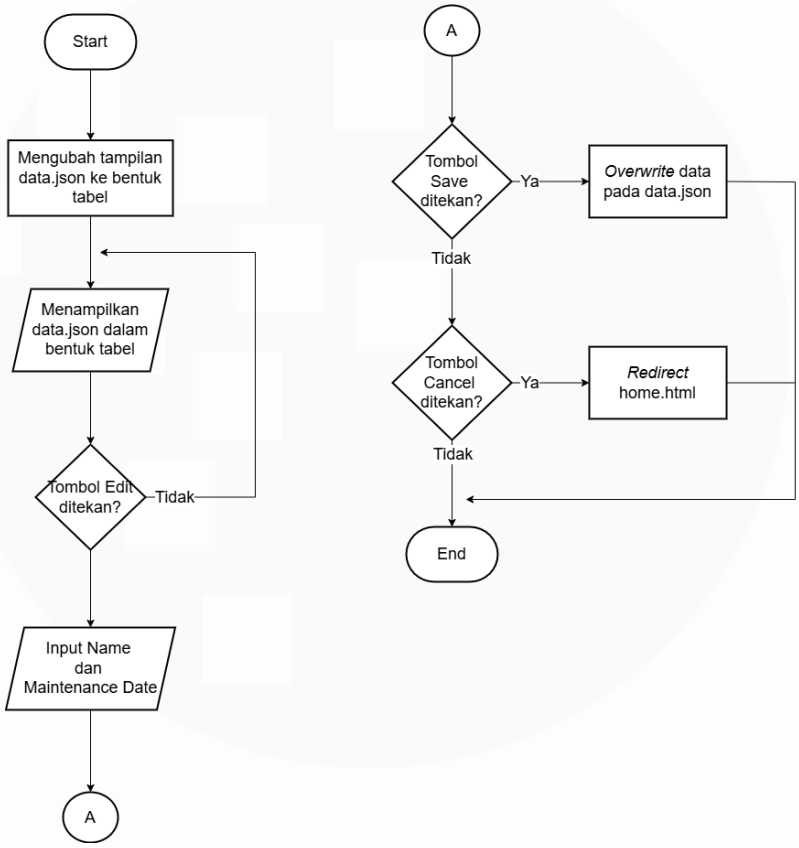
Gambar 3.9 merupakan dua *tag* unik yang dideklarasi sebagai zona A dan zona B. Sistem RTLS ini seharusnya dapat melacak lokasi barang dengan sistem zonasi dan menampilkan seluruh barang yang terdeteksi oleh RFID *reader* di setiap zona. Namun RFID *reader* yang digunakan pada tahap prototipe hanya berjumlah satu, oleh karena itu program disesuaikan agar dapat menampilkan zonasi barang-barang.



Gambar 3.10 *Flowchart* Subproses zonasi

Gambar 3.10 memperlihatkan subproses dari fungsi zonasi yang digunakan untuk menampilkan zonasi barang. Program akan memeriksa ID zona yang terdeteksi, jika ada ID dari *tag* zona A terdeteksi di detecting.json, maka seluruh ID dari *tag* yang ada di detecting.json akan ditampilkan di Zona A, begitu juga sebaliknya. Jika tidak ID yang terdeteksi di detecting.json tidak ada pada data.json maka program akan menampilkan pesan error karena ID belum didaftarkan oleh pengguna. Melalui program ini, sistem RTLS tidak dapat berfungsi dengan sempurna karena tidak dapat menampilkan kedua zona dalam waktu bersamaan.

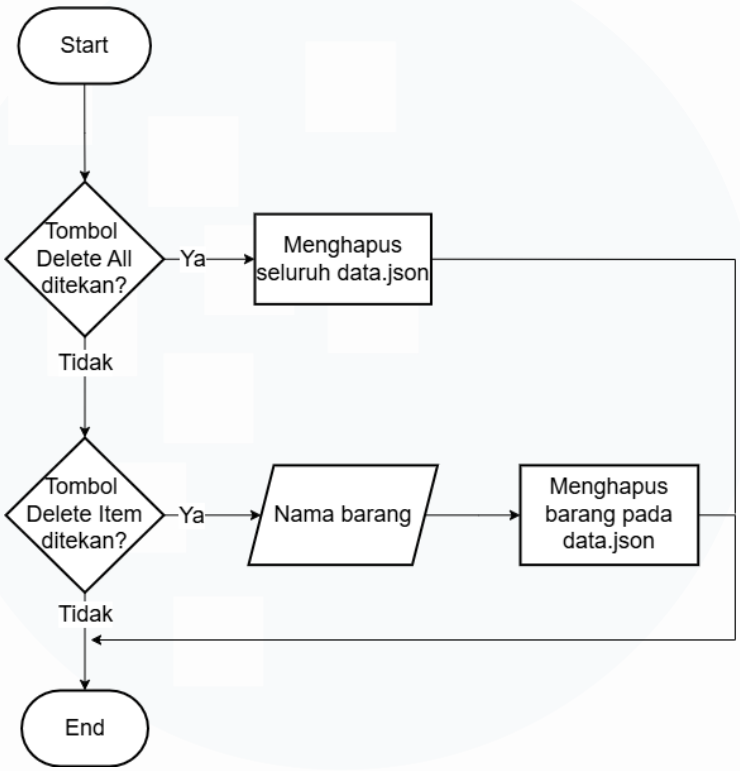
Fitur selanjutnya merupakan mengedit data dari barang yang ketika pengguna menekan tombol History.



Gambar 3.11 *Flowchart* Subproses history

Gambar 3.11 merupakan subproses dari fungsi history yang akan dilaksanakan setelah pengguna menekan tombol History. Program akan menampilkan data pada data.json dalam bentuk tabel dan pengguna dapat mengedit data barang yang dipilih untuk mengganti nama dan *maintenance date*. Data yang diubah akan menggantikan isi data pada *file* data.json.

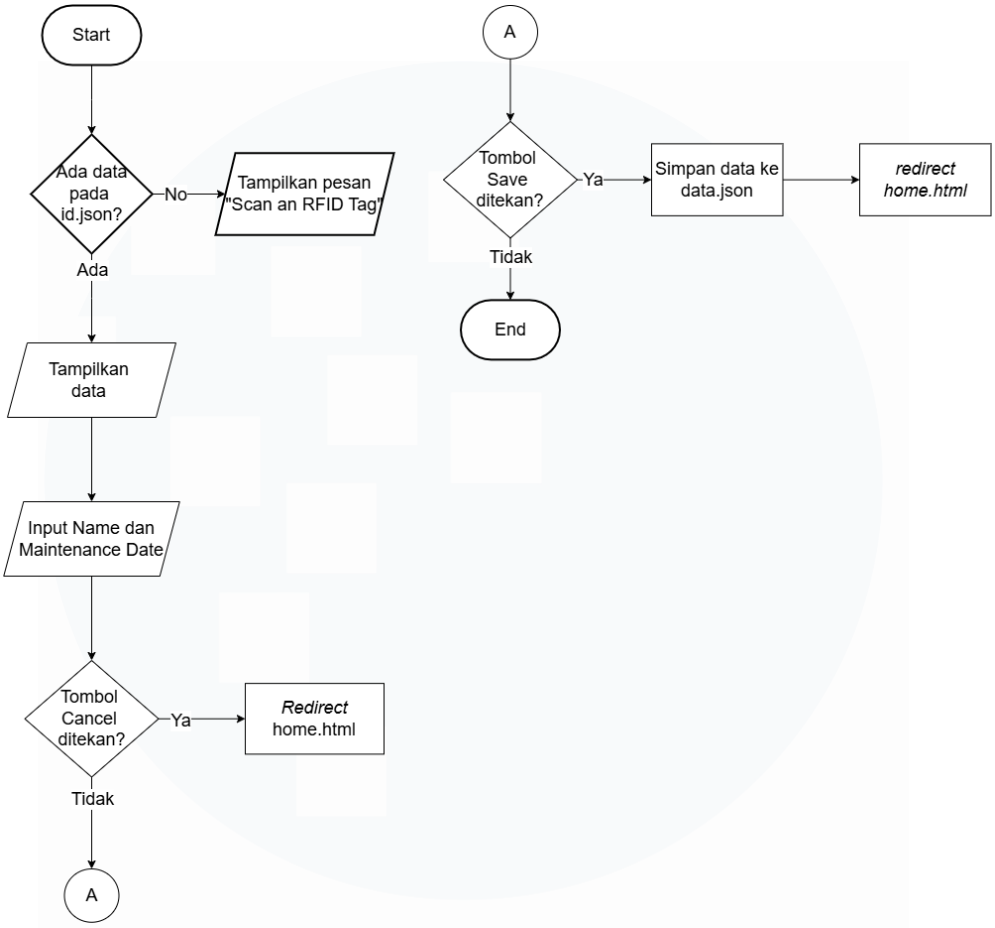
Fitur selanjutnya pada halaman *home* merupakan fitur menghapus barang sesuai dengan *input* dari pengguna.



Gambar 3.12 *Flowchart* Subproses delete

Gambar 3.12 menjelaskan subproses fungsi delete yang berfungsi untuk menghapus barang. Ketika pengguna menekan tombol Delete, program akan menampilkan dua pilihan seperti pada Gambar 3.12, pilihan pertama merupakan tombol Delete All yang akan menghapus seluruh data pada data.json dan pilihan kedua merupakan Delete Item untuk menghapus data dari barang yang dimasukkan oleh pengguna.

Fitur terakhir yang ada pada halaman ini merupakan mendaftarkan barang baru dengan pindah ke halaman baru.

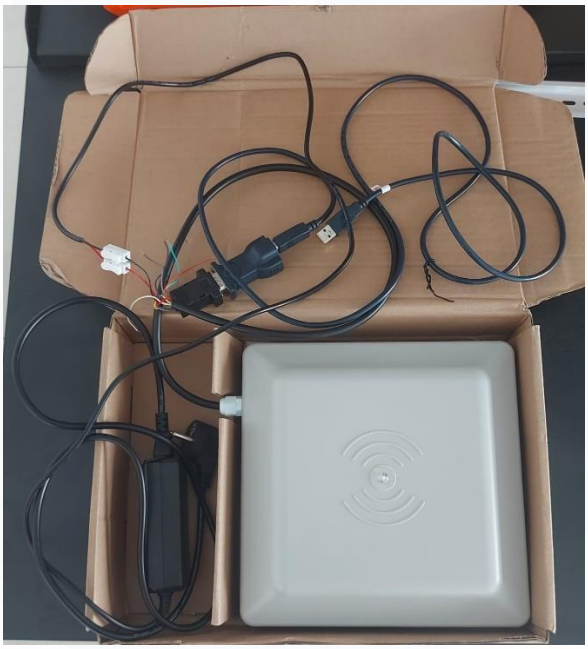


Gambar 3.13 *Flowchart* Halaman Pendaftaran Barang

Pada Gambar 3.13 ditunjukkan bahwa data yang terdapat pada id.json akan ditampilkan. Setelah itu pengguna dapat meng-*input* nama dan *maintenance date* untuk ID yang di *scan* oleh *reader*. Setelah tombol Save ditekan, data yang di-*input* akan disimpan ke data.json.

3.2.4 Skenario Pemakaian

RFID *reader* yang digunakan untuk prototipe adalah UHF RFID *reader* yang mampu mendeteksi hingga 6 meter.

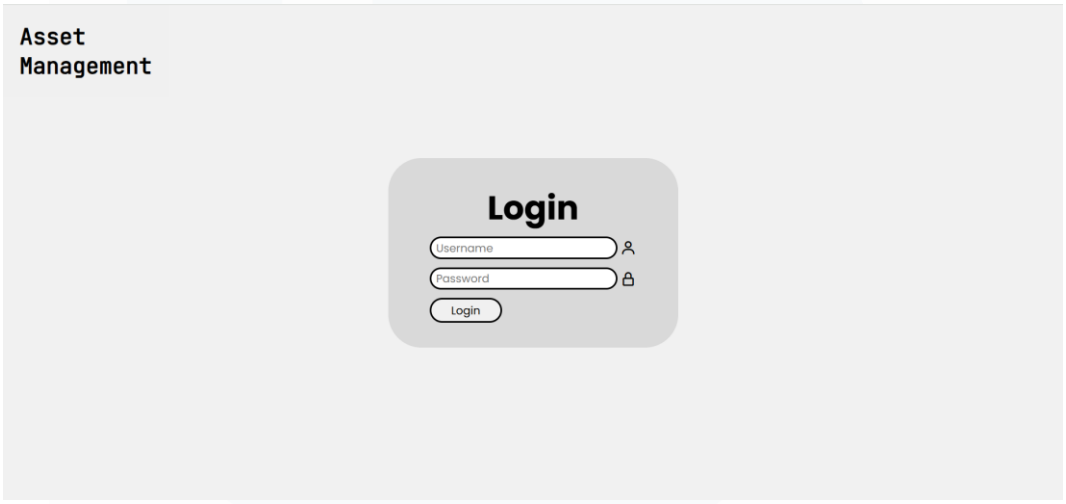


Gambar 3.14 Perangkat RFID *Reader*

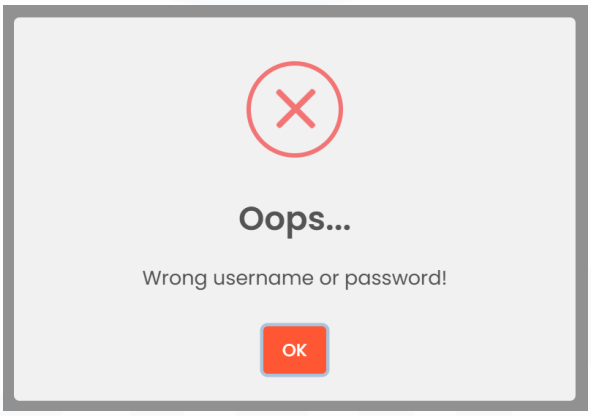
Pada Gambar 3.14 ditunjukkan perangkat RFID *reader* yang dipakai selama pengerjaan proyek. Perangkat dihubungkan demgan laptop atau komputer dengan menggunakan komunikasi serial melalui USB A. Percobaan pemakaian sistem ini berlangsung di gudang pabrik PT SA, Tbk yang berada di Solo. RFID *reader* akan digantung di ruangan dan dibiarkan dalam kondisi menyala dan terhubung dengan komputer. Pemakaian pada sistem final (bukan prototipe) yang akan dilaksanakan dalam tahun 2025, sistem akan memakai RFID *reader* Zebra FX7500 yang jumlahnya lebih dari satu, sesuai dengan kebutuhan yang akan ditetapkan di waktu yang akan mendatang.

3.2.5 Hasil Pengujian

Tampilan *website* sistem dibagi menjadi tiga tampilan utama, yaitu halaman *login*, halaman *home*, dan halaman pendaftaran barang. Kesalahan pengisian *username* dan *password* akan menampilkan peringatan dan tidak berhasil masuk ke halaman berikutnya.



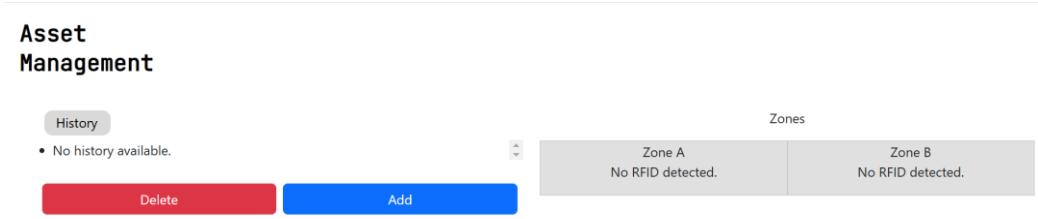
(a)



(b)

Gambar 3.15 (a) Tampilan *Login* Sistem (b) Tampilan *alert* Jika *username* atau *password* yang Salah

Gambar 3.15 (a) menunjukan tampilan ketika program dijalankan. Setelah *username* dan *password* yang dimasukkan telah sesuai dengan data yang ada di sistem, pengguna akan diarahkan ke halaman *home.* Jika data salah, maka tampilan seperti Gambar 3.15 (b) akan muncul dan pengguna tidak diarahkan ke halaman selanjutnya.



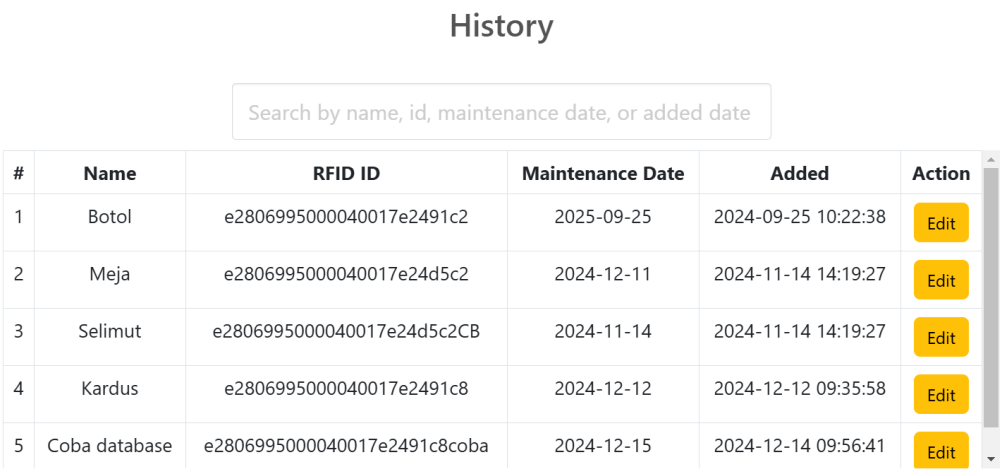
Gambar 3.16 Tampilan *Home* Tanpa Data

Tampilan *home* pada Gambar 3.16 adalah tampilan saat pertama kali program berjalan. Tidak ada data yang ditampilkan karena belum ada data yang disimpan oleh sistem.

Gambar 3.17 Tampilan *Home* dengan Data

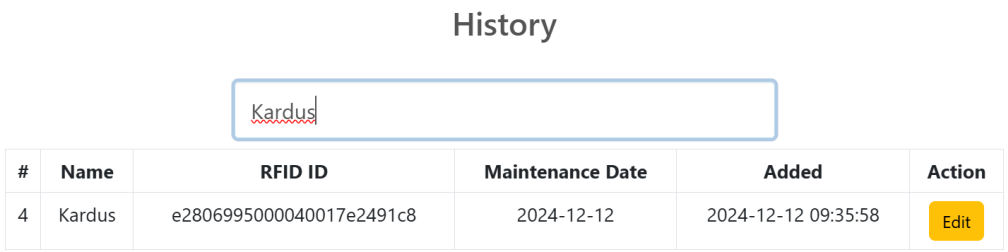
Gambar 3.17 adalah tampilan halaman *home* dengan data yang sudah dimasukkan oleh pengguna. Data yang ditampilkan merupakan ID dari *tag* RFID, *timestamp* saat *tag* didaftarkan, nama dari *tag* yang di-*input* oleh pengguna, dan *maintenance date* yang juga di-*input* oleh pengguna. Selain itu, sistem akan menampilkan zona di mana *tag* terdeteksi. Jika ada pemindahan barang dari Zona B ke Zona A atau sebaliknya, sistem juga akan menampilkannya. Tampilan warna dari bagian History akan berubah ketika *maintenance date* jatuh pada hari ini (warna kuning) dan ketika sudah lewat (warna merah). *Tag* yang sudah terdaftar namun tidak terdeteksi di zona manapun akan diberi peringatan.

Tombol History yang ada di halaman *home* dapat ditekan untuk melihat seluruh *tag* yang sudah didaftarkan dalam bentuk tabel dan dapat mencari barang dengan nama barang, *tag* ID, tanggal *maintenance,* dan tanggal pendaftaran *tag*.

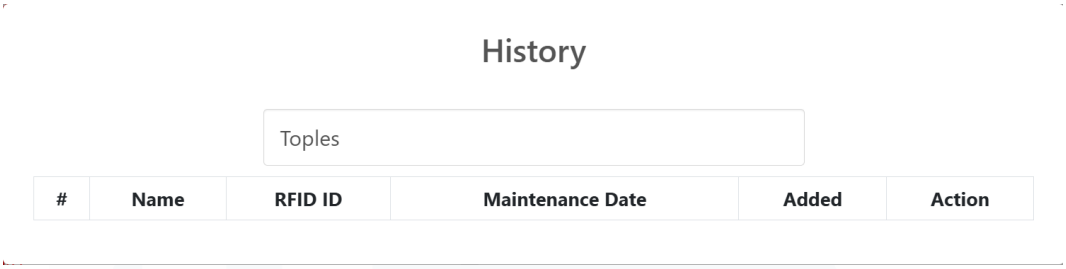


Gambar 3.18 Tampilan Tombol History

Gambar 3.18 menunjukkan tampilan sistem ketika tombol History ditekan oleh pengguna. Sistem juga menampilkan data-data sesuai dengan kata kunci pada filter yang di-*input* oleh pengguna.



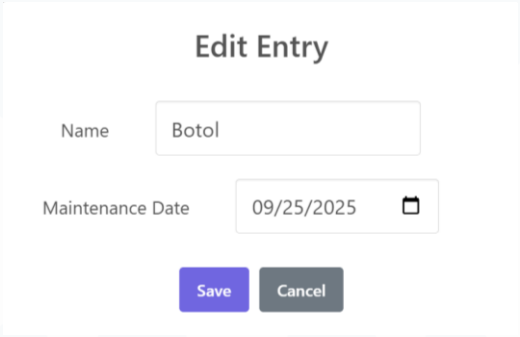
(a)



(b)

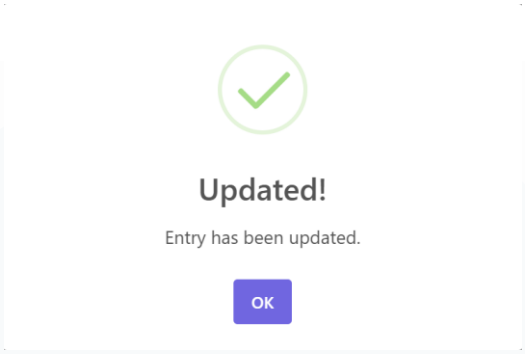
Gambar 3.19 (a) Tampilan dengan Kata Kunci yang Ada pada data.json (b) Tampilan dengan Kata Kunci yang Tidak Ada pada data.json

Seperti yang ditunjukkan pada Gambar 3.19 (a), pengguna dapat melihat daftar barang sesuai dengan *input*. Jika tidak ada barang dengan nama sesuai *input* Gambar 3.19 (b) akan muncul. Selanjutnya, pada tabel yang ada Gambar 3.18 terdapat tombol Edit yang digunakan untuk mengganti nama *tag* dan *maintenance date* dari barang yang dipilih.



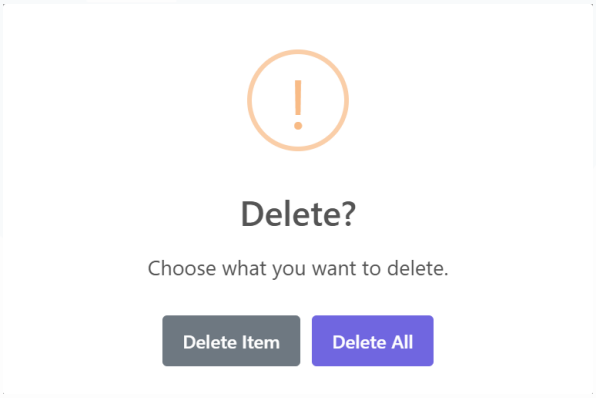
Gambar 3.20 Tampilan Tombol Edit

Seluruh *input* yang disimpan pada menu Edit seperti pada Gambar 3.20 akan disimpan pada data.json dan merubah data dari barang yang dipilih. Pengguna dapat menyimpan perubahan dengan menekan tombol Save, jika tombol Cancel ditekan maka akan kembali pada halaman *home* seperti pada Gambar 3.17.



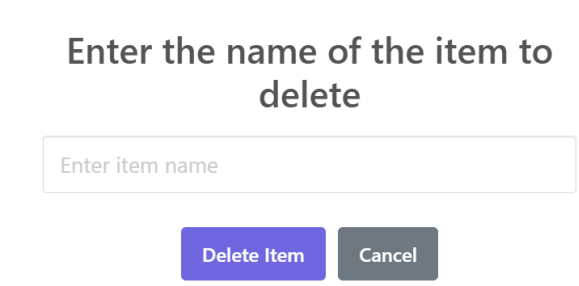
Gambar 3.21 Tampilan jika Perubahan Berhasil Disimpan

Gambar 3.21 akan ditampilkan ketika pengguna menekan tombol Save dan seluruh data yang diperlukan sudah diisi.

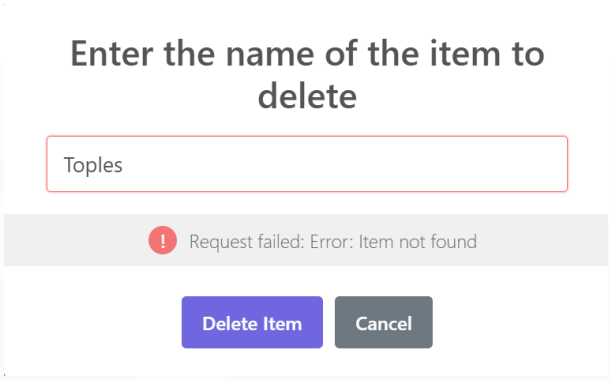


Gambar 3.22 Tampilan Pilihan Tombol Delete

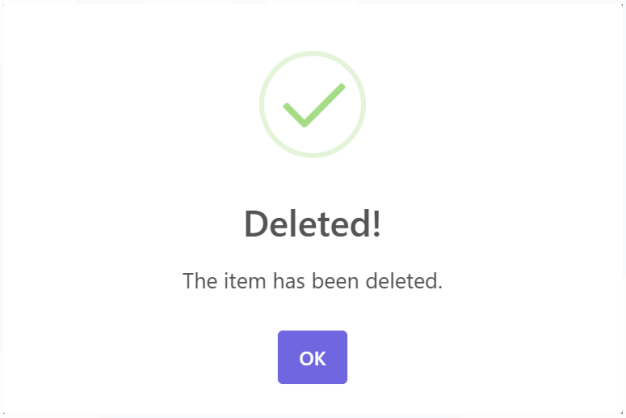
Tombol Delete digunakan untuk menghapus data yang ada pada data.json. Terdapat dua pilihan ketika menekan tombol Delete seperti pada Gambar 3.22. Tombol Delete Item digunakan untuk menghapus data yang sesuai dengan kata kunci yang berupa nama barang yang telah di-*input* oleh pengguna. Kata kunci yang tidak sesuai akan menampilkan peringatan.



(a)



(b)

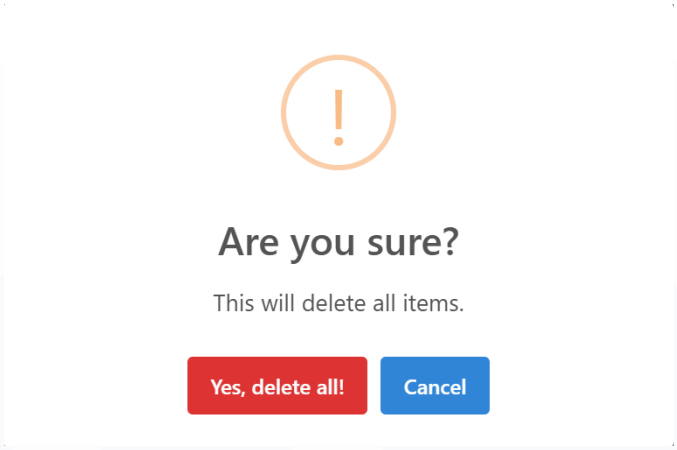


(c)

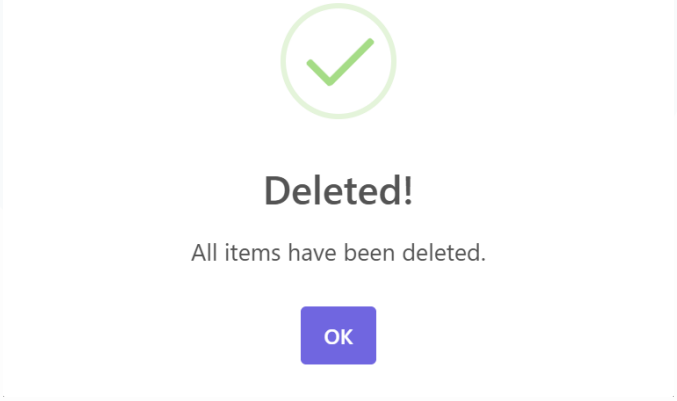
Gambar 3.23 (a) Tampilan ketika Delete Item Ditekan (b) Tampilan ketika Kata Kunci Tidak Sesuai (c) Tampilan Ketika Barang Berhasil dihapus

Jika pengguna menekan tombol Delete Item, maka sistem akan meminta *input* berupa nama dari barang yang sudah didaftarkan seperti pada Gambar 3.23 (a). Ketidakcocokan nama barang akan menampilkan Gambar 2.23 (b), sedangkan ketika penghapusan berhasil Gambar 2.23 (c) akan ditampilkan.

Tombol Delete All digunakan untuk menghapus seluruh data yang ada pada data.json.



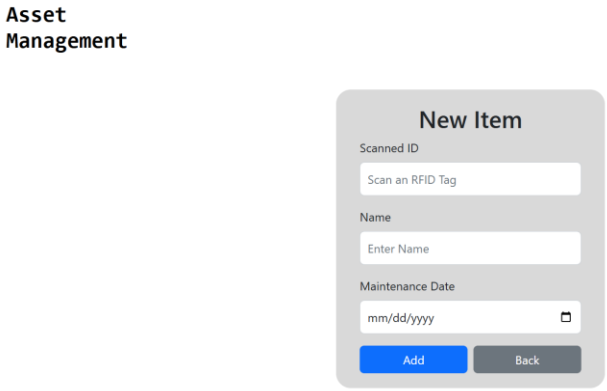
(a)



(b)

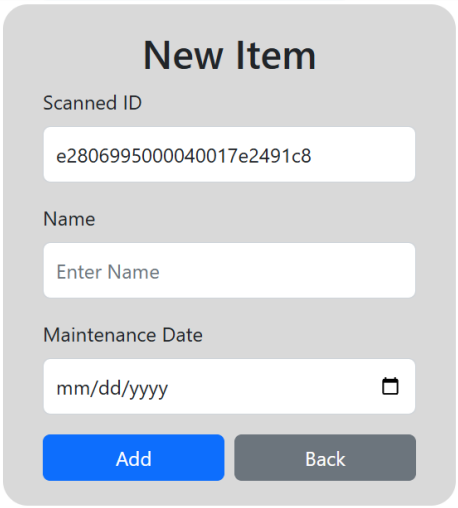
Gambar 3.24 (a) Tampilan Ketika Tombol Delete All Ditekan (b) Tampilan Ketika Berhasil dihapus

Pilihan akan ditampilkan seperti pada Gambar 3.24 (a) dan penghapusan berhasil dilakukan jika tampilan menunjukkan Gambar 3.24 (b). Setelah itu, fitur terakhir merupakan tombol Add, ketika ditekan sistem akan mengubah ke halaman lain untuk mendaftarkan *tag* baru.

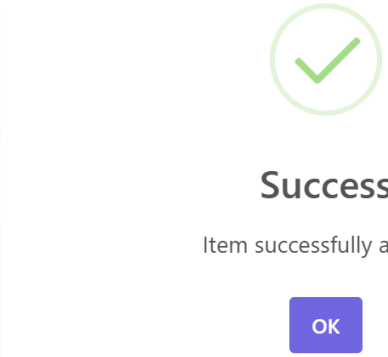


Gambar 3.25 Tampilan Halaman Pendaftaran Barang

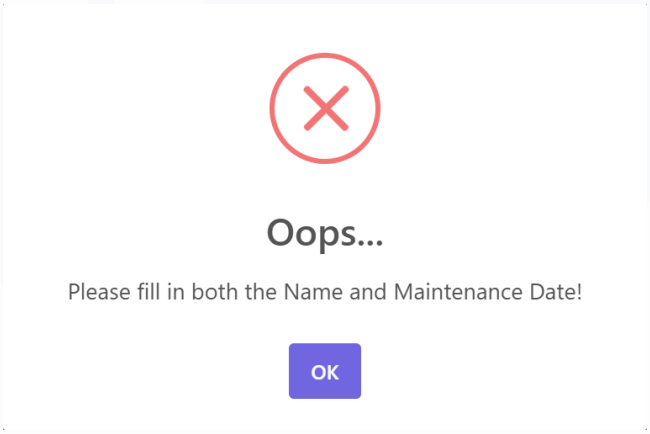
Jika sebuah *tag* RFID sudah ter-*scan* oleh RFID *reader*, maka ID dari *tag* tersebut akan muncul pada kotak Scanned ID seperti pada Gambar 3.25. Pengguna juga harus memasukkan nama dan tanggal *maintenance* untuk bisa mendaftarkan barang. Jika semua data sudah lengkap, maka ketika tombol *add* ditekan, barang yang didaftarkan akan ditampilkan pada halaman *home* dan sistem otomatis kembali ke halaman *home*.



(a)



(b)



(c)

Gambar 3.26 (a) Tampilan Ketika ID Terdeteksi (b) Tampilan Ketika Data Berhasil Disimpan (c) Tampilan Ketika Data Tidak Lengkap

Gambar 3.26 (a) menunjukkan ketika ID terdeteksi, maka ID tersebut akan ditampilkan pada kotak. Penambahan barang berhasil jika seluruh data terisi dan tombol Add ditekan dan tampilan Gambar 3.26 (b) muncul. Penambahan barang dinyatakan gagal ketika seluruh data tidak terisi dan tampilan Gambar 3.26 (c) muncul.

3.3 Kendala yang Ditemukan

Selama pengerjaan proyek, terdapat beberapa kendala yang ditemukan sebagai berikut:

1. Keterbatasan jumlah RFID *reader* yang membuat implementasi RTLS tidak akurat. Untuk mengetahui lokasi aset-aset pada pabrik dibutuhkan beberapa RFID *reader*, terlebih jika ingin mengetahui lokasi aset yang dapat bergerak. Selain itu agar kinerja optimal, RFID *reader* untuk melacak aset dan mendaftarkan aset dapat dipisah.
2. Kualitas RFID *reader* yang kurang optimal. Kualitas RFID *reader* yang digunakan untuk tahap prototipe dapat dibilang kurang optimal karena memiliki jangkauan yang tidak konsisten. *Tag* terkadang tidak dapat terbaca jika posisinya diagonal terhadap *reader*. sehingga peletakan *reader* harus tegak lurus dengan *tag* untuk mendapatkan pembacaan yang optimal.

3.4 Solusi atas Kendala yang Ditemukan

Berikut merupakan solusi-solsui yang diterapkan untuk semua kendala yang ditemukan saat mengerjakan proyek:

1. Menyesuaikan kode program dengan jumlah RFID *reader*, yaitu satu buah. Penyesuaian yang digunakan adalah menggunakan *tag* RFID sebagai perwakilan zona deteksi.
2. Pemasangan RFID *reader* yang harus tegak lurus dengan *tag* RFID untuk memastikan tidak ada *tag* yang tidak terbaca oleh *reader*.