TUGAS AKHIR – KI141502

IMPLEMENTASI MESSAGE QUEUE PADA SISTEM KEHADIRAN MENGGUNAKAN RASPBERRY PI DENGAN MODUL NFC

FIKRY KHAIRYTAMIM
NRP 05111440000192

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Dr.tech. Ir. R.V.Hari Ginardi, M.Sc.

Departemen Teknik Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – KI141502

IMPLEMENTASI MESSAGE QUEUE PADA SISTEM KEHADIRAN MENGGUNAKAN RASPBERRY PI DENGAN MODUL NFC

FIKRY KHAIRYTAMIM
NRP 05111440000192

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Dr.tech. Ir. R.V.Hari Ginardi, M.Sc.

Departemen Teknik Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

IMPLEMENTATION OF MESSAGE QUEUE ON ATTENDANCE SYSTEM USING RASPBERRY PI WITH NFC MODULE

FIKRY KHAIRYTAMIM
NRP 5113100192

First Advisor
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Second Advisor
Dr.tech. Ir. R.V.Hari Ginardi, M.Sc.

Department of Informatics
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI *MESSAGE QUEUE* PADA SISTEM KEHADIRAN MENGGUNAKAN RASPBERRY PI DENGAN MODUL NFC

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur dan Jaringan Komputer
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
FIKRY KHAIRYTAMIM
NRP: 05111440000192

Disetujui oleh Pembimbing Tugas Akhir:

1. Henning Titi Ciptaningtyas, S.Kom., M.Kom. .....
   (NIP. 19840708 201012 2 004) (Pembimbing 1)

2. Dr. Tech. Ir. R.V. Hari Ginardi, M.Sc .....
   (NIP. 19650518 199203 1 003) (Pembimbing 2)

SURABAYA
Januari, 2018

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI *MESSAGE QUEUE* PADA SISTEM KEHADIRAN MENGGUNAKAN RASPBERRY PI DENGAN MODUL NFC

Nama Mahasiswa : Fikry Khairytamim
NRP : 05111440000192
Jurusan : Teknik Informatika FTIK-ITS
Dosen Pembimbing 1 : Henning Titi Ciptaningtyas, S.Kom., M.Kom.
Dosen Pembimbing 2 : Dr.tech. Ir. R.V.Hari Ginardi, M.Sc

ABSTRAK

Di era yang serba digital ini tidak bisa dipungkiri kalau di dalam dunia perkuliahan teknologi terus berkembang. Dulunya absensi hanya menggunakan kertas, hingga saat ini sudah menggunakan teknologi sebagai penunjang. Jaringan yang digunakan merupakan Intranet dan hanya dapat dilakukan dalam jaringan. Koneksi di dalam jaringan juga dapat mempengaruhi kinerja dari transfer data yang dilakukan

Message Queue bisa dikatakan sebagai solusi dari permasalahan yang ada. Metode ini membantu seluruh data yang ada di aplikasi untuk ditampung kedalam *Message Queue*. Kelebihan dari *Message Queue* adalah membantu beberapa hal dalam penyimpanan data dan tidak berpengaruh apabila tidak ada koneksi ke database. Apabila *Message Queue* mati, state dan data yang ada akan terus di dalam. Sistem kerja *Message Queue* menerima setiap data yang di PUSH ke dalam dan menyimpannya pada memory, tapi juga menyimpannya di storage sebagai backup.

Dari hasil uji coba yang telah dilakukan, modul yang digunakan sangat handal, sehingga seberapa besar data yang dipindai akan tetap terbaca dan juga tertampung. Metode *Message Queue* yang digunakan berjalan dengan baik mulai dari menggunakan data sejumlah satuan hingga data besar sampai ratusan per mesin.

Kata kunci: *Message Queue, Sistem Kehadiran, Handal*

(Halaman ini sengaja dikosongkan)

IMPLEMENTATION OF MESSAGE QUEUE ON ATTENDANCE SYSTEM USING RASPBERRY PI WITH NFC MODULE

Student's Name : Fikry Khairytamim
Student's ID : 05111440000192
Department : Teknik Informatika FTIK-ITS
First Advisor : Henning Titi Ciptaningtyas, S.Kom., M.Kom.
Second Advisor : Dr.tech. Ir. R.V.Hari Ginardi, M.Sc.

ABSTRACT

In era of digital, it cannot be denied that the world of technology always improved. Formerly, attendance system only use paper, until recently used technology as supporting system. The network architecture used as an Intranet and can only be done in local area. Connection within the network may also affect the performance of data transfer.

Performed Message Queue can be said as the solution of the existing problems. This method helps to save all data in the application into Message Queue method. The advantages of Message Queue are to help data storage management and reliable if there is no connection to the database. When Message Queue is off, existing state and data will keep inside the machine. Message Queue work system accepts every data directly and stores it in memory, but also stores it in storage as a backup.

From the test results that have been done, the system is very reliable. so how big the data scanned, it will remain legible and also accommodated in the system. The Message Queue method runs well from small amount of data to hundreds of data up to 300 per machine.

Keywords : Message Queue, Attendance System, Reliable

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**IMPLEMENTASI *MESSAGE QUEUE* PADA SISTEM KEHADIRAN MENGGUNAKAN RASPBERRY PI DENGAN MODUL NFC**"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, Oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Allah SWT serta junjungan Nabi Muhammad SAW, karena limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Teknik Informatika ITS.
2. Kedua orangtua penulis, Papa dan Mama penulis, Dhea dan Davyna yang tiada hentinya memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Teman yang membantu pengerjaan, Muhammad Hilman, Syukron Rifa'il M. yang telah memberikan bantuan saat melakukan diskusi dalam pengerjaan tugas akhir ini.
4. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. dan Bapak Dr. Tech. Ir. R.V. Hari Ginardi, M.Sc selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknin Informatika ITS dan segenap dosen dan

karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di Teknik Informatika ITS.

6. Rekan seperjuangan Aviananda dan Glleen yang selalu memberikan bimbingan dan membantu ketika penulis menemukan kesusahan atau kesalahan dalam pengerjaan Tugas Akhir.
7. Orang-orang yanga ada di dalam Laboratorium Arsitektur dan Jaringan Komputer yang telah ramah menerima saya untuk fokus mengerjakan di ruangan tersebut.
8. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih memiliki banyak kekurangan. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis kedapannya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, Januari 2018

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 *Latar Belakang*

Di era yang serba digital ini tidak bisa dipungkuri kalau di dalam dunia perkuliahan teknologi terus berkembang. Dulunya absensi hanya menggunakan kertas, hingga saat ini sudah menggunakan teknologi sebagai penunjang.

Salah satu contoh yang ada, yaitu di Universitas nomor satu di dunia, Universitas Harvard. Di kampus terkemuka ini, mereka menggunakan kamera di setiap kelas untuk melakukan penganalisaan dari jumlah kursi yang kosong dan tidak. Setelah itu akan dilaporkan ke pihak yang bersangkutan [1].

Dari implementasi yang sudah berkembang saat ini, maka dibutuhkan suatu solusi. Solusi yang dibutuhkan adalah mengubah sistem autentifikasi yang masih menggunakan kertas dengan tanda tangan menjadi sesuatu yang sudah terotomatisasi dengan sistem yang dibuat.

Sistem yang dibuat merupakan alat yang sudah terintegrasi dengan aplikasi dan terhubung ke server, jadi semua data yang tercatat adalah ada yang transparan. Dan untuk autentifikasi menggunakan kartu mahasiswa yang memiliki NFC (*Near Field Communication*). Data yang sudah ada pun nantinya dapat diolah untuk kepentingan pengembangan lebih lanjut.

Dari sistem terbaru yang pernah diobservasi, sistem absensi menggunakan dua buah perangkat. Perangkat pertama sebagai pemberi kode dan perangkat kedua bertujuan untuk proses otentifikasi menggunakan foto atau tanda tangan. Perangkat pertama akan terus melakukan koneksi ke database untuk pengecekan jadwal. Perangkat kedua berbasis mobile apps dengan sistem operasi Android dan memiliki fitur pengecekan foto wajah dan tanda tangan.

Dari hal tersebut, muncul beberapa masalah yaitu pada koneksi yang dilakukan secara terus menerus yang membuat koneksi lain di dalam jaringan menjadi lambat dan absensi tidak dapat dilakukan apabila koneksi tidak ada.

Masalah tersebut memunculkan suatu solusi yaitu dengan membuat penampung data sementara di mesin kehadiran yang disebut *Message Queue*. Jika koneksi kembali tersedia, maka data akan dimasukan ke dalam database. Untuk pengecekan koneksi menggunakan *socket* dan nantinya saat koneksi tersedia, semua yang ada di dalam penampung akan dipindahkan ke dalam database.

Maka implementasi dari solusi yang ada adalah penggunaan *Message Queue* pada sistem kehadiran yang menggunakan modul NFC sebagai ID dari tiap mahasiswa. Dengan adanya alat ini, proses absensi menjadi lancar karena tidak memberatkan koneksi pada jaringan dan juga absensi dapat berlangsung walau terjadi kendala pada koneksi.

1.2 *Rumusan Masalah*

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana *Message Queue* dapat mengatasi masalah saat data banyak dan tidak ada koneksi ke database?
2. Apakah sistem kehadiran ini menggangu koneksi lain pada jaringan yang sama?
3. Apakah sistem kehadiran ini mempermudah proses absensi yang ada saat ini?

1.3 *Batasan Permasalahan*

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

1. Alat ini menggunakan modul dalam jumlah yang terbatas, dan untuk penggunaan awal hanya digunakan dalam lingkup MMT-ITS.
2. Jaringan yang digunakan untuk kebutuhan intranet, hal tersebut mempengaruhi performa kecepatan transfer data.
3. Kecepatan dari transfer data dipengaruhi oleh kekuatan dari intranet maupun internet masing-masing jaringan.
4. Alat yang digunakan adalah Raspberry Pi 3 Model B.
5. Teknologi utama yang digunakan oleh alat ini adalah Python, Node.js, Electron, PHP.
6. Kartu absen yang digunakan KTM elektronik, KTP, E-Card yang menggunakan tipe NFC ISO14443a.

1.4 *Tujuan*

Tujuan dari pembuatan tugas akhir selain memberikan data yang transparan, otomatisasi dalam penyimpanan data dan pelaporan juga dibuat di sistem ini. Hal paling penting adalah disaat koneksi tidak ada, data tetap dapat disimpan dan tidak memberatkan database maupun koneksi.

1.5 *Manfaat*

Tugas Akhir ini diharapkan dapat mempermudah dalam perekapan data karena transparansi dan otomatisasi yang diberikan. Selain itu, koneksi tidak harus selalu ada karena saat tidak ada koneksi, data masih dapat ditampung di mesin kehadiran dan tidak memberatkan koneksi di jaringan.

1.6 *Metodologi*

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir berisi pendahuluan, deskripsi dan gagasan metode – metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya Tugas Akhir, rumusan masalah yang diangkat, batasan masalah untuk Tugas Akhir, dan manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegitan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Tugas Akhir ini menggunakan literatur *paper* yang berasal dari jurnal internasional bereputasi yaitu IEEE untuk mencari informasi yang dapat dijadikan referensi dalam pengerjaan Tugas Akhir ini. Selain itu juga digunakan sejumlah referensi buku dan literatur lain yang berhubungan dengan metode enkripsi yang diusulkan pada Tugas Akhir ini termasuk tahap pembuatan sistem kehadiran pada MMT-ITS.

1.6.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal Tugas Akhir. Untuk membangun metode yang telah dirancang sebelumnya, maka dilakukan implementasi dengan membuat perangkat lunak dan perangkat keras. Perangkat lunak berada di mesin kehadiran dan server, sedangkan perangkat keras hanya berada pada sisi klien. Untuk implementasi Message Queue berada di antara aplikasi pemindai kartu dan server.

1.6.4 Pengujian dan Evaluasi

Selain uji coba implementasi sistem kehadiran, pada tahap ini dilakukan juga pengujian terhadap hasil implementasi dengan membandingkan performa dan keamanan metode *Message Queue.*

Uji coba yang dilakukan meliputi pemindaian kartu yang dapat diterima oleh server dalam kondisi koneksi ke server tersedia ataupun tidak tersedia.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 *Sistematika Penulisan Laporan*

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang teknologi, bahasa pemrograman, dan alat yang digunakan berdasarkan literatur.
3. Bab III. Perancangan Sistem
   Bab ini berisi pembahasan pembahasan secara detil dari sistem yang dibuat. Bentuk penjelasan berupa data gambaran dari implementasi seperti *use case*, flowchart, rangkaian modul, dan rangkaian sistem.
4. Bab IV. Implementasi

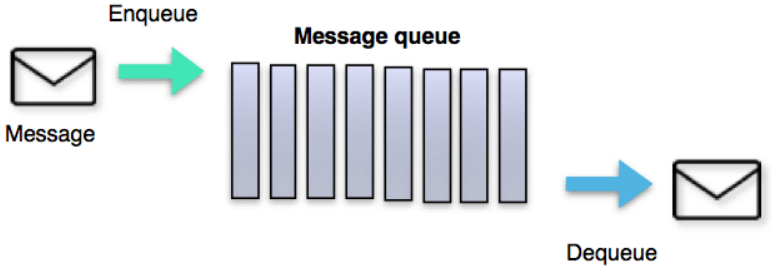
Bab ini menjelaskan implementasi dalam bentuk gambaran mesin dan sistem yang akan dibuat. Di dalamnya terdapat langkah-langkah pembuatan dari sistem kehadiran yang menerapkan *Message Queue*.

5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dari tes kasus yang disiapkan untuk menunjang dalam pembuktian metode yang telah ajukan dalam mengurangi suatu permasalahan.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan Kode Sumber 4.program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut diantaranya adalah penunjang untuk perangkat keras dan juga perangkat lunak. Beberapa contohnya yaitu *Message Queue,* Raspberry Pi 3 Model B, PN-532 dan beberapa teori lain yang mendukung pembuatan Tugas Akhir.

2.1 *Message Queue*



Gambar 2.1 Proses *Message Queue* [2]

Message Queue merupakan sebuah metode yang memungkinkan program-program untuk berkomunikasi diluar jaringan yang ada, tanpa memiliki hubungan koneksi. Dan ini merupakan sebuah cara yang sederhana dan sudah terbukti. Pada *Message Queue*, program berkomunikasi dengan menampung pesan atau data di *Message Queue*, dan kemudian mengambil pesan atau data dari *Message Queue* ke proses selanjutnya [1]. Ilustrasi dapat dilihat pada Gambar 2.1.

Tiga fakta kunci mengenai *Message Queue* yang membedakan dari gaya komunikasi lainnya:

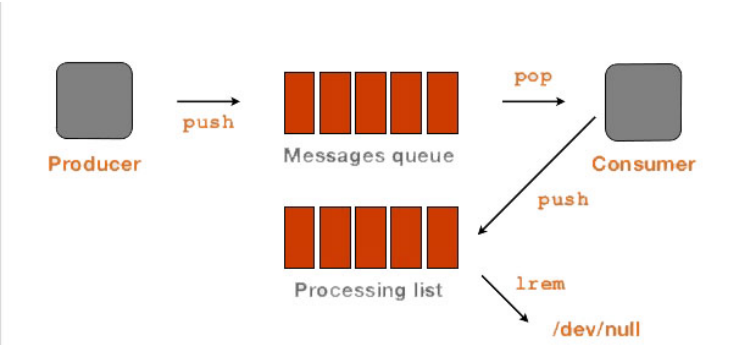
1. Program bagian komunikasinya dapat berjalan di waktu yang berbeda
2. Tidak ada batasan pada struktur aplikasi
3. Program terjaga dari kerumitan jaringan

Message Queue bisa dikatakan sebagai solusi dari permasalahan yang ada. Metode ini membantu seluruh data yang ada di aplikasi untuk ditampung kedalam *Message Queue.* Kelebihan dari *Message Queue* adalah membantu beberapa hal dalam penyimpanan data dan tidak berpengaruh apabila tidak ada koneksi ke database.

Apabila menerapkan metode *Message Queue*, *state* dan data yang ada akan terus di dalam. Sistem kerja *Message Queue* menerima setiap data yang dimasukkan ke dalam dan menyimpannya pada *memory*, tapi juga menyimpannya di *storage* sebagai *backup*.

Pada sistem kehadiran yang dibuat, *Message Queue* membantu proses absensi apabila terjadi kegagalan koneksi ke database dalam proses absensi. Saat proses request ke database gagal, maka data akan disimpan di dalam *Message Queue*. Saat koneksi tersedia kembali, data yang ada di dalam *Message Queue* akan di keluarkan dan dimasukan ke dalam Database.

2.2 *Redis*



Gambar 2.2 Message Queue Menggunakan Redis [4]

Redis adalah *open source* (BSD berlisensi), penyimpanan data dalam *memory*, digunakan sebagai database, *cache* dan *message broker*. Ini mendukung struktur data seperti *string*, *hash*, *list*, *set*, *set* diurutkan dengan berbagai *query*, bitmap, *hyperlog* dan *index geospacial* dengan *radius query*. Redis memiliki *built-in replication*, *scripting* Lua, *eviction* LRU, transaksi dan *on-disk level* yang berbeda, dan *high availability* melalui Redis Sentinel dan partisi otomatis dengan Redis Cluster [2].

Kelebihan dari penggunaan Redis yaitu memudahkan dalam proses penyimpanan data, koneksi, fitur database lainnya seperti query. Pada website Redis (https://redis.io/commands), sudah terdapat dokumentasi lengkap mengenai penggunaan dari *Message Queue*.

Dalam implementasi *Message Queue*, database Redis digunakan saat tidak tersedia koneksi ke database pusat. Perintah yang digunakan dalam sistem kehadiran yaitu *push* dan *pop*. Alurnya dimulai saat *request* ke database gagal, data akan di push oleh perintah Redis menggunakan python ke dalam database klien Redis. Saat data sudah di dalam database Redis, dan koneksi ada maka data satu per satu akan di pop ke database seperti pada Gambar 2.2 Masalah muncul apabila saat di pop gagal, yang dilakukan adalah melakukan *push* kembali ke dalam klien Redis.

2.3 *Python*

Python adalah *high-level programming language* yang banyak digunakan untuk pemrograman secara umum. Python menyediakan konstruksi yang dimaksudkan untuk memungkinkan penulisan program yang jelas baik dan mudah dalam skala kecil maupun besar.

Python memiliki sistem tipe data dinamis, manajemen memori otomatis, dan mendukung banyak paradigma pemrograman, termasuk pemrograman berorientasi objek,

imperatif, fungsional, dan prosedural. Ini memiliki *standart library* yang besar dan komprehensif [3].

Dalam sistem utama proses pengambilan data menggunakan python karena kekuatannya dalam proses data. Python secara garis besar mudah untuk dipelajari karena bahasanya yang lebih "manusia" dimana menggunakan kata kata yang umum digunakan. Di internet, banyak pembahasan yang menggunakan bahasa pemrograman Python dalam kasus NFC.

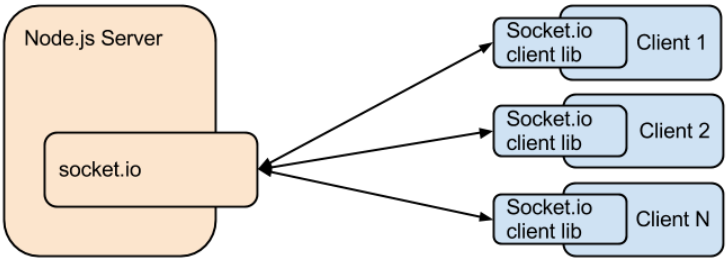
2.4 *Node.js*

Sebagai runtime JavaScript berbasis asinkron, runtime dirancang untuk membangun aplikasi jaringan terukur. Dikatakan asinkron karena proses yang dilakukan tidak harus menunggu proses sebelumnya selesai, selama data yang dibutuhkan sudah tersedia, maka proses akan berlangsung. Node.js bekerja dengan *thread* yang relatif lebih banyak dibanding pemrograman non-Javascipt lainnya.

Ini berbeda dengan model *concurrency* yang sekarang lebih umum digunakan. Jaringan berbasis thread ini relatif tidak efisien dan sangat sulit untuk digunakan. Sistem dari Node.js adalah asinkron yaitu tidak ada blok yang menunggu data kemudian proses baru dijalankan. Karena tidak ada blok, sistem terukur sangat masuk akal untuk dikembangkan di Node [4].

Node.js digunakan dalam membantu proses pengecekan koneksi ke server dan Penerimaan data dari Laravel. Pengecekan koneksi dilakukan dengan menginstall paket Socket.io yang berada di dalam Node.js menggunakan NPM. Untuk penerimaan data dari Laravel, Node.js digunakan oleh electron untuk mengambil data yang berada di laravel.

2.5 *Socket.io*



Gambar 2.3 Socket.io Berkomunikasi Antara Server Dan Klien

Socket.io, sejak awal aplikasi web, *developer* telah bekerja ke arah komunikasi dua arah antara server dan aplikasi [6].

Socket.io menyediakan *library* server dan klien yang mudah untuk dipahami. Hasil dari penggunaan Socket.io adalah data yang digunakan adalah *realtime* tanpa harus melakukan refresh pada browser. Socket.io adalah modul yang tersedia melalui npm Node.js seperti pada Gambar 2.3.

Setelah dilakukan penginstalan dari Socket.io di dalam *project*, maka socket.io dapat digunakan. Socket.io bertujuan untuk komunikasi antara Server dan Klien (Mesin Kehadiran). Tujuan akhirnya apabila Server dan Klien sudah saling terhubung, maka data yang ada di dalam klien Redis akan di POP ke dalam Database.

2.6 *Kerangka Kerja Laravel*

Laravel adalah kerangka kerja web PHP *open-source* yang gratis, dibuat oleh Taylor Otwell dan ditujukan untuk

pengembangan aplikasi web mengikuti pola arsitektur *model-view-controller* (MVC). Beberapa fitur Laravel adalah *module package system* yang menjadikannya mudah untuk digunakan hanya dengan melakukan instalasi pada paket [6].

Pengguna Laravel menjadi terbantu dalam proses pembuatan suatu aplikasi, banyak fungsi-fungsi yang membantu seperti *shortcut* dalam membentuk suatu sistem aplikasi.

Laravel digunakan dalam proses pengolahan data dari kartu yang sudah dideteksi dan juga sebagai API untuk nantinya didapatkan detil dari mahasiswa dan jadwal perkuliahan. Hasil dari proses API kemudian dikirim kembali ke mesin dan ditampilkan pada layar mesin kehadiran.

2.7 *Kerangka Kerja Electron*

Electron (sebelumnya dikenal sebagai Atom Shell) adalah *open-source framework* yang dibuat oleh Cheng Zhao, dan sekarang dikembangkan oleh GitHub. Hal ini memungkinkan untuk pengembangan aplikasi GUI desktop dengan menggunakan komponen *frontend* dan *backend* yang awalnya dikembangkan untuk aplikasi web: *runtime* Node.js untuk backend dan Chromium untuk frontend [8].

Electron adalah kerangka kerja GUI utama di balik beberapa proyek open-source yang terkenal termasuk *code editor* GitHub's Atom dan Microsoft Visual Studio Code. Electron digunakan karena kemudahan membuat aplikasi desktop dengan menggunakan bahasa pemrograman website. Untuk tampilan menggunakan HTML dan untuk *backend* menggunakan Node.js.

Dalam pengembangan Tugas Akhir, electron digunakan oleh klien (mesin kehadiran) untuk menampilkan detil mahasiswa dan jadwal perkuliahan yang sedang berlangsung.

2.8 *Kerangka Kerja Express.js*

Express.js adalah satu *web framework* paling populer di dunia Node.js. Dokumentasinya yang lengkap dan penggunaannya yang cukup mudah, dapat membuat kita mengembangkan berbagai produk seperti aplikasi web ataupun RESTful API. Express pun dapat digunakan menjadi dasar untuk membangun *web framework* yang lebih kompleks seperti, Sails.js, MEAN (MongoDB, Express.js, Angular.js, Node.js) dan MERN (MongoDB, Express.js, React.js, Node.js). Express dibuat oleh TJ Holowaychuk dan sekarang dikelola oleh komunitas [7].

Beberapa keunggulan yang dimiliki oleh Express antara lain:

1. Dukungan pembuatan *middleware*
2. Dukungan terhadap berbagai HTTP *verb* seperti POST, GET, PUT, DELETE, OPTION, HEAD, dan lainnya
3. Sudah terpasang *template engine* Jade
4. Manajemen *file* statik seperti CSS dan Javascript
5. Sangat bebas untuk dikostumisasi

Dalam pengimplementasiannya dalam sistem kehadiran ini, Express digunakan saat *request post* terjadi dari klien ke server. Dikarenakan express.js merupakan "web server' yang sangat dasar, dibutuhkan *plug-in body-parser* dari express.js untuk melakukan listen dari isi data yang dipost ke dalam server.

2.9 *Raspberry Pi 3 Model B*

Gambar 2.4 Raspberry Pi 3 Model B [11]

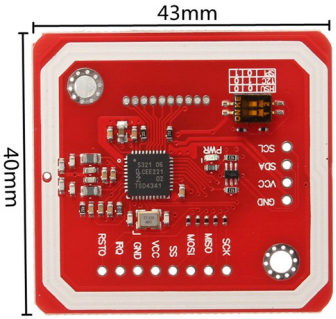
Raspberry Pi 3 Model B pada Gambar 2.4 adalah generasi ketiga Raspberry Pi. *Single Board Computer* bisa digunakan untuk banyak aplikasi dan menggantikan Model Raspberry Pi Model B + dan Raspberry Pi 2 yang asli B [9].

Dibanding *mini-computer* lainnya, Raspberry Pi memiliki kapabilitas yang paling besar karena memiliki komunitas yang banyak dan apabila terjadi kebingunan maka akan mudah untuk mencari masalah dan solusinya.

Dalam sistem kehadiran yang dibuat, Raspberry yang digunakan sebagai mesin utama. Dikatakan sebagai mesin utama karena semua proses dilakukan di Raspberry termaksud data yang terlempar database.

Proses yang terjadi di Raspberry termaksud pada proses pemindaian menggunakan modul NFC, pengiriman data ke server, proses penampungan data dengan *Message Queue*, dan menampilkan data ke layar Raspberry menggunakan Electron.

2.10 *PN-532*



Gambar 2.5 Modul NFC PN-532 [13]

PN532 di Gambar 2.5 menerapkan demodulator dan decoder untuk sinyal dari kartu dan transponder kompatibel ISO / IEC 14443A / MIFARE. PN532 menangani deteksi framing dan error ISO / IEC 14443A yang lengkap (Parity & CRC) [10].

PN532 mendukung mode emulasi kartu *MIFARE Classic 1K* atau *MIFARE Classic 4K*. PN532 mendukung komunikasi tanpa kontak menggunakan MIFARE Kecepatan transfer yang lebih tinggi, sampai 424 kbit / s di kedua arah. PN532 dapat mendemodulasi dan memecahkan kode sinyal kode FeliCa.

Pada PN-532, semua tipe dari kartu dapat digunakan, karena tiap kartu NFC memiliki tipenya masing masing. Modul NFC ini merupakan modul yang paling stabil dan paling banyak digunakan, ditambah modul ini cocok pada Raspberry Pi.

2.11 *Kartu NFC ISO14443a*

Gambar 2.6 Contoh Kartu NFC Yang Menggunakan Tipe ISO14443a [15]

Perangkat *Near Field Communication* menerapkan dukungan asli untuk tag ISO14443-A. Contoh *Anti-Crash* untuk mengurangi peluang, pesan inisialisasi yang digunakan untuk membuat saluran komunikasi dan untuk mengambil identifier dan fitur yang didukung dari sebuah tag. Selama fase pemindaian, terdapat tiga atau empat tipe berbeda yang dapat diterima dari sebuah tag (ATQA, UID, SAK dan ATS) [11].

Nilai ATQA, UID, SAK dan ATS dapat digunakan untuk mengidentifikasi produsen, jenis dan aplikasi tag. Tidak disarankan untuk menggunakan ATQA karena potensi *crash* bila lebih dari satu tipe dalam satu lokasi pemindaian [11].

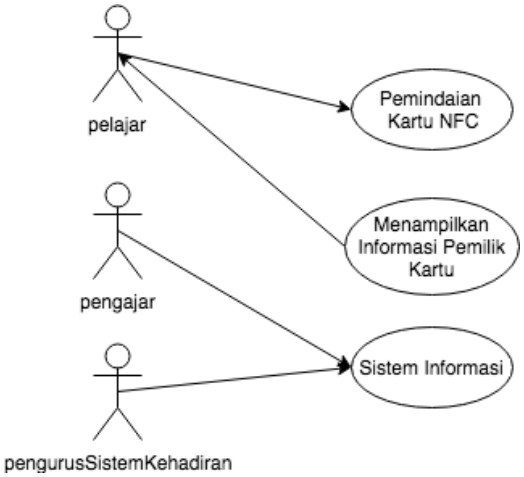
NFC ISO14443a digunakan karena sebagian besar kartu yang ada di Indonesia seperti pada Gambar 2.6, yaitu perbankan dan pemerintah menggunakan tipe ini. Karena dalam percobaan Sistem Kehadiran ini, Kartu NFC merupakan kartu mahasiswa yang juga sebagai kartu bank dan menggunakan NFC tipe ISO14443a. Apabila kehilangan kartu mahasiswa, pengguna dapat menggunakan e-KTP yang juga menggunakan tipe yang sama.

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan dan pembuatan sistem perangkat keras dan perangkat lunak. Sistem yang dibuat pada Tugas Akhir ini adalah sistem monitoring dari kehadiran menggunakan metode *Message Queue*. Pada bab ini pula akan dijelaskan gambaran umum sistem dalam bentuk diagram, skema, *flowchart,* maupun tabel deskripsi.

3.1 *Kasus Penggunaan*

Untuk penjelasan secara garis besar sistem akan ditampilkan dalam bentuk diagram. Penjelasan yang ada termasuk sistem di perangkat keras dan perangkat lunak. Aktor ada di dalamnya adalah pengurusSistemkehadiran, pengajar, dan pelajar. Untuk mengetahui kasus penggunaan secara umum dapat dilihat pada Gambar 3.1



Gambar 3.1 Diagram Kasus Penggunaan Pada Sistem Kehadiran

Deskripsi diagram kasus penggunaan pada mesin kehadiran dapat dijelaskan sesuai dengan ilustrasi diatas dinyatakan pada Tabel 3.1.

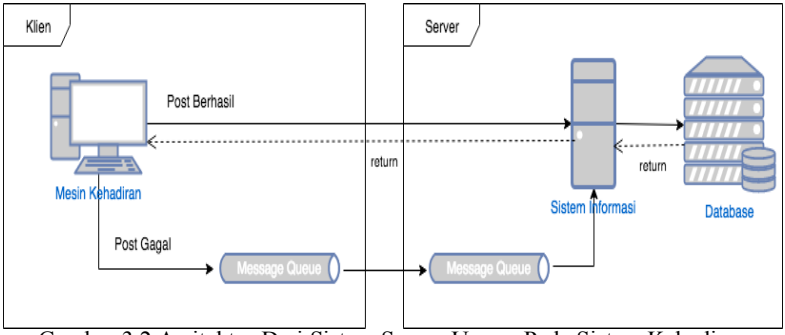
Tabel 3.1 Deskripsi Kasus Penggunaan Dari Sistem Kehadiran

No	Nama	Aktor	Deskripsi
UC001	Pemindaian Kartu NFC	Pelajar	Pelajar memindai kartu NFC dan dapat dipindai untuk selanjutnya akan di proses
UC002	Menampilkan Informasi Pemilik Kartu	Pelajar	Pelajar dapat melihat informasi yang didapatkan setelah pemindaian kartu NFC sebagai verifikasi
UC003	Sistem Informasi	Pengajar dan Pengurus Sistem Kehadiran	Aktor yang terlibat di dalam dapat melakukan CRUD pada data yang ada untuk mengelola sistem kehadiran sesuai dengan kebutuhannya

3.2 *Arsitektur Sistem*

Pada sub-bab ini akan membahas keseluruhan dari rancang bangun sistem kehadiran menggunakan metode *Message Queue* sebagai penampung dari data *real-time*.

Desain dari sistem dibagi menjadi dua bagian yaitu desain mesin kehadiran sebagai klien dan desain dari server. Di dalam desain mesin kehadiran terdapat penjelasan mendalam mengenai mesin kehadiran, *Message Queue,* dan aplikasi. Pada sisi server akan dijelaskan juga mengenai desain dari sistem informasi dan juga *web socket* untuk menghubungkan server ke klien-klien yang ada (mesin kehadiran).

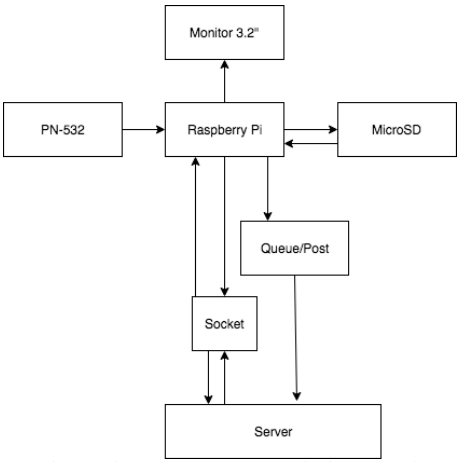


Gambar 3.2 Arsitektur Dari Sistem Secara Umum Pada Sistem Kehadiran

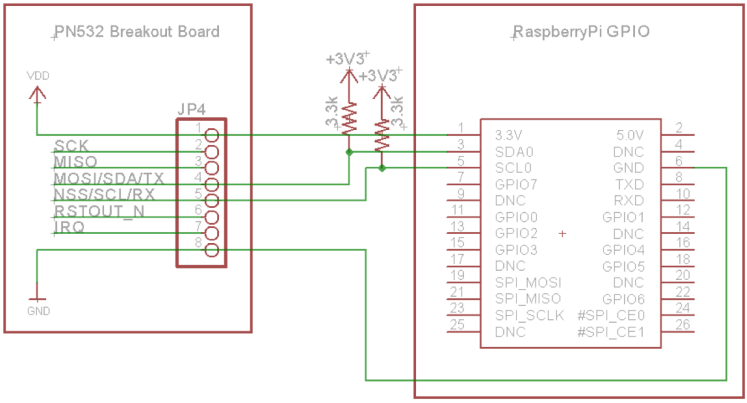
3.3 *Desain Umum Sistem Kehadiran*

Desain umum mesin kehadiran membahas mengenai proses yang terjadi pada awal sistem bekerja. Secara umum pembahasan yang dilakukan adalah pada proses awal pemindaian data yang terdapat di dalam kartu NFC ISO-14443A.

Dari kartu terpindai di dapatkan ID dan akan dimasukkan ke dalam proses *Message Queue* atau langsung kedalam server. Aplikasi di dalam mesin akan memberikan notifikasi jika data sudah tersimpan.



Gambar 3.3 Ilustrasi Secara Umum Arsitektur Mesin Kehadiran



Gambar 3.4 Skema PN-532 ke Raspberry Pi

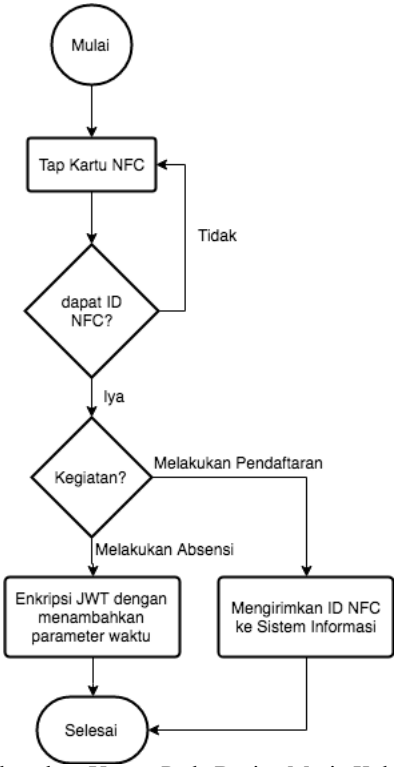
3.3.1 Desain Mesin Kehadiran

Dalam sistem mesin ada beberapa bagian yang menjadi pembahasan utama yaitu perakitan mesin kehadiran, pembacaan kartu NFC, penggabungan ID dan waktu sebagai pengenal, dan

juga proses memasukkan data yang tersimpan kedalam *Message Queue.*

Dalam tahap pembacaan NFC, python berperan dalam dalam pembacaan nfc dengan menjalankan program bahasa C yang telah di*compile* untuk mendapatkan ATS alias ID NFCnya.

Setelah ID didapatkan, kemudian disimpan untuk dilempar ke enkripsi dengan JWT dan ke server untuk didapatkan beberapa data yang nantinya akan dimunculkan kedalam layar mesin.



Gambar 3.5 Flowchart Umum Pada Bagian Mesin Kehadiran

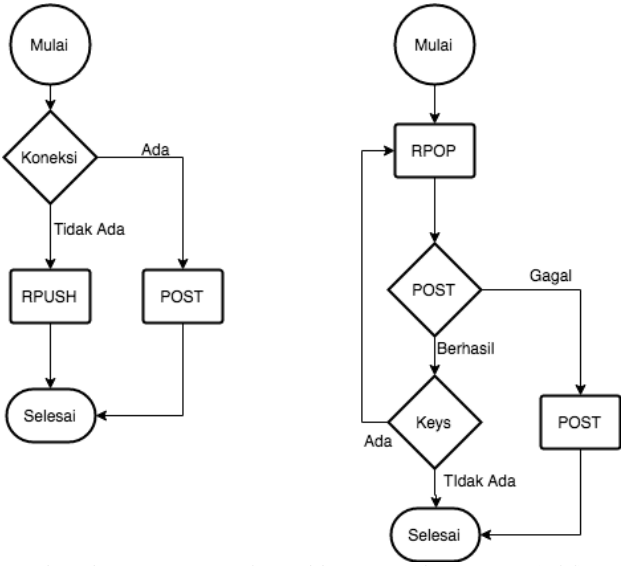
3.3.2 Desain *Message Queue*

Untuk metode *Message Queue*, aplikasi yang digunakan yaitu Redis sebagai *in-memory database*. Redis digunakan selain karena teknologinya yang *persistent* membuatnya memiliki performa yang konsisten dan maksimal sekalipun mesin kehadiran mati.

Di Redis, fungsi utama yang akan digunakan yaitu RPUSH, RPOP, dan LPUSH untuk mengeluarkan dan memasukkan data.

Jika data telah terdeteksi dan telah terenkrispi menggunakan JWT, maka akan masuk kedalam fungsi kirim untuk mengirimkan hasil enkripsi data NFC ke controller sistem informasi dan dilanjutkan ke database.

Apabila tidak ada respon dari server, maka data akan di *push* dengan RPUSH ke dalam *keys,* data yang tersimpan di dalam *keys* akan di pop dengan RPOP kedalam database. Jika terdapat koneksi maka semua data yang ada di dalam *keys* akan dimasukkan kedalam database. Dan saat koneksi terputus atau tidak ada koneksi maka data terakhir yang telah di pop akan di push dengan LPUSH ke antrian pertama.



Gambar 3.6 Flowchart Saat Data Dimasukkan Ke Dalam Queue (Kiri) Dan Saat Data Dikeluarkan Dari Queue Menuju Server (Kanan)

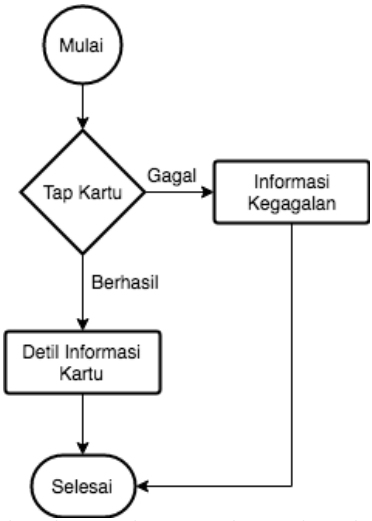
3.3.3 Desain Aplikasi

Aplikasi mulai bekerja saat data masuk dari server. Aplikasi yang digunakan berguna untuk menghubungkan data yang telah dibaca oleh python dan kemudian diteruskan ke server untuk didapatkan beberapa data.

Pada aplikasi ini menggunakan Electron sebagai kerangka kerjanya. Di dalamnya terdapat dua *web socket* sebagai server dan klien. Sebagai server berguna saat mendaftarkan kartu baru dan memunculkan datanya di sistem informasi. Untuk klien berguna disaat koneksi tersedia pada server, data yang sedang ditampung di dalam *Message Queue* akan pop dan dimasukkan kedalam database.

Hal lainnya yang dilakukan oleh aplikasi adalah menampilkan nama, ID, jadwal, dan informasi lainnya yang

dibutuhkan dengan melakukan permintaan dari aplikasi ke server dan mengembalikan dengan informasi yang sudah diolah.

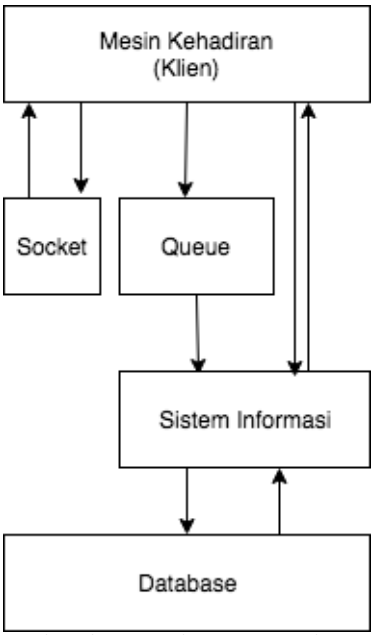


Gambar 3.7 Flowchart Dari Antarmuka Mesin Kehadiran

3.4 *Desain Umum Server*

Setelah banyak membahas pada bagian mesin yang ada disetiap ruangan, maka bahasan selanjutnya adalah mengenai penyimpanan dan proses pengolahan data.

Dalam hal ini ada dua hal yang akan dibahas yaitu desain dari sistem informasi dan desain *web socket*. Sistem informasi memiliki tugas besar dalam pengolahan data untuk disajikan dan dilaporkan, sedangkan *web socket* sebagai penghubung antara server dan mesin kehadiran.



Gambar 3.8 Flowchart Desain Umum Dari Server

3.4.1 Desain Sistem Informasi

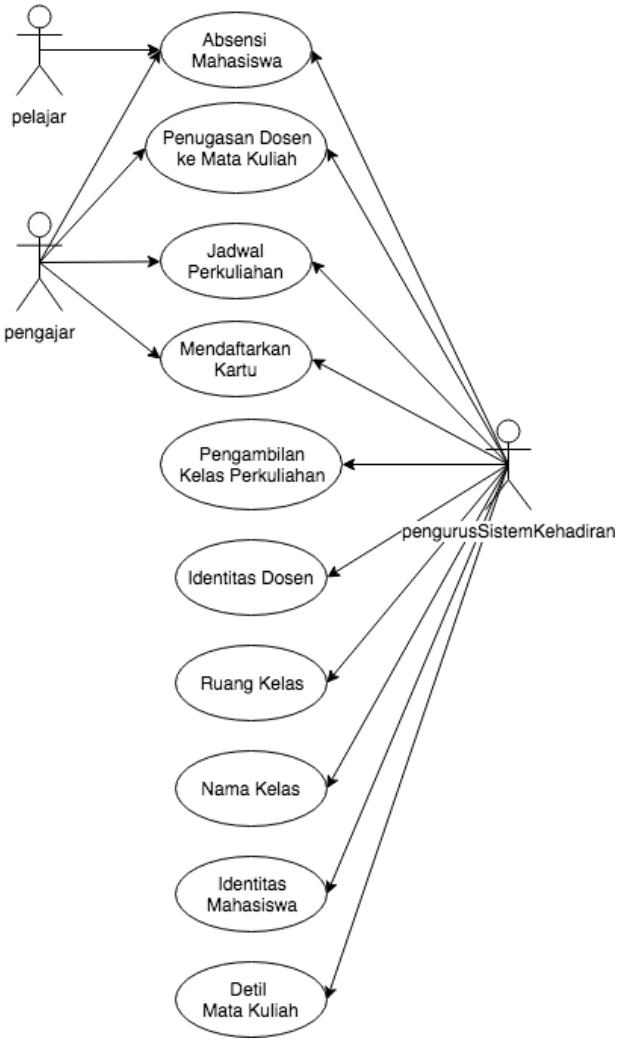
Sistem informasi merupakan pusat dari semua informasi berjalan, mulai dari proses ID NFC yang di dapat sampai data diproses untuk dijadikan laporan.

Membantu pada proses registrasi, melakukan pendaftaran terhadap banyak hal mulai dari pendaftaran mahasiswa hingga mendaftarkan matakuliah ke ruang kelas.

Menerima data dan dimasukan ke dalam database yang datanya berasal dari mesin kehadiran. Dengan mengetahui IP mesin, pendaftaran dapat dilakukan.

Melakukan pembaharuan juga dapat dilakukan secara menyeluruh, mulai penggantian jadwal matakuliah, penggantian ruangan, penambahan jadwal, penggantian dosen pengajar, hingga

rekap absensi mahasiswa sekalipun dapat dilihat di dalam sistem informasi ini. Bisa dilihat pada Gambar 3.9.

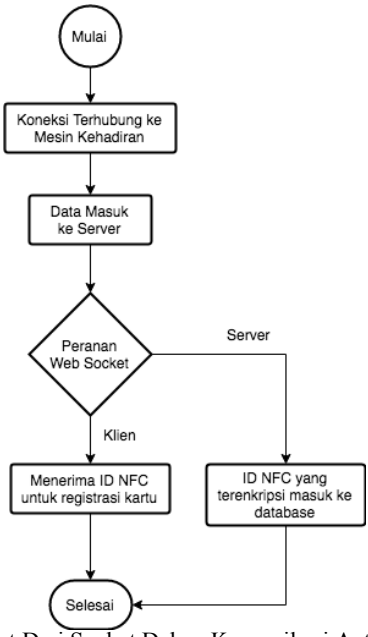


Gambar 3.9 Diagram Kasus Penggunaan Sistem Informasi

3.4.2 Desain Web Socket

Web Socket yang ada di dalam server membantu server untuk melakukan pengecekan koneksi pada tiap mesin kehadiran. Jika koneksi terjalin antara server dan mesin kehadiran, maka data yang ada di mesin kehadiran akan diambil dan kemudian di proses di dalam server.

Web Socket pada server dapat berperan sebagai klien maupun server. Server berperan sebagai sebagai socket klien saat mendaftarkan kartu baru, karena ID NFC akan muncul di dalam Sistem Informasi yang berada di server. Server berperan sebagai socket server saat server menunggu koneksi dari klien-kliennya, jika terdapat respon dari klien maka data yang ada di klien akan dimasukkan ke dalam database dan di proses di sistem informasi.



Gambar 3.10 Flowchart Dari Socket Dalam Komunikasi Antara Klien Dan Server

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi yang dilakukan adalah pembuatan dari dua buah perangkat mencangkup berangkat keras dan juga perangkat lunak. Perangkat keras yang dibuat berupa beberapa mesin kehadiran berperan sebagai klien dan perangkat lunak yang dibuat berperan dalam interface dari klien menuju ke server serta pengolahan data di dalamnya.

4.1 *Lingkungan Implementasi*

Implementasi pembuatan perangkat keras dan perangkat lunak mulai dari pembuatan hingga alat jadi. Pembagian dibagi menjadi 4 bagian yaitu perangkat keras untuk server dan klien, juga perangkat lunak untuk server dan klien.

Untuk bagian klien, perangkat yang digunakan merupakan hasil perakitan dari beberapa modul yaitu Raspberry Pi sebagai Perangkat utama, modul NFC, Monitor, Adaptor, dan SD Card untuk melakukan penyimpanan dari data.

Pada bagian server, yang digunakan adalah server dari Laboratorium Arsitektur dan Jaringan Komputer (AJK). Dengan alamat 10.151.36.98 dan sudah terinstall beberapa perangkat lunak untuk menjalankan aplikasi.

Tabel 4.1 Spesifikasi Untuk Mesin Kehadiran dan Server Dalam Proses Impelementasi

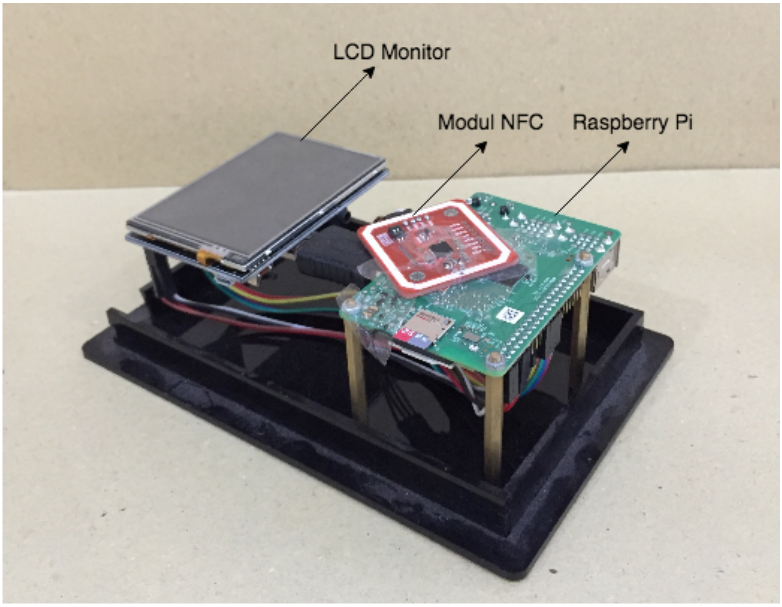
Perangkat	**Jenis**	**Spesifikasi**
Perangkat Keras Mesin Kehadiran	Mini Komputer	Raspberry Pi 3 Model B

Perangkat	Jenis	Spesifikasi
	Modul NFC	PN-532
	SD Card	SanDisk Ultra 16GB
	Monitor	KeDei 3.5 inch 480x320
	Adaptor	Part Resmi Raspberry Pi
Perangkat Lunak Mesin Kehadiran	Sistem Operasi	Raspbian
	Perangkat Pengembang	Python Versi 2.7.14
		Socket.IO 2.0
		Node.js v8.9.1
		Express.js 4.16
		Redis 4.0.2
		JSON Web Token
Perangkat Keras Server	Prosesor	Intel(R) Core(TM) I3 2120
	Memori	8 GB
Perangkat Lunak Server	Sistem Operasi	Linux Ubuntu 14.04.3
	Perangkat Pengembang	Python Versi 2.7.14
		Socket.IO 2.0
		Node.js v8.9.1
		Express.js 4.16
		Redis 4.0.2

Perangkat	Jenis	Spesifikasi
		JSON Web Token
		Laravel 5.4
		MySQL 5.7.19

4.2 *Implementasi Perangkat Keras Mesin Kehadiran*

Dalam Implementasi yang dilakukan pada bagian perangkat keras, beberapa bagian digabungkan menjadi satu. Secara garis besar, bagian yang digunakan dalam mesin kehadiran adalah Raspberry Pi, Modul NFC, SD Card, Adapter, dan Monitor.



Gambar 4.1 Bagian dari Mesin Kehadiran

Untuk lebih lanjutnya, implementasi mesin kehadiran akan dijelaskan secara langkah demi langkah.

Langkah awal yang dilakukan yaitu menghubungkan modul NFC ke Raspberry Pi. Solder bagian dari PN-532 dengan menggunakan 2 buah 3-pin dan sebuah 8-pin. Solder 3-pin ke **SEL0** dan **SEL1** pada modul NFC, kemudian siapkan Raspberry Pi untuk menghubungkan menggunakan kabel penghubung.

Hubungkam beberapa pin ke Raspberry Pi, dengan ketentuan:

1. PN532 3.3V ke Raspberry Pi 3.3V (kabel merah)
2. PN532 SCK ke Raspberry Pi GPIO # 25 (kabel kuning)
3. PN532 MISO ke Raspberry Pi GPIO # 24 (kabel hijau)
4. PN532 MOSI ke Raspberry Pi GPIO # 23 (kabel oranye)
5. PN532 SSEL ke Raspberry Pi GPIO # 18 (kabel biru)
6. PN532 GND ke Raspberry Pi GND (kabel hitam)

Setelah menyabungkan semua kabel, hubungkan PN532 ke Raspberry PI menggunakan aplikasi SPI yang hanya membutuhkan input dan output digital untuk koneksi.

Selain itu untuk layar menggunakan merek Kedei, yaitu monitor dengan layar sentuh yang berguna untuk memunculkan tampilan Raspberry Pi ke layar dengan menggunakan HDMI (*High Definition Multimedia Interface*). Untuk selanjutnya, yang dilakukan adalah mempelajari library python-nfc.

4.3 *Implementasi Perangkat Lunak Mesin Kehadiran*

Pada sub-bab implementasi mesin kehadiran, keseluruhannya akan membahas mengenai proses dari awal mesin dipindai sampai pada mengecekan koneksi ke server dan *push* data yang di dalam *queue* ke server saat koneksi tersedia. Dibawah ini akan dijelaskan secara detil mengenai proses langkah-langkah data yang ada di dalam kartu bisa menghasilkan sebuah data balik dan proses tidak berpengaruh saat koneksi ada maupun tidak.

4.3.1 Pemindaian Kartu NFC

Dalam proses ini bahasa yang digunakan adalah python, karena *library* untuk proses komunikasi langsung dari python ke modul NFC sudah tersedia. Langkah yang pertama kali dilakukan adalah mendeklarasikan tipe dan port dari modul NFC yang digunakan.

```
while True:
    clf = nfc.ContactlessFrontend("tty:AMA0:pn532")
```

Kode Sumber 4.1 Deklarasi Tipe Dan Port Menggunakan Library NFC Pada Python

Kemudian setelah koneksi dilakukan, pemilihan bitrate dan tipe modulasi agar modul dapat berkomunikasi dengan program yang dibuat.

Setelah berhasil dan cocok, process lain dilakukan yaitu kunci utama dari proses yang dilakukan yaitu memindai kartu dan mendapatkan *serial number* dari masing masing kartu. Untuk detil dari kartu NFC yang dapat digunakan adalah kartu NFC ISO14443a yang digunakan oleh bank dan pemerintahan Indonesia. Yang dilakukan oleh program python adalah menjalankan *compiled program* dengan nilai balik yaitu s*serial number* dari kartu.

```
target = clf.sense(nfc.clf.RemoteTarget("106A"),
nfc.clf.RemoteTarget("212F"),nfc.clf.RemoteTarget("424F
"))
clf.close()
tes=subprocess.check_output([dir_path+"/getats"])
```

Kode Sumber 4.2 Proses Menjalankan Aplikasi Untuk Mendapatkan Serial Number

Dalam proses perhitungan dari *serial number,* proses perulangan dari tiap string dilakukan. Pengambilan string dilakukan dengan kenaikan "+1" sampai menemukan "\n" pada saat melakukan perhitungan *serial number.*

Serial number didapatkan, kemudian ditampung pada suatu variabel yang nantinya akan disimpan untuk dikirimkan ke server dengan *request post.*

```
serial = serialnumber(tes)
       msg = {}
       msg['serial'] = serial
       msg['response'] = serial
```

Kode Sumber 4.4 Penyimpanan Serial Number Ke Dalam Array

```
       sys.stdout.flush()
       send = kirim(serial)
```

Kode Sumber 4.3 Langkah Awal Untuk Pengiriman Serial Number Ke Server

4.3.2 Request Post ke Server

Pada saat pemindaian data, nilai yang dikirim ke fungsi ini adalah serial number dari kartu NFC yang digunakan dan sudah melalui proses pemindaian.

```
def kirim(id):
       token = {
              'id' : id,
               'waktu' : int(time.time())
       }
```

Kode Sumber 4.5 Pendeklarasian ID Dan Waktu Dalam Bentuk JSON

Selanjutnya, data yang disimpan akan dimasukkan ke dalam fungsi "kirim". Sebelum dikirim data harus dienkripsi

terlebih dahulu menggunakan JWT. Data yang terenkripsi akan disimpan dalam fungsi *request* bawaan python.

Token pada kode diatas bertujuan untuk membuat token dengan menggunakan JWT. Data yang dibutuhkan meliputi ID dari kartu NFC dan juga waktu saat melakukan tap pada modul NFC.

```
try:
        r = requests.post
        ('http://10.151.64.194:9999/absen/add',
        data = {'token': SignEncrypt(id,
        time.time()+60)}, timeout=3)
        obj = r.text
```

Kode Sumber 4.6 Proses Pengiriman Data Ke Server

Untuk *request* yang dilakukan menggunakan timeout selama 3 detik. Jadi apabila selama 3 detik tidak didapatkan ID dari kartu NFC maka akan masuk bagian pengecualian yang menggunakan metode *Message Queue.*

4.3.3 Enkripsi Data Menggunakan JWT

Sebelum melakukan post ke server, data yang ada akan di enkripsi menggunakan JWT. Data dari token yang berisi id dan waktu akan dilempar ke fungsi SignEncrypt.

Langkah-langkah dalam pembuatan token dalam enkripsi menggunakan JWT yaitu melakukan *signature* dan enkripsi. *Signature* digunakan untuk memverifikasi jikalau pengirim JWT adalah orang yang memang mengirimkan data dan memastikan bahwa pesan yang dikirimkan tidak berubah selama perjalanan.

Yang pertama kali dilakukan adalah menspesifikasikan *key* yang digunakan untuk *signature.* Kemudian, setelah data di *signature* maka data akan di enkripsi. Ada banyak algoritma yang dapat digunakan di dalam JWT, oleh karena itu gunakan algoritma yang dibutuhkan.

Dalam implementasi enkripsi ini, digunakan 3 parameter dalam pengenkripsian yaitu id kartu NFC, waktu saat di tap, dan lama waktu token tersebut kadaluarsa.

Setelah *signature* dan enkripsi dilakukan, maka nilai balik akan dibalikkan dengan menampilkan hasil enkripsi atau *signature* dalam bentuk token.

```
def SignEncrypt(id, time):
    signkey = {"k":"QMeHEupswaLv5uFNPgqdZF-
PsAs9_emFG8g-aear8XM","kty":"oct"}
    signjwk = jwk.JWK(**signkey)
    Token = jwt.JWT(header={"alg": "HS256"},
                        claims={
                            "id": id,
                            "to": socket.gethostname(),
                            "exp": time
                        })
    Token.make_signed_token(signjwk)
    encryptkey = {"e":"AQAB","kty":"RSA","n":"uEJAcl-
SyK4fN6M8ugfK8U4um6uOrTB0BXcGJ7b2eizm6XGpC3QFAyTNmF15rw
54RaSZoBXj85oXRGPJVxLdEdX7vvxHH0w6UpgF-0dVw-
2_oiGbapev4bqJ4iSJYyORs2giQo4O4DLi2zF15WeqNJHVbEju5GvLo
6kgYeaYkmm0PBPJOgm3Ftmidmdku52l70MnnkZNxfbqQ5adDCf20l7_
x83-Vdn6M_bFSQlbZYO8KJEm0pK3l0GuLobERZCRuUAKF-
8weHnWYxKgQqhU2mg69dTr6L6MxVyk0vKLzhicb-
XaCOWbO97BJ6UgCwVix1qGMxLqjtrp4hjGjO-91Q"}
    encryptjwk = jwk.JWK(**encryptkey)
    EToken = jwt.JWT(header={"alg": "RSA-OAEP", 'enc':
"A256GCM"},
                        claims=Token.serialize())
    EToken.make_encrypted_token(encryptjwk)

    return EToken.serialize()
```

Kode Sumber 4.7 Proses Melakukan Enkripsi Menggunakan JWT

4.3.4 Implementasi *Message Queue* Pada Mesin Kehadiran

Data token didapat setelah dilakukan enkripsi dengan JWT sebelum melakukan *request post* ke server. Masalah akan muncul ketika data gagal untuk di *post* ke server, maka data akan hilang. Dikarenakan masalah tersebut, dibutuhkan pengecualian yaitu dengan penambahan penampung atau nama umumnya *Message Queue* di dalamnya.

Dalam pengembangannya database yang digunakan adalah Redis, yaitu database yang berbasis memori dimana data yang disimpan di dalam memori. Redis juga tangguh, dimana saat redis mati, data secara otomatis tersimpan ke *storage*.

Program dibuat dalam bahasa python, maka dibutuhkan library yang dapat menyambung kan python dengan Redis. Didapatkan *library* yang dibuat oleh Peter Hoffman dengan membuat fitur dasar dari *Message Queue* seperti PUSH, POP, DEL, maupun koneksi ke Redis.

Dari *library* yang telah dipublikasikan, python dan Redis dapat berkomunikasi dengan lebih mudah. Jika *request post* yang dilakukan gagal maka akan masuk ke pengecualian. Hal yang pertama kali dilakukan adalah melakukan koneksi dan juga membuat nama dari *key* untuk penyimpanan data.

Saat koneksi berhasil maka "database" kartu sudah dibuat. Sebagai gambaran dari proses sebelumnya, data gagal di *post* maka data yang gagal tersebut harus ditampung ke dalam Message Queue dengan melakukan PUSH. Data yang di PUSH ke dalam redis sudah dalam bentuk JSON dimana menyimpan data ID dan waktu saat tap kartu. Dalam library yang digunakan, fungsinya adalah put(), dimana data akan di PUSH ke dalam Redis ke key yang sudah dibuat sebelumnya. Saat berhasil maka akan menampilkan juga data balik ke layar mesin dalam bentuk objek.

```
import redis

class RedisQueue(object):
    def __init__(self, name, namespace='queue',
**redis_kwargs):
        self.__db= redis.Redis(**redis_kwargs)
        self.key = '%s' %(na)
    def qsize(self):
        return self.__db.llen(self.key)
    def empty(self):
        return self.qsize() == 0
    def put(self, item):
        self.__db.rpush(self.key, item)
    def pop(self):
        """Put item into the queue."""
        return self.__db.rpop(self.key)
    def get(self, block=True, timeout=None):
        if block:
            item = self.__db.blpop(self.key,
timeout=timeout)
        else:
            item = self.__db.lpop(self.key)

        if item:
            item = item[1]
        return item

    def get_nowait(self):
        """Equivalent to get(False)."""
        return self.get(False)
```

Kode Sumber 4.8 Library Penggunaan Python-Redis

```
except requests.exceptions.RequestException as e:
        q = RedisQueue('kartu')
```

Kode Sumber 4.9 Pengecualian Untuk Implementasi Message Queue

```
       q.put(json.dumps(token))
       obj = '{"response": "Tidak ada respon dari server,
data telah ditampung!"}'
    return obj
```

Kode Sumber 4.10 Proses Penampungan Data Ke Dalam Redis

Proses awal saat gagal *post* selesai, data telah ditampung di dalam key yang ada di dalam Redis. Untuk proses selanjutnya, data yang telah ditampung harus segera di POP jika ada koneksi. Hal yang selanjutnya dilakukan yaitu menggunakan library Python untuk Redis membuat koneksi dan pengecekan key. Kemudian akan masuk ke fungsi POP dari Redis.

```
if __name__ == "__main__":
       q = RedisQueue('kartu')
       main()
```

Kode Sumber 4.11 Inisiasi Python-Redis Ke Server

Dalam fungsi main akan menjalankan perintah jika isi dari key yang ada di dalam Redis tidak kosong, maka data akan di POP. Jika data di POP, maka akan didapatkan nilai baik berupa isi data yang telah di POP. Kemudian data nilai baik disimpan untuk nantinya akan di *post* ke server melalui socket. Sebelum *repost post*, data terlebih dahulu di enkripsi dengan JWT, langkah-langkah yang dilakukan sama seperti penjelasan sebelumnya.

```
while q.empty() is False:
        try:
            isi = q.pop()
            data = json.loads(isi)
            r = requests.post('http://10.151.36.116:9999', data =
{'token': SignEncryptWorker(data['id'], int(data['waktu']))}, timeout=3)
            print r.text
            sys.stdout.flush()
```

Kode Sumber 4.12 Melakukan Pop Data dan Request Post Ke Server

Untuk memastikan data tidak hilang saat sedang di pop dan *post* menuju server, apabila saat melakukan *post* dan tidak ada respon dari server selama 3 detik maka akan ditampung kembali ke dalam *Message Queue*.

Pengecualian dibuat apabila setelah 3 detik tidak ada respon dari server maka data akan di PUSH kembali dan worker selesai bekerja.

```
except requests.exceptions.RequestException as e:
            q.put(isi)
            exit()
```

Kode Sumber 4.13 Kasus Ketika Redis Gagal Melakukan Post

4.3.5 Komunikasi Dengan Server

Semua otomatisasi yang dilakukan mulai dari pemindaian, *post,* dan *Message Queue* berasal dari (electron. Di dalam file main.js, proses spawn dan socket beroperasi.

```
const electron = require('electron')
const app = electron.app
const BrowserWindow = electron.BrowserWindow
const path = require('path')
const url = require('url')
const os = require('os')
const spawn = require('child_process').spawn
const ip = require('./ip.js');
const io = require('socket.io')(8888);
```

Kode Sumber 4.14 Inisiasi Library Pada Electron Klien

Proses pertama adalah saat window dari electron berhasil di load, maka electron akan melakukan spawn untuk menjalankan program pemindaian kartu dengan nama file kelas.py. Di dalam fungsi ini data dapat berubah pada layar mesin kehadiran menggunakan emit data melalui socket.

Kemudian data akan dikirim kan melalui socket, jika terdapat koneksi maka data yang ditampung akan dikirim oleh worker ke server.

```
mainWindow.webContents.send('kelas', hostname.replace(/_/g,' '));
const py = spawn('python', [__dirname+'/kelas.py'])

py.stdout.on('data', function(data){
        msg = JSON.parse(data.toString());
        if(typeof msg.serial !== 'undefined'){
           io.emit('daftar', msg);
        }
        else{
           mainWindow.webContents.send('message', msg);
        }
    });
```

Kode Sumber 4.16 Melakukan Spawn File Python Dari Electron

```
const socket = require('socket.io-
client')('http://10.151.64.194:9999');
    socket.on('connect', function(){
    socket.emit('id', hostname);
    const worker = spawn('python',
[__dirname+'/worker.py'])
     worker.stdout.on('data', function(data){
        msg = JSON.parse(data.toString());
        console.log(msg);
        mainWindow.webContents.send('message', msg);
      });
      console.log('connected, running worker');
    });
```

Kode Sumber 4.15 Proses Penggunaan Socket.io Untuk Menjalankan Worker

Sekalipun data terputus, handling melalui socket tetap ada. Seperti saat koneksi putus, pada terminal dari program akan memberikan respon melalui console log. Atau apabila pemindaian kartu gagal karena tidak dapat komunikasi dengan modul NFC, repon akan muncul pada layar mesin kehadiran.

```
socket.on('disconnect', function(){
      console.log('disconnected');
    });
```

Kode Sumber 4.19 Respon Saat Socket Tidak Terhubung

```
py.stdout.on('end', function(data){
      console.log('modul terputus')
     mainWindow.webContents.send('app_closed', 'modul
nfc terputus');
    });
```

Kode Sumber 4.18 Respon Saat Modul NFC Gagal Di Jalankan

4.4 *Implementasi Perangkat Lunak Server*

Dalam sub-bab ini fokus utama dari proses yang di jelaskan adalah saat data yang dikirim dan diterima oleh server. Data yang telah diterima oleh server melalui *post* atau masuk dari socket akan diterima dan diproses lebih lanjut di dalam system informasi yang telah dibuat. Untuk kali ini nilai balik yang akan dikembalikan berupa data mahasiswa ataupun respon untuk mesin kehadiran.

Pada gambar diatas merupakan variabel dan *library* apa saja yang dibutuhkan. Teknologi yang dibutuhkan termaksud *Web Socket, Spawn, Web Server,* Redis, dan Promise. Semua yang digunakan menggunakan Node.js sebagai penghubungnya.

```
var client = {};
var ruanganlist = {};
let webServerPort;
var net = require('net');
var HOST = '127.0.0.1';
var PORT = 8000;
var cek=0;
const spawn = require('child_process').spawn
var app = require('express')();
var server = require('http').Server(app);
var io = require('socket.io')(server);
var jose = require('node-jose');
server.listen(9999);
var redis =require("redis");
var masukin = redis.createClient({
      enable_offline_queue:false,
    });
var bluebird = require("bluebird");
bluebird.promisifyAll(redis.RedisClient.prototype);
bluebird.promisifyAll(redis.Multi.prototype);
var bodyParser = require('body-parser')
app.use( bodyParser.json() );
app.use(bodyParser.urlencoded({
  extended: true
}));
```

Kode Sumber 4.20 Inisiasi Library Pada Electron Server

4.4.1 Komunikasi Dengan Mesin Kehadiran

Semua komunikasi yang terdapat di server terdapat di dalam file server.js. Di dalamnya terdapat express.js, spawn, socket, dan juga Promise untuk menangkal proses asinkron dari JavaScript. Ada juga fungsi body-parser dalam express.js untuk membantu menangkap body dari *request post* yang dikirim.

Terdapat dua bagian komunikasi yang bekerja, yaitu bagian koneksi broadcast ke semua klien untuk membantu mengecek koneksi pada server jikalau tersedia dan data dapat dikirim.

```
io.on('connection', function (socket) {
  console.log(socket.id+' connected');
  socket.on('disconnect', function () {
    console.log(socket.id+' disconnected');
    delete
ruanganlist[client[socket.id]].splice(ruanganlist[clien
t[socket.id]].indexOf(socket.id), 1);
    if (typeof ruanganlist[client[socket.id]][0] ===
'undefined') delete ruanganlist[client[socket.id]];
       delete client[socket.id];
       console.log(client);
       console.log(ruanganlist);
  });
  socket.on('connected', function(){
       console.log('konek');
  });
  socket.on('id', function (msg){
       client[socket.id] = msg;
       if (typeof ruanganlist[msg] !== 'undefined') {
         ruanganlist[msg].push(socket.id);
       }
       else ruanganlist[msg] = [socket.id];
       console.log(client);
       console.log(ruanganlist);
  });
});
```

Kode Sumber 4.21 Socket Pada Server Saat Ada Koneksi

Di dalam proses socket ke klien, koneksi selalu di cek apakah tersedia atau tidak dengan menerika array data yang dikirim. Untuk kasus mesin absensi, data yang dikirim dari klien ke server adalah data id socket dan nama ruang kelas yang diambil dari hostname tiap mesin absensi.

Bagian yang berikutnya adalah pengecekan koneksi ke aplikasi laravel sekaligus menjalankan worker untuk memasukkan data ke dalam aplikasi sistem informasi.

```
function b(){
    webServerPort = new net.Socket();
    webServerPort.setKeepAlive(true);
    webServerPort.connect(PORT, HOST, function(){
        if (cek == 1){
            console.log('cek sekali'+cek)
            cek = 0;
        }
        else{
            io.emit('worker', 'jalan');
            cek = 0;
        }
        console.log('CONNECTED TO: ' + HOST + ':' +
PORT);
    });

    webServerPort.on('error', function(e) {
        if(e.code == 'ECONNREFUSED') {
            webServerPort.destroy()
        }
        webServerPort.destroy()
    });
    webServerPort.on('data', function(data) {
        console.log('DATA: ' + data);
        webServerPort.destroy();
    });
    webServerPort.on('close', function() {
        ++cek;
        b();
    });
}
```

Kode Sumber 4.22 Proses Pengecekan Koneksi

Dalam proses pengecekan koneksi ke laravel, dilakukan pembuatan socket baru dan juga menjalankan fungsi agar pengecekan port dilakukan secara terus menerus. Penggunaan state

dilakukan dalam memberikan notifikasi untuk menjalankan worker.

Apabila worker sudah selesai benerkan maka koneksi akan diputus. Kasus berikutnya yang memungkinkan terjadi adalah saat koneksi tidak dapat dilakukan, maka koneksi akan diputus dengan menggunakan webServerPort.destroy().

4.4.2 Dekripsi Data Menggunakan JWT

Dari token yang telah dikirim melalui socket, akan diterima oleh express.js menggunakan body-parser yang merupakan fungsi dari express.js. Selanjutnya yang dilakukan adalah melakukan dekripsi dari token yang ada, kemudian hasil dekripsi diverifikasi.

```
var TESSS = '{"$key"}';
var kunciVerif = '{"k":"QMeHEupswaLv5uFNPgqdZF-
PsAs9_emFG8g-aear8XM","kty":"oct","alg":"HS256"}';

let VerifyDecrypt =  function(token){
    return jose.JWK.asKey(JSON.parse(TESSS)).
            then(function(key) {
              return key
            });
}
var JWK = VerifyDecrypt('dsadas');
JWK.then(function(result){
  jose.JWE.createDecrypt(result).
    decrypt(req.body.token).
    then(function(ea) {
          jose.JWS.createVerify(key).
            verify(ea.payload.toString())
        })
    })
```

Kode Sumber 4.23 Proses Dekripsi dan Verifikasi

Dari token yang sudah melalui dua langkah yaitu dekripsi dan verifikasi, data yang ada telah dikembalikan seperti sebelum

dilakukan enkrpsi oleh JWT. Selanjutnya, data akan diteruskan dengan melakukan PUSH ke dalam Redis dengan *Message Queue.*

4.4.3 Implementasi *Message Queue* Pada Server

Data yang telah didapatkan oleh server dan telah dienkripsi, kemudian akan disimpan dalam *payload*. Data selanjutnya akan di PUSH menggunakan Promise ke dalam Redis. Jika data yang di PUSH berhasil, worker akan bekerja dengan menggunakan fungsi *spawn*.

```
masukin.lpushAsync("queue",
hasil.payload.toString()).then(function(data) {
                      res.send('lulus');
                      const worker = spawn('python',
[__dirname+'/worker.py'])
                  })
```

Kode Sumber 4.25 Memasukkan Data Ke Redis Dan Request Post Ke Laravel

```
r = requests.post('http://10.151.36.98/absen/add', data
= {'token': SignEncrypt(id, time.time()+60)},
timeout=3)
```

Kode Sumber 4.24 Request Post Ke Laravel

4.4.4 Pengolahan Data Pada Sistem Informasi

Data yang selama ini dikirim melalui *request post* pada python di klien maupun pop dari server yang berasal dari proses *Message Queue* akan di kirim menuju laravel dan data akan diproses di dalamnya. Pada kasus ini, nilai balik yang diberikan berupa detil informasi bagi mahasiswa seperti jadwal, nama, dan NRP.

Saat di dalam laravel, data yang berasal dari klien langsung atau melalui proses *Message Queue* memilki proses yang hampir sama. Data yang diterima langsung dari server butuh dilakukan deskripsi dan verifikasi menggunakan JWT. Sedangkan untuk data yang didapatkan dari *Message Queue,* data tidak perlu didekripsi dan verifikasi karena sudah dilakukan saat data berada di dalam *Message Queue.*

Dalam proses selanjutnya, data yang disimpan dalam bentuk JSON dengan nama variable "payload" akan diproses. Data payload, kartu, dan jadwal akan digabung untuk menghasilkan data absensi. Kemudian data akan di kirim balik ke mesin kehadiran dalam bentuk JSON seperti pada Kode Sumber 4.26.

```
public function absenWorker(Request $request)
    {
        $payload = $this->signdecryptexp($request-
>token);
        if(!$payload) return '{"response" : "Curang,
silahkan ke kelas"}';

       $create = New Absen;
       $create->id_absen = UUID::generate(4);
       $create->id_kartu = $payload->id;
        $create->nrp = $kartu->nrp;
       $create->id_jadwal = $jadwal->id_jadwal;
       $create->waktu_absen = $waktu-
>toDateTimeString();
       $create->save();


              $result = '{"nama" : "'.$kartu->siswa-
>nama.'","nrp" : "'.$kartu->nrp.'","kelas" : "'.$ambil-
>kelasmatkul->matkul->nama_matkul.'","kode" :
"'.$ambil->kelasmatkul->kode->kode_kelas.'","mulai" :
"'.$jadwal->jam_mulai.'","selesai" : "'.$jadwal-
>jam_selesai.'","response" : "suksesWorker"}';
        return $result;

    }
```

Kode Sumber 4.26 Gambaran Pengecekan Data Pada Laravel

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA

Dalam bab implementasi, sudah dibahas secara detil bagaimana program yang dibuat berjalan. Program yang ada akan berjalan mulai dari data di tap ke mesin kehadiran sampai proses absensi dapat dilakukan. Dalam bab ini akan dibahas beberapa tahap yaitu lingkungan pengujian, skenario yang akan dilakukan, dan hasil percobaan dari skenario yang disiapkan.

5.1 *Lingkungan Pengujian*

Lingkungan untuk pengujian menggunakan dua buah komputer yang berfungsi sebagai *web server* dan penguji. Pada komputer *web server* hanya digunakan untuk menyalakan server. Sedangkan komputer penguji digunakan untuk mengakses aplikasi Tugas Akhir ini.

Proses pengujian dilakukan di Laboratorium Arsitektur dan Jaringan Komputer gedung Teknik Informatika ITS. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Pengujian

Perangkat	**Jenis**	**Spesifikasi**
Perangkat Keras Mesin Kehadiran	Mini Komputer	Raspberry Pi 3 Model B
	Modul NFC	PN-532
	SD Card	SanDisk Ultra 16GB
	Monitor	KeDei 3.5 inch 480x320

Perangkat	Jenis	Spesifikasi
	Adaptor	Part Resmi Raspberry Pi
Perangkat Lunak Mesin Kehadiran	Sistem Operasi	Raspbian
	Perangkat Pengembang	Python Versi 2.7.14
		Socket.IO 2.0
		Node.js v8.9.1
		Express.js 4.16
		Redis 4.0.2
		JSON Web Token
Perangkat Keras Server	Prosesor	Intel(R) Core(TM) I3 2120
	Memori	8 GB
Perangkat Lunak Server	Sistem Operasi	Linux Ubuntu 14.04.3
	Perangkat Pengembang	Python Versi 2.7.14
		Socket.IO 2.0
		Node.js v8.9.1
		Express.js 4.16
		Redis 4.0.2
		JSON Web Token

5.2 *Skenario Uji Coba*

Dalam pembahasan skenario ujicoba, hal yang dilakukan yaitu memastikan semua fungsi dari sistem yang dibuat bekerja sesuai dengan semestinya.

Akan dibagi menjadi 2 tahap besar yaitu tahap uji fungsionalitas dan uji performa. Dalam tahap uji fungsionalitas, fungsi dari mesin akan dilakukan uji coba dan juga uji coba untuk memastikan data telah berpindah dari mesin ke server menggunakan metode *Message Queue.*

Dalam proses uji coba yang dilakukan, jumlah mesin kehadiran yang digunakan adalah 3 buah dan menggunakan server dengan berbagai konfigurasi.

5.2.1 Skenario Uji Fungsionalitas

Dalam skenario pengujian pada fungsionalitas tujuan yang akan dicapai adalah sistem berjalan sesuai dengan perancangan pada pembahasan sebelumnya.

Uji fungsionalitas yang dilakukan di mesin kehadiran terfokus pada mesin kehadiran sebagai klien dapat memberikan respon apabila kartu NFC mendekat ke modul NFC maka ID dari kartu didapatkan. Setelah respon berhasil, maka proses akan memasuki pada tahap sistem yang bergerak untuk saling berkomunikasi antara mesin kehadiran sebagai klien dan server tempat data disimpan.

Berikutnya, untuk pengujian pada aplikasi merupakan pembahasan sistem perangkat lunak secara keseluruhan meliputi proses *request post, Message Queue,* sampai pada menyimpanan data ke *database* melalui laravel sebagai kerangka kerjanya.

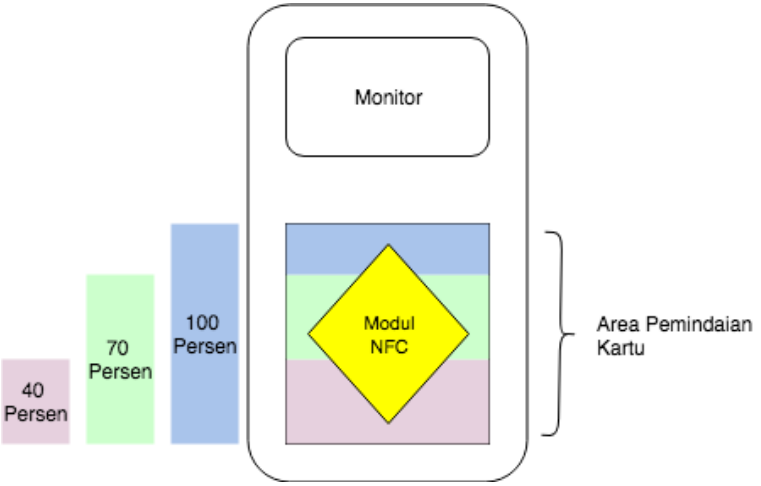
5.2.1.1 Skenario Uji Fungsionalitas Mesin Kehadiran

Sebelum memasuki skenario uji coba fungsionalitas dari mesin kehadiran, harus dipastikan port dan socket untuk monitor,

modul NFC, dan Raspberry Pi. Sebelumnya koneksi dari klien ke server harus dipastikan dapat terhubung.

Dalam penanggulangan saat mesin kehadiran rusak ataupun mati secara tiba-tiba, solusi yang diberikan dengan menambahkan skrip dalam **/home/pi/.config/lxsession/LXDE-pi/autostart** untuk menjalankan beberapa perintah dan menjalan electron sebagai kerangka kerja antarmuka pada mesin kehadiran.

Dalam uji coba fungsi dari mesin kehadiran yang dilakukan adalah memastikan seberapa banyak data yang hilang saat melakukan tap dan disimpan di dalam *queue* mesin kehadiran. Harapan yang diinginkan adalah data tidak ada yang hilang saat dilakukan pemindaian dan penyimpanan dalam *queue*. Jika data sudah aman semua, maka yang diperhitungkan adalah perilaku dari pengguna, berapa persentase dari kartu yang harus mengenai sensor dan seberapa jauh jarak dari mesin ke kartu yang dapat dilakukan untuk memindai.



Gambar 5.1 Skenario luasan pemindaian kartu

5.2.1.2 Skenario Uji Fungsionalitas Aplikasi

Pengujian pada fungsi aplikasi akan dilakukan pada system perangkat lunak yang dibuat, mulai dari penyimpanan *queue*, *request post* yang dilakukan, dan penyimpanan pada *queue* server.

Untuk skenario yang diajukan adalah saat klien tidak dapat berkomunikasi dengan server dan koneksi dapat dilakukan antara klien dan server. Dalam uji coba yang dilakukan akan menggunakan beberapa data yaitu pemindaian kartu dengan jumlah 50, 200, dan 500 pada tiga mesin kehadiran yang sudah sudah siap.

Uji coba memiliki dua kasus yaitu saat mesin tidak dapat terhubung ke server dan mesin dapat terhubung ke server. Dari kasus tersebut akan dilihat seberapa besar data yang dapat disimpan oleh queue. Dari hal tersebut akan didapatkan hasil berupa jumlah data yang ada. Berikutnya pembahasan akan melihat performa dari sistem kehadiran, dilihat dari lokasi sistem informasi sebagai pengolahan data dan server.

5.3 *Skenario Uji Performa Sistem Kehadiran*

Setelah menjelaskan mengenai skenario uji coba mesin kehadiran dan server, maka selanjutnya adalah melakukan uji konsistensi dari sistem meliputi performa dan kehandalan dalam berbagai kasus.

Berikutnya akan membahas bagaimana performa dari sistem kehadiran yang dibuat. Untuk uji coba yang dilakukan dalam percobaan ini yaitu menentukan kemungkinan terbesar dari delay yang terjadi apabila sistem informasi berada pada lingkungan yang berbeda. Sebagai gambaran dapat melihat yang merupakan perbedaan lama respon ke database yang berada di *local* komputer atau di server yang berbeda jaringan.

Untuk jumlah data yang digunakan tiga kali percobaan minimal sebesar 100, 500, dan 800 data.

```
64 bytes from 10.151.36.196: icmp_seq=130 ttl=64 time=199.064 ms
64 bytes from 10.151.36.196: icmp_seq=131 ttl=64 time=153.621 ms
64 bytes from 10.151.36.196: icmp_seq=132 ttl=64 time=279.655 ms
64 bytes from 10.151.36.196: icmp_seq=133 ttl=64 time=203.013 ms
64 bytes from 10.151.36.196: icmp_seq=134 ttl=64 time=17.421 ms
64 bytes from 10.151.36.196: icmp_seq=135 ttl=64 time=140.990 ms
64 bytes from 10.151.36.196: icmp_seq=136 ttl=64 time=157.796 ms
64 bytes from 10.151.36.196: icmp_seq=137 ttl=64 time=281.923 ms
64 bytes from 10.151.36.196: icmp_seq=138 ttl=64 time=203.862 ms
64 bytes from 10.151.36.196: icmp_seq=139 ttl=64 time=77.141 ms
64 bytes from 10.151.36.196: icmp_seq=140 ttl=64 time=137.614 ms
64 bytes from 10.151.36.196: icmp_seq=141 ttl=64 time=448.693 ms
64 bytes from 10.151.36.196: icmp_seq=142 ttl=64 time=327.429 ms
```

Gambar 5.2 Lama Respon Pada Database Beda Jaringan

```
64 bytes from 10.151.36.100: icmp_seq=0 ttl=64 time=0.090 ms
64 bytes from 10.151.36.100: icmp_seq=1 ttl=64 time=0.237 ms
64 bytes from 10.151.36.100: icmp_seq=2 ttl=64 time=0.160 ms
64 bytes from 10.151.36.100: icmp_seq=3 ttl=64 time=0.200 ms
64 bytes from 10.151.36.100: icmp_seq=4 ttl=64 time=0.130 ms
64 bytes from 10.151.36.100: icmp_seq=5 ttl=64 time=0.175 ms
64 bytes from 10.151.36.100: icmp_seq=6 ttl=64 time=0.182 ms
64 bytes from 10.151.36.100: icmp_seq=7 ttl=64 time=0.174 ms
64 bytes from 10.151.36.100: icmp_seq=8 ttl=64 time=0.165 ms
64 bytes from 10.151.36.100: icmp_seq=9 ttl=64 time=0.099 ms
64 bytes from 10.151.36.100: icmp_seq=10 ttl=64 time=0.240 ms
64 bytes from 10.151.36.100: icmp_seq=11 ttl=64 time=0.185 ms
```

Gambar 5.3 Lama Respon Pada Database Lokal

Dalam kasus ini uji coba kasus yang dilakukan meliputi beberapa kasus, diantara lain:

1. Sistem informasi dan *database* berada dalam satu komputer
2. Sistem informasi dan *database* berbeda komputer tapi masih dalam satu jaringan
3. Sistem informasi dan *database* berbeda jaringan

5.4 *Hasil Uji Coba*

Pada sub bab sebelumnya dibahas mengenai skenario pada saja yang akan dikerjakan pada uji coba pada sistem kehadiran ini. Maka pada sub bab hasil uji coba akan dibahas mengenai hasil dari uji coba yang dilakukan sesuai dengan skenario yang diberikan meliputi uji coba pada alat maupun arsitektur yang cocok untuk sistem yang dibuat.

Dalam hal ini akan diberikan beberapa lampiran foto bukti uji coba, nilai, dan hasil uji coba. Pada bagian terakhir juga akan di evaluasi hal-hal apa saja yang menarik untuk dikembangkan pada tahap selanjutnya.

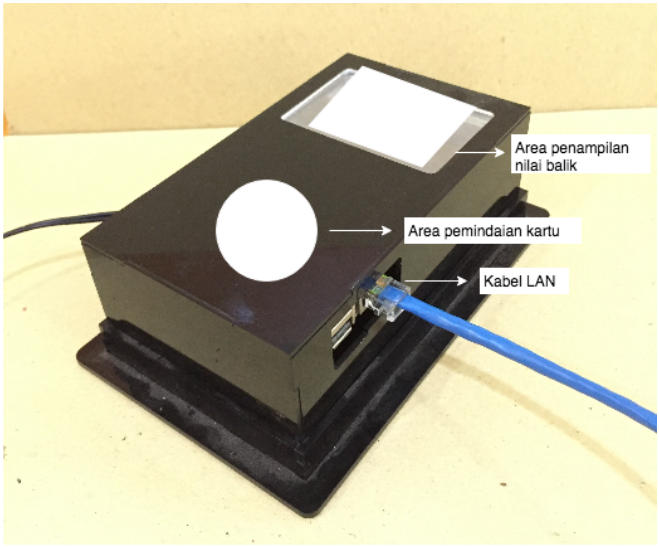
5.4.1 Hasil Uji Fungsionalitas

Saat uji coba dilakukan banyak hal-hal yang tidak terduga terjadi. Oleh sebab itu, dalam skenario disiapkan beberapa kasus yang dapat dicoba dalam hasil percobaan dan dilaporkan hasilnya dengan beberapa bukti foto.

5.4.1.1 Hasil Uji Fungsionalitas Mesin Kehadiran

Pengujian dilakukan beberapa kali dikarenakan banyak perubahan dan modifikasi yang terjadi setelah skenario dilakukan. Hal pertama yang dilakukan adalah dengan menyalakan ketiga model pertam mesin kehadiran, dipastikan semua part, socket, dan kabel sudah terpasang di lokasi yang tepat.

Hal berikutnya adalah menyalakan mesin, Raspberru Pi merupakan suatu mini-komputer yang di desain untuk IoT (Internet of Things) membuatnya langsung melakukan booting saat mni-komputer terhubung dengan *power-socket*.

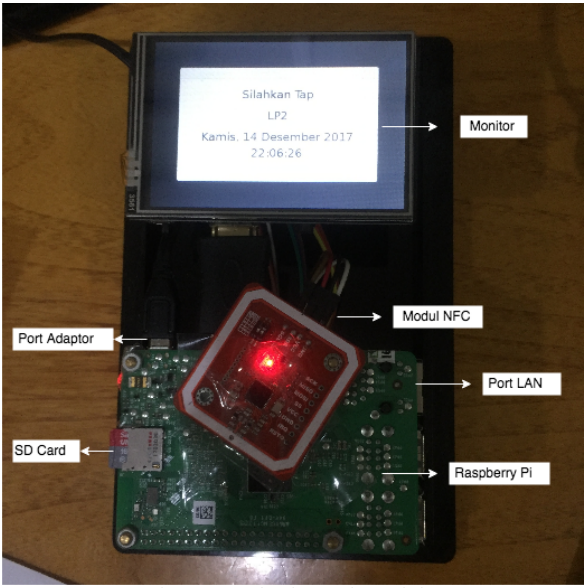


Gambar 5.4 Tampilan Mesin Yang Sudah Siap

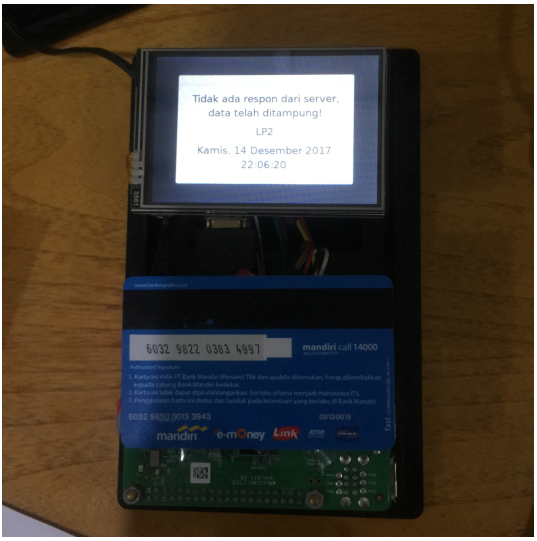
Untuk melakukan uji coba, dipastikan untuk mengubah kode pada file autostart agar aplikasi electron langsung jalan saat mesin telah booting up.

Saat melakukan uji keberhasilan dalam melakukan tap pada mesin, hasilnya adalah 100 persen berhasil. Semua data yang berhasil di deteksi oleh modul NFC merupakan kartu yang memiliki tipe ISO14443a.

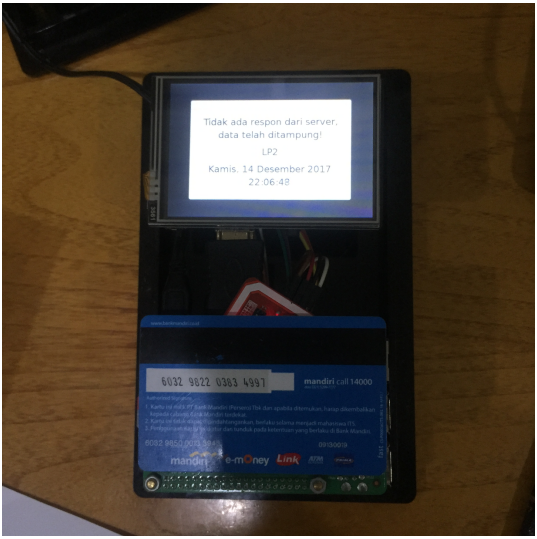
Oleh karena keberhasilannya yang 100 persen, maka jarak dan luasan daerah tap merupakan parameter berikutnya dalam percobaan. Dari percobaan yang dilakukan selama 1 jam, didapatkan hasil data seperti pada Gambar 5.5, Gambar 5.6, Gambar 5.7, Gambar 5.8.



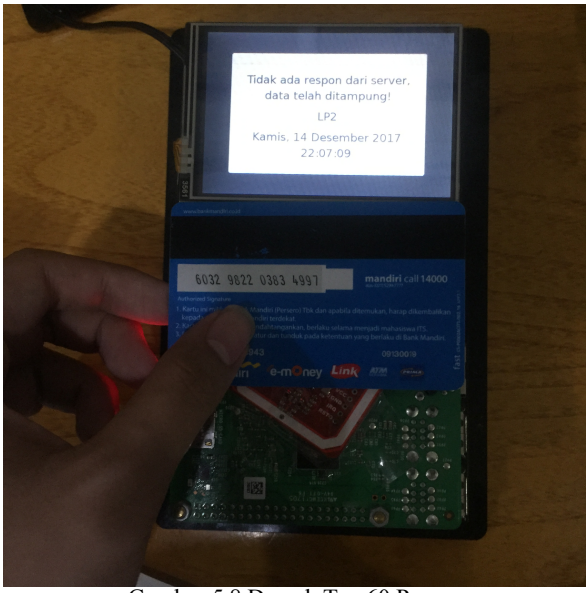
Gambar 5.5 Lokasi Awal Modul NFC



Gambar 5.6 Daerah Tap 100 Persen



Gambar 5.7 Daerah Tap 80 Persen



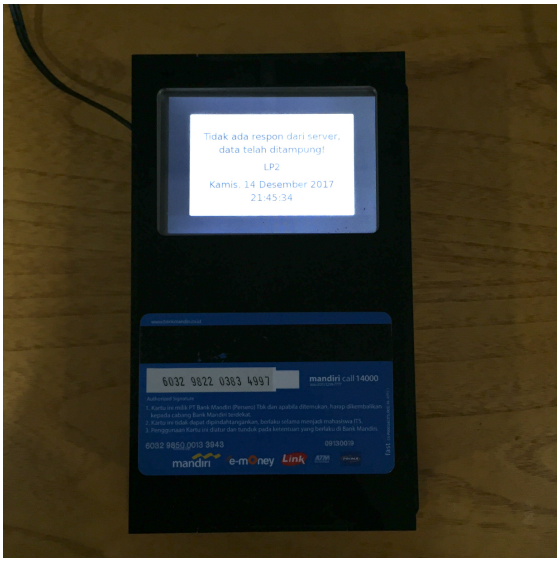
Gambar 5.8 Daerah Tap 60 Persen

Dari percobaan untuk lokasi melakukan tap yang tepat, seperti pada kumpulan foto diatas. Walaupun kartu secara garis besar tidak mengenai modul, trigger yang menggerakkan resistor tepat bisa berjalan. Karena serial number dari Kartu NFC di dalam dengan melakukan trigger elektrik dengan mesin.

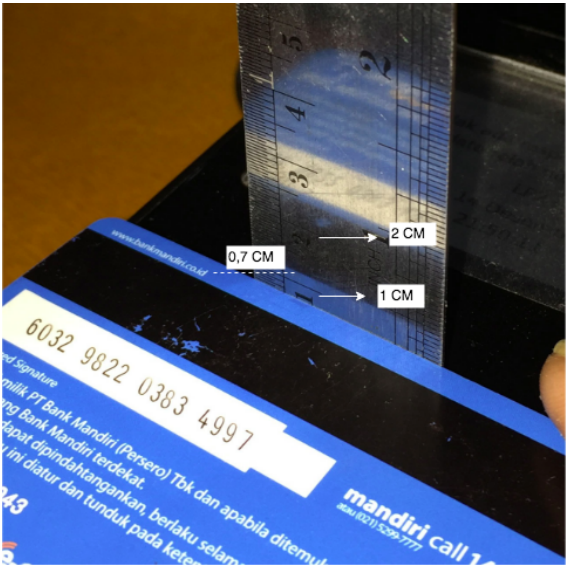
Berdasarkan uji coba yang dilakukan, untuk luasan lokasi tap yaitu seminimal mungkin adalah 60 persen dari luasan modul berada di kartu NFC yang akan dipindai.

Pengujian selanjutnya adalah dengan mengukur jarak maksimal dapat melakukan tap pada mesin selain harus menempelkan kartu ke mesin.

Dari pengujian yang dilakukan selama 1 jam tersebut dihasilkan data seperti pada Gambar 5.9, Gambar 5.10.



Gambar 5.9 Jarak 0 mm Dari Mesin



Gambar 5.10 Jarak 7 mm Dari Mesin

Dari data tersebut dapat dikatakan bahwa mesin akan mendeteksi 100 persen dari kartu NFC selama bertipe ISO14443a dan kartu diletakkan menempel dengan mesin kehadiran.

Tapi, kartu NFC sendiri dapat dipindai tanpa harus menempelkan kartu ke mesin. Untuk kasus dengan modul NFC yang sudah dibungkus oleh arklik, jarak maksimum untuk pendeteksian yaitu 7 mm. Untuk ringkasan dari seluruh percobaan jarak tap ke mesin kehadiran, dapat dilihat pada Tabel 5.2.

Tabel 5.2 Hasil Pengujian Jarak Tap Ke Mesin Kehadiran

Uji Coba	Jarak	Menggunakan *Casing*	Tanpa *Casing*
1	2 mm	Berhasil	Berhasil
2	6 mm	Berhasil	Berhasil
3	7 mm	Kadang Berhasil	Berhasil
4	8 mm	Kadang Berhasil	Berhasil
5	9 mm	Tidak Berhasil	Berhasil
6	12 mm	Tidak Berhasil	Berhasil
7	15 mm	Tidak Berhasil	Kadang Berhasil

5.4.1.2 Hasil Uji Fungsionalitas Aplikasi

Dari hasil uji coba sebelumnya, dapat dipastikan mesin kehadiran berjalan dengan semestinya dengan beberapa peraturan sesuai hasil uji coba yang dilakukan.

Hal selanjutnya yang akan diuji coba adalah melakukan pengujian pada metode *Message Queue*. Pengujian yang dilakukan adalah menampung semua data ke *queue* saat tidak ada koneksi ke server. Dari hal tersebut didapatkan hasil seperti pada Gambar 5.11.

Untuk uji coba kasus pertama data akan selalu tersimpan selama modul NFC dan lingkungan dari mesin kehadiran tidak ada masalah. Oleh karena itu dilakukan juga penampungan dalam jumlah besar untuk menunjukkan berapa banyak data yang dapat masuk ke dalam queue. Hasil dapat dilihat dari uji coba pada Gambar 5.12.

Dari hasil kedua uji coba dapat dikatakan data besar tidak terpengaruh oleh performa dari metode *Message Queue* yang diterapkan.

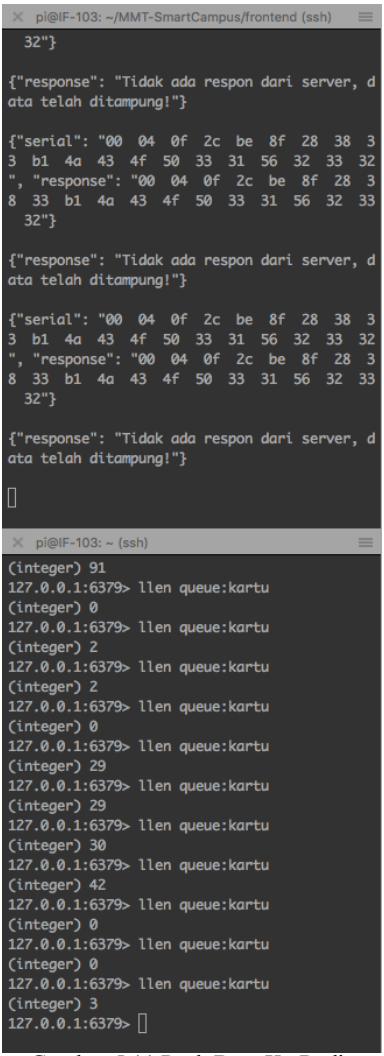
Proses berikutnya yang akan dilakukan adalah pada kasus terbaik, yaitu saat koneksi ada dan data dapat langsung diproses. Hasil dari uji coba dapat dilihat pada Gambar 5.13 & Gambar 5.14.

Dari kasus terbaik seperti pada Gambar 5.13 dan Gambar 5.14, uji coba dilanjutkan pada pengujian saat server tidak dapat terhubung dengan klien-klien kemudian ada saat dimana server kembali menyala.

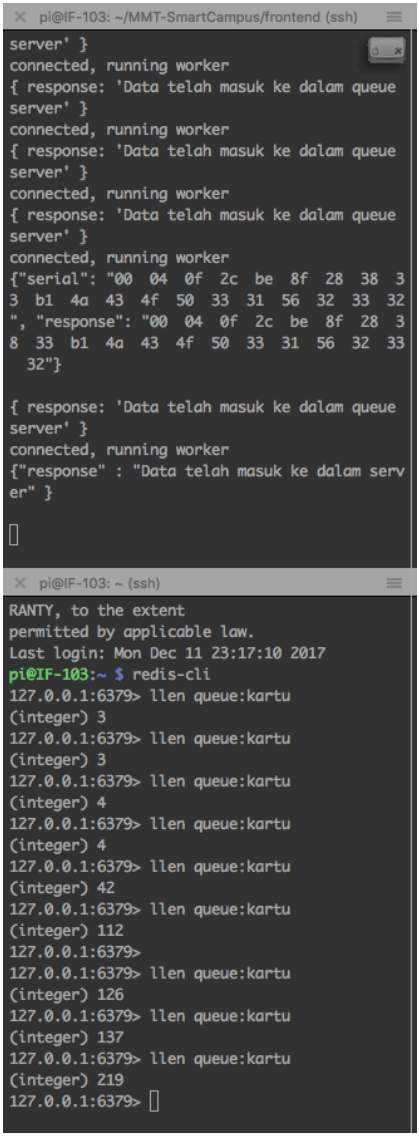
Langkah uji coba yang akan dilakukan adalah server mati dalam beberapa waktu, kemudian server akan dinyalakan sebagai pertanda jikalau koneksi sudah dapat dilakukan. Untuk langkah awal dapat dilihat pada Gambar 5.15.

Pada Gambar 5.16 menunjukkan bahwa saat koneksi terjadi antara klien dan server, semua data yang ada di dalam klien akan berpindah ke server. Saat data berpindah ke server dan gagal

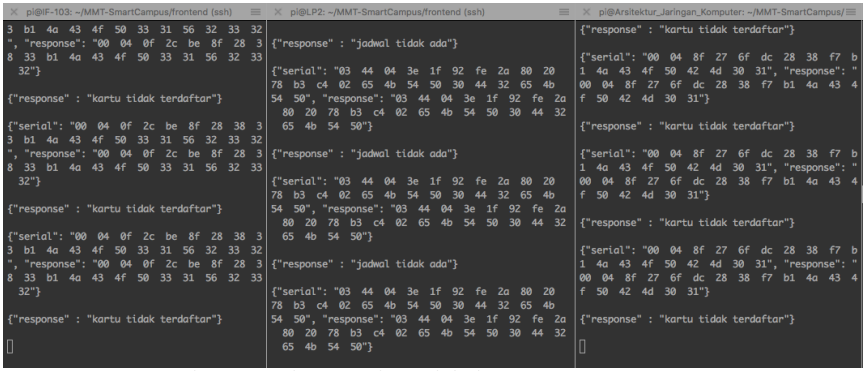
memasukkan kedalam database, data yang sudah berpindah ke server akan masuk ke dalam *queue* di Redis.



Gambar 5.11 Push Data Ke Redis

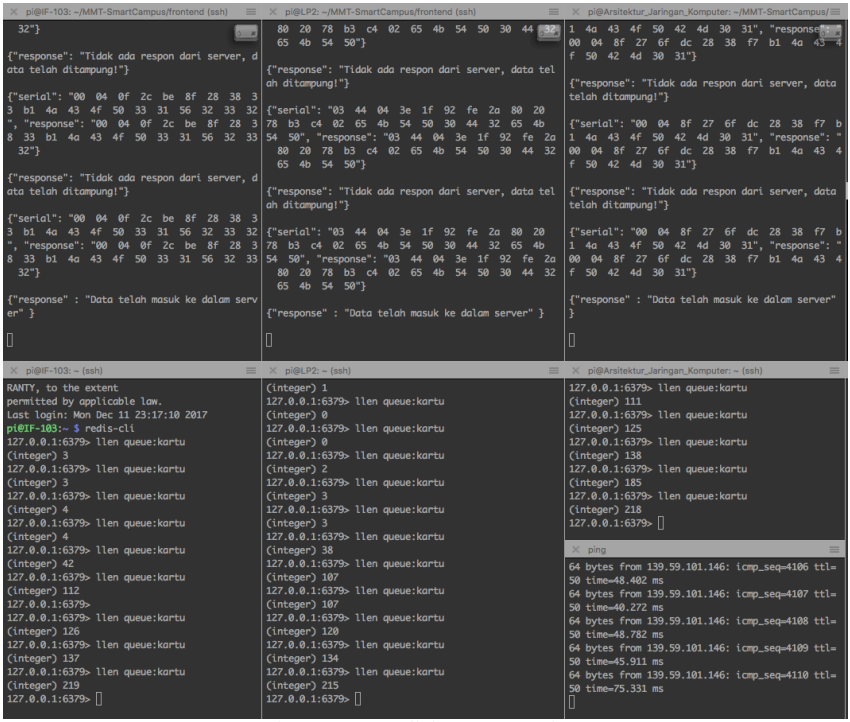


Gambar 5.12 Push Data Dalam Jumlah Besar

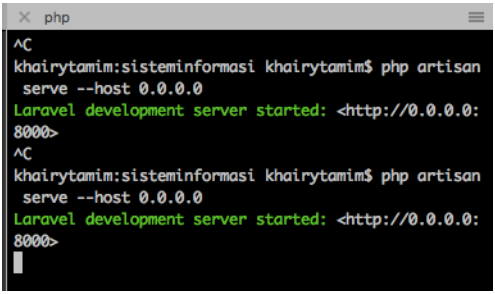


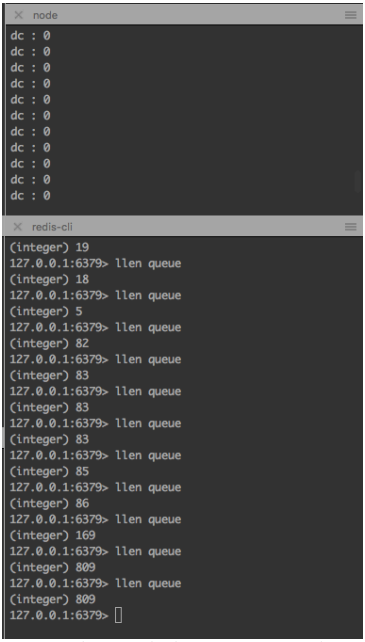
Gambar 5.13 Tigas Mesin Melakukan Request Post

Gambar 5.14 Waktu Respon Ke Database

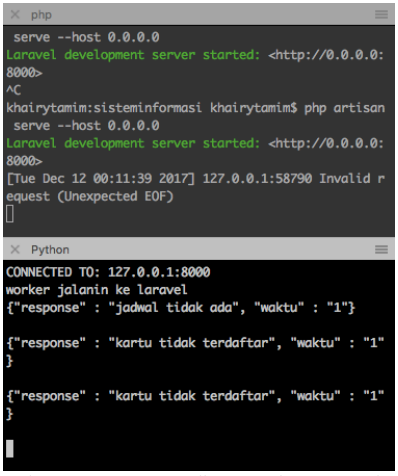


Gambar 5.15 Kondisi Server Mati





Gambar 5.16 Socket Terhubung Dan Database Terputus



Gambar 5.17 Data Redis Masuk Ke Database

Gambar 5.17 menunjukkan kondisi saat koneksi ke database dapat dilakukan. Worker yang berada di dalam server akan bergerak untuk memasukkan data yang ada di dalam Redis untuk di POP dan dimasukkan ke dalam Database.

Untuk ringkasan dari hasil ujicoba fungsionalitas aplikasi, akan ditampilkan dalam bentuk tabel pada Tabel 5.2 yang terdiri dari skenario yang dijelaskan di atas.

Tabel 5.3 Ringkasan Uji Coba yang dilakukan

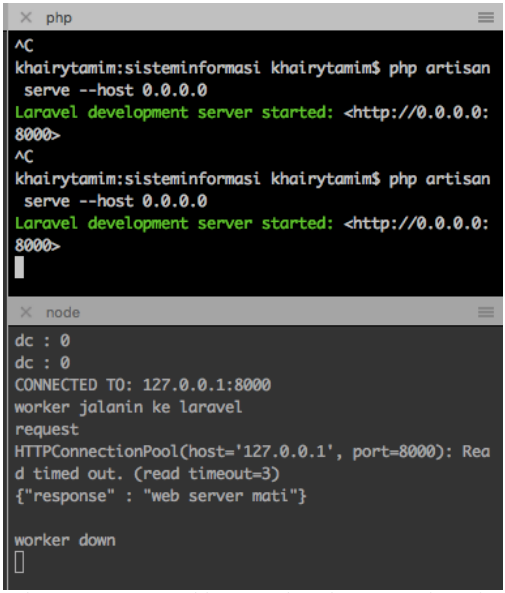
Skenario	Uji Coba	Keterangan
1	Semua klien dan server tidak dapat terhubung	Semua data akan ditampung di dalam queue
2	Semua klien dan server dapat terhubung	Request post akan dimasukkan ke server, jika timeout maka data akan masuk ke dalam queue kembali
3	Semua klien tidak dapat terhubung ke server, kemudian server akan dinyalakan kembali	Semua data yang berada di dalam queue dan juga request post akan

5.4.2 Hasil Uji Performa Mesin Kehadiran

Dari pengujian fungsionalitas dari aplikasi yang dilakukan untuk proses *Message Queue* yang terjadi. Maka terjadi anomali yang berfokus pada arsitektur dari aplikasinya.

Seperti pada Gambar 5.2 dan Gambar 5.3, lokasi dari jaringan berpengaruh dalam kehandalan aplikasi, semakin kecil latensi antara sistem informasi dan database maka semakin handal aplikasi yang ada. Dari data yang didapat, apabila sistem informasi

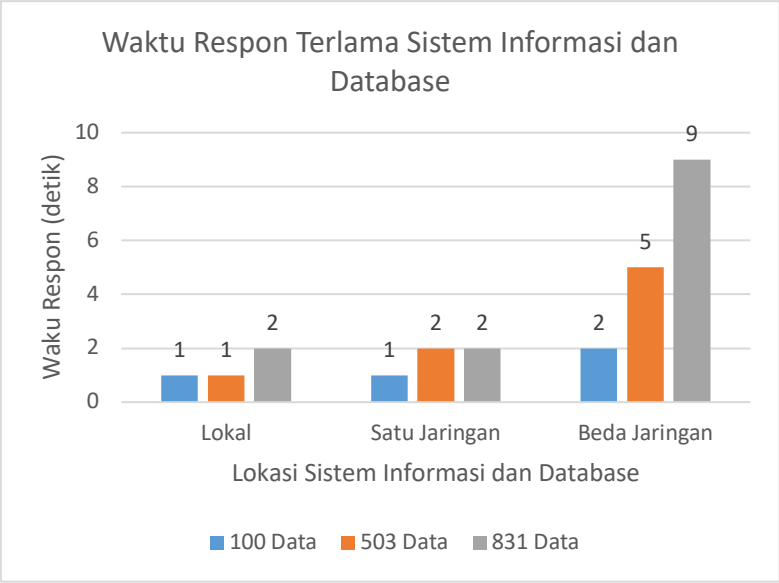
dan database berada pada beda jaringan, maka disitu terdapat kemungkinan untuk terjadi *Request Time Out* seperti pada Gambar 5.18.



Gambar 5.18 Kemungkinan Terburuk Saat Beda Jaringan

5.5 *Evaluasi Hasil Uji Coba*

Setelah semua data dilakukan pengujian, uji fungsionalitas pada mesin kehadiran terlihat lancar tanpa masalah. Masalah yang terjadi hanya dikarenakan perbedaan tipe kartu NFC maupun modul NFC terputus. Anomali yang terjadi adalah jarak maksimal pemindaian dipengaruhi oleh bahan *casing* mesin kehadiran. Untuk saat ini bahan yang digunakan adalah akrilik, kedepannya diharapkan bahan yang lebih tipis untuk meningkatkan jarak pemindaian.



Gambar 5.19 Waktu Respon Sistem Informasi dan Database

Untuk uji fungsionalitas dari aplikasi terlihat semua proses akan terlihat lancar selama sistem informasi dan database berada dalam satu jaringan. Dari percobaan yang dilakukan, terlihat bahwa memiliki sistem informasi yang berbeda jaringan dengan database dapat mengakibatkan *request timeout* dikarenakan untuk sekali request, waktu *timeout*nya adalah 3 detik. Setelah 3 detik maka data akan dikembalikan ke dalam *queue* Redis. Masalah yang terjadi apabila *request post* selalu lebih dari 3 detik, maka data tidak akan pernah masuk ke dalam database.

Oleh karena itu, berdasarkan data, jaringan lokal memiliki waktu respon yang relatif rendah dan terhindar dari *request timeout.*

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

6.1 *Kesimpulan*

Dalam pengerjaan Tugas Akhir ini melalui tahap perancangan aplikasi, implementasi metode, serta uji coba. Sehingga mendapatkan kesimpulan sebagai berikut:

1. Penggunaan *Message Queue,* merupakan solusi untuk menghandalkan sistem. Saat koneksi tidak ada, data akan tetap tersimpan *serial number* dan waktu tap dari kartu NFC.
2. Bahan untuk *casing* mesin kehadiran berpengaruh pada jarak tap kartu ke mesin.
3. Sistem tidak akan mengganggu jaringan di dalamnya, karena data yang digunakan hanya dalam bentuk string.
4. Sistem ini mempermudah proses absensi yang ada dan handal dalam menangani masalah koneksi. Dari uji coba, dihasilkan bahwa proses 100 persen berhasil saat sistem informasi dan database berada di satu lokal komputer.

6.2 *Saran*

Untuk memastikan bahwa semua data terproses secara aman, diharapkan lokasi dari sistem informasi dan database diusahakan berada dalam satu jaringan atau memiliki latensi yang paling rendah guna mengurangi potensi *request timeout.*

Diharapkan untuk bahan penutup dari mesin kehadiran yang digunakan merupakan alat yang tipis untuk meminimalisir kegagalan dalam proses awal pemindaian oleh modul NFC.

DAFTAR PUSTAKA

[1] J. Condliffe, "Harvard Is Privately Tracking Student Attendance With Secret Cameras," 11 10 2014. [Online]. Available: https://gizmodo.com/harvard-is-privately-tracking-student-attendance-with-c-1656730457. [Diakses 10 9 2017].

[2] R. Tiffany, "THE HIDDEN MESSAGE QUEUE ON YOUR WINDOWS PHONE," [Online]. Available: http://robtiffany.com/the-hidden-message-queue-on-your-windows-phone/.

[3] International Business Machines Corporation, "An Introduction to Messaging and Queuing," IBM, England, 1995.

[4] F. Laurita, "Building Scalable, Distributed Job Queues with Redis and Ruby," [Online]. Available: https://www.slideshare.net/francescolaurita/italians-coderjune13redisqueue.

[5] redislabs, "Introduction to Redis," redislabs, 2017. [Online]. Available: https://redis.io/topics/introduction. [Diakses 2017].

[6] C. R. Severance, Python for Everybody, Michigan: Creative Commons Attribution-NonCommercialShareAlike 3.0 Unported, 2009.

[7] Node.js Foundation, "About | Node.js," Joyent, Inc., 2017. [Online]. Available: https://nodejs.org/en/about/. [Diakses 2017].

[8] R. Rai, "Socket.IO Real-time Web Application Development," PACKT, Birmingham, 2013.

[9] T. Otwell, "Introduction," Taylor Otwell, 2017. [Online]. Available: https://laravel.com/docs/4.2/introduction. [Diakses 2017].

[10] CABOT, “Using Electron for Cross Platform Desktop Application Development: An Introduction,” 16 11 2017. [Online]. Available: https://www.cabotsolutions.com/2017/11/using-electron-for-cross-platform-desktop-application-development-an-introduction. [Diakses 12 12 2017].

[11] R. P. FOUNDATION, “RASPBERRY PI 3 MODEL B,” [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[12] Raspberry Pi, “Raspberry Pi 3 Model B,” RS Components.

[13] Adafruit, “PN532 NFC/RFID CONTROLLER BREAKOUT BOARD,” [Online]. Available: https://www.adafruit.com/product/364.

[14] NXP B.V., “PN532/C1,” NXP B.V., 2012.

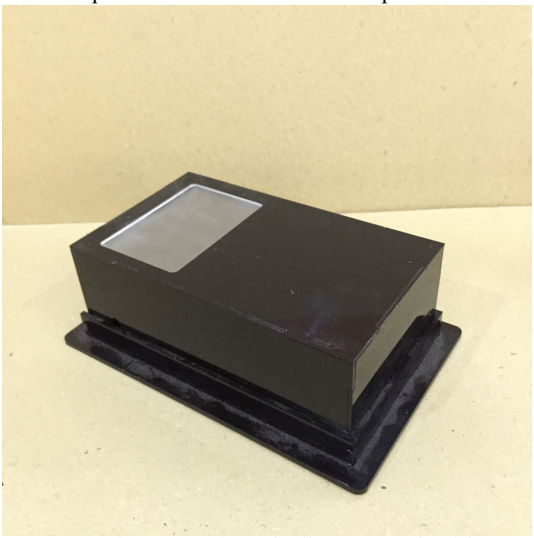
[15] Cermati, “Bayar Tol Wajib Gunakan E-Money Berlaku Oktober 2017,” [Online]. Available: https://www.cermati.com/artikel/bayar-tol-wajib-gunakan-e-money-berlaku-oktober-2017.

[16] Texas Instruments, ISO / NFC Standards and Specifications Overview, Texas: Texas Instruments.

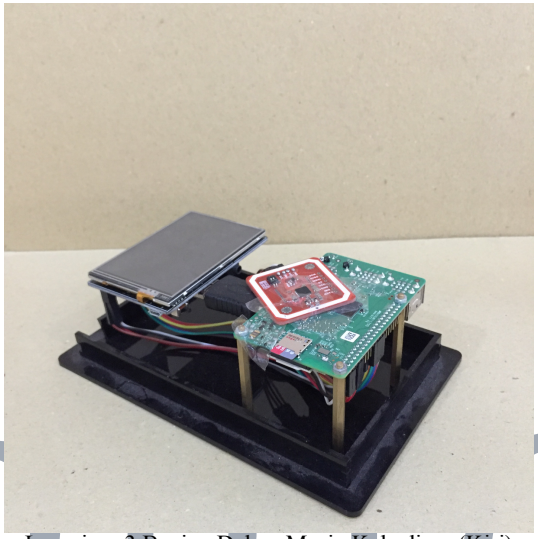
[17] J. Bradley, JSON Web Token (JWT), Microsoft, 2013.

LAMPIRAN

Lampiran 1 Mesin Kehadiran Tampak Kanan



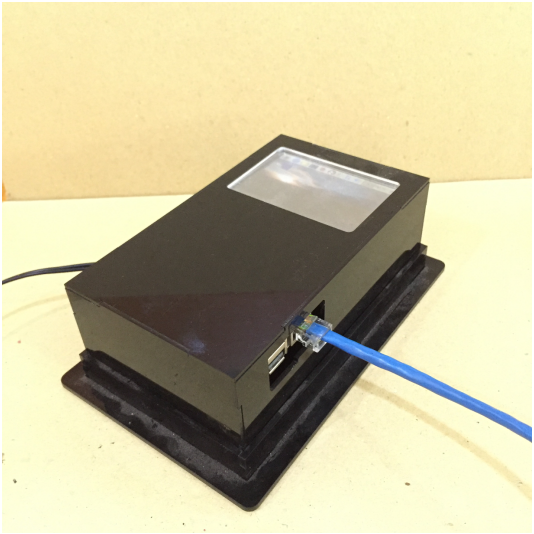
Lampiran 2 Mesin Kehadiran Tampak Kiri



Lampiran 3 Bagian Dalam Mesin Kehadiran (Kiri)

Lampiran 4 Tampak Dalam Mesin Kehadiran (Kanan)

Lampiran 5 Adaptor Mesin Kehadiran



Lampiran 6 Pengaturan Mesin Siap Digunakan

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Fikry Khairytamim, lahir di Jakarta pada tanggal 23 Juli 1996. Penulis menempuh pendidikan mulai dari TK Islam Al-Azhar 9 Kembangan (2000), SD Islam Al-Azhar 8 Kembangan (2002-2008), SMP Labschool Kebayoran (2008-2011), SMA Labschool Kebayoran (2011-2014), dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di ITS. Penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC) dan Badan Eksekutif Mahasiswa Fakultas (BEMF). Diantaranya adalah menjadi staff departemen kewirausahaan HMTC ITS 2015-2016 dan staff departemen hubungan luar BEMF FTIK 2015-2016. Penulis juga aktif dalam kegiatan riset dan magang. Diantaranya penulis pernah tiga kali lolos dalam 3 buat riset yang didanai oleh DIKTI dan dipercaya oleh beberapa dosen untuk menjadi asisten mata kuliah di Teknik Informatika. Penulis juga aktif dalam bidang pengembangan diri, yaitu melakukan magang di perusahaan dalam negeri maupun di luar negeri. Komunikasi dengan penulis dapat melalui telepon: +62 81703434379; *email*: **fikry.labsky08@gmail.com**