05-Perancangan Perangkat Lunak Pendukung Etc (Electronic Toll Collection) Dengan Notifikasi Melalui Social Messenger

by Aghus Sofwan

Submission date: 09-Apr-2019 07:11AM (UTC+0700)
Submission ID: 1108546748
File name: Trans-C05-Perancangan_Perangkat_Lunak_ETC.pdf (642.05K)
Word count: 3747
Character count: 21247

PERANCANGAN PERANGKAT LUNAK PENDUKUNG ETC (*ELECTRONIC TOLL COLLECTION*) DENGAN NOTIFIKASI MELALUI *SOCIAL MESSENGER*

Yusuf Baharuddin[*)], Aghus Sofwan, dan Wahyul Amien Syafei

Departemen Teknik Elektro, Universitas Diponegoro
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)]*E-mail: baharuddinucup@gmail.com*

Abstrak

Kemacetan yang terjadi di gerbang tol disebabkan oleh volume kendaraan yang melebihi kapasitas layanan gerbang tol. Salah satu teknologi untuk meningkatkan laju pelayanan pada gerbang tol adalah ETC (Electronic Toll Collection). ETC yang dikembangkan saat ini dapat memberikan notifikasi melalui SMS, tetapi biaya untuk mengirim SMS masih terbilang mahal. Penelitian ini mengusulkan solusi dengan menggunakan notifikasi melalui social messenger yang menggunakan internet yang lebih murah daripada SMS. Mobil yang terpasang OBU melewati gerbang tol dideteksi oleh OBU reader dan datanya dikirim ke cloud. Cloud mengirimkan data ke pengendara via social messenger. Data ini mejadi bukti pembayaran yang sah. Hasil pengujian sistem, notifikasi pembayaran dikirimkan via social messenger bekerja pada jaringan 2G, 3G, dan 4G dengan rata-rata waktu respon masing-masing jaringan adalah 5,24 detik, 4,99 detik, dan 4,05 detik. Jumlah pengguna kurang dari sama dengan 30, social messenger dapat mengirim notif dengan waktu kurang dari 1,7 detik.

Kata kunci: Social Messenger, perangkat bergerak, sistem bot, ETC

Abstract

The Traffics occur at the toll gate caused by the volume of vehicles that exceed the capacity of the service to toll gate. One of the technologies to increase the rate of service on toll gate is the ETC (Electronic Toll Collection). ETC are being developed at this time can give you a notification via SMS, but the cost to send an SMS is still relatively expensive. This thesis proposes solutions using notification via social messenger who use the internet are cheaper than SMS. The car's built-in OBU passing toll Gates detected by the OBU reader and data sent to the cloud. Cloud Rider to send the data via social messenger. These data become a valid proof of payment. The results of the testing system of payment, a notification is sent via social messenger works on 2 g, 3 g, and 4 g with an average response time of each network is 5.24 seconds, 4.05 seconds, and 4.99 seconds. The number of users is less than equal to 30, social messenger can send notif with less than 1.7 seconds.

Keywords : Social Messenger, mobile device, bot system, ETC

1. Pendahuluan

Dewasa ini jumlah pengguna jalan baik dari motor maupun mobil meningkat pesat. Tak jarang ditemui kemacetan di beberapa daerah. Upaya-upaya sudah dilakukan oleh pemerintah untuk mengatasi kemacetan ini. Salah satunya dengan membangun jalan tol atau jalan bebas hambatan. Hambatan yang dimaksud adalah kemacetan yang terjadi pada jalan-jalan biasa yang dilewati oleh berbagai jenis kendaraan. Diharapkan adanya jalan tol ini, para pengguna jalan tol dapat merasakan nyaman karena tidak adanya kemacetan jika melalui jalan tol dan dapat lebih cepat sampai ketujuan karena tidak terjebak macet.

Di Indonesia, sudah menerapkan gerbang tol elektronik yaitu GTO (Gerbang Tol Otomatis). GTO dalam perkembangan masih menggunakan *smart card* untuk proses identifikasi. Proses ini mengharuskan para pengendara kontak dengan *reader smart card* yaitu menempelkan kartu ID pada *reader smart card*. Dan bukti pembayaran yang masih konvensional yaitu dengan memberikan struk pembayaran berbentuk kertas kecil sehingga pengendara harus mengambil struk tersebut. Proses ini menjadi penyebab aliran di gerbang tol yang menjadi lama. Ditambah lagi, kenyataannya struk tersebut akan dibuang dan tidak terpakai lagi. Struk ini akan mengakibatkan kotor pada jalan tol ataupun pada gerbang tol [1].

Dengan memanfaatkan teknologi non kontak yang biasa disebut dengan *Radio Frequency Indentification* (RFID), diharapkan dapat meningkatkan pelayanan di gerbang tol [3][4][5]. RFID dapat mendeteksi sebuah kendaraan dari jarak jauh dan tanpa adanya kontak langsung karena RFID menggunakan detektor jarak jauh. Sistem identifikasi kendaraan sudah terjadi sebelum kendaraan melewati gerbang tol, sehingga para pengendara tidak akan berhenti untuk melakukan pembayaran ataupun mengambil struk pembayaran. Pengendara yang telah melewati gerbang tol dapat mengurangi waktu terbuang yang ada di gerbang tol.

ETC (*Electronic Toll Collection*) yang dikembangkan saat ini juga dapat memberikan notifikasi melalui SMS [2], sehingga dapat menghilangkan penggunaan kertas yang akan berdampak pada berkurangnya pengotoran lingkungan. Namun, biaya untuk mengirim SMS masih terbilang mahal. Solusi untuk ini adalah dengan menggunakan notifikasi melalui *social messenger* yang menggunakan paket data internet yang lebih murah daripada SMS. Notifikasi melalui *social messenger* ini menjadi bukti pembayaran sah karena terjadi transaksi antara pengguna dan pihak PT. Jasa Marga. Dari latar belakang ini, maka dibuat penelitian tentang perancangan Gerbang Tol Pintar berbasis RFID dengan notifikasi pembayaran via *social messenger*, yang akan berdampak pada pengurangan waktu yang dibutuhkan untuk transaksi masuk jalan tol dan pengurangan biaya operasional dengan menggunakan *social messenger* untuk mengirimkan bukti pembayaran.

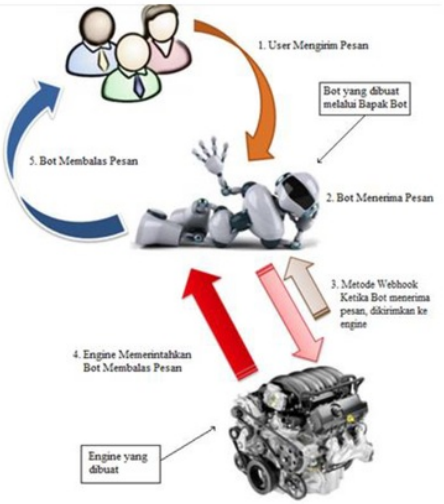
Penelitian ini bertujuan untuk merancang perangkat lunak Pendukung *ETC (Electronic Toll Collection)* dengan notifikasi melalui *social messenger* yaitu *Telegram* dengan menggunakan *bot* yang dimiliki *Telegram*. Aplikasi ini dapat memberikan notifikasi pembayaran tol setelah kendaraan masuk dalam gerbang tol yang menggunakan sistem ETC.

2. Metode

2.1. Webhook

Dengan Webhook, aplikasi dapat menerima pemberitahuan ketika terdapat perubahan terhadap serangkaian topik yang dipilih beserta kolomnya. Misalnya, mengatur webhook untuk topik User dan berlangganan ke kolom email, serta akan diberi tahu ketika pengguna memperbarui alamat emailnya. Hal ini mencegah agar tidak bergantung pada permintaan API berkelanjutan atau bahkan periodik untuk memeriksa pembaruan yang mungkin atau tidak mungkin terjadi. Hal ini juga membantu untuk menghindari pembatasan laju.

Pemberitahuan pembaruan webhook dikirimkan sebagai permintaan POST ke URL callback yang diberikan. Pemberitahuan dapat berupa hal yang sederhana, hanya mengindikasikan bahwa sebuah kolom sudah diperbarui, atau dapat menyertakan nilai yang baru diperbarui. Webhook merupakan salah satu metode mengambil pesan pada Telegram. Gambar 1 menggambarkan cara kerja bot menggunakan metode webhook.



Gambar 1. Cara Kerja Bot

2.2. Model-View-Controller (MVC)

MVC (*Model View Controller*) adalah konsep dasar yang harus diketahui sebelum mengenal Codelgniter. MVC adalah sebuah *pattern*/teknik pemrograman yang memisahkan bisnis *logic* (alur pikir), data *logic* (penyimpanan data) dan *presentation logic* (tampilan aplikasi) atau secara sederhana adalah memisahkan antara desain, data dan proses [6]. Adapun komponen-komponen MVC antara lain :

2.2.1. Model

Model berhubungan dengan data dan interaksi ke *database* atau *web service*. Model juga mempresentasikan struktur data dari aplikasi yang bisa berupa *database* maupun data lain, misalnya dalam bentuk file teks, file XML, maupun *webservice*. Model berisi *class* dan fungsi untuk mengambil, melakukan *update* dan menghapus data *website*. Sebuah aplikasi *web* biasanya menggunakan *database* dalam menyimpan data, maka pada bagian model biasanya berhubungan dengan perintah-perintah *query* SQL.

2.2.2. View

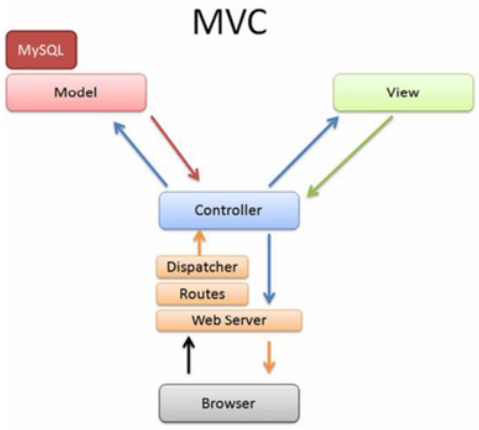
View berhubungan dengan segala sesuatu yang ditampilkan ke *end-user*. Bisa berupa halaman *web*, RSS, JavaScript dan lain-lain. *Programmer* menghindari adanya logika atau pemrosesan data di *view*. Di dalam *view* hanya berisi variabel-variabel yang berisi data yang ditampilkan. *View* dapat dikatakan sebagai halaman *website* yang dibuat

dengan menggunakan HTML dan bantuan CSS atau JavaScript. Di dalam *view* tidak diperbolehkan ada kode untuk melakukan koneksi ke *database*. *View* hanya dikhususkan untuk menampilkan data-data dari model atau *controller*.

2.2.3. Controller

Controller bertindak sebagai penghubung data dan *view*. Di dalam *controller* inilah terdapat *class-class* dan fungsi-fungsi yang memproses permintaan dari *view* ke dalam struktur data di dalam model. *Controller* juga tidak berisi kode untuk mengakses *database* karena tugas mengakses data telah diserahkan kepada model. Tugas *controller* adalah menyediakan berbagai variabel yang akan ditampilkan di *view*, memanggil model untuk melakukan akses ke *database*, menyediakan penanganan kesalahan/*error*, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap input.

Pada eksekusi PHP biasanya *programmer* akan me-*load* semua *library* dan fungsi yang dibutuhkan kemudian digabungkan ke dalam HTML untuk di eksekusi oleh PHP. Untuk kasus sederhana cara tersebut masih baik-baik saja, tetapi ketika aplikasi tersebut menjadi komplek/rumit maka kita akan sulit memeliharanya jika tidak didukung oleh arsitektur *software* yang bagus. Hal tersebut bisa terjadi disebabkan oleh kode yang sama namun dibuat berulang-ulang, kode tidak konsisten dan lain-lain. Sedangkan CodeIgniter menggunakan pola MVC, dipetakan menjadi model, *view* dan *controller*. Pola MVC dapat dilihat pada Gambar 2.

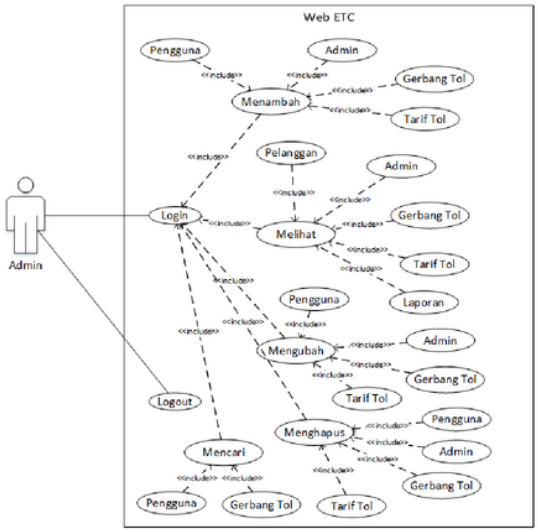


Gambar 2. *Flow* CodeIgniter

Jika dipetakan, alur kerja CodeIgniter digambarkan pada Gambar 2 *browser* berinteraksi melalui *controller*. *Controller*-lah yang menerima dan membalas semua *request* dari *browser*. Untuk data maka *controller* meminta ke model dan untuk UI/*template* meminta ke *view*. Jadi "otak" dari aplikasi ada di *controller*, "muka" aplikasi ada di *view* dan "data" aplikasi ada di model. Ketika *browser* meminta sebuah halaman *web* maka *router* mencarikan *controller* mana yang harus menangani *request* tersebut. *Controller* menggunakan model untuk mengakses data dan *view* untuk menampilkan data tersebut.

2.3. Diagram *Use Case*

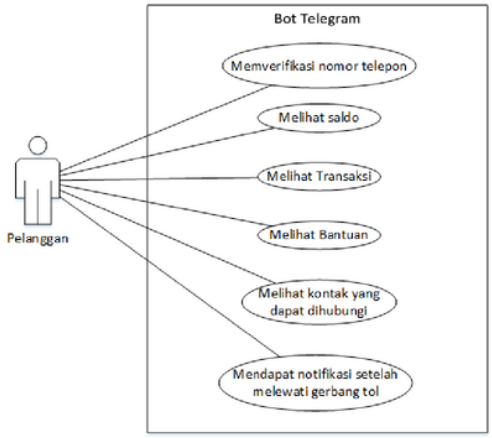
Diagram *use case* menggambarkan fungsi-fungsi yang ada pada sistem [7]. Diagram ini lebih berfokus pada fitur-fitur sistem dari sudut pandang pihak luar, yang dalam hal ini adalah *user* aplikasi baik admin yang mengelola *web* dan pelanggan yang mengggunakan Telegram. Diagram *use case* untuk perancangan sistem yang dibuat terdiri dari diagram *use case* untuk *web* dan diagram *use case* untuk pelanggan. Diagramnya dapat dilihat pada Gambar 3 dan Gambar 4.



Gambar 3. Diagram *Use Case Web*

Gambar 3 menggambarkan fitur-fitur yang dapat diakses oleh admin. Admin dapat mengakses semua menu yang teradapat dalam *web* ETC ini. Pertama yang harus dilakukan adalah login, setelah itu admin dapat menambah, melihat, mengubah, menghapus dan mencari. Jika admin ingin keluar dari *web*, menekan tombol logout.

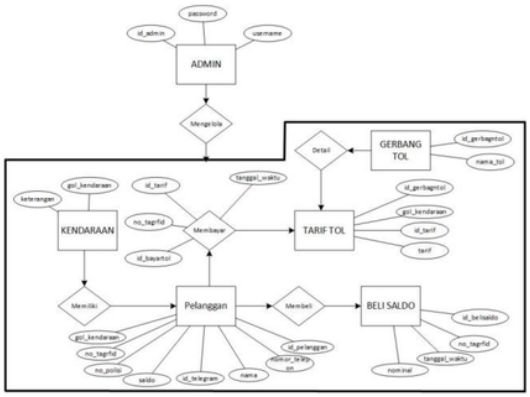
Gambar 4 menggambarkan fitur-fitur yang dapat diakses oleh pelanggan ketika dia menggunakan *bot* telegram. Pelanggan dapat mengakses semua fitur yang teradapat dalam *bot* telegram ini. Agar dapat menggunakan semuanya pelanggan harus menambah *bot* terlebih dahulu. Selanjutnya, adalah memverifikasi nomor telepon. Jika tidak demikian, maka pelanggan tidak dapat mengetahui saldo maupun transaksi saldo karena pelanggan belum terkoneksi dengan *database*.



Gambar 4. Diagram *Use Case* Pelanggan

2.4. Relasi *Database*

Rrelasi *database* menggambarkan hubungan antar tabel pada *database*. Bertujuan untuk memberikan gambaran *database* pada sistem yang dibuat. Diagram relasional *database* ini dapat dilihat pada Gambar 3.



Gambar 5. Diagram Relasional *Database*

Gambar 5 menggambarkan hubungan antar tabel pada sistem yang dibuat. Dari gambar 3 entitas atau tabel pada database meliputi admin, kendaraan, pelanggan, tarif tol, gerbang tol dan beli saldo. Relasi membayar menjadi suatu tabel tersendiri karena memiliki atribut sendiri. Atribut pada tabel kendaraan yaitu gol_kendaraan dan keterangan. Atribut pada tabel pelanggan yaitu no_*tag*rfid, nama, nomor_telepon, id_telegram, no_polisi, gol_kendaraan, dan saldo. Atribut pada tabel beli saldo yaitu id_belisaldo, no_*tag*rfid, tanggal_waktu, dan nominal. Atribut pada tabel tarif tol yaitu id_tarif, id_gerbangtol, gol_kendaraan, dan tarif. Atribut pada tabel gerbang tol yaitu id_gerbangtol dan nama_tol. Atribut pada tabel membayar yang dimaksud adalah membayar tol yaitu id_bayartol, no_*tag*rfid, tanggal_waktu, dan id_tarif.

3. Hasil & Analisa

3.1. Pengujian Fungsi Sistem Berdasarkan Berbagai *Browser Desktop*

Pengujian fungsi sistem dilakukan dengan membuka halaman yang memuat fungsi dan melakukan simulasi menjalankan sistem. Untuk pengujian, telah dipilih empat *browser* sebagai pembanding kerja fungsi sistem. Hasil pengujian fungsi sistem berdasarkan berbagai *browser desktop* dimuat pada Tabel 1.

Tabel 1. Hasil Pengujian Berbagai *Browser Desktop*

Fungsi Sistem	*Browser*			
	Google Chrome 61	Ms Edge 40	Mozilla Firefox 56	UC Browser 7
Login	berhasil	berhasil	berhasil	berhasil
Melihat halaman beranda	berhasil	berhasil	berhasil	berhasil
Melihat data pengguna	berhasil	berhasil	berhasil	berhasil
Melihat data gerbang tol	berhasil	berhasil	berhasil	berhasil
Melihat data tarif tol	berhasil	berhasil	berhasil	berhasil
Melihat data golongan kendaraan	berhasil	berhasil	berhasil	berhasil
Melihat data admin	berhasil	berhasil	berhasil	berhasil
Melihat laporan pembayaran tol	berhasil	berhasil	berhasil	berhasil
Melihat laporan pembelian saldo	berhasil	berhasil	berhasil	berhasil
Menambahkan pengguna	berhasil	berhasil	berhasil	berhasil
Menambahkan gerbang tol	berhasil	berhasil	berhasil	berhasil
Menambahkan tarif tol	berhasil	berhasil	berhasil	berhasil
Menambahkan admin	berhasil	berhasil	berhasil	berhasil
Mengubah pengguna	berhasil	berhasil	berhasil	berhasil
Mengubah gerbang tol	berhasil	berhasil	berhasil	berhasil
Mengubah tarif tol	berhasil	berhasil	berhasil	berhasil
Mengubah admin	berhasil	berhasil	berhasil	berhasil
Menghapus pengguna	berhasil	berhasil	berhasil	berhasil
Menghapus gerbang tol	berhasil	berhasil	berhasil	berhasil
Menghapus tarif tol	berhasil	berhasil	berhasil	berhasil
Menghapus admin	berhasil	berhasil	berhasil	berhasil
Mencari data pengguna	berhasil	berhasil	berhasil	berhasil
Mencari data gerbang tol	berhasil	berhasil	berhasil	berhasil
Logout	berhasil	berhasil	berhasil	berhasil

Seluruh *browser* bekerja dengan berhasil. Dapat ditarik hasil akhir bahwa sistem bekerja dengan baik dimana tingkat keberhasilannya mencapai 100%. *Browser* yang dipilih untuk pengujian adalah yang mendukung Css dan Javascript. Maka dari itu, dilakukan pengujian untuk mempertimbangkan kualitas *browser* yang digunakan. Pengujian kualitas *browser* dilakukan melalui dua alamat *web*. Pertama untuk kualitas halaman, pada html5test.com dengan nilai maksimal 555. Kedua untuk kualitas *style* halaman, pada css3test.com engan nilai maksimal 1802. Tabel 2 menunjukkan pengujian kehandalan *browser* melalui dua alamat *web* tersebut.

Tabel 2. Hasil Pengujian Kehandalan *Browser*

Web Pengujian	Google Chrome 61	Ms Edge 40	Mozilla Firefox 56	UC Browser 7
html5test.com (maksimum = 555)	518	468	478	492
css3test.com (maksimum = 1802)	1137	739	1229	918

Hasil pengujian pemuatan halaman, melalui alamat html5test.com, menunjukkan *browser* dengan kualitas terendah adalah Ms Edge. *Browser* tersebut dapat memuat semua fungsi halaman sistem secara sukses 100%. Dapat disimpulkan bahwa meskipun dimuat pada *browser* kualitas rendah, sistem dapat bekerja dengan baik.

Hasil pengujian pemuatan *style* halaman, melalui css3test.com, menunjukkan bahwa *browser* Ms Edge memiliki kualitas terendah di antara *browser* lain. Merujuk dari Tabel 1, menyatakan semua fungsi sistem yang dimuat *browser* Ms Edge menunjukkan hasil sukses. Maka, dapat ditarik kesimpulan bahwa sistem dapat bekerja dengan baik, meskipun kualitas *style browser* rendah.

3.2. Pengujian Waktu Respon Bot dengan Sistem Operasi Berbeda Jaringan Sama

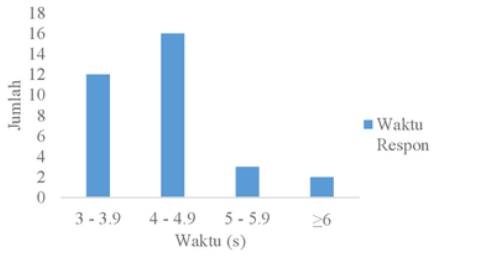
Pengujian waktu respon *bot* dengan sistem operasi berbeda jaringan sama dilakukan untuk mengetahui keandalan sistem operasi yang digunakan. Jaringan yang digunakan adalah 3G atau HSPA. Pengujian ini terhadap 3 perangkat yang berbeda, dengan menampilkan waktu antara pengguna mengirim pesan dengan bot membalasnya. Tabel 3 berikut adalah hasil pengujian respon bot dengan os berbeda jaringan sama.

Tabel 3 menunjukkan respon waktu *bot* dengan OS berbeda jaringan sama. Terdapat 11 fungsi yang masing-masing diuji waktu responnya. Dari Tabel 3 diatas dibuatlah grafik seperti yang ditunjukkan Gambar 6.

Dari Gambar 6 grafik menunjukkan waktu respon *bot*. Rata-rata waktu respon *bot* dengan OS sama jaringan berbeda ditunjukkan pada kisaran waktu 4 hingga 4,9 detik. Adapun rata-rata waktu respon masing-masing *smartphone* adalah Oppo A37 4,05 detik, Xiaomi Redmi 3x 4,53 detik, dan Iphone 6s 4,34 detik. Dari data rata-rata waktu respon *bot* dapat disimpulkan bahwa OS *smartphone* sama tidak mempengaruhi respon *bot* dan *bot* tergolong responsif sehingga pengguna tidak perlu menunggu lama balasan dari *bot*.

Tabel 3. Hasil Pengujian Waktu Respon *Bot* dengan OS Berbeda Jaringan Sama

Uji coba	Bentuk Pengujian	Hasil Pengujian		
		Oppo A37	Xiaomi Redmi 3x	Iphone 6s
1	Menambahkan bot dan memulai percakapan dengan menekan /start	4,9 s	6,5 s	6,3 s
2	Menampilkan *keyboard* untuk memverifikasi nomor telepon	4,2 s	4,1 s	4,8 s
3	Mengirim nomor telepon untuk mengambil *chat id.*	5,4 s	5 s	5,5 s
4	Mengecek saldo yang ada	3,8 s	4,4 s	4,8 s
5	Menampilkan *keyboard* cek transaksi	4,3 s	4,8 s	3,9 s
6	Melihat laporan pembayaran tol	3,7 s	4,5 s	4,5 s
7	Melihat laporan pembelian saldo	3,5 s	4,5 s	5 s
8	Menekan tombol "Hubungi Kami"	3,1 s	3,3 s	4,1 s
9	Menekan tombol "Bantuan"	3,5 s	4,4 s	4,7
10	Menekan tombol "Kembali"	4,3 s	4,3 s	4,3 s
11	Mobil melewati gerbang tol, pengguna mendapatkan notifikasi	3,8 s	4 s	4,5 s



Gambar 6. Grafik Waktu Respon Bot dengan OS Berbeda Jaringan Sama

3.3. Pengujian Waktu Respon Bot dengan Sistem Operasi Sama Jaringan Berbeda

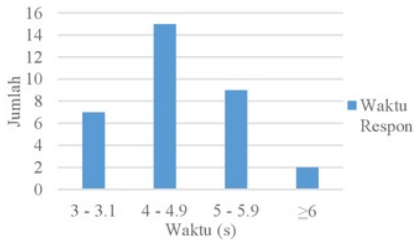
Pengujian waktu respon *bot* dengan sistem operasi sama jaringan berbeda dilakukan untuk mengetahui performa jaringan yang digunakan. Sistem operasi Android yang digunakan dalam pengujian ini adalah Lollipop pada *smartphone* Oppo A37. Jaringan yang digunakan dalam pengujian ini jaringan 2G, 3G dan 4G. Sebelum melakukan pengujian dilukakan *test ping* terlebih dahulu menggunakan aplikasi Terminal Emulator yang di-*install* di *smartphone*. Server yang dituju adalah server google yaitu 8.8.8.8 dan mengambil 4 paket yang dikirim sebagai hasil . Hasil rata-rata waktu dari *test ping* pada jaringan 2G 212,5 ms, 3G 183,5 ms dan 4G 71,63 ms. Pengujian ini dilakukan terhadap 3 jaringan yang berbeda, jaringan 2G, 3G dan 4G, kemudian dengan menampilkan waktu antara pengguna mengirim pesan dengan bot membalasnya. Tabel

4 berikut adalah hasil pengujian respon *bot* dengan OS sama jaringan berbeda.

Tabel 4. Hasil Pengujian Waktu Respon *Bot* dengan OS Sama Jaringan Berbeda

Uji coba	Bentuk Pengujian	Hasil Pengujian		
		Jaringan 2G	Jaringan 3G	Jaringan 4G
1	Menambahkan bot dan memulai percakapan dengan menekan /start	6,9 s	6,9 s	4,9 s
2	Menampilkan *keyboard* untuk memverifikasi nomor telepon	5,7 s	5,7 s	4,2 s
3	Mengirim nomor telepon untuk mengambil *chat id.*	5,9 s	5,1 s	5,4 s
4	Mengecek saldo yang ada	4,6 s	4,5 s	3,8 s
5	Menampilkan *keyboard* cek transaksi	4,8 s	4,8 s	4,3 s
6	Melihat laporan pembayaran tol	5,1 s	4,6 s	3,7 s
7	Melihat laporan pembelian saldo	5,2 s	4,7 s	3,5 s
8	Menekan tombol "Hubungi Kami"	4,7 s	3,7 s	3,1 s
9	Menekan tombol "Bantuan"	4,8 s	4,3 s	3,5 s
10	Menekan tombol "Kembali"	5,1 s	5,5 s	4,3 s
11	Mobil melewati gerbang tol, pengguna mendapatkan notifikasi	4,8 s	4,5 s	3,8 s

Tabel 4 menunjukkan respon waktu *bot* dengan OS sama jaringan berbeda. Terdapat 11 fungsi yang masing-masing diuji waktu responnya. Dari Tabel 4 di atas dibuatlah grafik seperti yang ditunjukkan Gambar 7.

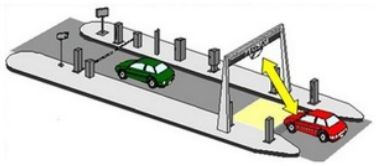


Gambar 7. Grafik Waktu Respon *Bot* dengan OS Sama Jaringan Berbeda

Dari Gambar 7 grafik menunjukkan waktu respon *bot*. Rata-rata waktu respon *bot* dengan OS sama jaringan berbeda ditunjukkan pada kisaran waktu 4 hingga 4,9 detik. Adapun rata-rata waktu respon masing-masing jaringan adalah jaringan 2G 5,24 s, jaringan 3G 4,99 s, dan jaringan 4G 4,05 s. Dari data rata-rata tersebut dapat disimpulkan bahwa waktu tercepat bot merespon ketika *smartphone* menggunakan jaringan 4G dan waktu terlambat ketika menggunakan jaringan 2G. Tetapi, secara umum *bot* tergolong responsif karena perbedaan waktu antar jaringan tidak begitu besar.

3.4. Pengujian Notifikasi Pembayaran Tol

Pengujian dilakukan dengan mendekatkan *tag* RFID dengan *reader* RFID. *Tag* yang digunakan menggunakan tiga *tag* yang sudah berisi id yang tersimpan pada *database*. Jarak antara *tag* RFID dan *reader* RFID sebesar kurang lebih satu meter. Data *tag* akan terkirim ke *database* dan *web* pada halaman laporan pembayaran tol. Pelanggan lalu akan menerima sebuah notifikasi karena *tag* RFID sudah terbaca oleh *reader* RFID. Ilustrasi saat mobil melewati gerbang tol dan *reader* RFID membaca *tag* RFID dapat dilihat pada Gambar 8. Gambar 9 merupakan data kode *tag* yang terdeteksi yang nantinya data-data ini akan dikirimkan ke pengguna. Gambar 10 merupakan pesan yang berisi data saat pengguna melewati gerbang tol dan pesan ini sebagai bukti pembayaran yang sah. Hasil pengujian notifikasi pembayaran tol dapat dilihat pada Tabel 5.



Gambar 8. Ilustrasi Mobil Melewati Gerbang Tol dengan Sistem ETC

no_tagrfid	nama	nama_tol	tarif	waktu_masuk
e2960207	Muhammad Arief Fatkhurrahman	Tembalang	4000	2017-09-28 20:39:36
86ffecd5	Yusuf Baharuddin	Tembalang	6000	2017-09-28 20:38:45

Gambar 9. Data Kode *Tag* yang Terdeteksi

Gambar 10. Pesan ketika Pengendara Melewati Gerbang Tol

Tabel 5. Hasil Pengujian Notifikasi *Bot* Telegram

No	Nama Pelanggan	Kode *Tag*	Kode Terbaca	Notifikasi
1	Yusuf Baharuddin	86ffecd5	86ffecd5	Diterima
2	M arief F	e2960207	e2960207	Diterima
3	Fuad Ashabus	e2f36e85	e2f36e85	Diterima
4	Arisla Choiruddin	5e29273b	5e29273b	Diterima
5	Rafi dhega	864ab2d5	864ab2d5	Diterima

Tabel 5 menunjukkan bahwa data identifikasi kendaraan dan notifikasi pembayaran telah berjalan dengan baik. Tabel ini menunjukkan ID tag dan ID terbaca dengan

benar dan sesuai dengan database. Notifikasi pembayaran juga telah diterima oleh pelanggan yang melewati reader RFID.

4. Kesimpulan

Setelah dilakukan pengujian terhadap sistem secara keseluruhan diketahui bahwa sistem sudah dapat bekerja dengan baik, terutama terhadap fungsi-fungsi baik di *web* maupun *bot* dan notifikasi pembayaran. Dapat dilihat pada Tabel 1, Tabel 2, Tabel 3, Tabel 4, dan Tabel 5. Sistem operasi pada *smartphone* tidak mempengaruhi pada kinerja *bot* karena waktu responnya tidak menunjukkan perbedaan yang cukup signifikan. Begitu pula pada jaringan yang digunakan pada *smartphone* baik 2G, 3G dan 4G tidak mempengaruhi pada kinerja *bot*. Karena selisih waktu respon antar jaringan sekitar 1 detik saja. Akan tetapi, jaringan yang bagus untuk digunakan adalah 4G dengan rata-rata waktu 4,05 detik. Notifikasi pembayaran juga berhasil dikirimkan setelah *reader* RFID membaca *tag* RFID. Sistem ini masih memiliki kekurangan, dan masih dapat dikembangkan lagi. Pertama, 1 membuat fungsi membeli saldo melalui *bot* Telegram sehingga pengguna atau pelanggan sistem ETC tidak perlu ke bank atau mitra terkait untuk membeli saldo. Kedua, sistem dilengkapi GIS (*Geographic Information System*) baik di sisi *server* (*web*) maupun pada *bot*. Salah satu tujuannya apabila pengguna atau pelanggan mengalami mogok atau masalah saat berada di jalan tol sehingga mudah dalam mengetahui lokasinya. Ketiga, keamanan sistem terutama pada keamanan jaringan. Karena sistem yang dibuat berbasis *web service*.

Referensi

[1] Sodikin, B. Riyanto dan B. Pudjianto, "*Kajian Masalah Antrian Pada Sistem Pengumpulan Tol Konvensional Terhadap Rancangan Sistem Pengumpulan Tol Elektronik*," Universitas Diponegoro, Semarang, 2005.

[2] Amol A. Chapate; D. D. Nawgaje, "*Electronic Toll Collection System Based on ARM*". International Journal of Science, Engineering and Technology Research (USETR)," vol. 4, no. 1, January 2015.

[3] K. Persad, C. M. Walton dan S. Hussain, "*Toll Collection Technology and Best Practices,*" Center for Transportation Research The University of Texas at Austin, Austin, Texas, 2006.

[4] R.-C. Jou, Y.-C. Chiou, C.-W. Kuo dan H.-I. Tan, "Freeway drivers' willingness to pay for an on board unit under," ELSEVIER, no. 27, pp. 16-24, 2013.

[5] V. V. Kommineni , K. R. Kumar dan W. J. Joseph, "*INTEGRATED HIGHWAY MANAGEMENT SYSTEM USING RADIO FREQUENCY TECHNOLOGY BASED ON ARM*," *IJERA,* vol. 2, no. 3, pp. 151-155, 2012.

[6] D.-P. Pop dan A. Altar, "*Designing an MVC Model for Rapid Web Application Development*," *ELSEVIER,* vol. 69, p. 1172 – 1179 , 2014.

[7] A. Y. Aleryani, "*Comparative Study between Data Flow Diagram and Use*," *IJSP,* vol. 6, no. 3, pp. 124-127, 2016.

05-Perancangan Perangkat Lunak Pendukung Etc (Electronic Toll Collection) Dengan Notifikasi Melalui Social Messenger

ORIGINALITY REPORT

8%	8%	0%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	paloegada.net Internet Source	8%

Exclude quotes	On	Exclude matches	< 5%
Exclude bibliography	On		