Pembuatan Aplikasi Android Driver Control Sebagai Sarana Memonitor Anak Berkendara Secara Waktu-Nyata

Lia Lidya Roza, R. Rizal Isnanto, Eko Didik Widianto
Program Studi Sistem Komputer, Fakultas Teknik, Universitas Diponegoro
Jl. Prof. Soedarto, SH, Kampus Undip Tembalang, Semarang, Indonesia 50275
e-mail: lialidya@ce.undip.ac.id

Abstract

Nowadays, motorcycle is one of the primary needs in Indonesia. Almost every family has at least one motorcycle, includes the children who is allowed by their parents to ride a motorcycle independently. However, without any realization, accidents and crimes such as motorcycle robbery are lurking children continuously. Because of that, a real time application is needed to monitor the children while riding motorcycle. This application is made as an effort to reduce the number of traffic accidents and motorcycle robbery. Waterfall model is used as research's method on developing the application. The application is developed on Android Studio, using Java programming language, database MySQL, and a web-based service. This research's results is an Android Driver Control Application which can provide motorcycle location information, speed, fallen motorcycle notifications, and driver record data. Users can turn off or turn on the motorcycle's engine automatically through this application. It also has login facility for admin to enable or disable the Driver Control tool and can be used by one or more user to monitor their motorcycle with Driver Control tool installed on it.

Keywords: *motorcycle; driver control; android studio*

Abstrak

Zaman sekarang motor merupakan salah satu kebutuhan primer di Indonesia, hampir setiap keluarga setidaknya memiliki minimal 1 kendaraan bermotor, tak terkecuali anak-anak yang juga sudah diizinkan orang tua untuk mengendarai sepeda motor agar lebih mandiri. Namun tanpa disadari bahaya kecelakaan dan kejahatan seperti pencurian sepeda motor terus mengintai sang anak. Untuk itu diperlukan sebuah aplikasi yang dapat memonitor anak berkendara secara waktu-nyata sebagai upaya mengurangi angka kecelakaan lalu lintas dan pencurian sepeda motor. Metodologi penelitian pada aplikasi ini dibuat menggunakan model *waterfall*. Pembuatan aplikasi ini menggunakan perangkat lunak *Android Studio* dengan bahasa pemrograman *Java*, basisdata *MySQL*, dan berbasis *Web Service*. Hasil penelitian ini adalah aplikasi *Android Driver Control* yang mampu memberikan informasi lokasi kendaraan, kecepatan, notifikasi motor jatuh, dan rekaman data pengemudi. Pengguna dapat mematikan atau menghidupkan mesin motor secara otomatis melalui aplikasi ini. Aplikasi ini juga memiliki fasilitas *login* untuk *admin* yang digunakan untuk mengaktifkan atau menonaktifkan alat *Driver Control*. Aplikasi ini dapat digunakan lebih dari satu pengguna untuk memantau satu sepeda motor yang sudah terpasang alat *Driver Control*.

Kata Kunci: motor; *driver control*; *android studio*

1. Pendahuluan

Mobilitas yang tinggi menuntut masyarakat agar tidak tertinggal oleh kemajuan zaman, salah satunya yaitu dengan memiliki kendaraan. Tetapi tidak sedikit orang terbunuh di jalan akibat kepadatan lalu lintas dan kesalahan manusia. Organisasi Kesehatan Dunia (WHO) memperkirakan bahwa di tahun 2020 penyebab terbesar ketiga kematian adalah kecelakaan

jalan raya, tepat di bawah penyakit jantung dan depresi. WHO mencatat bahwa 1 juta orang di seluruh dunia meninggal setiap tahunnya di jalan raya akibat kecelakaan, dimana 40% diantaranya berusia dibawah 25 tahun [1].

Di Indonesia angka pencurian sepeda motor cukup tinggi. Pada tahun 2012 di kota Malang, Jawa Timur terdapat 1.200 kasus pencurian sepeda motor, dan pada tahun 2013 mencapai 1.188 kasus [2].

Hal ini membuat masyarakat semakin khawatir untuk bepergian dan saat parkir kendaraan sepeda motor ditempat umum. Keinginan utama para orang tua tentu saja melindungi keluarga terutama anak-anak dari bahaya, namun hal ini sulit untuk dilakukan oleh orang tua yang sibuk bekerja karena mereka tidak memiliki waktu yang optimal untuk mengawasi anak mereka secara langsung terutama saat anak-anak mereka berkendara.

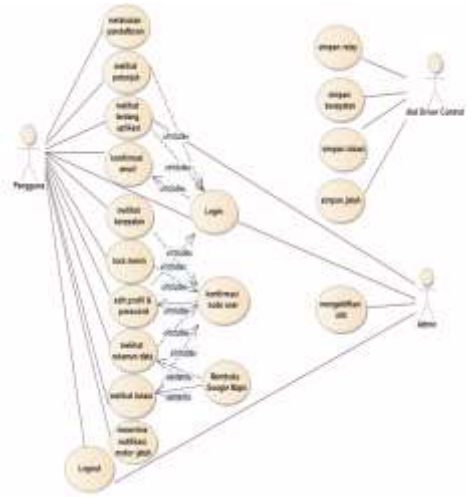
Penelitian berjudul *Simulasi Sistem Informasi Geografis (SIG) Pemantauan Posisi Kendaraan Via SMS Gateway* bertujuan untuk membangun simulasi sistem pemantauan posisi kendaraan dan dapat menampilkan jalur perjalanan kendaraan [3]. Penelitian berjudul *Sistem Pelacak Kendaraan Berbasis OpenGTS* bertujuan untuk membangun sistem pelacak kendaraan untuk meningkatkan keamanan komoditas industri pengiriman layanan barang dan jasa [4]. Penelitian berjudul *Pembuatan Aplikasi Memantau Lokasi Anak Berbasis Android Menggunakan Location Based Service* bertujuan untuk memantau lokasi anak melalui perangkat bergerak [5]. Berdasarkan latar belakang diatas diperlukan sebuah aplikasi yang mampu untuk membantu orang tua untuk mengawasi anak berkendara dan mendeteksi pencurian sepeda motor. Tujuan penelitian ini membuat aplikasi Android *Driver Control* yang dapat menampilkan data lokasi dengan bantuan GPS [6], kecepatan pengemudi, memberikan rekaman data, memberikan notifikasi motor jatuh, pengguna dapat menghidupkan atau mematikan mesin motor secara otomatis melalui aplikasi. Aplikasi juga memiliki fasilitas admin yang digunakan untuk mengaktifkan atau menonaktifkan alat *Driver Control*.

2. Metode Penelitian

Metode penelitian pada aplikasi Android *Driver Control* ini menggunakan model *waterfall* yang pengerjaan dari suatu sistem dilakukan secara berurutan atau secara linear yang memiliki lima tahap yang berbeda yaitu analisa kebutuhan, desain sistem, implementasi, pengujian program, dan pemeliharaan. Analisa kebutuhan aplikasi dilakukan dengan mengidentifikasi kebutuhan fungsional, dan kebutuhan non-fungsional meliputi kebutuhan pengguna, kebutuhan perangkat keras, dan perangkat lunak untuk membuat aplikasi. Desain pada aplikasi ini menggambarkan alur kerja sistem dengan UML versi 1.5 yaitu *use case diagram*, dan *Entity Relationship Diagram* [7]. Implementasi sistem ini menggunakan perangkat lunak Android Studio [8] dengan bahasa pemrograman Java [9] menggunakan basisdata MySQL [10] dan berbasis *Web Service* [11]. Pengujian sistem ini menggunakan pengujian kotak hitam [12] untuk menguji semua fungsi pada sistem. Pemeliharaan perangkat lunak yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan karena perangkat lunak harus menyesuaikan dengan kebutuhan pelanggan yang berubah dari waktu ke waktu.

3. Perancangan Sistem

Pada bagian ini membahas mengenai perancangan aplikasi *Driver Control*, yang pertama analisis kebutuhan yang terdiri dari analisis kebutuhan fungsional dan kebutuhan non fungsional. Kebutuhan fungsional yang berkaitan mengenai informasi apa saja yang harus ada pada sistem. Kebutuhan fungsional di antaranya aplikasi ini dapat digunakan beberapa pengguna untuk memantau satu motor, pengguna mendapatkan notifikasi *email* untuk verifikasi setelah registrasi akun baru, aplikasi dapat digunakan jika pengguna sudah berhasil Login. Aplikasi mampu menampilkan lokasi, kecepatan, rekaman data pengemudi dalam waktu 1 minggu, dan notifikasi motor jatuh. Pengguna dapat menghidupkan atau mematikan mesin motor secara automatis melalui aplikasi. Admin mampu mengaktifkan atau menonaktifkan alat melalui aplikasi. Kebutuhan non fungsional merupakan kebutuhan yang menunjang kinerja sistem meliputi perangkat keras dalam membuat aplikasi yaitu perangkat Android dan Laptop. Perangkat lunak meliputi Android Studio, JDK, dan Enterprise Architect. Berikutnya melakukan perancangan sistem dengan perangkat lunak Enterprise Architect.



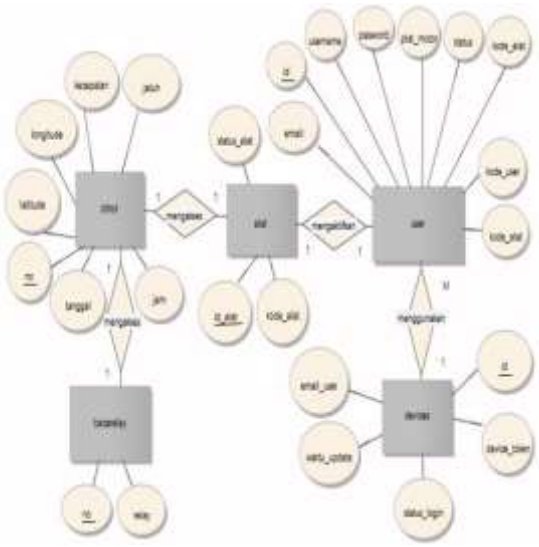
Gambar 1. *Use case* diagram sistem

Pada Gambar 1 merupakan *use case* diagram system atau gambaran umum aplikasi dimana terdapat tiga aktor yaitu pengguna, alat Driver Control, dan admin. Serta terdapat 19 *usecase* yaitu melakukan pendaftaran, login*,* melihat lokasi*,* melihat kecepatan*,* menerima notifikasi motor jatuh, konfirmasi *email*, konfirmasi kode user, melihat petunjuk, edit profil dan *password*, *lock* mesin, melihat rekaman data, Google Maps, melihat tentang aplikasi, simpan *relay*, simpan kecepatan, simpan lokasi, simpan jatuh, mengaktifkan alat, dan *Logout*.

Dari *use case* pada Gambar 1 dapat disimpulkan bahwa Pengguna dapat membuka aplikasi jika sudah melakukan pendaftaran dan konfirmasi *email* sebelum *Login*. Setelah berhasil *Login*, pengguna dapat melihat petunjuk penggunaan aplikasi, melihat tentang pengembang aplikasi, dan menu *Profile*. Untuk melihat lokasi kendaraan*,* melihat kecepatan kendaraan, *lock* mesin, dan melihat rekaman data dapat dilakukan jika pengguna sudah konfirmasi kode user pada menu *Profile*. Saat motor jatuh, pengguna dapat menerima notifikasi motor jatuh selama Internet aktif.

Google Maps digunakan untuk melihat lokasi kendaraan jika pengguna mengakses menu *Maps* dan halaman Rekaman data. Untuk keluar dari aplikasi pengguna dapat menggunakan menu *Logout* pada aplikasi. Alat *Driver Control* berfungsi untuk menyimpan data lokasi, kecepatan, jatuh, dan *relay* pada basisdata. Sebelum data ditampilkan pada aplikasi, *Admin* harus *Login* untuk mengaktifkan alat terlebih dahulu dengan mengecek kode alat dan status alat *Driver Control*.

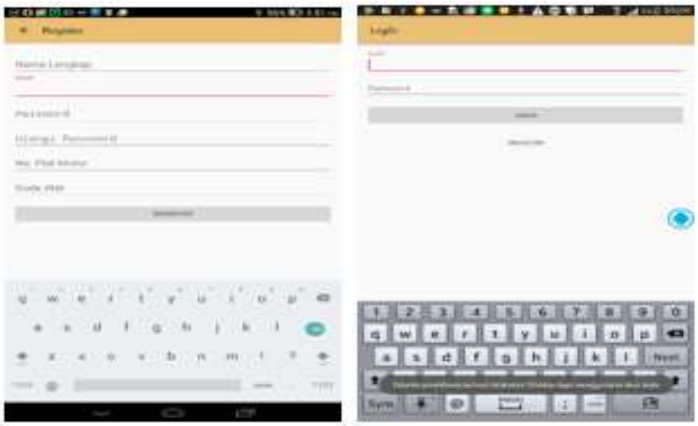
Gambaran yang digunakan untuk merancang basisdata yaitu menggunakan *Entity Relationship Diagram* (ERD) seperti Gambar 2. Gambar 2 merupakan desain ERD aplikasi *Android Driver Control,* dalam ERD terdapat lima entitas yaitu entitas *user*, *ditrol*, *bacarelay*, alat dan *devices*. Tabel user merupakan tabel untuk menyimpan data pengguna aplikasi, dan data admin. Tabel *ditrol* merupakan tabel untuk menyimpan data *latitude, longitude* untuk menampilkan lokasi motor, kemudian data kecepatan, dan nilai jatuh. Tabel *bacarelay* merupakan tabel untuk menyimpan data *relay* yang akan digunakan untuk mematikan atau menghidupkan mesin motor pada fitur *Lock*. Tabel *devices* merupakan tabel untuk menyimpan data *device_token* dari perangkat *Android* untuk fitur notifikasi. Tabel alat merupakan tabel untuk menyimpan data kode_alat, dan status_alat yang akan digunakan *admin* untuk mengaktifkan atau menonaktifkan alat. Tabel *user* berelasi ke tabel *devices* menyatakan hubungan pengguna dengan perangkat Android. Tabel *user* berelasi ke tabel alat menyatakan hubungan *admin* dengan alat *Driver Control*. Tabel alat berelasi ke tabel ditrol kemudian tabel ditrol berelasi ke tabel bacarelay untuk menampilkan data dari alat secara keseluruhan.



Gambar 2. Desain ERD aplikasi

4. Implementasi Perangkat Lunak

Pada bagian ini akan dijelaskan tahap pembuatan dan pengujian sistem. Tahap pembuatan merupakan langkah mengubah desain yang telah dibuat sebelumnya menjadi kode-kode program. Sedangkan untuk tahap pengujian dilakukan dengan menggunakan pengujian kotak hitam. Setelah membuat basisdata dan layanan *web service* dilakukan pembuatan aplikasi seperti berikut:



Gambar 3. Halaman masuk aplikasi

Gambar 3 bagian sebelah kiri merupakan halaman *Register* yang terdapat enam kolom masukan yang harus diisi pengguna terlebih dahulu agar mempunyai akun baru. Enam Kolom tersebut yaitu kolom untuk Nama Lengkap, *Email, Password*, Ulangi *Password,* No. Plat Motor, dan Kode Alat. Gambar 3 bagian sebelah kanan merupakan halaman *Login* yang terdapat dua kolom masukan yang wajib diisi oleh pengguna, kolom untuk *Email* dan kolom untuk *Password*. Melalui tombol *Login*, aplikasi akan melakukan autentikasi pengguna. Pengujian aplikasi ini menggunakan dua akun *email* berbeda, dan perangkat *Android* yang berbeda. Jika pengguna mengisi data dengan benar, sistem akan mengirim verifikasi *email* agar pengguna bisa masuk ke halaman utama aplikasi.

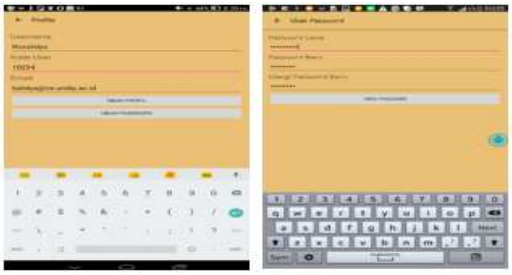
Gambar 4 menunjukkan verifikasi melalui pengguna dengan akun *email* yang berbeda. Setelah melakukan verifikasi pengguna dapat masuk ke halaman utama aplikasi.

Gambar 4. Verifikasi pengguna

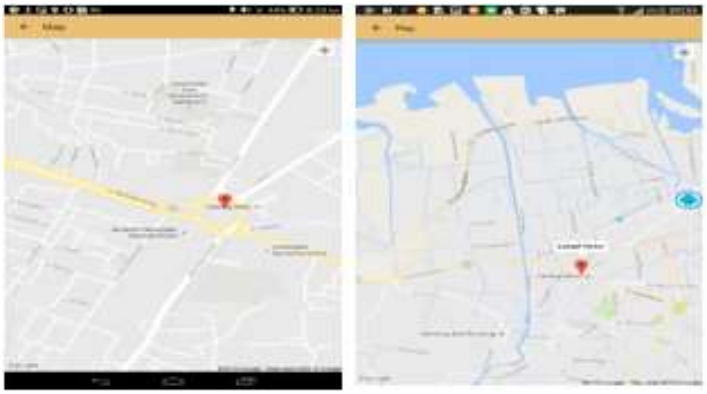
Gambar 5 merupakan tampilan halaman utama aplikasi *Driver Control*. Pada halaman ini terdapat empat tombol yaitu *Maps, Speed, Lock*, dan *Profile*. Pada *slidebar menu* terdapat empat menu *How to use, Data record, About*, dan *Logout*. Tombol *Maps* untuk melihat peta lokasi kendaraan motor. Tombol *Speed* untuk melihat grafik kecepatan rata-rata pengemudi, tombol *Profile* untuk mengedit *profile* dan mengubah *password*, tombol *Lock* untuk mematikan mesin motor secara otomatis dari aplikasi. Sedangkan pada *slidebar menu* terdapat menu *How to use* untuk petunjuk penggunaan aplikasi, menu *Data record* untuk rekaman data pengemudi, menu *About* untuk melihat tentang pengembang aplikasi, dan menu *Logout* untuk keluar dari aplikasi.

Gambar 5. Halaman utama

Gambar 6 bagian kiri merupakan tampilan dari halaman *Profile*. Pada halaman *Profile* terdapat tiga kolom masukan yaitu *Username,* Kode *User*, dan *Email.* Untuk dapat mengakses semua menu pengguna harus mengisi Kode *User* pada halaman *Profile*. Jika pengguna belum mengisi Kode *User* ketika menekan tombol *Maps, Speed,* dan *Lock* pada halaman utama akan tampil peringatan untuk mengisi Kode *User*. Pada halaman *Profile* juga terdapat tombol UBAH *PASSWORD* yang dapat digunakan untuk mengubah kata sandi seperti pada Gambar 6 bagian kanan.



Gambar 6. Halaman *profile*



Gambar 7. Halaman *map*

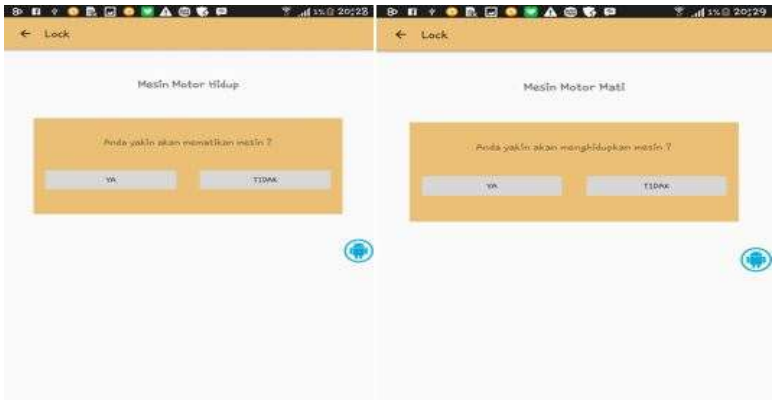
Setelah mengisi Kode *User* pengguna dapat mengakses seluruh menu aplikasi. *Maps* adalah salah satu menu aplikasi *Driver Control* yang dapat menunjukkan lokasi kendaraan dalam waktu terbaru, dan menggunakan layanan *Google Maps*. Gambar 7 merupakan tampilan halaman *Map* dari dua perangkat *Android* yang berbeda. Lokasi motor pada *Map* ditunjukkan oleh sebuah penanda. Halaman *Map* juga terdiri dari tombol ← yang berfungsi untuk kembali ke halaman sebelumnya.

Speed merupakan fitur aplikasi *Driver Control* yang dapat menampilkan grafik kecepatan pengemudi. Data kecepatan yang ditampilkan dalam satu halaman merupakan data kecepatan pengemudi dalam satu hari terakhir. Halaman Kecepatan juga terdiri dari tombol ← yang berfungsi untuk kembali ke halaman sebelumnya.

Gambar 8. Halaman kecepatan

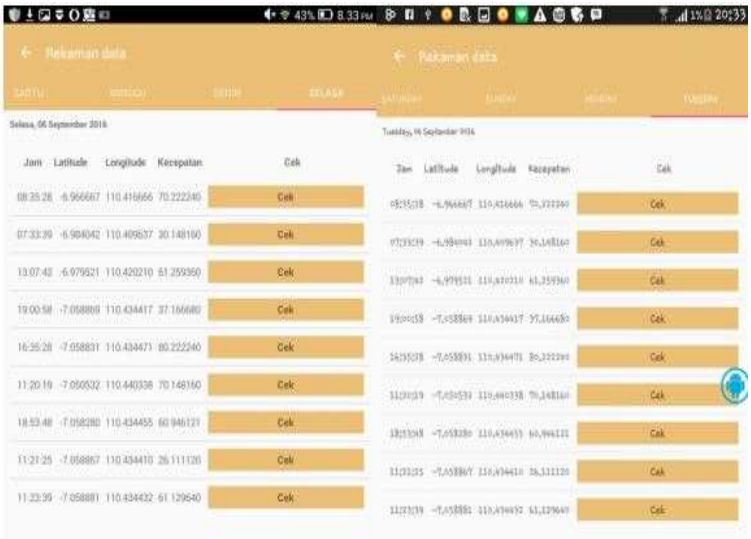
Gambar 8 bagian kiri menunjukkan grafik kecepatan rata-rata motor pada tanggal 24 Oktober 2016 dari pukul 10.00 sampai dengan 22.00, dan Gambar 8 bagian kanan menunjukkan grafik kecepatan rata-rata motor pada tanggal 25 Oktober 2016 dari pukul 10.00 sampai dengan 22.00. Aplikasi dapat menunjukkan kecepatan rata-rata sesuai dengan hasil pengujian alat *Driver Control*. *Lock* merupakan fitur aplikasi *Driver Control* yang dapat mematikan mesin motor secara otomatis. Ketika mesin motor hidup, halaman *Lock* akan menampilkan teks Mesin Motor Hidup dan konfirmasi untuk mematikan mesin motor seperti pada Gambar 9 bagian kiri.

Jika pengguna memilih YA maka aplikasi akan mengirimkan perintah untuk mematikan mesin motor, dan jika memilih TIDAK maka aplikasi tetap berada pada halaman *Lock* dan mesin motor tetap hidup. Sebaliknya, ketika mesin motor mati, halaman *Lock* akan menampilkan teks Mesin Motor Mati dan konfirmasi untuk menghidupkan mesin motor seperti pada Gambar 9 bagian kanan. Halaman *Lock* juga terdiri dari tombol ← yang berfungsi untuk kembali ke halaman sebelumnya. Aplikasi dapat berjalan dengan baik pada kedua perangkat *Android* dengan pengguna aplikasi yang berbeda, terlihat pada Gambar 9 dimana aplikasi dapat menunjukkan status mesin motor dan konfirmasi untuk mematikan atau menghidupkan mesin motor secara otomatis.



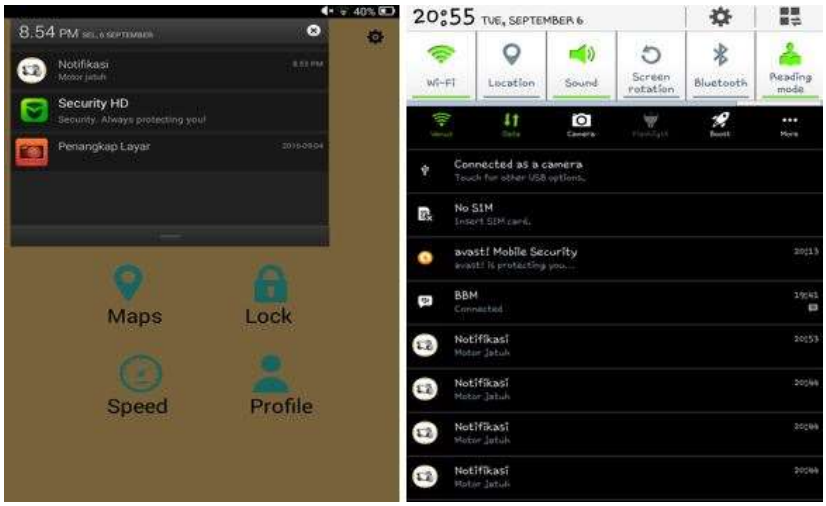
Gambar 9. Halaman *Lock*

Data record merupakan menu aplikasi *Driver Control* yang merekam data aktivitas kendaraan selama digunakan, data yang direkam berupa *latitude, longitude*, kecepatan, jam, dan tanggal. Data direkam selama satu minggu terakhir saat kendaraan digunakan. Untuk menuju halaman Rekaman data, pengguna harus menekan tombol *Data record* pada *slidebar menu*. Pada halaman Rekaman data terdapat tombol aksi Cek yang berfungsi untuk melihat peta lokasi kendaraan seperti pada Gambar 10.



Gambar 10. Halaman Rekaman data

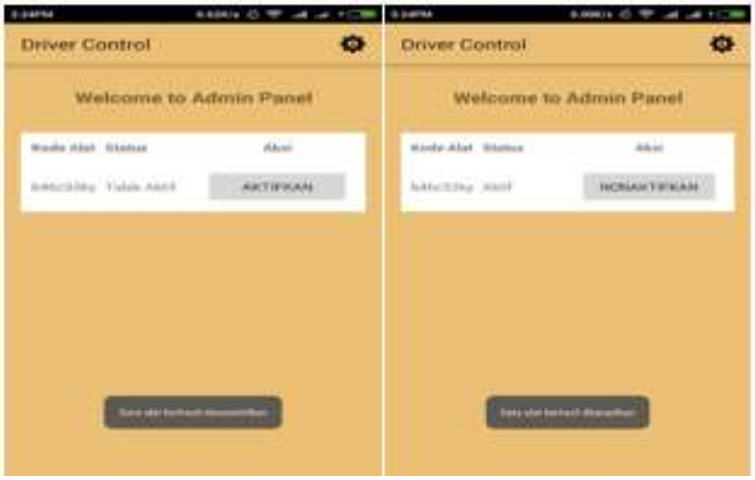
Notifikasi motor jatuh adalah salah satu menu aplikasi *Driver Control* yang dapat diterima pengguna ketika motor jatuh. Notifikasi dapat dilihat pengguna selama Internet pada perangkat *Android* dalam keadaan aktif. Aplikasi dapat berjalan dengan baik pada kedua perangkat *Android* dengan pengguna aplikasi yang berbeda, terlihat pada Gambar 11 bagian kiri, dan bagian kanan yang menunjukkan aplikasi dapat menampilkan notifikasi pada perangkat *Android* di saat yang bersamaan ketika motor jatuh pada pukul 20.53.



Gambar 11. Notifikasi motor jatuh

Pada aplikasi ini *admin* bertugas untuk menyaring data dengan cara mengaktifkan atau menonaktifkan alat. Pertama, *admin* harus melakukan *Login* terlebih dahulu agar dapat masuk halaman *Admin*. Kemudian, *admin* dapat melihat kode alat dan status alat untuk mengambil keputusan untuk mengaktifkan alat atau menonaktifkan alat menggunakan tombol aksi pada halaman *Admin*.

Gambar 12 menunjukkan halaman *Admin*, *admin* dapat melihat kode alat dan status alat yang menunjukkan alat dalam kondisi tidak aktif pada Gambar 12 bagian kiri, dan alat aktif pada Gambar 12 bagian kanan. Jika *admin* ingin mengaktifkan alat, *admin* dapat menekan tombol AKTIFKAN sebaliknya, jika *admin* ingin menonaktifkan alat, *admin* dapat menekan tombol NONAKTIFKAN.



Gambar 12. Halaman *Admin*

5. Pembahasan

Pengujian pada pembuatan aplikasi ini akan menggunakan pengujian kotak hitam. Pengujian ini dilakukan untuk menunjukkan fungsi program yang dibuat tentang cara operasi dan kegunaannya, apakah keluaran data sesuai dengan yang diharapkan. Pengujian aplikasi dibuat dalam bentuk tabel pengujian kotak hitam dari masing-masing menu yang ada dalam aplikasi. Pengujian yang akan dilakukan dengan metode pengujian kotak hitam yang berfokus pada persyaratan fungsional perangkat lunak. Pengujian dimulai dengan pengujian menu *Login*

yang muncul pertama kali ketika membuka aplikasi. Tabel 1 menunjukkan tabel pengujian *Login*. Hasil pengujian Tabel 1 menunjukkan bahwa fungsi yang ada pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan mulai dari menampilkan halaman *Login*, mengirimkan *email* verifikasi saat pengguna menekan tombol *Login*, menampilkan pesan kesalahan ketika pengguna salah memasukkan data, menampilkan halaman *Register* saat pengguna menekan tombol *Register*.

Tabel 1. Pengujian *login*

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Tampil halaman Login	Membuka aplikasi	Menampilkan halaman Login	Berhasil
Berhasil masuk	Mengklik tombol Login setelah mengisi nama pengguna dan kata sandi dengan benar	Masuk halaman utama aplikasi	Berhasil
Gagal masuk	Mengklik tombol Login setelah mengisi nama pengguna atau kata sandi salah atau sedang tidak terhubung jaringan	Menampilkan pesan kesalahan	Berhasil
Daftar akun baru	Mengklik tombol Register untuk menuju halaman pembuatan akun baru	Menampilkan halaman Register	Berhasil

Hal ini terjadi karena aplikasi mengirimkan data ke *Web Service* kemudian *server* mencocokkan data yang diisi pengguna dengan tabel *user* yang ada pada basisdata. Jika data pengguna tidak sama dengan data yang sudah terdaftar pada basisdata akan tampil pesan kesalahan. Sebelum melakukan *Login* pengguna harus mendaftar akun baru terlebih dahulu, setelah registrasi akun baru pada halaman *Register* sistem akan mengirimkan *email* untuk diverifikasi oleh pengguna. Ketika belum diverifikasi status pengguna masih bernilai 0 pada tabel *user* dalam basisdata, sehingga pengguna harus melakukan verifikasi *email* terlebih dahulu agar pengguna dapat masuk ke dalam halaman utama aplikasi. Setelah berhasil verifikasi, kolom status pada tabel user dalam basisdata akan bernilai 1 dan pengguna dapat masuk ke halaman utama aplikasi.

Tabel 2. Pengujian *register*

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Daftar pengguna baru	Mengisi data pengguna	Data disimpan dalam basisdata	Berhasil
Berhasil daftar	Mengklik tombol *Register* setelah mengisi data nama, *email*, *password*, dan no. plat motor.	Mengirim *email* verifikasi agar pengguna dapat *Login*	Berhasil
Gagal daftar	Mengklik tombol *Register* setelah mengisi data nama, *email*, *password*, dan no. plat motor, tetapi terdapat data yang salah atau gagal terhubung ke jaringan.	Menampilkan pesan kesalahan	Berhasil

Tabel 2 menunjukkan hasil pengujian *Register*. Hasil pengujian Tabel 2 menunjukkan bahwa fungsi yang ada pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan mulai dari menampilkan halaman *Register*, mengisi data pengguna baru untuk

disimpan ke basisdata, mengirim *email* verifikasi agar pengguna dapat melakukan *Login*, dan menampilkan pesan kesalahan jika terjadi kesalahan dalam pengisian data atau gagal terhubung ke jaringan. Pada proses pendaftaran akun baru aplikasi akan melakukan autentikasi, jika data sudah benar maka data akan disimpan pada tabel *user* dan tabel *devices* yang ada pada basisdata. Kemudian sistem akan mengirimkan *email* verifikasi agar pengguna dapat melakukan *Login*.

Hasil pengujian Tabel 3 menunjukkan bahwa fungsi pengujian halaman Utama pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan, mulai dari menampilkan halaman *Map* ketika tombol *Maps* ditekan, menampilkan halaman Kecepatan ketika tombol *Speed* ditekan, menampilkan halaman *Lock* ketika tombol *Lock* ditekan, menampilkan halaman *Profile* ketika tombol *Profile* ditekan, Menampilkan daftar menu *How to use, Data record*, *About,* dan *Logout* ketika *icon slidebar menu* ditekan. Hal ini terjadi karena pemanggilan fungsi *Intent* pada *mainactivity.java* sudah dihubungkan sesuai dengan setiap *activity.java* yang dibuat.

Tabel 3. Pengujian halaman utama

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Tombol *Maps*	Mengklik tombol *Maps*	Menampilkan halaman *Map* , dan menunjukan lokasi motor	Berhasil
Tombol *Speed*	Mengklik tombol *Speed*	Menampilkan halaman Kecepatan	Berhasil
Tombol *Lock*	Mengklik tombol *Lock*	Menampilkan halaman *Lock*	Berhasil
Tombol *Profile*	Mengklik tombol *Profile*	Menampilkan halaman *Profile*	Berhasil
Slidebar menu	Mengklik *icon slidebar* menu	Menampilkan daftar menu *How to use*, *Data record*, *About*, dan *Logout*	Berhasil

Tabel 4 menunjukkan hasil pengujian menu *Lock* saat motor hidup, dan Tabel 5 menunjukkan hasil pengujian menu *Lock* saat motor mati. Hasil pengujian Tabel 4 menunjukkan fungsi *Lock* saat motor hidup yang ada pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan, mulai dari menampilkan status mesin motor hidup, konfirmasi untuk mematikan mesin motor, memberi perintah untuk mematikan mesin motor, dan membatalkan perintah untuk mematikan mesin motor. Saat mesin motor hidup, alat *Driver Control* mengirimkan data status *relay* 1 ke basisdata yang terdapat dalam tabel *bacarelay*, kemudian ketika pengguna membuka menu *Lock* pada aplikasi, akan ditampilkan status mesin motor hidup sekaligus konfirmasi untuk mematikan mesin motor. Jika pengguna memilih Ya, maka status *relay* pada tabel *bacarelay* yang ada pada basisdata akan berubah menjadi 0, kemudian *server* mengirim perintah untuk memutus *relay* yang ada pada alat *Driver Control* sehingga mesin motor mati. Proses pengiriman perintah untuk mematikan mesin membutuhkan waktu kurang lebih 54 detik. Ketika mesin motor mati, pengemudi tidak bisa menyalakan mesin motor kembali sebelum pengguna aplikasi menghidupkan mesin motor melalui aplikasi *Driver Control*.

Tabel 4. Pengujian menu *lock* saat motor hidup

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Status mesin hidup	Memasuki halaman *Lock*	Menampilkan status mesin motor hidup	Berhasil
Konfirmasi mematikan mesin	Memasuki halaman *Lock* saat mesin motor hidup	Menampilkan konfirmasi untuk mematikan mesin motor	Berhasil
Tombol Ya	Mengklik tombol Ya	Memberi perintah untuk mematikan mesin motor	Berhasil
Tombol Tidak	Mengklik tombol Tidak	Menampilkan status mesin motor hidup	Berhasil

Hasil pengujian Tabel 5 menunjukkan fungsi *Lock* saat motor mati yang ada pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan, mulai dari menampilkan status mesin motor mati, konfirmasi untuk menghidupkan mesin motor, memberi perintah untuk menghidupkan mesin motor, dan membatalkan perintah untuk menghidupkan mesin motor. Saat mesin motor mati, alat *Driver Control* mengirimkan data status *relay* 0 ke basisdata yang terdapat dalam tabel bacarelay, kemudian ketika pengguna membuka menu *Lock* pada aplikasi, akan ditampilkan status mesin motor mati sekaligus konfirmasi untuk menghidupkan mesin motor. Jika pengguna memilih Ya, maka status *relay* pada tabel *bacarelay* yang ada pada basisdata akan berubah menjadi 1, kemudian *server* mengirim perintah untuk menyambung *relay* yang ada pada alat *Driver Control* dengan aki sehingga mesin motor hidup. Proses pengiriman perintah untuk menghidupkan mesin motor membutuhkan waktu kurang lebih 54 detik.

Tabel 5. Pengujian menu *lock* saat motor mati

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Status mesin mati	Memasuki halaman *Lock*	Menampilkan status mesin motor mati	Berhasil
Konfirmasi menghidupkan mesin motor	Memasuki halaman *Lock* saat mesin motor mati	Menampilkan konfirmasi untuk menghidupkan mesin motor	Berhasil
Tombol Ya	Mengklik tombol Ya	Memberi perintah untuk menghidupkan mesin motor	Berhasil
Tombol Tidak	Mengklik tombol Tidak	Menampilkan status mesin motor mati	Berhasil

Hasil pengujian Tabel 6 menunjukkan bahwa fungsi menu *Profile* pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan, mulai dari menampilkan halaman Ubah Profil, mengubah data profil, menampilkan konfirmasi ubah profil, menyimpan perubahan data profil, dan membatalkan perintah ubah profil. Menu *Profile* berkaitan langsung dengan tabel *user* yang ada pada basisdata, ketika pengguna menekan tombol *Profile* pada halaman utama, aplikasi akan meminta data ke s*erver*, kemudian *server* akan mengirimkan data pengguna dari tabel *user* yang ada pada basisdata untuk ditampilkan pada aplikasi. Setelah pengguna mengubah data, aplikasi akan mengirimkan perubahan data ke *server* kemudian disimpan ke dalam basisdata.

Tabel 6. Pengujian menu *profile*

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Halaman Ubah Profil	Mengklik tombol Profile pada halaman utama	Menampilkan halaman Ubah Profil	Berhasil
Mengubah data profil	Mengubah *username*, *email*, atau kode *user*	Data berhasil diubah dan disimpan ke dalam basisdata	Berhasil
Tombol Ubah Profil	Mengklik tombol Ubah Profil	Menampilkan konfirmasi untuk ubah profil	Berhasil
Tombol OK	Mengklik tombol OK	Menyimpan perubahan data, dan menampilkan pesan berhasil	Berhasil
Tombol BATAL	Mengklik tombol BATAL	Tetap berada pada halaman Profil	Berhasil

Hasil pengujian Tabel 7 menunjukkan bahwa fungsi Ubah *Password* pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan, mulai dari menampilkan halaman Ubah *Password*, mengubah kata sandi, menampilkan konfirmasi ubah kata sandi,

menyimpan perubahan kata sandi, membuat pengguna menuju halaman *Login* ketika sudah mengubah kata sandi, dan membatalkan perintah ubah kata sandi. Menu Ubah *Password* berkaitan langsung dengan tabel *user* yang ada pada basisdata, ketika pengguna menekan tombol Ubah *Password* pada halaman *Profile*, aplikasi akan menampilkan kolom kata sandi dalam kondisi kosong sehingga harus diisi oleh pengguna. Setelah pengguna mengisi kata sandi lama dan kata sandi baru, aplikasi akan mengirimkan perubahan data ke *server* kemudian disimpan ke dalam basisdata.

Tabel 7. Pengujian ubah *password*

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Tombol Ubah *Password*	Mengklik tombol UBAH *PASSWORD*	Menampilkan halaman Ubah *Password*	Berhasil
Mengubah kata sandi	Memasukkan kata sandi lama, kata sandi baru, dan ulangi kata sandi baru	Mencocokkan kata sandi	Berhasil
Tombol UBAH PASSWORD	Mengklik tombol UBAH *PASSWORD*	Menampilkan konfirmasi ubah kata sandi	Berhasil
Tombol *OK*	Mengklik tombol *OK*	Menyimpan perubahan kata sandi, dan menuju halaman Login	Berhasil
Tombol BATAL	Mengklik tombol BATAL	Tidak ada perubahan kata sandi	Berhasil

Hasil pengujian Tabel 8 menunjukkan bahwa fungsi Rekaman data pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan, mulai dari menampilkan rekaman data pengemudi, dan menampilkan halaman *Map* yang menunjukkan lokasi motor ketika tombol Cek ditekan. Ketika pengguna menekan menu *Data record*, aplikasi akan menampilkan halaman Rekaman data dengan data terbaru. Pada halaman Rekaman data terdapat pilihan waktu selama satu minggu terakhir, Jika pengguna ingin melihat rekaman data pengemudi pada hari sebelumnya, pengguna harus memilih hari, maka aplikasi akan meminta data ke *server*. *Server* mengirimkan rekaman data berupa *latitude, longitude*, kecepatan, dan waktu dari tabel ditrol pada basisdata. Jika pengguna ingin melihat lokasi motor, pengguna dapat menekan tombol Cek, aplikasi akan meminta layanan *Google Map* untuk menampilkan lokasi motor pada halaman *Map*.

Tabel 8. Pengujian menu *data record*

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Tombol Hari	Memilih hari	Menampilkan rekaman data	Berhasil
Tombol Cek	Mengklik tombol Cek	Menampilkan halaman *Map*	Berhasil

Hasil pengujian Tabel 9 menunjukkan bahwa fungsi Notifikasi pada tampilan sebagai pengguna berhasil dijalankan sesuai dengan harapan, ketika alat *Driver Control* diguncangkan, alat akan mengirim nilai jatuh 1 pada tabel ditrol yang ada pada basisdata. Ketika nilai jatuh sama dengan 1 sistem akan mengirimkan notifikasi motor jatuh pada perangkat *Android* yang terdaftar pada tabel *devices* yang ada dalam basisdata. Sedangkan ketika nilai jatuh 0 pada tabel ditrol aplikasi tidak mengirim notifikasi karena motor terdeteksi dalam kondisi normal.

Tabel 9. Pengujian notifikasi motor jatuh

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Notifikasi Motor Jatuh	Mengguncangkan alat *Driver Control*	Menampilkan notifikasi motor jatuh pada perangkat *Android*	Berhasil

Hasil pengujian Tabel 10 menunjukkan bahwa menu *Speed* berhasil dijalankan sesuai dengan harapan mulai dari menampilkan tanggal dalam seminggu, dan memperbarui data kecepatan pada grafik. Data tanggal dalam seminggu dapat ditampilkan karena menggunakan fungsi *spinner* pada *Android Studio*, dan proses saat memperbarui grafik kecepatan menggunakan fungsi *reset array* dan *update Speed Task()* pada *Android Studio*.

Tabel 10. Pengujian *speed*

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Pilih tanggal	Menekan tombol V	Menampilkan tanggal dalam seminggu	Berhasil
Tombol *UPDATE*	Mengklik tombol *UPDATE*	Memperbarui data kecepatan pada grafik	Berhasil

Pengujian berikutnya dilakukan untuk mengecek fungsi *admin* dalam aktivasi alat *Driver Control*. Hasil pengujian Tabel 11 menunjukkan bahwa fungsi yang ada pada tampilan sebagai *admin* berhasil dijalankan sesuai dengan harapan mulai dari *Login admin*, menampilkan halaman *Admin*, mengaktifkan alat, dan menonaktifkan alat. Fungsi menu ini bertujuan agar sistem tidak disalahgunakan oleh orang yang tidak bertanggung jawab dengan cara menyaring data yang dikirim oleh alat.

Tabel 11. Pengujian *admin*

Nama Pengujian	Bentuk Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Login	Menekan tombol *Login*	Menampilkan halaman *Admin* yang berisi kode alat dan status alat	Berhasil
Mengaktifkan alat	Menekan tombol AKTIFKAN	Menampilkan konfirmasi untuk mengaktifkan alat	Berhasil
Menonaktifkan alat	Menekan tombol NONAKTIFKAN	Menampilkan konfirmasi untuk menonaktifkan alat	Berhasil
Tombol *OK* konfirmasi	Menekan Tombol *OK* pada konfirmasi	Merubah status alat	Berhasil
Tombol Batal konfirmasi	Menekan Tombol Batal pada konfirmasi	Membatalkan perintah merubah status alat	Berhasil

6. Kesimpulan

Berdasarkan hasil pengujian dan analisis pembuatan aplikasi *Driver Control* dapat disimpulkan aplikasi ini mampu menampilkan peta lokasi kendaraan motor, grafik kecepatan pengemudi, profil pengguna aplikasi, rekaman data pengemudi, mengirimkan notifikasi ketika motor jatuh, dan pengguna dapat mematikan atau menghidupkan mesin motor secara otomatis melalui aplikasi. Aplikasi ini dapat digunakan oleh beberapa pengguna untuk memantau sebuah sepeda motor yang sudah terpasang alat *Driver Control*, pengguna membutuhkan koneksi Internet, dan perangkat keras *Driver Control* untuk bisa mengakses menu aplikasi.

Referensi

[1] World Health Organization, Global status report on road safety 2015. Italy. 2015: vii-x.

[2] Azmi, N.A., 2014, Modus Operandi Kejahatan Pencurian Kendaraan Bermotor Roda Dua. Skripsi S-1, Universitas Brawijaya, Malang.
[3] Hanifah, Raidah., 2010, Simulasi Sistem Informasi Geografis (SIG) Pemantauan Posisi Kendaraan Via SMS Gateway*,* Skripsi S-1, Teknik Elektro, Universitas Diponegoro, Semarang.
[4] Rusnandar, dkk., 2013, Sistem Pelacak Kendaraan Berbasis Open GTS, Jurnal Fakultas Teknologi Industri Universitas Ahmad Dahlan, Yogyakarta.
[5] Pria Utama, Hendra., 2015, Pembuatan Aplikasi Memantau Lokasi Anak Berbasis Android Menggunakan Location Based Service*",* Skripsi S-1, Departemen Sistem Komputer, Universitas Diponegoro, Semarang.
[6] Rompas, B.R, 2012, Aplikasi Location Based Service Pencarian Tempat Di Kota Manado Berbasis Android, Jurnal Teknik Elektro dan Komputer Universitas Sam Ratulangi Vo. 1.
[7] Rosa, A.S., dan Shalahuddin, Rekayasa Perangkat Lunak, Bandung: Modula, 2011.
[8] A. A. Huda, Live Coding: 9 Aplikasi Android Buatan Sendiri, Yogyakarta: Andi, 2013.
[9] Nugroho, Adi., Ebook: Rekayasa Perangkat Lunak Menggunakan UML dan Java, Andi Offset, 2009.
[10] Solichin, A., MySQL 5: Dari Pemula Hingga Mahir*,* Jakarta: Univ. Budi Luhur, 2010.
[11] Surendra, M. R. S.,2014 Implementasi PHP Web Service Sebagai Penyedia Data Aplikasi Mobile, vol. VI, no. 2, pp. 85–93.
[12] Rouf, A., Pengujian Perangkat Lunak Dengan Menggunakan Metode White Box dan Black Box, STMIK HIMSYA, Semarang, 2010.