SISTEM PENENTUAN TARGET PROMOSI PADA KONSUMEN MENGGUNAKAN *GEOFENCE*

SKRIPSI

Oleh:
AHMAD RIZA
NIM. 16650053

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2020

SISTEM PENENTUAN TARGET PROMOSI PADA KONSUMEN MENGGUNAKAN *GEOFENCE*

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
AHMAD RIZA
NIM. 16650053

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2020

LEMBAR PERSETUJUAN

SISTEM PENENTUAN TARGET PROMOSI PADA KONSUMEN MENGGUNAKAN *GEOFENCE*

SKRIPSI

Oleh :
AHMAD RIZA
NIM. 16650053

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal : Desember 2020

Pembimbing I	Pembimbing II
A'la Syauqi, M.Kom	M. Imamuddin, Lc., M.A
NIP. 19771201 200801 1 007	NIP. 19740602 200901 1 010

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

SISTEM PENENTUAN TARGET PROMOSI PADA KONSUMEN MENGGUNAKAN *GEOFENCE*

SKRIPSI

Oleh:
AHMAD RIZA
NIM. 16650053

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal Desember 2020

Susunan Dewan Penguji :			**Tanda Tangan**
Penguji Utama	:	Fajar Rohman Hariri, M. Kom NIP. 19890515 201801 1 001	( )
Ketua Penguji	:	Agung Teguh Wibowo Almais, M.T NIDT. 19860103 20180201 1 235	( )
Sekertaris Penguji	:	A'la Syauqi, M.Kom NIP. 19771201 200801 1 007	( )
Anggota Penguji	:	M. Imamuddin, Lc., M.A NIP. 19740602 200901 1 010	( )

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Ahmad Riza
NIM	: 16650053
Fakultas/Jurusan	: Sains dan Teknologi/Teknik Informatika
Judul Skripsi	: Sistem Penentuan Target Promosi pada Konsumen Menggunakan *Geofence*

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan menyertakan sumber kutipan pada daftar pustaka.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa skripsi saya hasil plagiasi, maka saya bersedia menerima sanksi atas perbuatan tersebut

Malang, 25 November 2020
Yang membuat pernyataan,

Ahmad Riza
NIM. 16650053

MOTTO

"Life is a blank paper, so write your own story"

HALAMAN PERSEMBAHAN

Alhamdulillah, puji syukur kepada Allah SWT yang telah memberikan kekuatan dan kesempatan kepada saya sehingga dapat menyelesaikan studi saya di Jurusan Teknik Informatika UIN Maulana Malik Ibrahim Malang. *Shalawat* salam kepada Rasulullah SAW, semoga kita senantiasa mendapat syafaatnya di *yaumil akhir*.

Ucapan terima kasih kepada orang tua saya, yang telah berjuang keras demi saya bisa malanjutkan studi saya hingga menjadi sarjana. Ucapan terima kasih juga saya berikan kepada segenap keluarga besar di Tuban yang selalu memberikan dukungan dan doa demi kesuksesan saya.

Terima kasih kepada dosen pembimbing saya Bapak A'la Syauqi M.Kom, dan Bapak M. Imamuddin, Lc., M.A yang telah memberikan bimbingan dan ilmunya sehingga skripsi ini dapat diselesaikan dengan sebaik-baiknya. Terima kasih juga kepada seluruh dosen dan staff di Jurusan Teknik Informatika UIN Maulana Malik Ibrahim Malang khususnya kepada dosen wali saya Ibu Hani Nurhayati, M.T yang telah membantu studi saya selama kurang lebih 4 tahun ini.

Terima kasih kepada teman-teman khususnya teman satu kontrakan yang telah menemani saya dalam suka dan duka saat menyelesaikan kuliah selama ini. Semoga kita menjadi yang terbaik di mana pun kita berada, *aamiin*.

KATA PENGANTAR

Assalamu'alaikum Warohmatullaahi Wabarakaatuh

Puji syukur kepada Allah SWT karena berkat rahmat-Nya yang tak terhingga, penulis bisa menyelesaikan skripsi ini. *Shalawat* serta salam kepada junjungan nabi Muhammad SAW, pembawa kabar gembira dan pemberi peringatan, yang membimbing umat Islam menuju kebenaran dengan izin-Nya. Semoga kita termasuk golongan yang senantiasa mendapat berkah Allah SWT dan mendapat pertolongan Nabi Muhamad SAW. *Aamiin.*

Selama proses pengerjaan skripsi, penulis mendapatkan banyak bantuan dan dukungan dari berbagai pihak. Maka dari itu, ucapan syukur dan terima kasih penulis sampaikan kepada:

1. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika yang telah berkenan menyetujui pengajuan skripsi ini serta memberikan bimbingan dalam penulisan skripsi.
2. A'la Syauqi, M.Kom, selaku dosen pembimbing I yang telah memberikan bimbingan dan saran yang sangat membantu penulis selama proses penelitian maupun penulisan skripsi.
3. M. Imamuddin, Lc., M.A, selaku dosen pembimbing II yang telah berkenan membimbing penulis secara religius baik di dalam maupun di luar proses pengerjaan skripsi
4. Agung Teguh Wibowo Almais, MT dan Fajar Rohman Hariri, M. Kom, selaku dosen penguji yang terlibat secara langsung menguji dan memberi saran pada proses pengerjaan skripsi dari seminar proposal hingga sidang skripsi, sehingga skripsi ini dapat diselesaikan dengan lancar dengan hasil yang lebih baik.

5. Seluruh staf dan dosen jurusan Teknik Informatika yang terlibat secara langsung maupun tidak langsung di dalam proses pengerjaan skripsi.
6. Seluruh teman-teman Teknik Informatika angkatan 2016, anggota komunitas Mocap, kakak tingkat, adik tingkat dan seluruh teman seperjuangan yang telah memberikan ilmu dan pengalaman yang memudahkan penulis dalam proses pengerjaan skripsi.
7. Ibu, bapak, kakak dan keluarga besar penulis yang telah menjadi memberikan dukungan dan doa yang tak terhingga kepada penulis.

Penulis menyadari bahwa hasil penelitian yang ditulis masih jauh dari kata sempurna. Oleh karena itu, penulis membuka kesempatan untuk setiap saran dan kritik yang membangun dengan senang hati. Terlepas dari itu semua, penulis berharap skripsi ini bermanfaat dan mampu membuat dunia menjadi tempat yang lebih baik.

Wassalamu'alaikum Warohmatullaahi Wabarakaatuh

Malang, 25 November 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Riza, Ahmad. 2020. **Sistem Penentuan Target Promosi pada Konsumen Menggunakan *Geofence***. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
Pembimbing: (I) A'la Syauqi, M.Kom. (II) M. Imamuddin, Lc., M.A.

Kata kunci: *Geofencing, Crossing Number, Winding Number, Geographic Information System.*

Promosi adalah salah satu strategi pemasaran yang bertujuan untuk mendapatkan pelanggan baru, mempengaruhi keputusan pembelian pelanggan dan meningkatkan kunjungan pelanggan. Oleh karena itu, promosi menjadi salah satu aspek penting bagi perusahaan yang ingin meningkatkan penjualannya. Namun, promosi memerlukan banyak biaya dalam pelaksanaannya, sehingga akan menyulitkan bagi perusahaan perintis yang hanya mempunyai modal terbatas. Biaya promosi dapat ditekan dengan mengoptimalkan target konsumen, salah satunya dengan menargetkan pelanggan pada wilayah tertentu. *Geofencing* dapat digunakan untuk menyeleksi pelanggan berdasarkan area poligon tertentu secara spesifik. Penelitian ini menggunakan *geofenced* area dengan metode winding point dan *crossing point* sebagai metode penentuan apakah suatu titik di dalam poligon atau tidak. Berdasarkan pengujian yang dilakukan dengan menggunakan data set acak yang diperoleh dari perhitungan dan aturan tertentu, metode *winding point* memperoleh hasil yang lebih memuaskan yaitu dengan akurasi 100% pada semua jenis poligon dan menghabiskan waktu rata-rata 30,8 ms dalam mengolah 100 ribu data. Sementara itu, metode *crossing point* memperoleh akurasi 100% kecuali pada poligon *convoluted*, yaitu 96% dan menghabiskan waktu rata-rata 51,2 ms.

ABSTRACT

Riza, Ahmad. 2020. **Promotion Target Determination System for Consumers Using Geofence**. Undergraduate Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University of Malang.
Supervisors: (I) A'la Syauqi, M.Kom. (II) M. Imamuddin, Lc., M.A.

Keyword: Geofencing, Crossing Number, Winding Number, Geographic Information System.

Promotion is a marketing strategy that aims to acquire new customers, influence customer purchasing decisions and increase customer visits. Therefore, promotion is an important aspect for companies that want to increase their sales. However, promotion requires a lot of money in its implementation, so it is difficult for startup companies that only have limited capital to do it. Promotion costs can be reduced by optimizing target consumers, one thing that can be done is targeting consumers in certain areas. Geofencing can be used to select customers based on a specific polygon area. This study uses a geofenced area with winding point and crossing point methods as a method of determining whether a point is in a polygon or not. Based on tests carried out using random data sets obtained from certain calculations and rules, the winding point method obtained more satisfactory results, namely with 100% accuracy on all types of polygons and spent an average of 30.8 milliseconds in processing 100 thousand data. Meanwhile, the crossing point method obtains 100% accuracy except for convoluted polygons, which is 96% and takes an average of 51.2 milliseconds in processing 100 thousand data.

ملخص

رضا أحمد. 2020. نظام تحديد هدف الترويج للمستهلكين باستخدام .Geofence مقال. قسم المعلوماتية ، كلية العلوم والتكنولوجيا ، جامعة الدولة الإسلامية (UIN) مولانا مالك إبراهيم مالانج.
المستشار: (أنا) آلاء سياوقي ، محمد كوم. (2) محمد إمام الدين ، ماجستير ، ماجستير

الكلمات المفتاحية: السياج الجغرافي ، رقم العبور ، رقم الملف ، نظام المعلومات الجغرافية.

الترويج هو استراتيجية تسويقية تهدف إلى اكتساب عملاء جدد والتأثير على قرارات شراء العملاء وزيادة زيارات العملاء. لذلك ، يعد الترويج جانبًا مهمًا للشركات التي ترغب في زيادة مبيعاتها. ومع ذلك ، يتطلب الترويج الكثير من المال في تنفيذه ، لذلك من الصعب على الشركات الناشئة التي لديها رأس مال محدود فقط القيام بذلك. يمكن تخفيض تكاليف الترويج من خلال تحسين المستهلكين المستهدفين ، والشيء الوحيد الذي يمكن القيام به هو استهداف المستهلكين في مناطق معينة. يمكن استخدام المبارزة الجغرافية لتحديد العملاء بناءً على منطقة مضلع معينة. تستخدم هذه الدراسة منطقة مسيجة جغرافيًا مع طرق نقطة الالتفاف ونقاط العبور كطريقة لتحديد ما إذا كانت النقطة في مضلع أم لا. بناءً على الاختبارات التي تم إجراؤها باستخدام مجموعات البيانات العشوائية التي تم الحصول عليها من حسابات وقواعد معينة ، حصلت طريقة نقطة اللف على نتائج مرضية أكثر ، أي بدقة 100٪ على جميع أنواع المضلعات وقضت ما معدله 30.8 مللي ثانية في معالجة 100 ألف بيانات. وفي الوقت نفسه ، تحصل طريقة نقطة العبور على دقة 100٪ باستثناء المضلعات الملتفة ، والتي تبلغ 96٪ وتستغرق معالجة 100 ألف بيانات في المتوسط 51.2 مللي ثانية.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi menghasilkan banyak inovasi pada bidang industri. Akibatnya banyak perusahaan-perusahaan perintis (*start-up*) yang tumbuh dengan menawarkan produk-produk inovatif yang beragam. Hal tersebut menjadikan pasar industri semakin kompetitif. Salah satu cara paling efektif dan termudah untuk bersaing di antara dunia pasar yang kompetitif adalah dengan menerapkan promosi penjualan (Jean & Yazdanifard, 2015). Promosi selalu menjadi strategi pemasaran yang penting untuk memastikan pertumbuhan dan keuntungan bagi organisasi ritel mana pun. Tujuan pemasaran dapat dibagi menjadi tiga arah yang berbeda. Tujuan pertama adalah untuk menarik pelanggan (akuisisi). Tujuan kedua adalah untuk mempengaruhi keputusan pembelian pelanggan dan tujuan ketiga adalah untuk meningkatkan frekuensi kunjungan dan tingkat pengeluaran pelanggan (Estelami & O'Connor, 2009).

Selain untuk mendapatkan pengguna, promosi juga dapat digunakan sebagai bentuk apresiasi suatu perusahaan untuk para penggunanya. Memberikan sebagian penghasilan kepada pengguna dalam bentuk promosi memungkinkan penghasilan perusahaan bertambah. Jika dilakukan dengan benar dan tepat sasaran, maka pengguna semakin loyal menggunakan produk dan terdapat kemungkinan jumlah pengguna bertambah. Allah ﷻ berfirman:

إِنَّ ٱلَّذِينَ يَتْلُونَ كِتَٰبَ ٱللَّهِ وَأَقَامُواْ ٱلصَّلَوٰةَ وَأَنفَقُواْ مِمَّا رَزَقْنَٰهُمْ سِرًّا وَعَلَانِيَةً يَرْجُونَ تِجَٰرَةً لَّن تَبُورَ

"Sesungguhnya orang-orang yang selalu membaca kitab Allah dan mendirikan salat dan menafkahkan sebahagian dari rezeki yang Kami anugerahkan kepada mereka dengan diam-diam dan terang-terangan, mereka itu mengharapkan perniagaan yang tidak akan merugi" (QS. Fatir:29).

Senantiasa membaca Al-Qur'an dan beribadah dengan tidak lupa menafkahkan sebagian penghasilan usaha atau perniagaan adalah tindakan yang dapat dilakukan oleh manusia untuk mendapatkan berkah Allah ﷻ dalam bentuk perniagaan yang tidak bangkrut. Salah satu bentuk menafkahkan sebagian penghasilan usaha adalah dengan memberikan sesuatu terhadap masyarakat sekitar dalam bentuk promosi. Jika dilakukan dengan baik dan benar maka nama perusahaan akan lebih dikenal masyarakat sehingga menguntungkan perusahaan.

Promosi adalah strategi pemasaran yang membutuhkan biaya yang besar. Laporan keuangan Spotify pada kuartal ketiga tahun 2019 menyatakan bahwa perusahaan ini menghabiskan biaya sebesar 178 juta Euro di bidang *sales* dan *marketing*. Biaya tersebut bahkan naik sebesar 22% atau 32 juta Euro dari bulan yang sama di tahun 2018. Biaya tersebut tidak menjadi masalah bagi Spotify karena uang tersebut hanya senilai 10% dari total penghasilan pada kuartal ketiga tahun 2019 (Spotify Technology S.A., 2019). Biaya promosi yang besar akan menjadi masalah bagi perusahaan perintis yang ingin bersaing. Menurut catatan data Statista sepanjang Agustus 2018 hingga Agustus 2019, sebuah perusahaan perlu mengeluarkan biaya senilai 1.75 USD untuk mendapatkan 1 orang pengguna mengistal aplikasi *mobile* mereka. Sementara untuk mendapatkan 1 orang pengguna yang mendaftar, diperlukan biaya sebesar 3.52 USD (Statista, 2019).

Data tersebut menunjukkan bahwa target akuisisi pengguna berbanding lurus dengan biaya promosi yang dikeluarkan.

Biaya promosi bisa ditekan dengan mempersempit target promosi. Namun, langkah tersebut akan berimbas pada potensi jumlah konsumen yang diakuisisi. Untuk itu, langkah terbaik adalah dengan mengoptimalkan target konsumen dengan melakukan personalisasi promosi ke segmen pasar tertentu yang paling potensial bagi perusahaan. Sebagai contoh, pada perusahaan yang bergerak di bisnis transportasi *online*, maka target promosi optimal adalah kelompok konsumen yang berpotensi melakukan banyak transaksi pada wilayah tertentu. Ketika personalisasi diterapkan di fragmentasi pasar, produk atau layanan standar dapat berubah menjadi solusi khusus untuk seorang individu (Changchien, et al., 2004). Oleh karena itu, penelitian ini mengusulkan suatu sistem personalisasi promosi berdasarkan lokasi pengguna (*Location Based Service*). Sistem ini memanfaat teknologi GPS pada perangkat *mobile* pengguna dan menggunakan metode *Geofencing* untuk mengelompokkan pengguna pada wilayah tertentu.

Geofencing adalah *Location Based Service* (LBS) proaktif yang paling banyak didiskusikan dan diminta saat ini. Metode ini memungkinkan untuk menentukan hubungan topologi antara objek bergerak dan suatu set area geografis terbatas, memungkinkan untuk mendeteksi dan memantau ketika perangkat *mobile* memasuki, meninggalkan, memotong atau memotong area geografis yang tepat dibatasi oleh perimeter virtual, yang disebut *geofence*, kemudian memperingatkan pengguna melalui notifikasi. *Geofence* dapat dibuat secara dinamis, seperti area melingkar yang mengelilingi posisi perangkat seluler atau seperangkat garis pembatas yang telah ditentukan, yang dapat digambar secara bebas oleh pengguna

untuk suatu tempat atau bangunan yang spesifik (Carchiolo, et al., 2018). *Geofence* untuk tempat atau bangunan yang spesifik adalah berbentuk poligon. Terdapat dua metode yang sering digunakan untuk mengetahui apakah suatu titik termasuk di dalam poligon atau di luar poligon, yaitu metode *crossing number (cn)* dan metode *winding number (wn)* (Sunday, 2020).

Penelitian serupa dengan menggunakan *geofence* pernah dilakukan oleh Puspa M. N. S. A. Basid pada tahun 2017 dengan judul *Pengembangan Sistem E-Complaint Berbasis Social Crowdsource Geotagging Studi Kasus Kota Malang* (Basid, et al., 2017). Penelitian tersebut menggunakan metode *raycasting*/*crossing number/even odd*, yaitu penentuan titik dengan menghitung apakah perpotongan garis proyeksi berjumlah ganjil atau genap. Akurasi algoritma yang digunakan pada penelitian tersebut hanya 94%. Penelitian ini akan mencoba untuk menyempurnakan penelitian tersebut dengan menggunakan algoritma berbeda, yaitu algoritma *winding number*.

Studi kasus dalam penelitian ini adalah sistem untuk mendapatkan target promosi mahasiswa di UIN Malang aplikasi Deliv. Deliv adalah aplikasi penyedia layanan transportasi *online* yang berbasis di Malang dan Bojonegoro. Pada penelitian ini akan dibuat sistem yang terdiri dari 3 aplikasi yaitu aplikasi administrator berbasis Android, aplikasi pengguna berbasis Android, dan API untuk mengolah algoritma yang berbasis *webservice*. Sistem akan memonitor lokasi pengguna terkini melalui perangkat *mobile* kemudian melaporkannya ke server dan selanjutnya akan dilaporkan daftar pengguna yang termasuk target promosi ke administrator.

Penelitian ini bertujuan untuk mengetahui bagaimana perbandingan performa metode *crossing number* dan *winding number* dalam mengidentifikasi target promosi pada wilayah tertentu yang dibatasi dengan *geofence*. Parameter pengujian yang dipakai adalah akurasi dan waktu yang dihabiskan dalam proses identifikasi lokasi pengguna oleh sistem yang dibuat. Hasil akhir yang diharapkan pada penelitian ini adalah metode yang dapat menargetkan promosi pada wilayah tertentu secara otomatis berbasis sistem informasi geografis.

1.2 Identifikasi Masalah

Berdasarkan permasalahan pada latar belakang yang telah dijelaskan di atas, masalah yang diidentifikasi pada penelitian ini adalah berapa akurasi dan kecepatan algoritma *crossing number* dan *winding number* dalam menentukan target promosi pada pengguna yang berada di wilayah tertentu yang dibatasi dengan *geofence*?

1.3 Tujuan Penelitian

Berdasarkan identifikasi masalah yang telah dipaparkan atas, tujuan yang ingin dicapai pada penelitian ini adalah menghitung akurasi dan kecepatan algoritma *crossing number* dan *winding number* dalam menentukan target promosi pada pengguna yang berada di wilayah tertentu yang dibatasi dengan *geofence.*

1.4 Manfaat Penelitian

Hasil akhir dari penelitian ini adalah metode yang dapat menargetkan promosi pada wilayah tertentu secara otomatis berbasis sistem informasi geografis.

1.5 Batasan Masalah

Adapun batasan masalah pada penelitian ini adalah sebagai berikut.

1. *Input* garis *geofence* digambar dari aplikasi administrator.
2. Data yang digunakan adalah data lokasi *random* di sekitar wilayah target.

3. Pengujian menggunakan aplikasi testing *client* yang dibuat untuk memungkinkan menghasilkan lokasi *random* berdasarkan kondisi pengujian tertentu.

BAB II

TINJAUAN PUSTAKA

2.1 Sistem Informasi Geografis

Sistem informasi geografis/*Geographic Information System* (GIS) adalah sistem berbasis komputer yang mempunyai kemampuan untuk menangani *georeferenced data.* GIS mempunyai empat set fungsi yaitu pengambilan dan persiapan data, manajemen data yang termasuk penyimpanan dan *maintenance*, manipulasi dan analisis data, dan presentasi data. Sehingga pengguna GIS dapat memasukkan data, menganalisis data tersebut dengan berbagai cara, dan menampilkan data tersebut dalam bentuk peta dan media presentasi lainnya (Huisman & By, 2009).

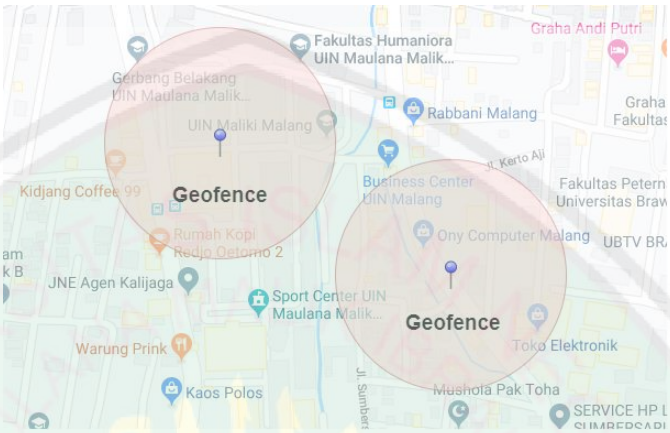
GIS telah diaplikasikan secara luas pada perangkat lunak program yang banyak tersedia untuk pengguna. GIS juga merupakan subjek penelitian yang serius dari para ilmuwan ilmu alam dan ilmu komputer. Entitas GIS ditentukan oleh kombinasi sumber daya teknis, program, dan informasi yang tidak hanya menyediakan akuisisi, penyimpanan, pemrosesan, akses, representasi, dan penyebaran, tetapi juga pemodelan kartografi matematis dan representasi data integral untuk memecahkan masalah perencanaan dan kontrol teritorial. Oleh karena itu GIS menggabungkan sumber daya teknis, perangkat lunak, regulasi dan aturan akuisisi, penyimpanan, analisis dan transfer informasi tentang proses dan fenomena yang memiliki pengikatan dan penyebaran spasial. Akibatnya mereka dikembangkan pada pergantian banyak cabang ilmu pengetahuan dan digunakan dalam berbagai bidang kegiatan manajemen (Tarasov & Pikhtin, 2007).

2.2 Geofencing

Geofencing adalah *Location Based Service* (LBS) proaktif yang paling banyak didiskusikan dan diminta saat ini. Metode ini memungkinkan untuk menentukan hubungan topologi antara objek bergerak dan suatu set area geografis terbatas, memungkinkan untuk mendeteksi dan memantau ketika perangkat *mobile* memasuki, meninggalkan, memotong atau memotong area geografis yang tepat dibatasi oleh perimeter virtual, yang disebut *geofence*, kemudian memperingatkan pengguna melalui notifikasi. *Geofence* dapat dibuat secara dinamis, seperti area melingkar yang mengelilingi posisi perangkat seluler atau seperangkat garis pembatas yang telah ditentukan, yang dapat digambar secara bebas oleh pengguna untuk suatu tempat atau bangunan yang spesifik (Carchiolo, et al., 2018).

Geofencing menggabungkan kesadaran tentang lokasi pengguna saat ini dengan kesadaran tentang kedekatan pengguna dengan lokasi yang mungkin menarik. Untuk menandai lokasi yang tepat, titik *latitude* dan *longitude* harus ditentukan. Untuk menyesuaikan kedekatan lokasi, suatu *radius* harus ditambahkan. *Latitude*, *longitude*, dan *radius* tersebut menentukan suatu *geofence* dan menciptakan area melingkar, atau pagar, di sekitar lokasi yang diinginkan. *Geofencing* dieksekusi di perangkat seluler yang mencakup pemosisian dan / atau pelacakan terus-menerus dari perangkat seluler serta pencocokan terus-menerus dari posisi perangkat seluler dengan seperangkat batas virtual yaitu *geofence*. *Geofence* dapat berupa pagar koordinat melingkar atau *poligonal*. Seperti yang ditunjukkan pada **Gambar 2.1** dan **Gambar 2.2**. Perangkat seluler dianggap sebagai klien yang bertanggung jawab terutama untuk mengirim lokasinya dan perbandingan posisi ponsel secara berkelanjutan dengan sejumlah besar *geofence*

untuk durasi minimum akan dihitung. Dengan demikian, sistem *Geofencing* untuk lingkungan ritel yang cerdas juga harus dapat memantau waktu tinggal target dalam *geofence* dan membatalkan pengiriman kupon jika prasyarat spasial-temporal dilanggar (Rahate & Saikh, 2016).



Gambar 2.1 *Circular Geofence*

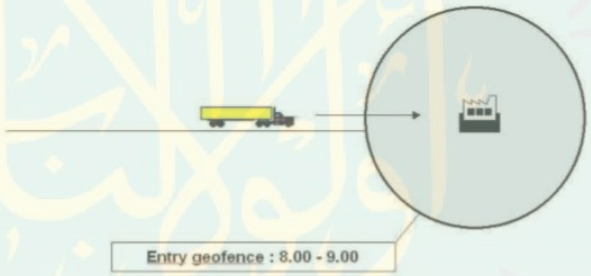
Gambar 2.2 *Poligonal Geofence*

2.2.1 Teknik-teknik *Geofencing*

Geofencing dapat dimanfaatkan pada banyak bidang dan memiliki banyak fungsi seperti pemantauan aset seluler dan orang-orang di dalam wilayah geografis, deteksi intrusi, dan perlindungan terhadap pencurian. Berbagai teknik *geofencing* telah dikembangkan untuk memenuhi kebutuhan pragmatis yang berbeda. Berikut teknik *geofencing* yang dapat digunakan (Reclus & Drouard, 2009).

A. *Proximity with A Point of Interest* (POI)

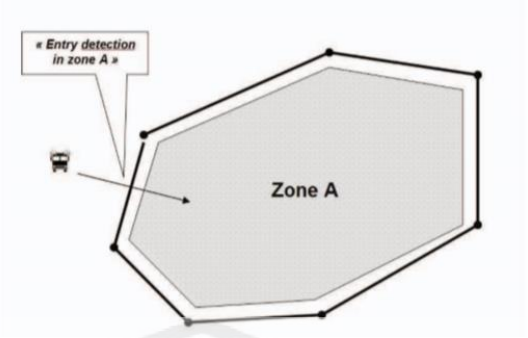
Teknik ini dimaksudkan untuk mendeteksi kedekatan kendaraan dalam kaitannya dengan tempat tujuan (POI). Dalam praktiknya, *geofence* yang dipakai adalah lingkaran, dan POI terletak di tengah. Radius diukur berdasarkan jarak yang dianggap "kedekatan" terhadap POI, dari beberapa meter hingga beberapa kilometer. Metode ini adalah cara paling sederhana untuk mengimplementasikan *geofencing*, karena hanya membutuhkan 2 parameter, koordinat pusat dan nilai jari-jari. Algoritma ini menghitung jarak antara objek seluler dan pusat lingkaran apakah lebih rendah atau lebih tinggi dari nilai jari-jari seperti yang terlihat pada **Gambar 2.3**, objek bergerak masing-masing akan ditentukan apakah di dalam atau di luar *geofence* (Reclus & Drouard, 2009).



Gambar 2.3. Deteksi Kedekatan dengan *Circular Geofence*

B. *Geofenced Area*

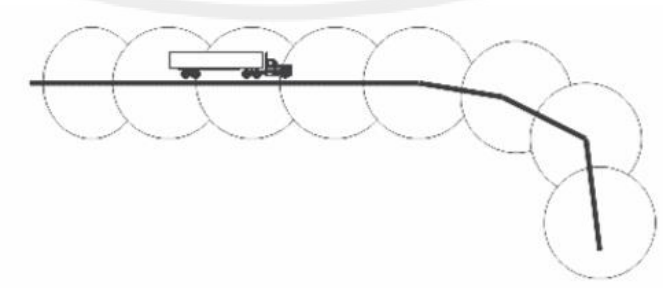
Teknik ini menyediakan pemantauan otomatis objek bergerak yang bergerak di sekitar atau di dalam area *geofenced*. Perangkat *mobile* akan terdeteksi ketika masuk atau keluar dari batas. Bentuk *geofence* dapat berupa figur geometris sederhana, seperti persegi atau persegi panjang, atau yang lebih rumit, seperti poligon kompleks. **Gambar 2.4** menjelaskan contoh model *geofenced area* dengan bentuk *geofence* poligon kompleks (Reclus & Drouard, 2009).



Gambar 2.4 *Geofence Aktif (Garis Hitam) di Sekeliling Sebuah Wilayah (Wilayah A)*

C. Route adherence

Teknik ini ditujukan untuk pemantauan objek bergerak dalam perjalanan, mulai dari titik keberangkatan hingga ke tujuan akhir. *Geofence* memungkinkan untuk memastikan bahwa kendaraan tidak menyimpang dari rute yang ditentukan. Rute dibuat dengan seperangkat koordinat, dan disimpan dalam aplikasi perangkat lunak sebelum keberangkatan kendaraan. Sebuah set *geofence* melingkar diterapkan di sepanjang seluruh rute, satu demi satu, seperti yang ditunjukkan pada **Gambar 2.5**. Jika kendaraan menyimpang dari rute, *mungkin* melintasi salah satu *geofences* dan melebihi toleransi deviasi yang telah ditetapkan, peringatan akan dihasilkan dan dikirim ke pusat kontrol, dengan lokasi di mana kendaraan telah keluar dari rute yang ditetapkan (Reclus & Drouard, 2009).



Gambar 2.5 *Kontrol Route Adherence Menggunakan Geofencing*

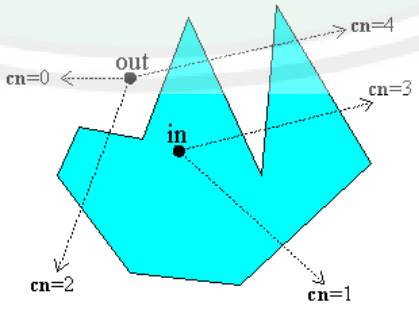
2.3 Point in Poligon Inclusion

Point in Poligon Inclusion adalah teknik untuk mengetahui apakah suatu titik termasuk di dalam poligon atau di luar poligon. Terdapat dua metode yang sering digunakan untuk menyelesaikan masalah ini, yaitu metode *Crossing Number (cn)* dan metode *Winding Number (wn)*.

2.3.1 *Crossing Number*

Dasar algoritma crossing number atau dapat disebut algoritma *even-odd* adalah menggunakan proyeksi tak terbatas dari titik yang dimaksud dan hitung berapa banyak tepi poligon yang dipotong oleh proyeksi (Galetzka & Glauner, 2017). Metode ini menghitung berapa kali sebuah proyeksi (*ray*) mulai dari titik P hingga melintasi tepi batas poligon. Ketika jumlah persimpangan/*crossing number* (cn) ganjil maka titik P dianggap di dalam. Sedangkan jika jumlah persimpangan genap maka titik P dianggap di luar poligon (Sunday, 2020)

Setiap sebuah proyeksi melewati garis poligon maka statusnya berubah jadi masuk (*in*) atau keluar (*out*). Maka dapat disimpulkan bahwa jika sebuah titik di dalam poligon maka urutan persimpangan proyeksinya adalah in > out > ... > in > out. Sedangkan jika di luar maka urutannya out > in > ... > in > out (Sunday, 2020). Urutan proyeksi ini digambarkan pada **Gambar 2.6.**



Gambar 2.6 Proyeksi *Crossing Number*

Algoritma *crossing number* secara langsung memilih proyeksi horizontal yang memanjang ke kanan P dan sejajar dengan sumbu x positif. Persimpangan menjadi lebih mudah dihitung dengan menggunakan proyeksi ini. Bahkan akan lebih mudah untuk menentukan kapan persimpangan semacamnya tidak mungkin terjadi. Untuk menghitung total penyeberangan, cn, algoritma hanya melakukan perulangan melalui semua tepi poligon, menguji masing-masing persimpangan, dan menambahkan cn ketika terjadi persimpangan. Selain itu, tes persimpangan harus menangani kasus dan titik khusus pada tepian. Hal ini dapat dipenuhi dengan menggunakan aturan berikut:

1. *Endpoint* akhir pada tepi atas diabaikan, dan *endpoint* awalnya tetap dimasukkan
2. *Endpoint* awal pada tepi bawah diabaikan, dan *endpoint* akhirnya tetap dimasukkan
3. Tepi horizontal diabaikan
4. Perpotongan antara tepian dan proyeksi harus di sebelah kanan titik *P* (Sunday, 2020).

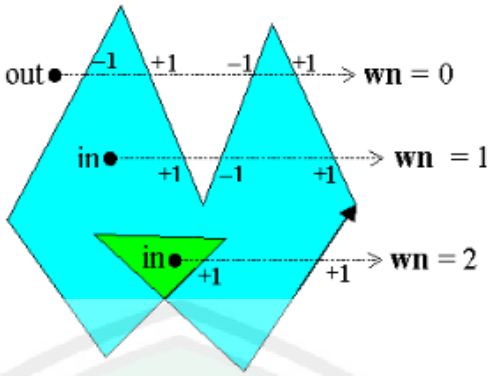
2.3.2 *Winding Number*

Banyaknya belitan/*winding number* (wn) secara akurat dapat menentukan apakah suatu titik berada di dalam suatu poligon tertutup yang tidak sederhana. Algoritma ini menghitung berapa kali poligon berputar di sekitar titik. Winding number (wn) didefinisikan sebagai berapa kali poligon ‘S’ berputar di sekitar titik ‘P’. Jika ‘P 'ada di dalam poligon ‘S’, maka wn bukan nol. Demikian pula, jika titik 'P' di luar, hanya ketika poligon tidak berputar di sekitar titik sama sekali yang berarti wn = 0. Angka belitan wn dari poligon 'S' dan dengan titik arbiter 'P' tidak

hanya mengukur seberapa 'S' melingkupi titik 'P', tetapi juga memberikan informasi tentang orientasi dan menghitungnya, berapa kali poligon 'S' ' membelit di sekitar titik ‘P’ (Kumar & Bangi, 2018).

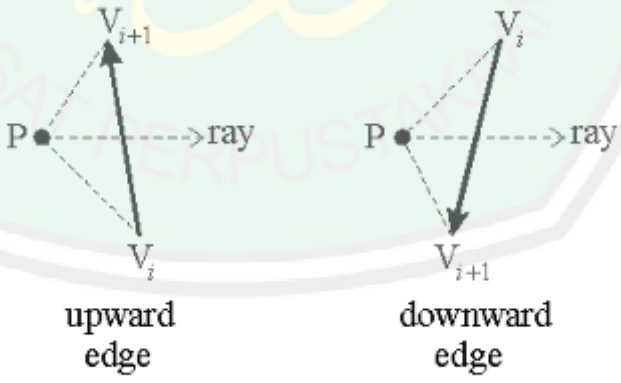
$$wn = \begin{cases} 0; & Jika\ P\ di\ luar\ S \\ n > 0; & Jika\ S\ membelit\ P\ searah\ jarum\ jam \\ n < 0; & Jika\ S\ membelit\ P\ berlawanan\ jarum\ jam \end{cases}$$

Secara umum, angka belitan wn (P, C) dapat ditentukan dari setiap kurva kontinu tertutup C di sekitar titik P di bidang 2D. Namun formula tersebut tidak efisien karena menggunakan fungsi trigonometri yang mahal secara komputasi. Pengamatan sederhana memungkinkan untuk mengganti formula ini dengan formula yang lebih efisien. Pilih titik Q pada S1. Kemudian, ketika kurva W (P) membungkus S1, ia akan melewati Q beberapa kali. Jika ketika melewati Q berlawanan arah jarum jam dihitung (+1), dan (–1) ketika melewati searah jarum jam, maka jumlah akumulasi persis jumlah total yang W (P) membungkus S1, dan sama dengan jumlah belitan wn (P, C). Lebih lanjut, jika proyeksi *infinite* R diambil mulai dari P dan memanjang ke arah vektor Q, maka persimpangan di mana R memotong kurva C sesuai dengan titik di mana W (P) melewati Q. Jika tepi memotong proyeksi positif dari bawah ke atas, persimpangan dianggap positif (+1); tetapi jika melintasi dari atas ke bawah, persimpangan dianggap negatif (–1). Langkah terakhir adalah menambahkan semua nilai persimpangan untuk mendapatkan wn (P, C). Adapun penggambarannya bisa dilihat pada **Gambar 2.7** (Sunday, 2020).



Gambar 2.7 Perhitungan *Winding Number*

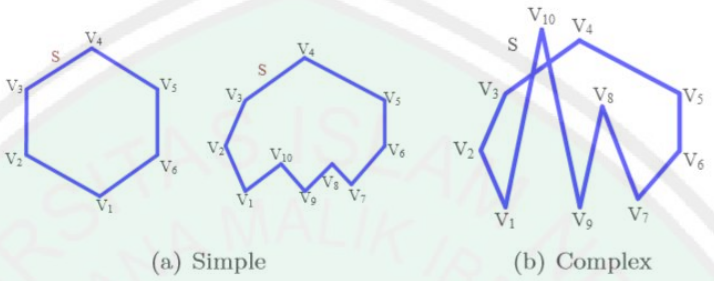
Perhitungan titik persimpangan tepi dengan proyeksi yang sebenarnya dapat dihindari dengan menggunakan atribut isLeft (). Namun, atribut ini perlu diterapkan secara berbeda untuk tepi naik dan turun. Jika tepi atas melewati proyeksi ke kanan P, maka P berada di sisi kiri tepi karena segitiga Vi V(i + 1) P berorientasi berlawanan arah jarum jam. Di sisi lain, jika tepi bawah melintasi proyeksi positif maka akan memiliki P di sisi kanan karena segitiga Vi V(i + 1) P berorientasi searah jarum jam (Sunday, 2020). Adapun penggambaran untuk mendapatkan atribut isLeft() terdapat pada **Gambar 2.8.**



Gambar 2.8 Perhitungan atribut isLeft()

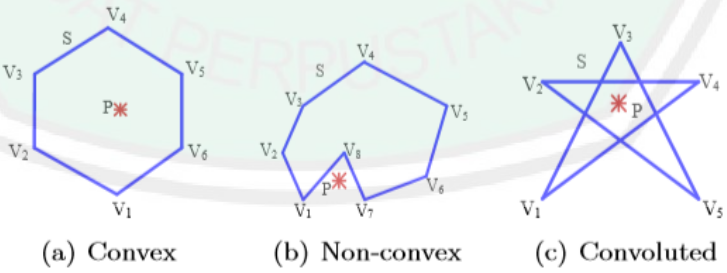
2.3.3 Poligon Classification

Poligon pada umumnya diklasifikasikan ke dalam dua kategori seperti pada **Gambar 2.9**. Sebuah poligon diklasifikasikan sederhana (*simple*) jika garis-garis yang menyusunnya tidak saling berpotongan. Sementara itu, sisi-sisi dari poligon kompleks (*complex*) saling memotong satu sama lain (Kumar & Bangi, 2018).



Gambar 2.9 Klasifikasi poligon

Tipe Poligon dapat dibagi menjadi *convex*, *non-convex*, dan *convoluted* seperti pada **Gambar 2.10**. Poligon *convex* adalah poligon yang semua sudut dalamnya kurang dari 180˚. Poligon dikategorikan *non-convex* jika setidaknya mempunya satu sudut dalam yang lebih dari 180˚. Poligon *convoluted* adalah bentuk bintang yang mempunyai beberapa sisi yang berpotongan sehingga membentuk satu lubang di dalamnya.



Gambar 2.10 Tipe-tipe poligon

2.4 Penelitian Terkait

Penelitian dengan menggunakan *geofence* pernah dilakukan sebelumnya oleh Puspa M. N. S. A. Basid pada tahun 2017 dengan judul *Pengembangan Sistem E-Complaint Berbasis Social Crowdsource Geotagging Studi Kasus Kota Malang*. Penelitian tersebut memanfaatkan *geofenced area* untuk membatasi wilayah pengguna yang dapat melapor pada sistem *e-complaint*. Algoritma yang dipakai adalah dengan menggunakan teknik *raycasting* dengan metode *crossing number/even odd*. Algoritma yang dibuat mampu memperoleh akurasi sebesar 94% dengan menggunakan 50 titik secara acak pada tahap pengujian (Basid, et al., 2017). Kelebihan pada penelitian ini adalah metode yang dipakai adalah mampu mendeteksi titik dalam *geofence* yang berbentuk poligon yang kompleks, sehingga cakupan wilayah yang bisa dibatasi dengan *geofence* lebih spesifik daripada menggunakan metode *circle geofence*. Kekurangan pada penelitian ini adalah kurangnya pengujian dengan menggunakan poligon yang berbeda jenis seperti *convex* dan *convoluted*. Kekurangan lain adalah titik-titik pada poligon yang dipakai harus sudah ditentukan sebelumnya, tidak terdapat tampilan untuk menggambar poligon secara dinamis yang memungkinkan pengguna mengubah wilayah secara bebas.

Penerapan konsep *geofencing* juga dilakukan oleh Afrizal Fath Rahman pada tahun 2017 dalam *Rancang Bangun Aplikasi Geofence Marketing Cafe Berbasis Android Studi Kasus: Ice Ah!*. Penelitian ini menggunakan metode *circular geofencing* untuk sistem *marketing* dengan informasi yang disampaikan berbentuk notifikasi. Pengujian yang dilakukan menghasilkan 6 skala dengan nilai *Attractiveness* sebesar 1.45, *Perspicuity* sebesar 1.725, *Efficiency* sebesar 1.4,

Dependability sebesar 1.275, *Stimulation* sebesar 1.4375 dan *Novelty* sebesar 1.0125. (Rahman, 2017). Kelebihan pada penelitian ini adalah karena *geofence* yang dipakai berjenis *circular*, maka proses komputasi akan lebih ringan dibandingkan dengan *geofenced area* dan pengguna dimudahkan dengan hanya perlu memasukkan titik tengah dan radius yang diinginkan. Kekurangan pada penelitian ini adalah metode yang dipakai hanya dapat mendeteksi titik di dalam area lingkaran, sehingga untuk mencakup target yang kompleks seperti wilayah kampung tertentu metode ini belum memadai. Kekurangan lainnya adalah kurangnya pengujian terhadap performa algoritma yang dipakai, penelitian ini lebih berfokus pada pengujian *user experience*.

Roly Segara pada penelitiannya tahun 2017 yang berjudul *Sistem Pemantauan Lokasi Anak Menggunakan Metode Geofencing pada Platform Android*, memanfaatkan *geofence* sebagai perimeter virtual pada wilayah geografis yang menggunakan layanan berbasis lokasi dan digunakan untuk pembatasan wilayah pengawasan anak. Sistem yang dibuat akan mengirimkan notifikasi ke perangkat orang tua jika perangkat anak keluar dari area *geofence* (Segara & Subari, 2017). Kelebihan pada penelitian ini anatara lain: metode yang dipakai sudah menggunakan *geofence* berjenis *geofenced area* sehingga area yang dibatasi lebih spesifik, dan pengguna dapat mengubah *geofence* yang dipakai secara dinamis pada tampilan yang disediakan dalam sistem yang dibuat. Kekurangan pada penelitian adalah kurangnya penjelasan pada metode yang dipakai, penelitian ini menggunakan *geofenced area* namun tidak dijelaskan bagaimana cara mendeteksi titik dalam poligon *geofence*, pengujian performa metode yang digunakan juga tidak dilakukan sehingga akurasi dan kecepatannya tidak diketahui.

Algoritma *geofencing* digunakan oleh Yoga Resta H. pada penelitiannya yang berjudul *Development of Blood Donor Application Using Geofencing and Firebase Technology on Android Platform* pada tahun 2019. Pada penelitian ini, jenis *geofence* yang digunakan adalah *circle geofence* dengan metode *Haversine*. Hasil penelitian ini adalah sistem untuk mempermudah pasien mencari donor darah dan mengetahui kegiatan donor darah dan stok darah yang berada di PMI Kota Bandung. (Handayanto & Dewi, 2019). Kelebihan pada penelitian ini adalah dengan *circle geofence* maka proses komputasi yang dilakukan relatif ringan. Kekurangan pada penelitian ini adalah area yang dipakai hanya terbatas berbentuk lingkaran sehingga tidak dapat mencakup pada wilayah yang spesifik. Kekurangan lain adalah tidak ada pengujian performa algoritma yang digunakan.

Haditya Dwi Perkasa dalam penelitian yang berjudul *Pembangunan Aplikasi Bandung near You Untuk Pemberitahuan Tempat-Tempat Wisata Di Kota Bandung Memanfaatkan Teknologi Geofence Studi Kasus DISBUDPAR Kota Bandung*, memanfaatkan algoritma *geofencing* untuk untuk menemukan tempat wisata dan memberikan rekomendasi tempat wisata terdekat di Kota Bandung. Peneliti mengandalkan fitur GPS A-GPS di perangkat *mobile* dalam aplikasi yang dibuat (Haditya, 2017). Kelebihan pada penelitian ini adalah penggunaan aplikasi pihak ketiga memungkinkan proses pengembangan lebih cepat tanpa mengurangi fungsionalitasnya. Kekurangan pada penelitian ini adalah digunakannya aplikasi pihak ketiga mengakibatkan kustomisasi dan performa yang terbatas bergantung pada pengembang aplikasi pihak ketiga tersebut.

Januar Irawan dalam skripsinya yang berjudul *Penerapan Absen Mahasiswa Berbasis Android Menggunakan Teknologi Or Code dan Geofence*

(Studi Kasus: Ti Uin Syarif Hidayatullah Jakarta) menerapkan algoritma *geofencing* untuk membuat aplikasi absensi mahasiswa. Kelebihan pada penelitian ini adalah *geofence* dapat meningkatkan keamanan sistem yang dibuat, namun jenis *geofence* yang digunakan adalah *circle geofence* sehingga mempunyai kekurangan yaitu *geofence* tidak bisa mencakup gedung secara khusus. *Geofence circle*/lingkaran akan memungkinkan area di luar gedung ikut tercakup (Irawan, 2018).

M. Romdon Nurdin S dalam jurnal yang berjudul *Pembangunan Aplikasi Appedestrian Untuk Meningkatkan Jumlah Pejalan Kaki Memanfaatkan Teknologi Geofencing dan Fitur Gamification* memanfaatkan algoritma *geofence* untuk memberi informasi promosi suatu tempat usaha di sekitar kepada pengguna. Pada penelitian ini penulis menggunakan *circle geofence* dengan bantuan algoritma Haversine untuk menghitung radius pengguna dengan target lokasi. Penggunaan *circle geofence* berpengaruh terhadap area yang dibatasi tidak bisa berbentuk spesifik (Nurdin, 2018). Seperti pada penelitian-penelitian sebelumnya yang menggunakan *circle geofence*, metode yang dipakai mempunyai kelebihan proses komputasi yang lebih ringan dengan *input* lebih sederhana, namun memiliki kekurangan pada wilayah cakupan yang tidak spesifik. Penelitian ini juga tidak menyertakan pengujian performa pada algoritma yang dipakai.

2.5 Deliv

Deliv adalah sebuah aplikasi *mobile* berbasis Android yang memberikan layanan ojek dan pesan antar atau *delivery* order makanan, minuman, dan jajanan khas daerah. Aplikasi Deliv dapat didapatkan secara gratis di Google Play Store. Deliv menghubungkan para penggunanya dengan restoran-restoran dan tukang ojek yang telah menjadi mitranya. Fitur andalan Deliv antara lain Deride (pemesanan

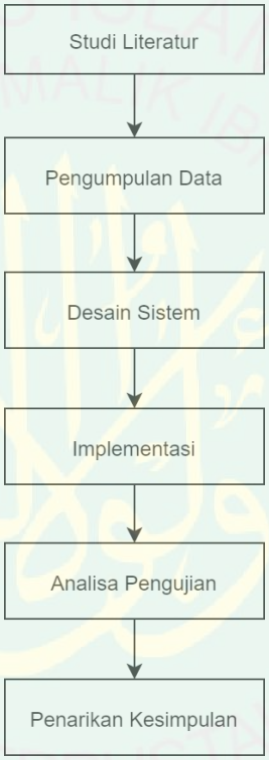
ojek *online*), Decar (pemesanan taksi *online*), dan Defood (pemesanan makanan *online*). Deliv dibuat oleh PT. Deliv Tehnologi Indoraya pada tahun 2015 dan beroperasi di Kota Malang dan Kabupaten Bojonegoro dengan kantor pusat yang terletak di Villa Sengkaling Blok T No. 4, Malang, Jawa Timur (Hidayat, 2019).

BAB III

METODOLOGI PENELITIAN

3.1 Desain Penelitian

Penelitian ini dilakukan dengan enam tahap secara berurutan. Adapun tahap yang dilakukan pada penelitian ini adalah studi literatur, pengumpulan data, desain sistem, implementasi, analisa pengujian, dan penarikan kesimpulan. Langkah-langkah penelitian telah dimuat dalam desain penelitian pada **Gambar 3.1**.

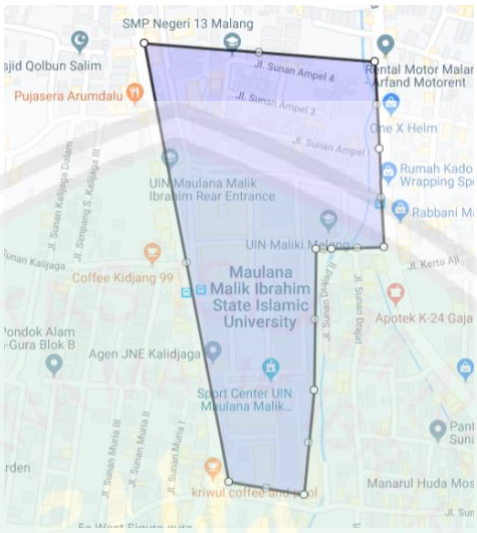


Gambar 3.1 Desain Penelitian

3.2 Pengumpulan Data

Pengumpulan data pada penelitian ini dilakukan untuk mendapatkan data sebagai acuan untuk proses analisa pengujian. Data yang dikumpulkan adalah data spasial dari lokasi target promosi. Lokasi target promosi yang akan diuji pada penelitian ini adalah area poligon yang digambar manual menggunakan aplikasi

administrator yang dibuat. Data spasial tersebut kemudian akan diekstrak menjadi *geofence* pada sistem yang dibuat. Adapun contoh data yang didapatkan adalah dari *poligon* pada **Gambar 3.2** dapat diperoleh data spasial pada **Tabel 3.1**.



Gambar 3.2 *Poligon* UIN Maulana Malik Ibrahim Malang

Node	Latitude	Longitude
1	-7.94885	112.60609
2	-7.94906	112.60879
3	-7.95007	112.60884
4	-7.95121	112.60889
5	-7.95124	112.60829
6	-7.95124	112.60810
7	-7.95287	112.60808
8	-7.95408	112.60796
9	-7.95393	112.60708

Tabel 3.1 Data Spasial UIN Maulana Malik Ibrahim Malang

Data yang didapatkan kemudian akan diubah menjadi format *Javascript Object Notation* (JSON). Tujuan pengubahan data ke format JSON adalah supaya data dapat dikirim ke REST API yang dibuat melalui *form body*. Terdapat dua *filed* yang dikirimkan yaitu *name* berisi nama area dan *area* berisi daftar koordinat

latitude longitude yang menyusun *poligon* area. Data yang telah diubah ke dalam bentuk JSON dapat dilihat pada **Gambar 3.3**.

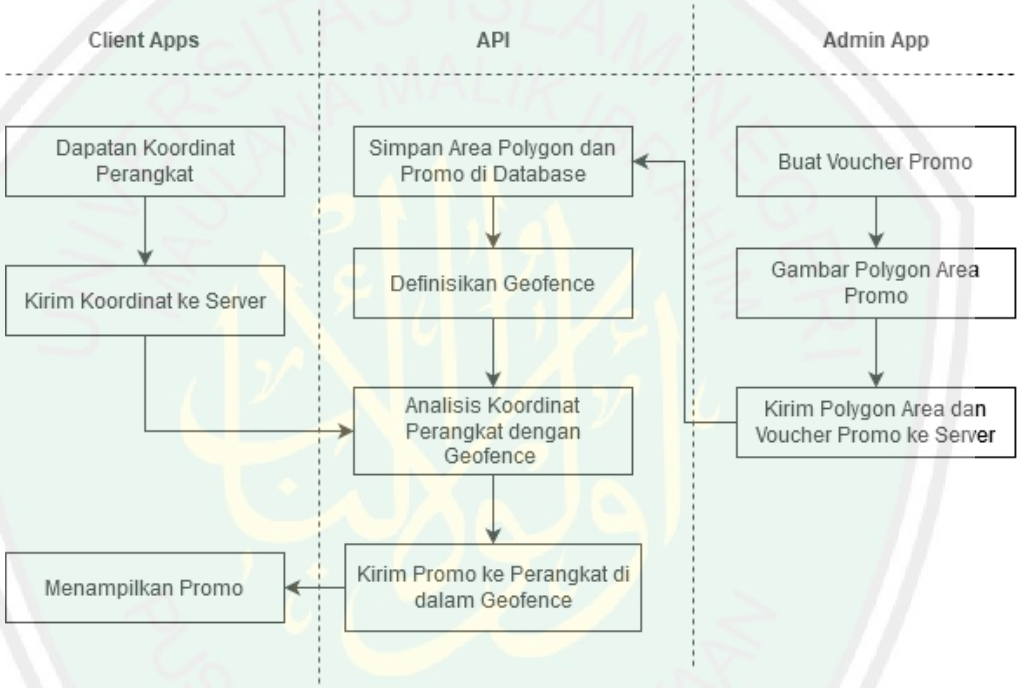
Gambar 3.3 Contoh Data JSON Area

3.3 Desain Sistem

Sistem yang akan dibuat pada penelitian ini terdiri dari tiga perangkat lunak, yaitu aplikasi *client*, *webservice* API, dan aplikasi admin. Pada aplikasi *client* terjadi proses untuk mendapatkan koordinat dari perangkat yang kemudian akan dikirim ke server. Pada aplikasi admin terdapat proses untuk membuat promo, menggambar poligon area promo, dan mengirimkan poligon area dan data promo ke server. *Webservice* API akan melakukan proses menyimpan area poligon dan data promo ke *database*, mendefinisikan *geofence*, analisa koordinat perangkat dengan geofencing, dan melakukan *broadcast* promo ke perangkat yang ada di dalam geofence. Semua proses yang terjadi pada sistem yang akan dibuat telah dimuat dalam desain sistem pada **Gambar 3.4**.

Jenis teknik *geofencing* yang digunakan pada penelitian ini adalah *geofenced area*. Teknik ini dipakai karena target promosi adalah area berbentuk

khusus menyesuaikan dengan tempat yang akan dipilih menjadi target. Bentuk lingkaran tidak dipakai karena akan memperbesar kemungkinan didapatkannya target promosi yang tidak sesuai, oleh karena itu bentuk yang dapat mencakupnya adalah poligon. Penentuan keputusan apakah titik pengguna termasuk ke dalam poligon atau tidak dengan menggunakan algoritma *crossing number* dan *winding number*. Kedua algoritma tersebut akan diuji pada tahap pengujian untuk menentukan algoritma yang paling cepat dan akurat.



Gambar 3.4 Desain Sistem

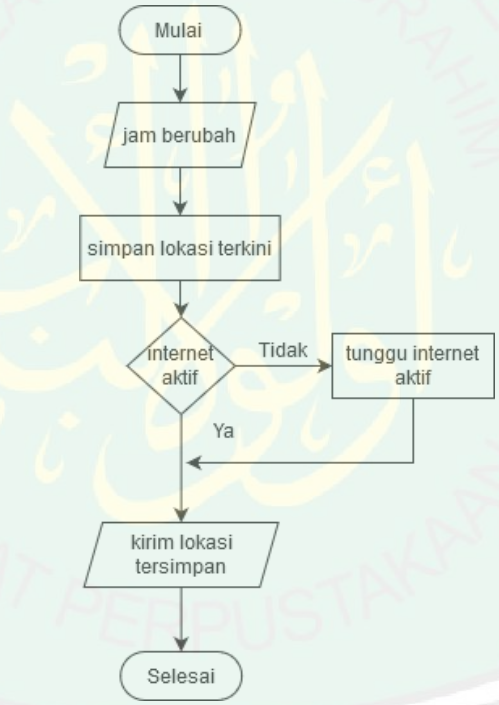
3.3.1 Mendapatkan Koordinat Perangkat

Koordinat pada perangkat Android dapat didapatkan dengan memanfaatkan Google Play Service yang sudah tertanam pada semua perangkat Android resmi. Koordinat yang akan diambil adalah titik *latitude* (garis lintang) dan titik *longitude* (garis bujur). Aplikasi *client* akan menyimpan lokasi pengguna secara rutin setiap jam dan jika internet pada perangkat aktif maka data lokasi tersimpan akan segera di kirim ke server sehingga ditemukan variabel waktu pada tabel 3. Tabel variabel

waktu tersebut akan mewakili data lokasi pengguna setiap harinya dan akan disimpan pada *database* sistem. Algoritma pengambilan koordinat perangkat dimuat pada **Gambar 3.5**.

T1	01:00
T2	02:00
T3	03:00
T4	04:00
...	...
T21	21:00
T22	22:30
T23	23:00
T24	00:00

Tabel 3.2 Variabel waktu pengambilan lokasi

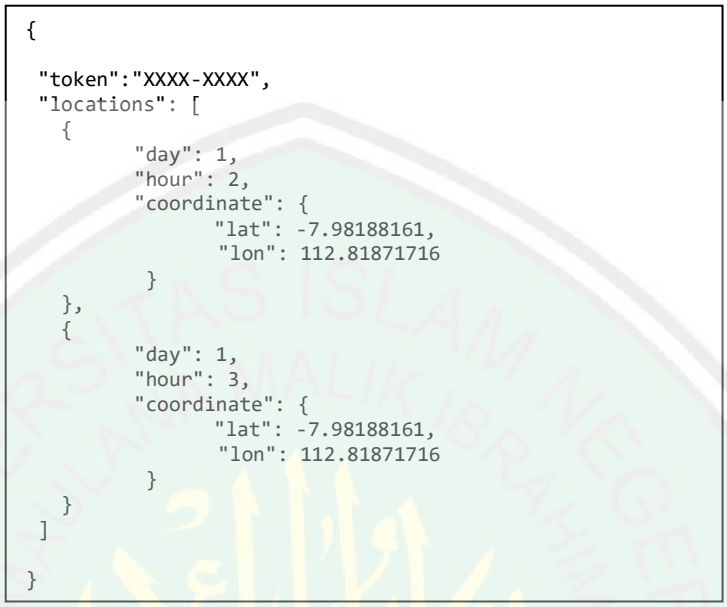


Gambar 3.5 *Flowchart* Pengambilan Koordinat Perangkat

3.3.2 Mengirim Koordinat Perangkat

Koordinat perangkat akan dikirimkan ke server setelah ditemukan titik tengah melalui tahap sebelumnya. Koordinat akan dikirimkan bersamaan dengan data hari dan jam koordinat tersebut didapatkan dan token *session*. Token

didapatkan pengguna melalui proses log in melalui aplikasi *client*. Data koordinat yang dikirim pada proses pengiriman perangkat adalah data berformat JSON. Adapun contoh data yang dikirimkan dapat dilihat pada **Gambar 3.6**.



```
{
 "token":"XXXX-XXXX",
 "locations": [
   {
          "day": 1,
          "hour": 2,
          "coordinate": {
                 "lat": -7.98188161,
                 "lon": 112.81871716
           }
   },
   {
          "day": 1,
          "hour": 3,
          "coordinate": {
                 "lat": -7.98188161,
                 "lon": 112.81871716
           }
   }
 ]

}
```


Gambar 3.6 Contoh Data Pengiriman Lokasi JSON

3.3.3 Membuat *Voucher* Promo

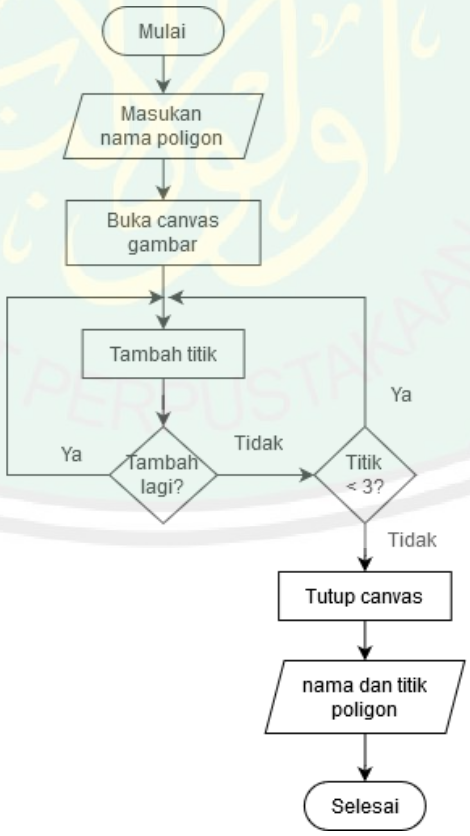
Voucher promo memuat informasi detail promosi yang akan dikirimkan kepada pengguna. *Voucher* promo memuat nama promosi, jenis promosi, jumlah potongan promosi, deskripsi promosi, lokasi promosi, kode promosi dan masa berlaku promosi. Lokasi pada promo bisa lebih dari 1 area, sehingga digunakan daftar ID area untuk mewakilinya. *Voucher* promo akan dibuat melalui aplikasi *admin* dan kemudian akan dikirimkan ke server dalam bentuk format JSON. Adapun contoh data promo dapat dilihat pada **Gambar 3.7**.

```
{

 "name": "Promo Mahasiswa UIN",
 "type": "DISCOUNT",
 "serviceType": "DERIDE",
 "value": 10,
 "startDate": "2020-01-01"
 "endDate": "2020-01-31",
 "description": "Promo ojek diskon 50% khusus mahasiswa UIN Malang"
 "code": "MHSUIN",
 "areaIds": [ 1, 2]
}
```

Gambar 3.7 Contoh Data Promo JSON

3.3.4 Menggambar Poligon *Geofence*

Penelitian ini menggunakan *geofence* berjenis poligon yang dapat dibuat secara bebas di aplikasi admin. Poligon yang digambar terdiri dari banyak titik-titik koordinat yang dihubungkan dengan garis lurus satu sama lain. Poligon dapat dibuat dengan setidaknya tiga titik yang menyusunnya. Adapun algoritma penggambaran poligon *geofence* dimuat pada **Gambar 3.8**.

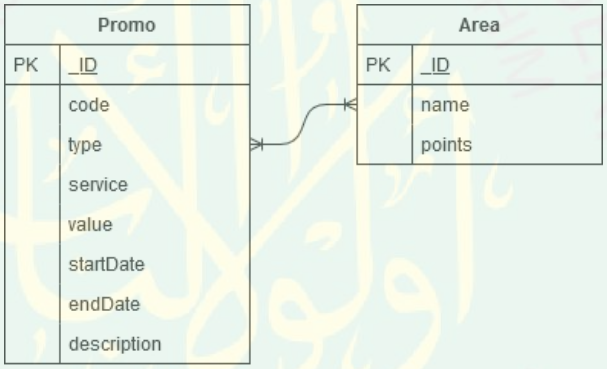


Gambar 3.8 *Flowchart* Penggambaran Poligon *Geofence*

Poligon yang selesai dibuat selanjutnya akan diambil titik-titik koordinatnya. Titik-titik koordinat yang terkumpul akan dikirimkan beserta nama area ke dalam bentuk JSON ke server. Adapun contoh data poligon yang dikirimkan ke server ada pada **Gambar 3.3**.

3.3.5 Menyimpan Poligon dan *Voucher* Promo ke *Database*

Voucher promo dan poligon akan disimpan ke *database* dengan kondisi setiap promo bisa diterapkan di banyak area poligon dan setiap poligon dapat diterapkan di banyak *voucher* promo. Menurut kondisi tersebut maka poligon dan *voucher* promo akan disimpan kedua tabel dengan relasi *many-to-many*. Adapun ERD tabel area poligon dan *voucher* promo dapat dilihat pada **Gambar 3.9**.

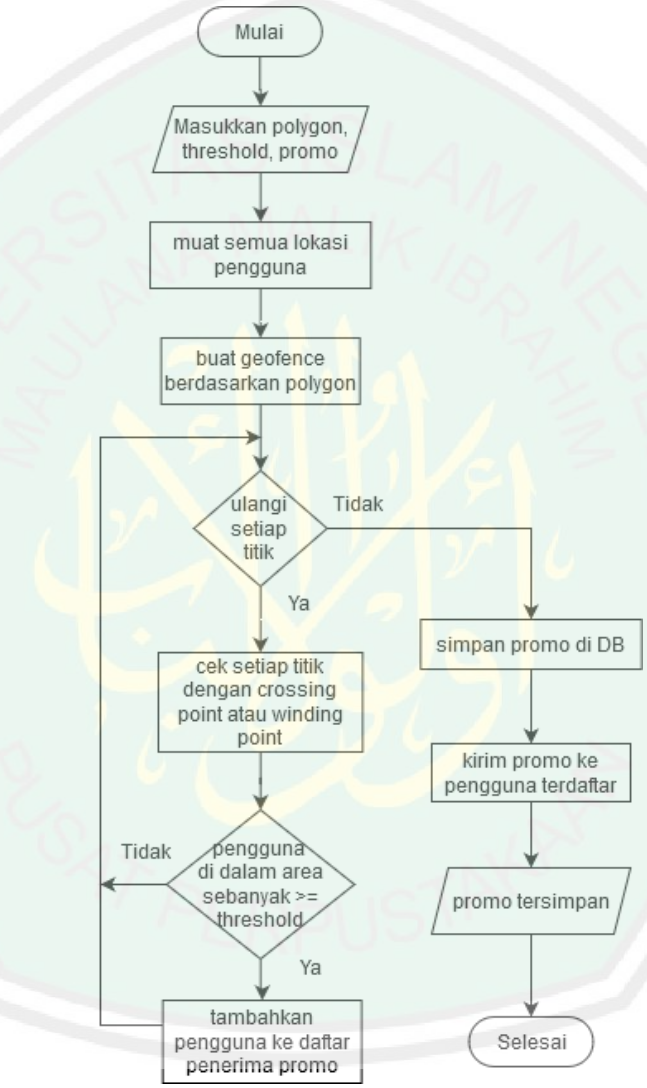


Gambar 3.9 ERD *Voucher* Promo dan Area

3.3.6 Analisis Koordinat Perangkat dengan *Geofencing*

Tahap *geofencing* adalah tahap inti pada sistem yang dibuat. Tahap ini menentukan apakah pengguna termasuk di dalam area atau di luar area. Proses ini dilakukan di server setelah data lokasi dan pengguna terkumpul. Aplikasi admin akan mengirimkan data promo, area poligon, dan *threshold*. *Threshold* pada tahap ini adalah jumlah minimal pengguna berada dalam area poligon dalam 7 hari terakhir. Lokasi pengguna/*point* yang tersimpan di *database* akan dihitung menggunakan metode *crossing number* atau *winding number* terhadap area poligon

yang dikirimkan. Setelah semua pengguna yang memasuki area ditemukan, maka langkah selanjutnya adalah mendaftarkan semua pengguna pada promo yang ditemukan, kemudian server akan mengirimkan promo ke semua perangkat pengguna tersebut. Adapun keseluruhan algoritma pada tahap *geofencing* dapat dilihat pada **Gambar 3.10**.



Gambar 3.10 *Flowchart* Analisis Koordinat Perangkat dengan *Geofencing*

3.3.7 *Point Inclusion* dengan *Crossing Number*

Pada tahap ini *point* (*latitude* dan *longitude*) yang dimasukkan oleh pengguna akan dicek apakah masuk ke dalam poligon di *database*. Metode *crossing*

number menentukannya dengan menghitung jumlah perpotongan (cn) proyeksi dari titik yang dimasukkan ke arah tepi poligon. Perhitungan *crossing number* yang digunakan pada penelitian ini adalah dengan menerapkan algoritma di bawah (O'Rourke, 1998).

```
definisikan {
        Point adalah koordinat dengan titik x, y integer
        Polygon adalah array dari Point
}

cn_Poly(Point p, Polygon poly)
{
        int cn = 0;   //variabel untuk menyimpan jumlah cn

        //perulanagan ke setiap sisi poly
        for(setiap sisi pada poly){
                if(sisi[i] memotong ke atas memenuhi aturan #1
                || sisi[i] memotong ke bawah memenuhi aturan #2
                ){
                        // perpotongan harus di kanan memenuhi aturan #4
                        if(p.x < x perpotongan antara sisi[i] dan y=P.y )
                                ++cn; //perpotongan yang valid

                }

        }

        return (cn&1);
        // kembali nilai 0 jika genap (di luar), dan 1 jika ganjil (di
dalam)

}
```

Berdasarkan *pseudocode* di atas dapat disimpulkan langkah-langkah untuk menentukan *point inclusion* dengan metode *crossing number*.

1. Point P direpresentasikan sebagai objek dengan 2 atribut x dan y, poligon direpresentasikan sebagai *array* Point V[n+1] dengan V[0] = V[n].
2. cn diinisiasi sebagai variabel untuk menyimpan jumlah perpotongan dengan nilai awal 0.
3. Lakukan *looping* untuk semua tepi (*edge*) pada poligon.
4. Jika tepi memotong ke atas sesuai aturan 1 (*endpoint* akhir pada tepi atas diabaikan, dan *endpoint* awalnya tetap dimasukkan) atau jika tepi

memotong ke bawah sesuai aturan 2 (*endpoint* awal pada tepi bawah diabaikan, dan *endpoint* akhirnya tetap dimasukkan) maka;

5. Jika nilai x pada *point* kurang dari nilai x pada perpotongan antara proyeksi y=P.y dengan tepi sesuai aturan 4 (perpotongan antara tepian dan proyeksi harus tetap di sebelah kanan titik *P*), maka tambahkan nilai cn.
6. Lanjutkan langkah 4 dan 5 hingga semua tepi dicek
7. Jika cn genap maka *point* P dianggap di luar dan jika cn ganjil maka *point* dianggap di dalam.

3.3.8 *Point Inclusion* dengan *Winding Number*

Metode *winding number* menentukan apakah titik yang dimasukkan oleh pengguna masuk di dalam poligon dengan menghitung berapa kali garis poligon membelit titik. Banyaknya belitan tersebut disebut *winding number* (wn). Perhitungan *winding number* yang diterapkan pada penelitian ini adalah dengan menerapkan algoritma berikut (Sunday, 2020).

```
definisikan {
    Point adalah koordinat dengan titik x, y integer
    Poligon adalah array dari Point
}

wn_Poly(Point p, Poligon poly)
{
    int wn = 0;   //variabel untuk menyimpan jumlah wn
    for(setiap sisi pada poly){

            if(sisi[i] memotong keatas memenuhi aturan #1 ){

                    if(p berada di kiri sisi[i]) //memenuhi aturan #4
                            ++wn; // perpotongan keatas yang valid

            }else if(sisi[i] memotong kebawah memenuhi aturan #2){
                    if(p berada di kanan sisi[i]) // memenuhi aturan #4
                            --wn; // perpotongan kebawah yang valid

            }

    }

    return wn; // jika wn tidak = 0  maka p di dalam poly

}
```

Algoritma di atas mengadaptasi algoritma *crossing mumber* sebelumnya dengan mengganti tahap penambahan angka perpotongan menjadi penambahan jika perpotongan di kanan, dan pengurangan jika perpotongan di kiri. Berdasarkan *pseudocode* di atas dapat disimpulkan langkah-langkah untuk menentukan *point inclusion* dengan metode *winding number*.

1. *Point* (P) direpresentasikan sebagai objek dengan 2 atribut x dan y, poligon direpresentasikan sebagai *array* Point V[n+1] dengan V[0] = V[n].
2. wn diinisiasi sebagai variabel untuk menyimpan jumlah belitan dengan nilai awal 0.
3. Lakukan *looping* untuk semua tepi (*edge*) pada poligon.
4. Jika tepi memotong ke atas sesuai aturan 1 (*endpoint* akhir pada tepi atas diabaikan, dan *endpoint* awalnya tetap dimasukkan) dan jika point P berada di kiri dari tepi yang dicek maka tambah nilai wn.
5. Jika tepi memotong ke bawah sesuai aturan 2 (*endpoint* awal pada tepi bawah diabaikan, dan *endpoint* akhirnya tetap dimasukkan) jika *point* P berada di kanan dari tepi yang dicek maka kurangi nilai wn.
6. Lanjutkan langkah 4 dan 5 hingga semua tepi dicek.
7. Jika wn bernilai 0 maka *point* P dianggap di luar, selain itu maka dianggap di dalam.

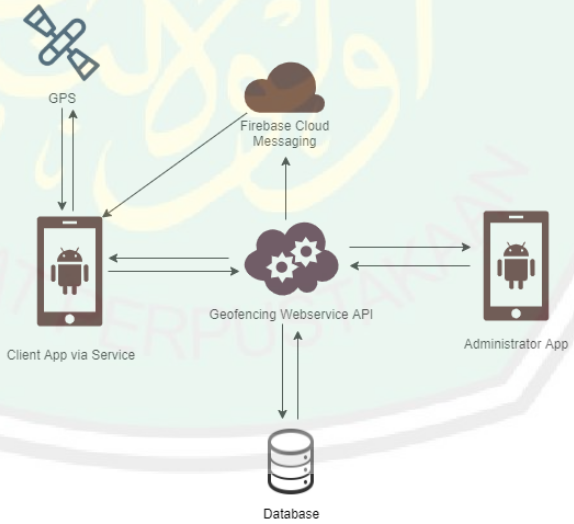
3.4 Desain Implementasi

Implementasi dilakukan berdasarkan desain sistem yang telah dibuat sebelumnya. Implementasi meliputi membangun sistem aplikasi yang dapat menjalankan semua proses yang telah dijabarkan pada desain sistem dan

menjalankannya di perangkat fisik. Adapun aplikasi yang akan dibangun antara lain:

1. Service pada aplikasi *client* berbasis Android yang bertugas mendapatkan lokasi dari GPS, mengirim lokasi pengguna ke server dan menerima *broadcast* promo. Service ini berjalan di latar belakang.
2. Aplikasi *dashboard admin* berbasis Android yang dilengkapi *user interface* untuk mengelola promo dan menggambar *geofence*.
3. Webservice API Berbasis Java Springboot yang bertugas untuk mendapatkan target promo menggunakan algoritma *geofencing*, mengelola *database*, dan melakukan *broadcast* promo ke aplikasi *client* menggunakan Firebase Cloud Messaging (FCM).

Tiga aplikasi tersebut saling terhubung menjadi suatu sistem dengan alur komunikasi yang tergambar pada **Gambar 3.11**.



Gambar 3.11 Alur Komunikasi Sistem

3.4.1 Desain Antarmuka Aplikasi Administrator

Terdapat lima halaman pada aplikasi administrator, yaitu tampilan *log in*, beranda, buat promo, buat area, dan kelola area. Semua tampilan tersebut dibuat

untuk memenuhi fungsi aplikasi admin dalam mengelola promo dan area *geofence*. Berikut desain antarmuka aplikasi administrator.

1. Halaman beranda

Halaman beranda mempunyai desain seperti pada **Gambar 3.12**. Halaman ini berfungsi untuk menampilkan statistik sistem dan kontrol utama untuk menuju halaman lain.

Gambar 3.12 Desain Antarmuka Beranda pada Aplikasi Admin

Berikut penjelasan untuk **Gambar 3.12.**

a. *Header*

Bagian *header* berisi informasi nama akun admin yang masuk.

b. Statistik promo

Statistik promo memuat informasi tentang jumlah promo yang sedang berjalan, jumlah pengguna yang mendapatkan promo, dan jumlah area yang terdaftar.

c. Daftar promo berjalan

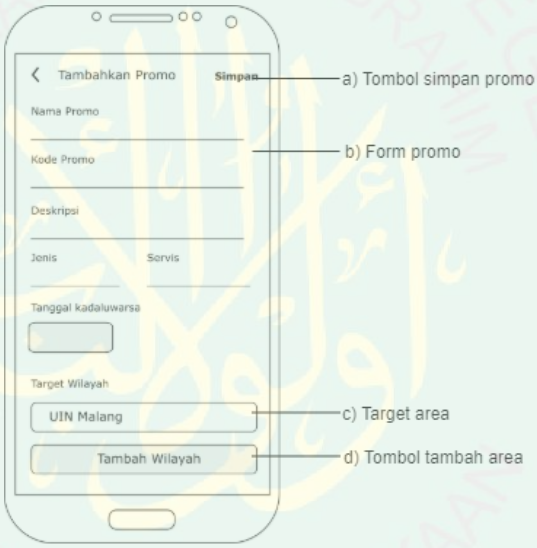
Daftar ini berisi semua promo yang sedang berjalan disertai dengan jumlah lokasi yang tercakup dalam promo tersebut.

d. Tombol tambah promo

Tombol ini berfungsi untuk membuka halaman membuat promo baru.

2. Halaman buat promo

Halaman buat promo mempunyai desain seperti pada **Gambar 3.13**. Halaman ini berfungsi untuk membuat *voucher* promo baru yang berisi nama promosi, jenis promosi, jumlah potongan promosi, deskripsi promosi, lokasi promosi, dan masa berlaku promosi.



Gambar 3.13 Desain Antarmuka Buat Promo pada Aplikasi Admin

Berikut penjelasan untuk **Gambar 3.13.**

a. Tombol simpan promo

Tombol ini berfungsi untuk menyimpan promo setelah semua formulir promo terisi.

b. Formulir promo

Formulir ini terdiri dari tujuh isian yang wajib diisi, yaitu nama promo, jenis promo (*discount/price*), tanggal kadaluwarsa, deskripsi promo, dan kode promo.

c. Target area

Target area berisi daftar area yang dipilih untuk penyebaran promo.

d. Tombol tambah area

Tombol ini berfungsi untuk menambahkan target area pada promo dengan menampilkan halaman pilih area.

3. Halaman pilih area

Halaman pilih area mempunyai desain seperti pada **Gambar 3.14**. Halaman ini berfungsi untuk menampilkan pilihan area yang tersedia dan mengelola area seperti menghapus dan membuat area baru.

Gambar 3.14 Desain Antarmuka Pilih Area pada Aplikasi Admin

Berikut penjelasan untuk **Gambar 3.14.**

a. Daftar area

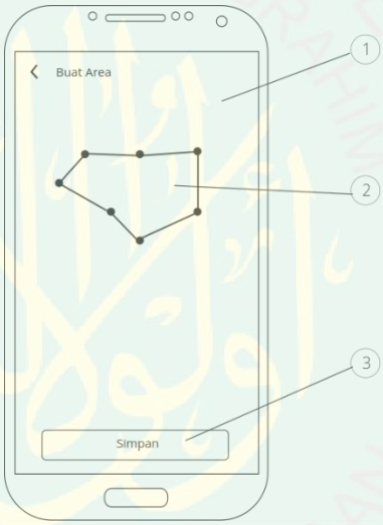
Daftar ini berisi semua area yang tersedia di *database* server. Setiap item pada daftar ini dilengkapi dengan tombol silang yang berfungsi untuk menghapus area tersebut.

b. Tombol tambah area

Tombol ini berfungsi untuk menampilkan halaman buat area baru.

4. Halaman buat area

Halaman pilih area mempunyai desain seperti pada **Gambar 3.15**. Halaman ini berfungsi untuk membuat poligon area baru sebagai *geofence* pada sistem yang dibuat.



Gambar 3.15 Desain Antarmuka Buat Area pada Aplikasi Admin

Berikut penjelasan untuk **Gambar 3.15.**

a. Palet area

Palet area adalah tampilan Google Maps SDK Android yang direkayasa sedemikian rupa sehingga dapat digunakan untuk menggambar poligon melalui *input* pengguna.

b. Tombol simpan

Tombol ini berfungsi untuk menyimpan poligon area yang telah dibuat.

3.5 Desain Pengujian

Tahap pengujian bertujuan untuk mendapatkan data sebagai pertimbangan penarikan kesimpulan. Data yang diperlukan adalah akurasi dan kecepatan sistem dalam mengidentifikasi lokasi pengguna menggunakan algoritma *geofencing* dengan teknik *crossing number* dan *winding number*. Nilai akurasi dan kecepatan dapat dihitung dengan rumus berikut:

$$akurasi = \frac{total\ hasil\ benar}{total\ percobaan} \times 100\% \quad \text{.....(Persamaan 1)}$$

$$kecepatan = \frac{total\ waktu\ yang\ dihabiskan}{total\ \ percobaan} \quad \text{.....(Persamaan 2)}$$

Pengujian yang dilakukan dibagi menjadi 2 jenis yaitu pengujian algoritma dan pengujian sistem. Setiap pengujian membutuhkan data uji yang tertentu sesuai dengan tujuan pengujian. Berikut pengujian dan pengumpulan data uji yang dilakukan pada penelitian ini:

3.5.1 Data Uji

Data yang digunakan adalah data acak yang dihasilkan dari parameter tertentu saat pengujian. Jenis data yang akan digunakan ada 2 yaitu *circular random data* dan *walking random data*. Masing-masing jenis data pengujian tersebut sama-sama menggunakan perhitungan titik *random* pada area lingkaran tertentu. Titik *random* (x2, y2) antara suatu titik pusat lingkaran (x1, y1) dengan radius tertentu (r) ditentukan dengan rumus berikut (R adalah angka desimal acak antara 0 hingga 1):

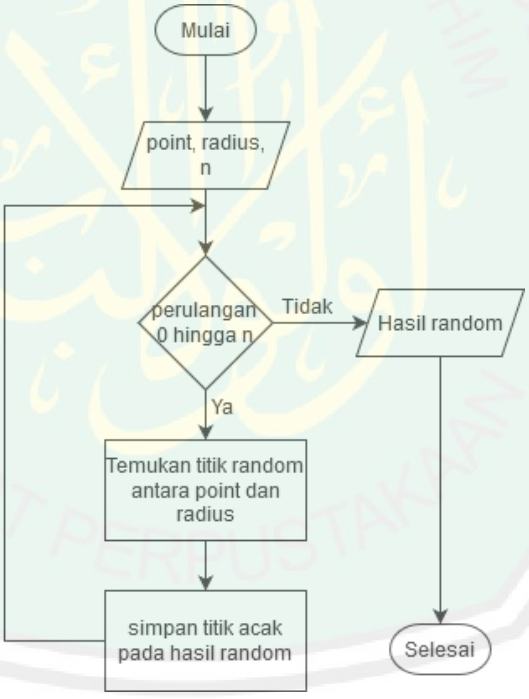
$$x2 = x1 + \left(r\ .\ \sqrt{R}\ .\ \cos(R\ .\ 2\ \pi)\right) \text{.....} \quad \text{(Persamaan 3)}$$

$$y2 = y1 + \left(r\ .\ \sqrt{R}.\ \sin(R\ .\ 2\ \pi)\right) \text{.....} \quad \text{(Persamaan 4)}$$

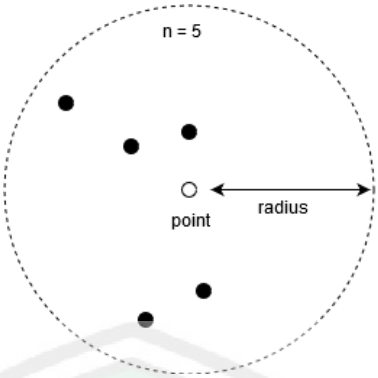
Titik *random* yang dihasilkan pada perhitungan di atas kemudian disusun menjadi satu set titik *circular random data* dan *walking random data*, berikut langkah-langkahnya:

1. *Circular Random Data*

Data jenis ini adalah kumpulan titik-titik koordinat acak pada wilayah dengan radius tertentu. Algoritma untuk menghasilkan data ini dapat dilihat pada **Gambar 3.16**. *Point* adalah titik tengah dari wilayah yang dipilih, radius adalah jari-jari area lingkaran yang akan dibuat, dan n adalah jumlah data yang diinginkan. Bentuk data yang dihasilkan digambarkan dengan titik-titik hitam pada **Gambar 3.17.** Data ini akan dipakai sebagai *input* pada pengujian algoritma.



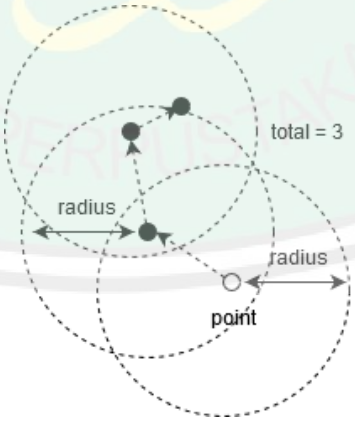
Gambar 3.16 *Flowchart Circular Random Data*



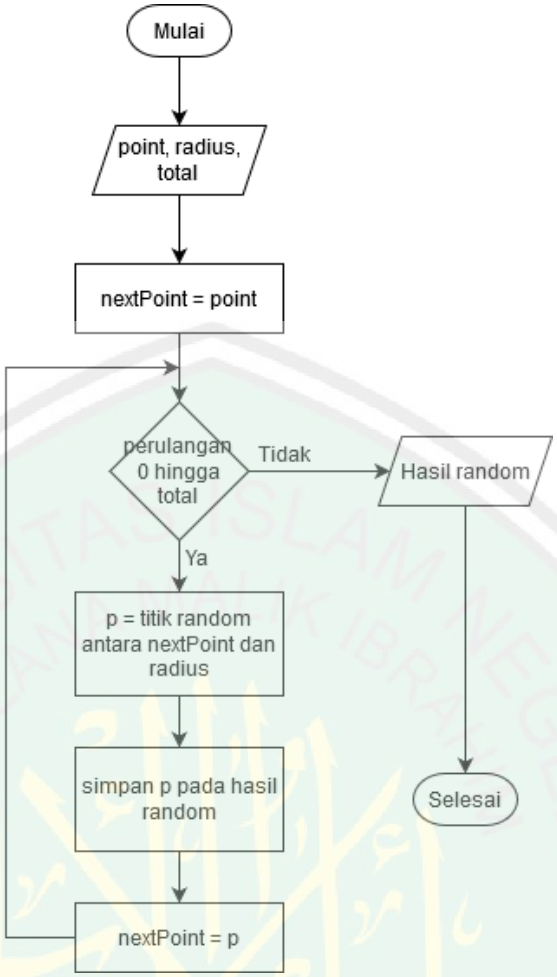
Gambar 3.17 Hasil *Circular Random Data*

1. *Walking Random Data*

Data jenis ini adalah data yang menggambarkan posisi pengguna yang berpindah-pindah. Data ini terdiri dari titik-titik koordinat acak yang saling terhubung antara titik satu dengan yang lain. Algoritma untuk mendapatkan *walking random data* dapat dilihat pada **Gambar 3.19**. *Point* adalah titik awal posisi pengguna, radius adalah jarak acak antara titik satu dengan yang lainnya, dan total adalah jumlah titik yang diinginkan. Formasi data yang dihasilkan digambarkan dengan titik-titik hitam pada **Gambar 3.18.** Data ini akan dipakai sebagai *input* pada pengujian sistem.



Gambar 3.18 Hasil *Walking Random Data*

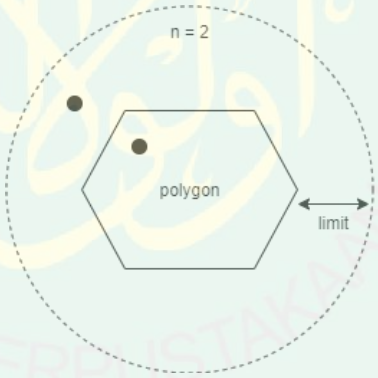


Gambar 3.19 *Flowchart Walking Random Data*

3.5.2 Pengujian Algoritma

Pengujian algoritma bertujuan untuk memperoleh nilai akurasi dan kecepatan algoritma *crossing number* dan *winding number*. Data yang digunakan adalah titik *random* pada jumlah dan radius tertentu yang diperoleh dengan perhitungan data *circular random* pada **Gambar 3.16**. Langkah-langkah pengujian algoritma dapat dilihat pada **Gambar 3.20.** Nilai *input* poligon dan limit akan digunakan sebagai *input* dalam pembuatan data *circular random* dengan aturan titik tengah poligon sebagai *point*, jarak titik tengah poligon dengan titik terluar limit sebagai radius dan n sebagai jumlah data *random* yang diinginkan, sehingga akan terbentuk tampilan seperti gambar pada **Gambar 3.21**.

Gambar 3.20 *Flowchart* Pengujian Algoritma



Gambar 3.21 Tampilan *Input* Pengujian Algoritma

3.5.3 Pengujian Sistem

Pengujian algoritma bertujuan untuk melihat hasil penerapan algoritma *crossing number* dan *winding number*. Data yang digunakan adalah data lokasi pengguna pada waktu seminggu. Data pengguna pada pengujian ini akan di-

generate menggunakan perhitungan data *walking random* pada **Gambar 3.18**. Langkah-langkah pengujian sistem dapat dilihat pada **Gambar 3.22.**

Gambar 3.22 *Flowchart* Pengujian Sistem

BAB IV

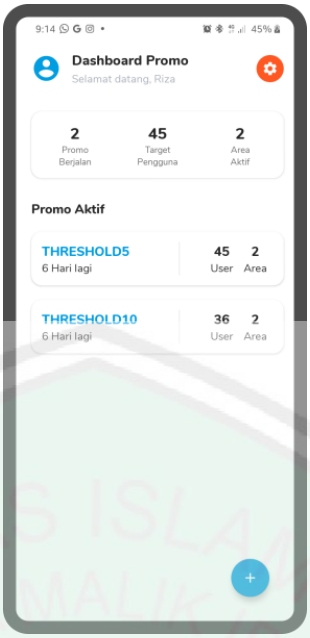
PEMBAHASAN

4.1 Implementasi Antarmuka Aplikasi Administrator

Implementasi antarmuka membahas tampilan-tampilan yang diimplementasikan pada aplikasi administrator Android berdasarkan desain antarmuka aplikasi administrator yang telah dibahas sebelumnya. Aplikasi administrator mempunyai 6 tampilan, yaitu tampilan beranda, tambah promo, kelola area, buat area baru, detail target promo, dan detail lokasi *user*. Berikut hasil implementasi antarmuka pada aplikasi administrator:

4.1.1 Halaman Beranda

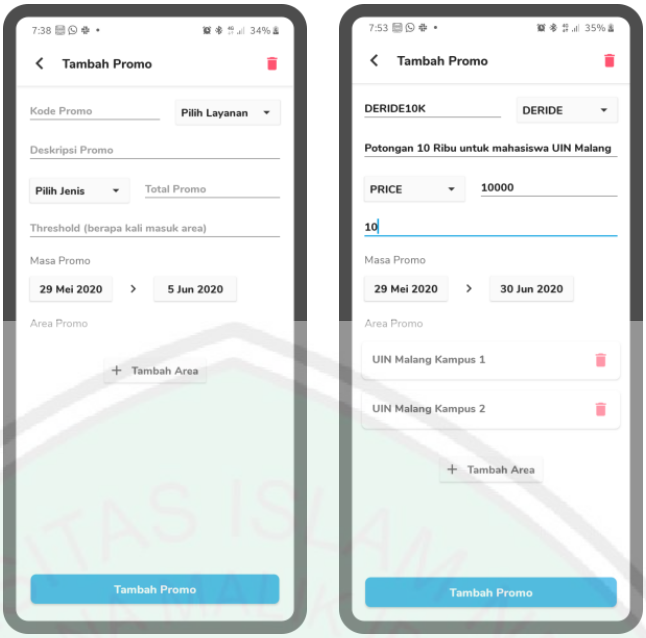
Halaman beranda adalah halaman pertama yang ditampilkan saat aplikasi administrator dibuka. Halaman ini terdiri dari 4 bagian yaitu: header, panel statistik, daftar promo yang sedang berjalan, dan tombol tambah promo seperti pada **Gambar 4.1**. Header terdiri dari 1 ImageView, 2 *TextView, dan 1 FloatingActionButton* di dalam *AppBarLayout*. Panel statistik terdiri dari 6 *TextView* di dalam *CardView*. Daftar promo terdiri dari 1 *TextView* dan 1 *RecyclerView* yang memuat *CardView* yang berisi 6 *TextView* dan 2 *View*. Tombol tambah promo adalah *FloatingActionButton* untuk membuka halaman tambah promo.



Gambar 4.1 Tampilan Halaman Beranda

4.1.2 Halaman Tambah Promo

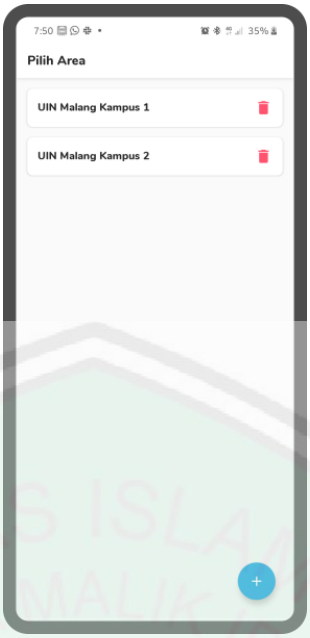
Halaman tambah promo berfungsi untuk menambahkan promo baru. Halaman tambah promo memiliki 9 masukan yaitu: *EditText* kode promo, dialog *Selector* jenis layanan promo , deskripsi promo berupa *EditText*, dialog *Selector* jenis promo, *EditText* total promo, *EditText threshold* promo, 2 *DateSelector* masa promo mulai sampai selesai, *Button* tambah area, dan *Button* konfirmasi. Hasil implementasi halaman tambah promo dapat dilihat pada **Gambar 4.2.** Gambar sebelah kiri adalah tampilan sebelum data dimasukkan, sedangkan gambar sebelah kanan adalah contoh data yang dimasukkan saat akan menambah promo. Jumlah area yang dipilih tidak di batasi namun tidak boleh memuat area yang sama.



Gambar 4.2 Tampilan Halaman Tambah Promo

4.1.3 Halaman Pilih Area

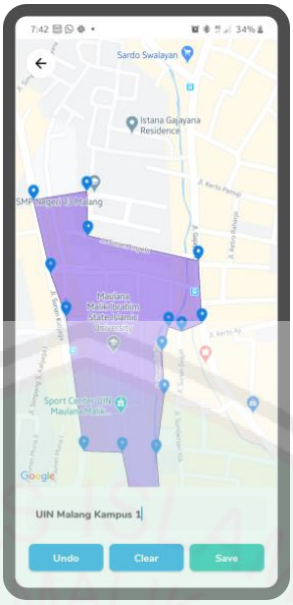
Halaman pilih area berfungsi untuk memilih dan mengelola area pada promo. Halaman ini terdiri dari *FloatingActionButton* tambah area dan daftar area berupa *RecylerView* dengan item *CardView* yang berisi 1 *TextView* dan 1 ImageButton. *FloatingActionButton* tambah area berfungsi untuk membuka halaman buat area baru. Hasil implementasi halaman ini dapat dilihat pada **Gambar 4.3.** Pada saat salah satu item area ditekan maka akan mengarah pada *Activity* sebelumnya dan mengembalikan objek area yang dipilih. Pengaplikasian halaman ini adalah pada saat proses penambahan promo pada tampilan tambah promo. Jika area berhasil ditambahkan maka akan area akan ditampilkan pada daftar area seperti pada **Gambar 4.2** sebelah kanan.



Gambar 4.3 Tampilan Halaman Pilih Area

4.1.4 Halaman Buat Area Baru

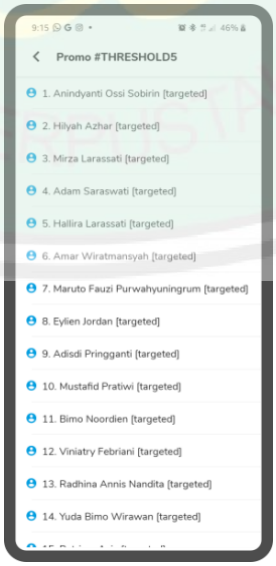
Halaman buat area baru terdiri dari 2 bagian yaitu: palet untuk menggambar poligon area dan *form* kontrol seperti pada **Gambar 4.4**. Palet area adalah *MapsFragment* dari Google Maps SDK untuk menampilkan peta dan menggambar poligon sesuai input pengguna. Form kontrol terdiri dari *EditText* nama area dan 3 *Button* untuk *undo* bersihkan area dan simpan area. Skema penggambaran area poligon adalah dengan *tap* palet area maka titik koordinat akan ditambahkan, jika dilakukan tap selanjutnya maka poligon akan dibentuk berdasarkan titik yang di-*tap*. Tombol *clear* akan membersihkan semua titik koordinat dan poligon yang ditambahkan jika ditekan. Tombol *undo* akan membatalkan penambahan titik terakhir jika ditekan dan tombol *save* akan menyimpan poligon yang terbentuk jika ditekan.



Gambar 4.4 Tampilan Halaman Buat Area

4.1.5 Halaman Detail Target Promo

Halaman ini akan menampilkan daftar target pengguna yang didapatkan pada suatu promo seperti pada **Gambar 4.5**. Halaman ini terdiri dari *RecyclerView* dengan item yang memuat 1 *TextView* dan 1 *ImageView* untuk menampilkan nama pengguna. Item *RecyclerView* akan mengarah ke halaman detail lokasi pengguna saat diklik.



Gambar 4.5 Tampilan Halaman Detail Target Promo

4.1.6 Halaman Detail Lokasi Pengguna

Halaman detail lokasi pengguna menampilkan riwayat lokasi pengguna dalam 7 hari terakhir dan area promo yang dipilih. Setiap hari terdiri dari 24 titik lokasi yang terhubung dengan garis dengan warna berbeda pada setiap harinya. Tampilan halaman detail lokasi pengguna dapat dilihat pada **Gambar 4.6.**

Gambar 4.6 Tampilan Halaman Detail Lokasi Pengguna

4.1 Implementasi Algoritma *Geofencing*

Implementasi algoritma ke dalam bentuk program menggunakan bahasa Kotlin. Algoritma *Crossing Number* dibuat berdasarkan algoritma pada **Bab 3.3.7** dan algoritma *Winding Number* dibuat berdasarkan *pseudocode* pada **Bab 3.3.8**. Proses *bounding box* ditambahkan untuk membatasi jumlah koordinat yang diperiksa. Berikut masing-masing implementasinya:

4.2.1 Implementasi *Bounding Box*

Bounding box digunakan sebagai pembatas kasar antara titik koordinat yang berada di dalam dan di luar poligon. Setiap poligon yang diproses akan dicari titik-

titik terluarnya dan sebuah kotak (*bounding box*) dibuat berdasarkan titik-titik tersebut. *Input* koordinat hanya akan diperiksa lebih lanjut dengan algoritma *crossing number* atau dengan *winding number* jika berada di dalam kotak yang dibuat. *Source code bounding box* dapat dilihat pada **Gambar 4.7.** *Method* `isInside(point: Point)` akan memeriksa apakah *input* koordinat di dalam *bounding box.*

```
class BoundingBox(p: Polygon) {
    var xMin : Double
    var xMax : Double
    var yMax : Double
    var yMin : Double

    init {
        xMin = p.points[0].x
        xMax = p.points[0].x
        yMin = p.points[0].y
        yMax = p.points[0].y

        p.points.forEach {
            if (it.x > xMax) xMax = it.x
            if (it.x < xMin) xMin = it.x
            if (it.y > yMax) yMax = it.y
            if (it.y < yMin) yMin = it.y
        }
    }

    fun isInside(point: Point): Boolean {
        return (point.x >= xMin && point.x <= xMax
                && point.y >= yMin && point.y <= yMax)
    }
}
```

Gambar 4.7 *Source Code Bounding Box*

4.2.2 Implementasi *Crossing Number*

Hasil implementasi algoritma *crossing number* dapat dilihat pada **Gambar 4.8**. Method `analyzePointByCN(poly: Poligon, p: Point)` akan mengembalikan nilai `true` jika *input* koordinat p berada di dalam poligon `poly`, jika sebaliknya maka akan mengembalikan nilai `false`. Proses yang dijalankan adalah membuat *bounding box* lalu menghitung jumlah `cn`. Perhitungan `cn` dilakukan dengan perulangan pada setiap tepi poligon, `cn` akan bertambah jika *ray* memotong tepi poligon dan memenuhi aturan/*rule* 1 sampai 4 sesuai dengan **Bab 2.3.1**. *rule* 4

dipenuhi dengan melakukan perulangan mulai dari indeks terkecil. *Rule* 3 diterapkan dengan ekspresi `V[i].x == V[i + 1].x`, yaitu jika tepi yang diperiksa adalah tepi tegak lurus maka tepi tersebut diabaikan. *Rule* 1 dan 2 dipenuhi dengan ekspresi `(V[i].y <= p.y && V[i + 1].y > p.y) || (V[i].y > p.y && V[i + 1].y <= p.y)`, yaitu jika tepi diperiksa bukan tepi yang berada di atas atau di bawah titik koordinat *input*.

Setelah keempat *rule* dipenuhi maka *ray*/garis proyeksi akan dibuat dengan perintah `(p.y - V[i].y) / (V[i + 1].y - V[i].y)`. Koordinat *input* dianggap memotong *ray* jika memenuhi ekspresi `p.x < V[i].x + ray * (V[i + 1].x - V[i].x)`, yaitu jika x *input* kurang dari x pada tepi ditambah perkalian *ray* dengan selisih nilai x tepi. Nilai cn akan bertambah 1 jika dianggap memotong tepi yang diperiksa.

```
data class Point(
        val x: Double = 0.0,
        val y: Double = 0.0
)

data class Polygon(
       val name: String,
       val points : ArrayList<Point>
){
    init {
        points.getOrNull(0)?.let { points.add(it) }
    }
}

fun analyzePointByCN(poly: Polygon, p: Point): Boolean {

    val boundingBox = BoundingBox(poly)

    if (boundingBox.isInside(p)) {
        val cn = countCN(poly, p)
        println("cn is $cn")
        return (cn % 2 != 0)
    } else {
        return false
    }
}

private fun countCN(poly: Polygon, p: Point): Int {

        var cn = 0
        val V = poly.points
        val n = V.size - 1

        for (i in 0 until n) {

            if (V[i].x == V[i + 1].x) continue //rule 3

            if (
                (V[i].y <= p.y && V[i + 1].y > p.y)  // rule1
                ||
                (V[i].y > p.y && V[i + 1].y <= p.y) // rule 2
            ) {
                val ray = (p.y - V[i].y) / (V[i + 1].y - V[i].y)

                if (p.x < V[i].x + ray * (V[i + 1].x - V[i].x)) {
                    ++cn
                }

            }

        }

        return cn

}
```

Gambar 4.8 *Source Code Crossing Number*

4.2.3 Implementasi *Winding Number*

Hasil implementasi algoritma *winding number* dapat dilihat pada **Gambar 4.9**. *Method* `analyzePointByWN()` mengembalikan nilai `true` jika *input* koordinat

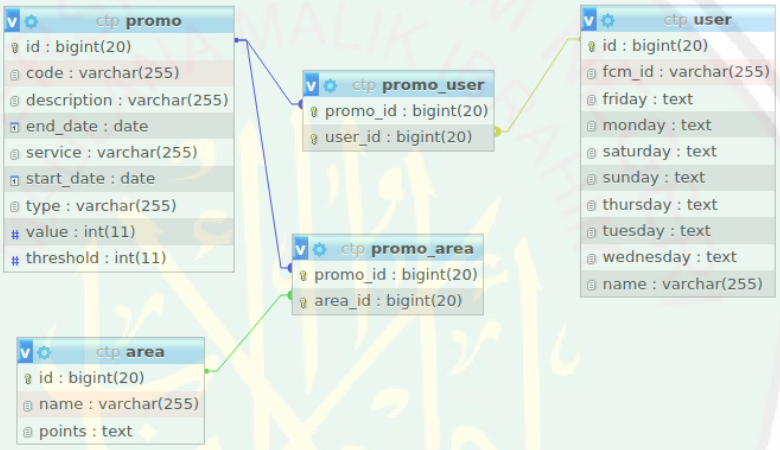
berada di dalam poligon. Langkah awal yang dikerjakan sama dengan pada algoritma *crossing number*, hanya saja algoritma ini menghitung wn untuk menentukan apakah *input* koordinat berada di dalam poligon. Perhitungan wn dilakukan dengan memeriksa semua tepi poligon yang memenuhi *rule* 3 melalui ekspresi `V[i].x == V[i + 1].x`, yaitu tepi yang diperiksa bukan tepi tegak lurus. Selanjutnya adalah jika tepi mengarah ke atas dan berada di kiri titik *input* maka nilai wn bertambah 1. Kemudian jika tepi mengarah ke bawah dan berada di kanan titik *input* maka nilai wn berkurang 1. Penentuan apakah titik *input* terletak di kiri atau kanan tepi dihitung oleh *method* `isLeft()`.

```
fun analyzePointByWN(poly: Polygon, p: Point): Boolean {
    val boundingBox = BoundingBox(poly)
    if (boundingBox.isInside(p)) {
       val wn = countWN(poly, p)
       return (wn != 0)
    } else {
       return false
    }
}

private fun countWN(poly: Polygon, p: Point): Int {

    var wn = 0
    val v = poly.points
    val n = v.size - 1

    for (i in 0 until n) {

        if (v[i].x == v[i + 1].x) continue //rule 3

        if(v[i].y<=p.y){

            if(v[i+1].y > p.y)
                if(isLeft(v[i], v[i+1], p)) ++wn

        }else{
            if(v[i+1].y <= p.y)
                if(!isLeft(v[i], v[i+1], p)) --wn
        }
    }
    return wn
}

private fun isLeft(p0: Point, p1: Point, p2: Point): Boolean {
    val n = (p1.x - p0.x) * (p2.y - p0.y) - (p2.x - p0.x) * (p1.y - p0.y)
    return (n > 0)
}
```

Gambar 4.9 *Source Code Winding Number*

4.3 Implementasi *Database*

Pemodelan tabel pada aplikasi yang dibuat menggunakan bantuan *library* JPA (Java Persistence API) yang terpasang pada *framework* Springboot. JPA akan secara otomatis memuat tabel *database* sesuai dengan tipe data yang digunakan dan akan membuat tabel relasi secara otomatis jika diperlukan. Semua tabel dan relasi akan di-*generate* ke dalam *database* MySQL pada saat di-*compile*. Hasil desain *database* dapat dilihat pada **Gambar 4.10**.



Gambar 4.10 Hasil Desain *Database*

Desain di atas dapat dibentuk dengan membuat model tabel dalam bentuk POJO (*Plain Old Java Object*) dengan anotasi tertentu. Berikut 3 model tabel yang dibuat:

4.3.1 Tabel Area

Tabel area memiliki 3 kolom yaitu *id (primary key)*, *name*, dan *points*. Kolom *name* memuat nama area dan kolom *points* memuat data *list* koordinat yang menyusun area poligon, data tersebut diformat ke dalam bentuk JSON sebelum

dimasukkan. Tabel area memiliki relasi *many-to-many* dengan tabel promo. *Source code* POJO tabel area dapat dilihat pada **Gambar 4.11.**

```
@Entity(name = "area")
public class Area {
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;

    @Column(columnDefinition = "text")
    private String points;

    @ManyToMany(fetch = FetchType.LAZY, mappedBy = "areas")
    @JsonIgnoreProperties("areas")
    private Set<PromoDTO> promos;

    /* constructor & setter getter */

}
```

Gambar 4.11 *Souce Code Entity* Area

4.3.2 Tabel *User*

Tabel *user* terdiri dari 10 kolom yaitu *id (primary key), name, fcmId, monday, tuesday, wednesday, thursday, friday, saturday, sunday*. Kolom *name* memuat nama pengguna, kolom *fcmId* memuat *id* Firebase Cloud Messaging untuk keperluan pengiriman notifikasi, kolom *monday-sunday* memuat data *list* riwayat koordinat pengguna setiap harinya yang berformat JSON. Tabel *user* memiliki 1 *many-to-many relationship* dengan tabel promo. *Script* POJO JPA untuk tabel *user* dapat dilihat pada **Gambar 4.12.**

```
@Entity(name = "user")
public class UserDTO {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;
    private String fcmId;

    @Embedded
    private UserLocation locations;

    @ManyToMany(fetch = FetchType.LAZY, mappedBy = "users")
    @JsonIgnore
    private Set<PromoDTO> promos;

    /* constructor & setter getter */
}

@Embeddable
public class UserLocation {

    @Column(columnDefinition = "text")
    private String monday;
    @Column(columnDefinition = "text")
    private String tuesday;
    @Column(columnDefinition = "text")
    private String wednesday;
    @Column(columnDefinition = "text")
    private String thursday;
    @Column(columnDefinition = "text")
    private String friday;
    @Column(columnDefinition = "text")
    private String saturday;
    @Column(columnDefinition = "text")
    private String sunday;

    /* constructor & setter getter */

}
```

Gambar 4.12 *Source Code Entity User*

4.3.3 Tabel *Promo*

Tabel promo memiliki 9 kolom *yaitu id, code, startDate, endDate, type, value, service, description,* dan *threshold*. Kolom *code* berisi kode promo, *startDate* berisi tanggal promo dimulai, *endDate* berisi tanggal promo berakhir, *type* berisi tipe promo yaitu potongan diskon atau potongan harga, *value* berisi nilai promo yang dikeluarkan, *service* berisi jenis servis yang diberi promo, *description* berisi deskripsi promo, dan *threshold* berisi batas berapa kali pengguna harus masuk ke dalam area promo. Tabel promo memiliki 2 *many-to-many relationship* dengan tabel *user* dan area. **Gambar 4.13** menunjukkan *source code* POJO tabel promo.

```
@Entity(name = "promo")
public class PromoDTO {
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String code;
    @Temporal(TemporalType.DATE)
    private Date startDate;
    @Temporal(TemporalType.DATE)
    private Date endDate;
    @Enumerated(EnumType.STRING)
    private PromoType type;
    private Integer value;
    private String service;
    private String description;
    private Integer threshold;

    @ManyToMany(
            fetch = FetchType.LAZY,
            cascade = CascadeType.ALL
    )
    @JoinTable(
            name = "promo_area",
            joinColumns = @JoinColumn(name = "promo_id"),
            inverseJoinColumns = @JoinColumn(name = "area_id")
    )
    @JsonIgnoreProperties("promos")
    private Set<AreaDTO> areas;

    @ManyToMany(
            fetch = FetchType.LAZY,
            cascade = CascadeType.ALL
    )
    @JoinTable(
            name = "promo_user",
            joinColumns = @JoinColumn(name = "promo_id"),
            inverseJoinColumns = @JoinColumn(name = "user_id")
    )
    @JsonIgnoreProperties({"locations", "fcmId", "promos"})
    private Set<UserDTO> users;

    /* constructor & setter getter */

}
```

Gambar 4.13 *Source Code Entity* Promo

4.4 Implementasi *Geofencing Webservice* API

Implementasi dilakukan menggunakan bahasa Java dan Kotlin dengan bantuan *framework* Springboot. Terdapat 7 *endpoint* API yang dibuat untuk memenuhi rancangan sistem pada bab sebelumnya. Berikut implementasi yang dilakukan:

4.4.1 Menyimpan Area

Endpoint ini berfungsi untuk menyimpan area baru yang dimasukkan pada tampilan buat area baru pada **Gambar 4.4**. Sesuai dengan struktur data pada model

tabel area, area disimpan dengan nama dan data koordinat yang menyusunnya. Sebelum disimpan koordinat di-*encode* ke dalam format JSON dengan perintah `writeValueAsString()`. *Source code* selengkapnya pada proses menyimpan area dapat dilihat pada **Gambar 4.14**. *Endpoint* menyimpan area diakses degan *method* HTTP POST dengan *request body* seperti pada **Gambar 4.15**.

```
@PostMapping("add")
@ResponseBody
fun add(@RequestBody body: AreaRequest): BaseResponse<AreaDTO> {
    val points = objectMapper.writeValueAsString(body.points)
    val area = AreaDTO(
            body.name,
            points,memuat
            emptySet()
    )
    val saved = areaRepository.save(area)
    return BaseResponse(
            message = "Berhasil menambah Area",
            data = saved
    )
}
```

Gambar 4.14 *Source Code* Simpan Area

```
{
    "name": "UIN Malang",
    "points": [
        {
            "y": -7.971241538425769,
            "x": 112.63209448865541
        },
        {
            "y": -7.980251584948508,
            "x": 112.62505637220033
        },
        {
            "y": -7.98220656882575,
            "x": 112.64119254163393
        }
    ]
}
```

Gambar 4.15 *Request Body* Simpan Area

4.4.2 Menampilkan Semua Area

Endpoint menampilkan area dapat diakses menggunakan *method* HTTP GET. *Endpoint* ini berfungsi untuk menampilkan semua area yang tersimpan di database untuk keperluan pemilihan area promo sesuai tampilan pada **Gambar 4.3**.

Source code endpoint ini dapat dilihat pada **Gambar 4.16**. Hasil *output endpoint* ini adalah *list* objek area dalam bentuk JSON.

```
@GetMapping("all")
@ResponseBody
fun all(): BaseResponse<Iterable<AreaDTO>> {

    return BaseResponse(
            message = "semua area",
            data = areaRepository.findAll()
    )

}
```

Gambar 4.16 *Source Code* Menampilkan Semua Area

4.4.3 Menambah Pengguna

Pengguna yang ditambahkan pada sistem ini hanya memiliki identitas nama sesuai dengan tabel *User*, karena hanya ditujukan untuk menyimpan *id* dan *fcmId* sebagai keperluan notifikasi. *Source code* pada *webservice* untuk menambah pengguna dapat dilihat pada **Gambar 4.17**. Secara *default* kolom selain nama akan dikosongkan dan kolom *id* adalah *auto-increment*.

```
@PostMapping("add")
@ResponseBody
fun addUser(@RequestBody request: AddUserRequest):
BaseResponse<UserDTO> {

    val user = userRepository.save(
            UserDTO(request.name)
    )

    return BaseResponse(
            data = user
    )
}
```

Gambar 4.17 *Source Code* Menambah Pengguna

4.4.4 Memperbarui Lokasi Pengguna

Lokasi pengguna perlu diperbarui setiap harinya secara kolektif. *Endpoint* ini akan melakukan *update* lokasi pada hari yang dipilih pada tabel *User*. Data yang dikirimkan adalah daftar lokasi pada perangkat pengguna yang dikumpulkan pada

setiap jam. *Source code* memperbarui lokasi pengguna dapat dilihat pada **Gambar 4.18**. *Endpoint* ini dapat diakses dengan *method* HTTP POST dengan *request body* seperti pada **Gambar 4.19.**

```
@PostMapping("update-location")
@ResponseBody
fun editLocation(@RequestBody request: UpdateLocationRequest):
BaseResponse<UserDTO> {

    val response = BaseResponse<UserDTO>()
    val user = userRepository.findById(request.idUser)

    user.ifPresentOrElse({
        val location = objectMapper.writeValueAsString(request.location)
        when (request.day) {

            "monday" -> {
                it.locations.monday = location
            }

            /* Hari Lain*/

            else -> {
                throw BadRequestException("Hari tidak valid")
            }

        }

        userRepository.save(it)
        response.data = it

    }, {
        throw BadRequestException("User tidak ditemukan")
    })

    return response
}
```

Gambar 4.18 *Source Code* Memperbarui Lokasi Pengguna

```
{
    "idUser": 10,
    "day": "monday",
    "location":[
        {
            "y": -7.971241538425769,
            "x": 112.63209448865541
        },
        {
            "y": -7.980251584948508,
            "x": 112.62505637220033
        },
        {
            "y": -7.98220656882575,
            "x": 112.64119254163393
        }
    ]

}
```

Gambar 4.19 *Request Body* Memperbarui Lokasi Pengguna

4.4.5 Menampilkan Semua Pengguna

Daftar pengguna beserta riwayat lokasinya akan ditampilkan melalui tampilan ini. Riwayat lokasi yang ditampilkan adalah daftar lokasi yang sebelumnya telah diperbarui pada setiap harinya atau kosong jika tidak ada data yang tersimpan. Hasil implementasi API untuk menampilkan semua pengguna dapat dilihat pada **Gambar 4.20.**

```
@GetMapping("all")
@ResponseBody
fun allUser(): BaseResponse<Iterable<UserDTO>> {
    val users = userRepository.findAll()
    return BaseResponse(
            data = users
    )
}
```

Gambar 4.20 *Source Code* Menampilkan Semua Pengguna

4.4.6 Menambah Promo

Endpoint menambah promo memiliki proses terbanyak, karena proses analisa dengan menggunakan *geofencing* berlangsung pada *endpoint* ini. Algoritma *crossing number* dan *winding number* digunakan pada proses penambahan promo untuk mendapatkan daftar pengguna yang menjadi target promo. Hasil implementasi penambahan promo pada *webservice* API dapat diihat pada **Gambar 4.22**. Data yang dikirimkan sebagai *request body* dapat dilihat pada **Gambar 4.21.**

```
{
    "code" : "PROMO10",
    "startDate" : 199188266,
    "endDate" : 19982992991,
    "type": "PERCENT",
    "value" : 10,
    "service": "DERIDE",
    "description" : "PROMO DISKON DERIDE 10%",
    "areaIds" : [ 1 , 2 ],
    "threshold": 10
}
```

Gambar 4.21 Request Body Menambah Promo

```
@Transactional
@PostMapping("add")
@ResponseBody
fun addPromo(
        @RequestBody promoRequest: AddPromoRequest
): BaseResponse<PromoDTO> {

    if (promoRequest.areaIds.isEmpty()) {
        throw BadRequestException("Area ID Kosong")
    }

    val areas: Iterable<AreaDTO> =
areaRepository.findAllById(promoRequest.areaIds)

    val user = getUserLocIn(areas, promoRequest.threshold)

    val promo = PromoDTO(
            promoRequest.code,
            Date(promoRequest.startDate),
            Date(promoRequest.endDate),
            promoRequest.type,
            promoRequest.value,
            promoRequest.service,
            promoRequest.description,
            areas.toHashSet(),
            user.toHashSet(),
            promoRequest.threshold
    )

    val result = promoRepository.save(promo)

    return BaseResponse(
            data = result
    )

}
```

Gambar 4.22 *Source Code* Menambah Promo

Proses pertama yang dilakukan adalah mendapatkan semua area berdasarkan *area id* yang dikirimkan. Kemudian target *user* akan dicari melalui *method* `getUserLocIn()` dengan parameter area yang didapatkan pada proses pertama dan nilai threshold yang dimasukkan. *Source code method* `getUserLocIn()` dapat dilihat pada **Gambar 4.23**. Proses yang terjadi adalah dilakukan perulangan pada setiap pengguna, kemudian pengguna tersebut ditentukan apakah masuk ke dalam kriteria target promosi dengan menggunakan algoritma *crossing number* atau *winding number*. Penentuan target promosi melibatkan nilai *threshold* sebagai jumlah masuk area minimal yang harus dipenuhi oleh seorang pengguna dari semua titik koordinat yang tercatat pada kolom hari (*monday-sunday*). Jika pengguna

memasuki area sejumlah *threshold*, maka titik lokasi selanjutnya diabaikan dan pengguna tersebut disimpan sebagai target promosi. *Method* `getUserLocIn()` mengembalikan daftar pengguna yang terpilih, yang kemudian disimpan ke tabel promo beserta data promo dan area yang dimasukkan dengan relasi *many-to-many*.

```
private fun getUserLocIn(areas: Iterable<AreaDTO>, threshold: Int = 1):
List<UserDTO> {
    val result = arrayListOf<UserDTO>()
    val users = userRepository.findAll()

    users.forEach { user: UserDTO ->

        var isInside = false
        val locationDay = arrayListOf<String>()
        val locationHistory = arrayListOf<Point>()
        locationDay.add(user.locations.monday)
        locationDay.add(user.locations.tuesday)
        locationDay.add(user.locations.wednesday)
        locationDay.add(user.locations.thursday)
        locationDay.add(user.locations.friday)
        locationDay.add(user.locations.saturday)
        locationDay.add(user.locations.sunday)

        locationDay.forEach {
            val point = objectMapper.readValue<List<Point>>(it)
            locationHistory.addAll(point)
        }

        for (area: AreaDTO in areas) {
            val polygon = Utils.area2Polygon(area, objectMapper)
            var insideCount = 0

            for (it in locationHistory) {

                when (ALGORITHM) {
                    "CN" -> if (pointInclusion.analyzePointByCN(polygon, it))
                                insideCount++
                    else -> if (pointInclusion.analyzePointByWN(polygon, it))
                                insideCount++
                }

                if (insideCount >= threshold) {
                    isInside = true
                    break
                }
            }

            if (isInside) {
                result.add(user)
                break
            }
        }
    }
    return result
}
```

Gambar 4.23 *Source Code Method* Mendapatkan Target Pengguna

4.4.7 Menampilkan Semua Promo

Endpoint menampilkan promo dibuat untuk menyuplai data pada tampilan beranda. Data yang dihasilkan adalah daftar promo dengan dilengkapi daftar *id area* dan pengguna yang terhubung. *Endpoint* ini dapat diakses dengan *request method* HTTP POST sesuai *source code* pada **Gambar 4.24**.

```
@GetMapping("all")
@ResponseBody
fun getAllPromo(): BaseResponse<Iterable<PromoDTO>> {
    var result: Iterable<PromoDTO>? = null
    validate {
        result = promoRepository.findAll()
    }
    return BaseResponse(data = result)
}
```

Gambar 4.24 *Source Code* Menampilkan Semua Promo

4.5 Proses Pengujian

Proses pengujian terdiri dari 3 langkah yaitu: *generate random* data, *algorithm testing*, dan pengujian sistem. *Generate random* data adalah proses untuk menghasilkan *user* dan riwayat lokasinya secara *random* dengan jumlah tertentu untuk menyuplai data pada proses *algorithm testing* dan pengujian sistem. *Algorithm* testing adalah pengujian terhadap algoritma *crossing number* dan *winding number* tanpa melibatkan sistem yang dibuat untuk mengetahui akurasi dan kecepatan masing-masing algoritma. Pengujian sistem adalah pengujian terhadap keseluruhan sistem yang dibuat untuk menentukan hasil penerapan *geofencing* pada sistem penentuan target promosi.

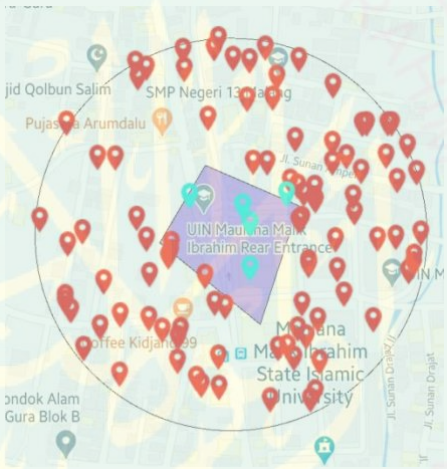
4.5.1 *Generate Circular Random Data*

Data hasil proses ini adalah data lokasi *random* pada radius lokasi tertentu sebagai data pendukung untuk *algorithm testing*. Data dihasilkan dari mengolah *input point* dan radius dengan *method* `generateRandomPointFrom()` pada **Gambar**

4.25. *Point* adalah titik tengah lingkaran area *random* dan *radius* adalah jari-jari lingkaran area *random*. Hasil dari proses ini yang diulang-ulang dengan jumlah tertentu dan ditampilkan dengan Google Maps SDK dapat dilihat pada **Gambar 4.26.**

```
fun generateRandomPointFrom(point: Point, radiusInDegree: Double): Point {
        val a = random() * 2 * PI
        val r = radiusInDegree * sqrt(random())

        val x = r * cos(a)
        val y = r * sin(a)

        return Point(point.x + x, point.y + y)
}
```

Gambar 4.25 *Source Code Generate Circular Random Data*



Gambar 4.26 Hasil *Generate Circular Random Data*

4.5.2 *Generate Walking Random Data*

Data yang dihasilkan pada proses ini adalah data lokasi *random* yang kontinu, sehingga menghasilkan lokasi berjalan dengan rute tertentu. Proses ini adalah lanjutan dari proses *generate circular random data* dengan *input point* saling terhubung dari suatu titik ke titik berikutnya. *Input point*, *radius*, dan total diproses dengan *method* `generateWalkPoints()` pada **Gambar 4.27** sehingga menghasilkan satu set koordinat. *Point* adalah titik awal rute *random*, *radius* adalah jarak *random*

antara titik, dan total adalah jumlah *point* yang akan dihasilkan. Hasil dari proses ini yang telah diproses dengan Google Maps SDK pada perangkat Android dapat dilihat pada **Gambar 4.28**.

```
fun generateWalkPoints(point: Point, radius: Double, total: Int): List<Point> {

        val result = arrayListOf<Point>()

        var nextPoint = point
        for (i in 0 until total) {
            val p = generateRandomPointFrom(nextPoint, meterToDegree(radius))
            result.add(p)
            nextPoint = p
        }

        return result
}
```

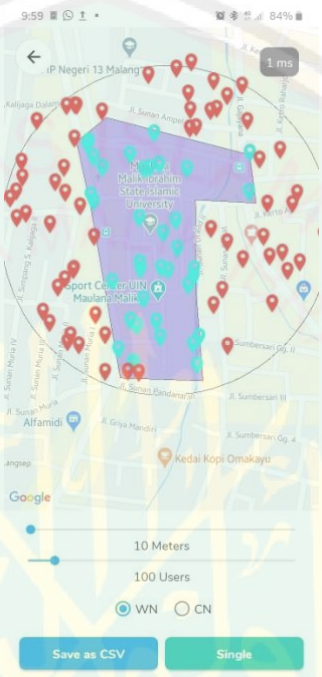
Gambar 4.27 *Source Code Generate Walking Random Data*

Gambar 4.28 Hasil *Generate Walking Random Data*

4.5.3 Pengujian Algoritma

Proses pengujian algoritma bertujuan untuk mendapatkan nilai akurasi dan kecepatan dari algoritma *crossing number* dan *winding number*. Tampilan khusus untuk melakukan proses ini dapat dilihat pada **Gambar 4.29**. Tampilan tersebut dibuat untuk memudahkan pengguna menentukan jumlah titik dan luas area yang diuji. Tampilan Google Maps akan mempermudah melakukan evaluasi hasil uji. Titik koordinat akan berwarna merah jika dianggap di luar area dan berwarna hijau jika dianggap di dalam area oleh algoritma yang dipilih.

Data kecepatan algoritma diperoleh dari pengujian masing-masing algoritma dengan 100 ribu data. Pengujian akan dilakukan sebanyak 10 kali dengan menggunakan skrip pengujian pada **Gambar 4.30.** Hasil akhir dari pengujian kecepatan adalah rata-rata yang dibutuhkan setiap algoritma dalam mengolah 100 ribu data dalam satuan *milisecond* (ms).



Gambar 4.29 Tampilan Pengujian Algoritma

```
@RunWith(JUnit4::class)
class TestSpeedAlgorithm() {
    @Test
    fun testSpeed() {

        println("Testing Speen on WN & CN Algorithm with 100k data")

        val pointInclusion = PointInclusion()
        val centre = PolygonUtils.calculateCentroid(polygon.points)
        val nUser = 100000
        val target = getRandomPoint(centre, polygon, 10.0, nUser)

        var now = System.currentTimeMillis()

        target.forEach {
            pointInclusion.analyzePointByCN(polygon, it)
        }

        println("1. CN total time = ${System.currentTimeMillis() - now} ms")

        now = System.currentTimeMillis()

        target.forEach {
            pointInclusion.analyzePointByWN(polygon, it)
        }

        println("2. WN total time = ${System.currentTimeMillis() - now} ms")

    }
}
```

Gambar 4.30 Skrip Pengujian Kecepatan

4.5.4 Pengujian Sistem

Proses pengujian sistem bertujuan untuk mendapatkan hasil penerapan algoritma *crossing number* dan *crossing number* dalam sistem penentuan target konsumen yang dibuat. Proses sistem testing dimulai dengan melakukan *generate* pengguna dengan lokasi *random* dari hasil proses *generate random walking point*. Kemudian dilanjutkan dengan menambah suatu promo dengan *threshold* tertentu yang selanjutnya akan dievaluasi daftar pengguna yang dianggap menjadi target. Evaluasi hasil memanfaatkan tampilan detail lokasi pengguna pada **Gambar 4.6.**

4.6 Hasil Pengujian

Hasil pengujian dibagi menjadi beberapa bagian untuk mendapatkan nilai yang diharapkan pada situasi yang berbeda-beda. Hasil pengujian dibagi menjadi

hasil pengujian algoritma dan hasil pengujian sistem. Berikut hasil pengujian yang didapat.

4.6.1 Hasil Pengujian Algoritma

Hasil akhir dari pengujian algoritma adalah nilai akurasi dan kecepatan pada algoritma *crossing number* dan *winding number*. Data yang digunakan adalah 100 titik lokasi *random* yang diperoleh dari proses pada **bab 4.5.1**. Radius yang digunakan adalah 10 meter dari sisi terluar area poligon. Pengujian dilakukan dengan 3 klasifikasi jenis poligon sesuai pada **bab 2.3.3** dan telah dilampirkan detail titik dan gambarnya pada **Lampiran I**. Berikut hasil pengujian algoritma.

a. Hasil pengujian *Crossing Number*

Akurasi CN pada poligon *convex* = $\frac{100}{100} \times 100\ \% = 100\%$

Akurasi CN pada poligon non-*convex* = $\frac{100}{100} \times 100\ \% = 100\%$

Akurasi CN pada poligon *convulated* = $\frac{96}{100} \times 100\ \% = 96\%$

Waktu rata-rata CN dalam mengolah 100 ribu koordinat = 51,1 ms

b. Hasil pengujian *Winding Number*

Akurasi WN pada poligon *convex* = $\frac{100}{100} \times 100\ \% = 100\%$

Akurasi WN pada poligon non-*convex* = $\frac{100}{100} \times 100\ \% = 100\%$

Akurasi WN pada poligon *convulated* = $\frac{100}{100} \times 100\ \% = 100\%$

Waktu rata-rata WN dalam mengolah 100 ribu koordinat = 30.8 ms

4.6.2 Hasil Pengujian Sistem

Hasil akhir pengujian sistem adalah akurasi sistem dalam menentukan target konsumen dengan memanfaatkan *geofencing*. Pengujian ini menggunakan skenario untuk mendapatkan target konsumen mahasiswa UIN Maulana Malik Ibrahim Malang yang berada di area kampus utama dan Gedung D. Visualisasi area yang digunakan dapat dilihat pada **Gambar 4.31.** Pengguna yang didapatkan dari proses *generate random walking data* diperiksa berapa kali berada pada area promo kemudian keputusan sistem dicatat. Detail pengujian ini telah dilampirkan pada **Lampiran II**. Berikut hasil pengujian sistem:

Akurasi Sistem menggunakan *crossing number* = $\frac{100}{100} \times 100\ \% = 100\%$

Akurasi Sistem menggunakan *winding number* = $\frac{100}{100} \times 100\ \% = 100\%$

Gambar 4.31 Visualisasi Area Pengujian

4.7 Evaluasi Hasil Pengujian

Angka-angka dari hasil pengujian algoritma dan pengujian sistem dijelaskan perlu dievaluasi untuk membantu menyelesaikan masalah pada penelitian ini. Hasil analisa berpengaruh pada kesimpulan penelitian ini. Berikut evaluasi pengujian:

4.7.1 Evaluasi Hasil Pengujian algoritma

Pengujian algoritma *crossing number* dan *winding number* dengan menggunakan 100 titik acak menghasilkan nilai akurasi dan kecepatan kedua algoritma saat dijalankan. Kedua algoritma mempunyai akurasi sempurna 100% jika di terapkan pada poligon *convex* dan *non-convex*. Namun terdapat perbedaan hasil akurasi jika diterapkan pada poligon *convulated* di mana algoritma *winding number* memiliki akurasi 100% dan algoritma *crossing number* memiliki akurasi 96%. Algoritma *winding number* menghabiskan waktu rata-rata 30,8 ms dalam mengolah 100 ribu koordinat sementara algoritma *crossing number* menghabiskan waktu rata-rata 51,1 ms.

Hasil akurasi *crossing number* yang mencapai 100% pada poligon *convex* dan *non-convex* menyempurnakan akurasi pada penelitian sebelumnya yang hanya mencapai 94% dengan menggunakan metode yang sama dan hanya mengujinya pada poligon *convex* dan *non-convex* (Basid, et al., 2017). Berdasarkan analisa yang dilakukan, perbedaan tersebut diakibatkan oleh penggunaan tipe data yang berbeda pada saat implementasi. **Gambar 4.32** menunjukkan salah satu kode dari penelitian sebelumnya. **Gambar 4.8** menunjukkan kode yang digunakan pada penelitian ini. Tipe data yang digunakan pada penelitian sebelumnya adalah *float*, sedangkan tipe data yang digunakan pada penelitian ini adalah *double*.

```
malang = Polygon.Builder()
            .addVertex(new Point(-7.909614f, 112.629033f))
            .addVertex(new Point(-7.923216f, 112.636758f))
            .addVertex(new Point(-7.914035f, 112.645169f))
            .addVertex(new Point(-7.926616f, 112.662335f))
            .addVertex(new Point(-7.965379f, 112.657529f))
            .addVertex(new Point(-7.974900f, 112.684308f))
            .addVertex(new Point(-7.984420f, 112.694951f))
            .addVertex(new Point(-8.001419f, 112.679158f))
            .addVertex(new Point(-7.983400f, 112.671949f))
            .addVertex(new Point(-7.992239f, 112.653409f))
            .addVertex(new Point(-8.003459f, 112.647916f))
            .addVertex(new Point(-8.041705f, 112.659417f))
            .addVertex(new Point(-8.046294f, 112.641049f))
            .addVertex(new Point(-8.041875f, 112.632466f))
            .addVertex(new Point(-8.035454f, 112.630828f))
            .addVertex(new Point(-8.020967f, 112.613303f))
            .addVertex(new Point(-8.011278f, 112.618733f))
            .addVertex(new Point(-8.009899f, 112.616644f))
            .addVertex(new Point(-8.019097f, 112.608434f))
```

Gambar 4.32 S*ource Code* Tipe Data Penelitian Sebelumnya (Basid, et al., 2017)

Tipe data *float* mempunyai presisi 2,4 meter saat digunakan untuk perhitungan *geo-coordinate* sedangkan tipe data *double* mempunyai presisi hingga satuan *nano* meter (Ernerfeldt, 2017). Hal tersebut mengakibatkan akurasi pada penelitian sebelumnya tidak dapat mencapai 100%, karena titik yang berjarak kurang dari 2,4 meter dari tepi poligon tidak dapat dideteksi secara akurat. penggunaan tipe data *double* dapat meningkatkan akurasi, namun juga dapat meningkatkan *memory* yang digunakan pada saat perhitungan karena *double* adalah tipe data 8 *Byte* sementara *float* adalah tipe data 4 *Byte* (Kahan, 1997).

Algoritma *winding number* ditawarkan untuk menggantikan algoritma *crossing number/raycasting* pada penelitian sebelumnya. Hasil pengujian algoritma membuktikan bahwa algoritma ini memiliki akurasi yang lebih tinggi dari algoritma *crossing number*. *Crossing number* hanya mempunyai akurasi 96% jika diterapkan pada poligon *convoluted* sementara *winding number* mempunyai akurasi 100%. Hal ini disebabkan karena algoritma *crossing number* menentukan suatu titik berada di dalam poligon hanya jika perpotongan proyeksi terhadap tepi poligon berjumlah

ganjil. **Gambar 4.33** menjelaskan bahwa jika terdapat perpotongan pada poligon, maka belum tentu titik yang berada di dalam poligon mempunyai perpotongan garis proyeksi ganjil. Sementara itu, *winding number* menentukan suatu titik berada di dalam dengan menghitung berapa kali tepi poligon membelitnya, sehingga penentuan titik lebih akurat.

Gambar 4.33 Analisa Kekurangan Algoritma *Crossing Number*

Algoritma *winding number* merupakan algoritma *crossing number* yang telah dimodifikasi dengan aturan baru seperti yang dijelaskan pada **Bab 2.3.2**. Hasil pengujian membuktikan bahwa selain lebih akurat, algoritma *winding number* terbukti lebih cepat jika dibandingkan dengan algoritma *crossing number* pada saat digunakan untuk mendeteksi 100 ribu titik *random*. *Winding number* membutuhkan waktu rata-rata 30,8 ms sementara *crossing number* membutuhkan waktu lebih lama, yaitu 51,2 ms. Algoritma yang lebih cepat dan akurat adalah algoritma yang lebih baik untuk diterapkan pada sistem yang dibuat.

4.7.2 Evaluasi Hasil Pengujian Sistem

Sistem dapat menentukan target konsumen dengan akurasi 100%. Hal ini dipengaruhi oleh hasil pengujian algoritma pada poligon *convex* dan *non-convex* yang dapat mendeteksi titik dengan akurasi 100%. Kesalahan penentuan tidak ditemukan karena *input* area yang digunakan pada **Gambar 4.30** adalah poligon *non-convex* (Gedung utama UIN Maulana Malik Ibrahim Malang) dan poligon *convex* (Gedung D UIN Maulana Malik Ibrahim Malang). Kesalahan pada algoritma hanya terjadi jika poligon yang digunakan berjenis *convoluted*, sedangkan pada studi kasus yang digunakan tidak ada poligon berjenis *convoluted*.

4.8 Integrasi penelitian Sistem Penentuan Target Promosi pada Konsumen dengan Islam

Tujuan utama berdagang adalah memperoleh keuntungan. Memberikan sebagian keuntungan dari berdagang untuk bersedekah tentu akan mengurangi jumlah keuntungan yang diterima, namun hal itu dianjurkan oleh Islam dan justru kegiatan tersebut berimbas pada perniagaan yang tidak akan merugi. Kegiatan tersebut dapat dilakukan dalam bentuk promosi ke pada pelanggan atau konsumen. Berikut penjelasan pada Al-Quran surah Fatir ayat 29 beserta tafsirnya.

إِنَّ ٱلَّذِينَ يَتۡلُونَ كِتَٰبَ ٱللَّهِ وَأَقَامُواْ ٱلصَّلَوٰةَ وَأَنفَقُواْ مِمَّا رَزَقۡنَٰهُمۡ سِرّٗا وَعَلَانِيَةٗ يَرۡجُونَ تِجَٰرَةٗ لَّن تَبُورَ

"Sesungguhnya orang-orang yang selalu membaca Kitab Allah (Al-Qur'an) dan melaksanakan salat dan menginfakkan sebagian rezeki yang Kami anugerahkan kepadanya dengan diam-diam dan terang-terangan, mereka itu mengharapkan perdagangan yang tidak akan rugi," *(QS. Fatir:29).*

1. Penjelasan Surah Fatir ayat 29 menurut Tafsir Jalalain:

(*Sesungguhnya orang-orang yang selalu membaca)* selalu mempelajari *(kitab Allah ﷻ dan mendirikan salat*) yakni mereka melaksanakannya secara rutin dan memeliharanya (*dan menafkahkan sebagian dari rezeki yang Kami anugerahkan kepada mereka dengan diam-diam dan terang-terangan*) berupa zakat dan lain-lainnya (*mereka itu mengharapkan perniagaan yang tidak akan merugi*) tidak bangkrut.

2. Penjelasan Surah Fatir ayat 29 menurut Tafsir Ringkas Kemenag RI

Pada ayat ini Allah ﷻ menyebutkan sebagian tanda orang yang takut kepada-Nya. Sesungguhnya orang-orang yang selalu membaca Kitab Allah ﷻ, yakni Al-Qur'an, lalu mereka mengkaji dan mengamalkan kandungannya, dan melaksanakan salat dengan sempurna syarat dan rukunnya, dan menginfakkan sebagian rezeki yang Kami anugerahkan kepadanya dengan diam-diam dan terang-terangan, baik dalam keadaan lapang maupun sempit, mereka itu mengharapkan perdagangan dengan Allah ﷻ yang tidak akan pernah rugi.

Dua tafsir di atas menjelaskan bahwa orang-orang yang menginfakkan sebagian rezeki secara diam-diam maupun terang-terangan maka tidak akan merugi. Tidak akan merugi dalam hal ini karena Allah ﷻ akan membalasnya baik di dunia maupun di akhirat. Balasan di dunia salah satunya adalah keuntungan pada perniagaan atau usaha. Konsep tersebut sejalan dengan promosi, yaitu dengan memberikan sebagian penghasilan perusahaan kepada konsumen maka diharapkan konsumen menjadi pelanggan setia yang pada akhirnya akan menguntungkan perusahaan.

Menginfakkan harta memang dianjurkan, namun harus dilakukan dengan benar. Infak seharusnya dilakukan dengan secukupnya dan tidak berlebihan karena

Allah ﷻ menyayangi orang-orang yang tidak boros. Berikut penjelasan surah Al-Furqon ayat 67 beserta terjemahan dan artinya:

وَالَّذِينَ إِذَا أَنْفَقُوا لَمْ يُسْرِفُوا وَلَمْ يَقْتُرُوا وَكَانَ بَيْنَ ذَٰلِكَ قَوَامًا

"Dan (termasuk hamba-hamba Tuhan Yang Maha Pengasih) orang-orang yang apabila menginfakkan (harta), mereka tidak berlebihan, dan tidak (pula) kikir, di antara keduanya secara wajar," (QS. Al-Furqan:67).

1. Penjelasan surah Al-Furqan ayat 67 menurut Tafsir Jalalain

(Dan orang-orang yang apabila membelanjakan) hartanya kepada anak-anak mereka (mereka tidak berlebih-lebihan dan tidak pula kikir) dapat dibaca *Yaqturuu* dan *Yuqtiruu*, artinya tidak mempersempit perbelanjaannya (dan adalah) nafkah mereka (di antara yang demikian itu) di antara berlebih-lebihan dan kikir (mengambil jalan pertengahan) yakni tengah-tengah.

2. Penjelasan surah Al-Furqan ayat 67 menurut Tafsir Ringkas Kemenag RI

Sifat berikutnya adalah tidak berlebih-lebihan dalam berinfak. Dan di antara sifat hamba-hamba Tuhan Yang Maha Pengasih adalah orang-orang yang apabila menginfakkan harta, mereka tidak berlebihan dengan menghambur-hamburkannya, karena perilaku seperti inilah yang dikehendaki setan dan tidak pula kikir yang menyebabkan dibenci oleh masyarakat. Mereka berinfak di antara keduanya secara wajar, inilah agama yang pertengahan, moderat, seimbang antara kepentingan individu dan masyarakat.

Dua tafsir di atas menjelaskan bahwa menginfakkan harta secara tidak berlebihan adalah sifat terpuji. Penjelasan tersebut sejalan dengan manfaat penelitian ini, yaitu terciptanya sistem informasi geografis untuk mendapatkan target promosi pada wilayah tertentu. Promosi yang dilakukan secara berlebihan

dengan target seluruh konsumen dapat menghabiskan biaya yang besar. Target promosi yang tepat sasaran dapat mengurangi biaya pengeluaran tanpa menghilangkan tujuan untuk mendapatkan pelanggan.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian, implementasi dan uji coba yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Akurasi algoritma *crossing number* (CN) dalam menganalisis titik pada poligon *convex* dan *non-convex* adalah 100%
2. Akurasi algoritma *crossing number* dalam menganalisis titik pada poligon *convoluted* adalah 96%
3. Akurasi algoritma *winding number* (WN) dalam menganalisis titik pada poligon *convex, non-convex,* dan *convoluted* adalah 100%
4. Waktu rata yang dibutuhkan dalam mengolah 100 ribu koordinat dengan menggunakan algoritma *crossing number* adalah 51,1 ms
5. Waktu rata yang dibutuhkan dalam mengolah 100 ribu koordinat dengan menggunakan algoritma *winding number* adalah 30,8 ms

Penelitian ini membuktikan bahwa algoritma *winding number* lebih unggul dari segi akurasi dan kecepatan dibandingkan dengan algoritma *crossing number*. Penggunaan algoritma *crossing number* dalam menganalisis titik pada poligon tidak dapat mencapai akurasi sempurna karena tidak dapat mendeteksi suatu titik di dalam poligon yang mempunyai perpotongan di dalamnya.

5.2 Saran

Masih terdapat kekurangan dalam pembuatan sistem penentuan target promosi pada konsumen menggunakan geofence. Terdapat beberapa optimasi yang dapat dilakukan untuk meningkatkan kinerja sistem yang dibuat antara lain:

1. Pengujian masih menggunakan data *random* hasil dari komputasi tertentu, sehingga perlu dilakukan uji coba dengan menggunakan data riil di lapangan.
2. Jika terdapat lebih dari 1 poligon yang dianalisis sistem, maka sistem akan melakukan analisis satu per satu setiap poligonnya. Proses ini dapat dihindari dengan melakukan komputasi geografis tertentu, yaitu menyatukan semua poligon sebelum mengolahnya.
3. Saran nomor 2 akan mengakibatkan kemungkinan terbentuknya poligon *convoluted*, sehingga algoritma yang dapat digunakan hanya *winding number* mengingat algoritma *crossing number* tidak dapat digunakan untuk mendeteksi titik di dalam poligon *convoluted* secara akurat.
4. Algoritma *winding number* terbukti lebih baik daripada algoritma *crossing number*, namun tidak menutup kemungkinan kedua algoritma tersebut dikembangkan lebih lanjut untuk mendapat hasil yang lebih memuaskan.

DAFTAR PUSTAKA

O'Rourke, J., 1998. Point in Polygon. Dalam: *Computational Geometry in C (2nd Edition).* Cambridge (NY): Springer.

Basid, P. M., Tolle, H. & Ramdani, F., 2017. Pengembangan Sistem E-Complaint Berbasis Social Crowdsource Geotagging Studi Kasus Kota Malang. *Magister thesis, Universitas Brawijaya.*

Carchiolo, V. et al., 2018. An Adaptive Algorithm for Geofencing. *Information Technology for Management Emerging Research and Applications*, pp. 115-135.

Changchien, S., Lee, C.-F. & Hsu, Y.-J., 2004. On-line personalized sales promotion in electronic commerce. *Expert Systems with Applications Volume 27 Issue 1,* pp. 35-42.

Ernerfeldt, E., 2017. *I LIKE BIG BITS.* [Online]
Available at: http://www.ilikebigbits.com/2017_06_01_float_or_double.html
[Diakses 20 June 2020].

Estelami, H. & O'Connor, G., 2009. The Effects of Cognitive Style,Shopping Experience and Consumer Demographics on Consumer Reactions to Quantity Surcharges. *Journal of Promotion Management,* pp. 161-180.

Galetzka, M. & Glauner, P., 2017. A Simple and Correct Even-Odd Algorithm for the Point-in-Polygon Problem for Complex Polygons. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* , Volume 1, pp. 175-179.

Haditya, D. P., 2017. *Pembangunan Aplikasi Bandung near You Untuk Pemberitahuan Tempat-Tempat Wisata Di Kota Bandung Memanfaatkan Teknologi geofence Studi Kasus DISBUDPAR Kota Bandung.* UNIKOM: Undergraduate Theses from JBPTUNIKOMPP.

Handayanto, Y. R. & Dewi, K. E., 2019. Development Of Blood Donor Application Using Geofencing and Firebase Technology on Android Platform. *Doctoral dissertation, Universitas Komputer Indonesia.*

Hidayat, M. A., 2019. *Profil Deliv* [Wawancara] (1 Desember 2019).

Huisman, O. & By, R. A. d., 2009. *Principles of Geographic Information Systems.* Enschede: The International Institute for Geo-Information Science and Earth Observation.

Irawan, J., 2018. *Penerapan Absen Mahasiswa Berbasis Android Menggunakan Teknologi Or Code dan Geofence (Studi Kasus: Ti Uin Syarif Hidayatullah Jakarta).* Jakarta: Skripsi S1 UIN Syarif Hidayatullah Jakarta.

Jean, A. W. & Yazdanifard, R., 2015. The Review of how Sales Promotion Change the Consumer's Perception and Their Purchasing Behavior of a Product. *Global Journal of Management and Business Research: E-Marketing,* pp. 33-37.

Kahan, W., 1997. IEEE Standard 754 for Binary Floating-Point Arithmetic. *Lecture Notes on the Status of IEEE 754,* pp. 1-30.

Kumar, G. N. & Bangi, M., 2018. An Extension to Winding Number and Point-in-Polygon Algorithm. *IFAC PapersOnLine,* 51(1), pp. 548-554.

Nurdin, M. R., 2018. *Pembangunan Aplikasi Appedestrian Untuk Meningkatkan Jumlah Pejalan Kaki Memanfaatkan Teknologi Geofencing Dan Fitur Gamification.* Bandung: Thesis D3 Unikom.

Rahate, S. W. & Saikh, M., 2016. Geo-fencing Infrastructure: Location Based Service. *International Research Journal of Engineering and Technology,* pp. 1095-1098.

Rahman, A. F., 2017. Rancang Bangun Aplikasi Geofence Marketing Cafe Berbasis Android Studi Kasus: Ice Ah!. *Sarjana thesis, Universitas Brawijaya.*

Reclus, F. & Drouard, K., 2009. Geofencing for fleet & freight management. *2009 9th International Conference on Intelligent Transport Systems Telecommunications, (ITST),* pp. 353-356.

Segara, R. & Subari, 2017. Sistem Pemantauan Lokasi Anak Menggunakan Metode Geofencing pada Platform Android. *Jurnal Teknologi dan Manajemen Informatika STIKI Malang,* 7(1), pp. 72-85.

Spotify Technology S.A., 2019. *Spotify Investors.* [Online]
Available at: https://investors.spotify.com/financials/default.aspx
[Diakses 20 November 2019].

Statista, 2019. *Statista.* [Online]
Available at: https://www.statista.com/statistics/185736/mobile-app-average-user-acquisition-cost/
[Diakses 19 November 2019].

Sunday, D., 2020. *Inclusion of a Point in a Polygon.* [Online]
Available at: http://geomalgorithms.com/a03-_inclusion.html
[Diakses 20 January 2020].

Tarasov, I. S. & Pikhtin, N. A., 2007. Information Fusion of Land Laser Scanning for Geographic Information System. *Information Fusion and Geographic Information Systems*, pp. 194-201.

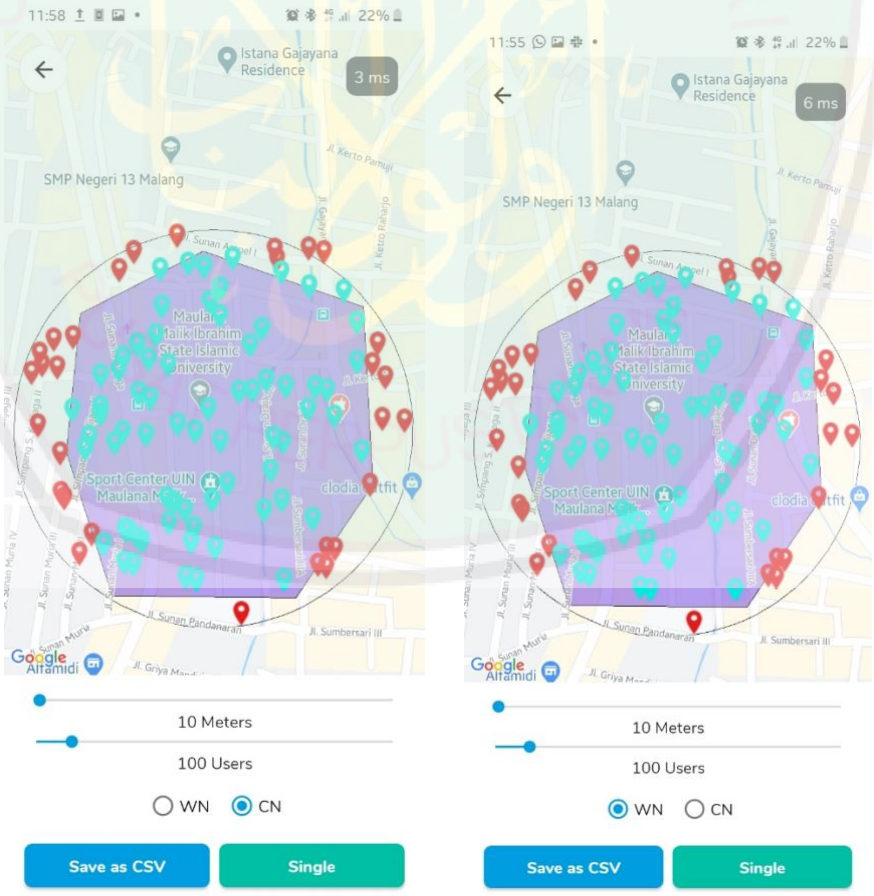
LAMPIRAN I

1. Pengujian Algoritma pada Poligon Convex

a. Tabel titik poligon convex yang digunakan

No	Latitude	Longitude
1	-7.94999354339083	112.607493884861
2	-7.95061813619070	112.609304040670
3	-7.95261343983215	112.609411664307
4	-7.95388353569633	112.608541958034
5	-7.95386195240076	112.606465592980
6	-7.95224685456891	112.605946250259
7	-7.95069417639753	112.606104165315
8	-7.94999354339083	112.607493884861

b. Hasil Visual Pengujian Algoritma pada Poligon Convex



c. Tabel hasil pengujian pada poligon convex

No	Latitude	Longitude	CN		WN	
			Inside	Correct	Inside	Correct
1	-7.95196579896493	112.606333911181	TRUE	TRUE	TRUE	TRUE
2	-7.95325691935105	112.606605930963	TRUE	TRUE	TRUE	TRUE
3	-7.95093761181989	112.609234682852	TRUE	TRUE	TRUE	TRUE
4	-7.95346656516455	112.607624260132	TRUE	TRUE	TRUE	TRUE
5	-7.95141177366167	112.605672177232	FALSE	TRUE	FALSE	TRUE
6	-7.95208190639862	112.607202176826	TRUE	TRUE	TRUE	TRUE
7	-7.95034763625868	112.606539938507	FALSE	TRUE	FALSE	TRUE
8	-7.95109847363749	112.606017482746	FALSE	TRUE	FALSE	TRUE
9	-7.95293315200817	112.607591848721	TRUE	TRUE	TRUE	TRUE
10	-7.95274495379174	112.607764711927	TRUE	TRUE	TRUE	TRUE
11	-7.95199092017004	112.608974377606	TRUE	TRUE	TRUE	TRUE
12	-7.95330963404449	112.606230702684	FALSE	TRUE	FALSE	TRUE
13	-7.95118171368016	112.606746625130	TRUE	TRUE	TRUE	TRUE
14	-7.95022677843607	112.608326510403	FALSE	TRUE	FALSE	TRUE
15	-7.95138269877622	112.609232240985	TRUE	TRUE	TRUE	TRUE
16	-7.95055282666227	112.607687159698	TRUE	TRUE	TRUE	TRUE
17	-7.95216620753635	112.606585264128	TRUE	TRUE	TRUE	TRUE
18	-7.95228852671368	112.606491335267	TRUE	TRUE	TRUE	TRUE
19	-7.95362577289194	112.606603439316	TRUE	TRUE	TRUE	TRUE
20	-7.95030984352496	112.607501828864	TRUE	TRUE	TRUE	TRUE
21	-7.95364991908247	112.608792179949	FALSE	TRUE	FALSE	TRUE
22	-7.95169189357204	112.608886169843	TRUE	TRUE	TRUE	TRUE
23	-7.95340535667152	112.607353486441	TRUE	TRUE	TRUE	TRUE
24	-7.95240183913533	112.605873396655	FALSE	TRUE	FALSE	TRUE
25	-7.95172349896736	112.606758669583	TRUE	TRUE	TRUE	TRUE
26	-7.95191845385681	112.606011247822	TRUE	TRUE	TRUE	TRUE
27	-7.95290231039181	112.605923174564	FALSE	TRUE	FALSE	TRUE
28	-7.95108049718981	112.606957330552	TRUE	TRUE	TRUE	TRUE
29	-7.95228421307477	112.608395528960	TRUE	TRUE	TRUE	TRUE
30	-7.95259757223295	112.608252787748	TRUE	TRUE	TRUE	TRUE
31	-7.95190336664113	112.607547682252	TRUE	TRUE	TRUE	TRUE
32	-7.95013917368488	112.606706059415	FALSE	TRUE	FALSE	TRUE
33	-7.95086541014825	112.607679730282	TRUE	TRUE	TRUE	TRUE
34	-7.95313886440125	112.608732918988	TRUE	TRUE	TRUE	TRUE
35	-7.95379756149549	112.607294735316	TRUE	TRUE	TRUE	TRUE
36	-7.95342033865408	112.606363648159	TRUE	TRUE	TRUE	TRUE
37	-7.95058134089220	112.609080009427	TRUE	TRUE	TRUE	TRUE
38	-7.95188423008990	112.606329731951	TRUE	TRUE	TRUE	TRUE
39	-7.95027034828797	112.607323341046	TRUE	TRUE	TRUE	TRUE

40	-7.95352957962161	112.606091389546	FALSE	TRUE	FALSE	TRUE
41	-7.95290782094982	112.607084422890	TRUE	TRUE	TRUE	TRUE
42	-7.95201303831101	112.609520719563	FALSE	TRUE	FALSE	TRUE
43	-7.95365051147027	112.606701773486	TRUE	TRUE	TRUE	TRUE
44	-7.95010490937252	112.608685638812	FALSE	TRUE	FALSE	TRUE
45	-7.95165397275516	112.608429851927	TRUE	TRUE	TRUE	TRUE
46	-7.95318525935916	112.607106005457	TRUE	TRUE	TRUE	TRUE
47	-7.95021312683558	112.607826842421	TRUE	TRUE	TRUE	TRUE
48	-7.95149434759159	112.605550407567	FALSE	TRUE	FALSE	TRUE
49	-7.95112310222633	112.605805047620	FALSE	TRUE	FALSE	TRUE
50	-7.95052840138747	112.608689934081	TRUE	TRUE	TRUE	TRUE
51	-7.95419237510892	112.607923655631	FALSE	TRUE	FALSE	TRUE
52	-7.95171353346702	112.607899506447	TRUE	TRUE	TRUE	TRUE
53	-7.95334997714277	112.606750385766	TRUE	TRUE	TRUE	TRUE
54	-7.95100317705707	112.608153633815	TRUE	TRUE	TRUE	TRUE
55	-7.95324514469903	112.607391710223	TRUE	TRUE	TRUE	TRUE
56	-7.95326901683121	112.606642449580	TRUE	TRUE	TRUE	TRUE
57	-7.95132759137053	112.606586149694	TRUE	TRUE	TRUE	TRUE
58	-7.95146927608974	112.606534431213	TRUE	TRUE	TRUE	TRUE
59	-7.95368005549187	112.608883110216	FALSE	TRUE	FALSE	TRUE
60	-7.95382115381489	112.607412507443	TRUE	TRUE	TRUE	TRUE
61	-7.95304500822954	112.607281355547	TRUE	TRUE	TRUE	TRUE
62	-7.95221675703725	112.606202530852	TRUE	TRUE	TRUE	TRUE
63	-7.95011906965188	112.608304074579	FALSE	TRUE	FALSE	TRUE
64	-7.95238435932879	112.609286484152	TRUE	TRUE	TRUE	TRUE
65	-7.95346924065337	112.608889526670	FALSE	TRUE	FALSE	TRUE
66	-7.95158465742912	112.608207905400	TRUE	TRUE	TRUE	TRUE
67	-7.95222590839204	112.608290824009	TRUE	TRUE	TRUE	TRUE
68	-7.95303933660030	112.608181301520	TRUE	TRUE	TRUE	TRUE
69	-7.95295048108060	112.607156253319	TRUE	TRUE	TRUE	TRUE
70	-7.95203679186547	112.609767817155	FALSE	TRUE	FALSE	TRUE
71	-7.95179430554117	112.606893365646	TRUE	TRUE	TRUE	TRUE
72	-7.95119087829804	112.608020174131	TRUE	TRUE	TRUE	TRUE
73	-7.95190735607840	112.608678528240	TRUE	TRUE	TRUE	TRUE
74	-7.95293286481574	112.608379049836	TRUE	TRUE	TRUE	TRUE
75	-7.95284284721553	112.605884298962	FALSE	TRUE	FALSE	TRUE
76	-7.95153005070290	112.606333978438	TRUE	TRUE	TRUE	TRUE
77	-7.94995931027233	112.607195578406	FALSE	TRUE	FALSE	TRUE
78	-7.95143654300269	112.605839239270	FALSE	TRUE	FALSE	TRUE
79	-7.95275744782751	112.609376549391	FALSE	TRUE	FALSE	TRUE
80	-7.95134842639615	112.606871042495	TRUE	TRUE	TRUE	TRUE
81	-7.95235315391819	112.606156020255	TRUE	TRUE	TRUE	TRUE
82	-7.95339643369710	112.606799136098	TRUE	TRUE	TRUE	TRUE

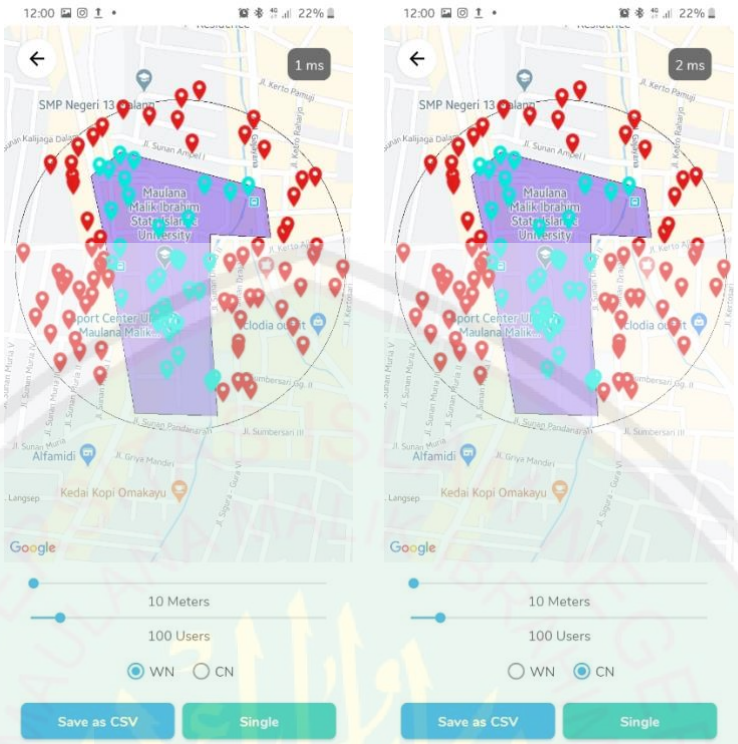
83	-7.95165898189746	112.608754048012	TRUE	TRUE	TRUE	TRUE
84	-7.95139620672529	112.609407967530	FALSE	TRUE	FALSE	TRUE
85	-7.95155066517544	112.609549290911	FALSE	TRUE	FALSE	TRUE
86	-7.95170210959285	112.608048237029	TRUE	TRUE	TRUE	TRUE
87	-7.95014456299887	112.608854166821	FALSE	TRUE	FALSE	TRUE
88	-7.95070226342926	112.607565372328	TRUE	TRUE	TRUE	TRUE
89	-7.95208425532464	112.605622756052	FALSE	TRUE	FALSE	TRUE
90	-7.95127846343820	112.605645271421	FALSE	TRUE	FALSE	TRUE
91	-7.95034133063742	112.607008267293	TRUE	TRUE	TRUE	TRUE
92	-7.95347693114734	112.608980259863	FALSE	TRUE	FALSE	TRUE
93	-7.95215895287815	112.607402535995	TRUE	TRUE	TRUE	TRUE
94	-7.95141261816783	112.607096592905	TRUE	TRUE	TRUE	TRUE
95	-7.95116886355491	112.609462858664	FALSE	TRUE	FALSE	TRUE
96	-7.95219963031116	112.606849468615	TRUE	TRUE	TRUE	TRUE
97	-7.95036856086789	112.608376068674	TRUE	TRUE	TRUE	TRUE
98	-7.95075794285065	112.607025460323	TRUE	TRUE	TRUE	TRUE
99	-7.95381976876721	112.608405843226	TRUE	TRUE	TRUE	TRUE
100	-7.95226887667827	112.607687493127	TRUE	TRUE	TRUE	TRUE
Total Correct			100		100	

2. Pengujian Algoritma pada Poligon Non-Convex

a. Tabel titik poligon non-convex yang digunakan

No	Latitude	Longitude
1	-7.95030965852588	112.606297284364
2	-7.94993277753338	112.606763653457
3	-7.95051121500268	112.609009332954
4	-7.95125966273363	112.609110921621
5	-7.95121184712528	112.608162425458
6	-7.95264498474438	112.608173154294
7	-7.95402133671039	112.608278766274
8	-7.95397883423389	112.606981582939
9	-7.95130349370306	112.606515213847
10	-7.95030965852588	112.606297284364

b. Hasil Visual Pengujian Algoritma pada poligon non-convex



c. Tabel hasil pengujian pada poligon non-convex

No	Latitude	Longitude	CN		WN	
			Inside	Correct	Inside	Correct
1	-7.94954018085603	112.607209264450	FALSE	TRUE	FALSE	TRUE
2	-7.95319319501941	112.606114583575	FALSE	TRUE	FALSE	TRUE
3	-7.95302055793400	112.606046005732	FALSE	TRUE	FALSE	TRUE
4	-7.95186356660947	112.606442657564	FALSE	TRUE	FALSE	TRUE
5	-7.94926737580585	112.607709929173	FALSE	TRUE	FALSE	TRUE
6	-7.95366024837574	112.608598426134	FALSE	TRUE	FALSE	TRUE
7	-7.95181192062564	112.608150179017	TRUE	TRUE	TRUE	TRUE
8	-7.95029446232112	112.605661512211	FALSE	TRUE	FALSE	TRUE
9	-7.95078819159256	112.609468467951	FALSE	TRUE	FALSE	TRUE
10	-7.95219218303502	112.610118720057	FALSE	TRUE	FALSE	TRUE
11	-7.95053816638678	112.609597658836	FALSE	TRUE	FALSE	TRUE
12	-7.95290042066925	112.607518398363	TRUE	TRUE	TRUE	TRUE
13	-7.95190833458417	112.610233904794	FALSE	TRUE	FALSE	TRUE
14	-7.95197147063039	112.605785057288	FALSE	TRUE	FALSE	TRUE
15	-7.95157743889481	112.606310177848	FALSE	TRUE	FALSE	TRUE
16	-7.95269074665482	112.605949661114	FALSE	TRUE	FALSE	TRUE
17	-7.95157189381725	112.609826805530	FALSE	TRUE	FALSE	TRUE
18	-7.95269594138981	112.608635299449	FALSE	TRUE	FALSE	TRUE

19	-7.95226425674930	112.607990774048	TRUE	TRUE	TRUE	TRUE
20	-7.95205977933504	112.606561985208	FALSE	TRUE	FALSE	TRUE
21	-7.95225055810059	112.607832614445	TRUE	TRUE	TRUE	TRUE
22	-7.95179327636886	112.607682345843	TRUE	TRUE	TRUE	TRUE
23	-7.95184170927243	112.606621584135	TRUE	TRUE	TRUE	TRUE
24	-7.95114634838818	112.607812910868	TRUE	TRUE	TRUE	TRUE
25	-7.95048329715978	112.609805489676	FALSE	TRUE	FALSE	TRUE
26	-7.95219236255461	112.609580275322	FALSE	TRUE	FALSE	TRUE
27	-7.95236493060400	112.606766817182	TRUE	TRUE	TRUE	TRUE
28	-7.95060929250621	112.608071885813	TRUE	TRUE	TRUE	TRUE
29	-7.95317529131242	112.606458268339	FALSE	TRUE	FALSE	TRUE
30	-7.95198360088986	112.605798952788	FALSE	TRUE	FALSE	TRUE
31	-7.95223736740566	112.606108588239	FALSE	TRUE	FALSE	TRUE
32	-7.95178557511382	112.606499526717	FALSE	TRUE	FALSE	TRUE
33	-7.95128525919561	112.607359385608	TRUE	TRUE	TRUE	TRUE
34	-7.95233233082705	112.606312109927	FALSE	TRUE	FALSE	TRUE
35	-7.95002294646112	112.606090770569	FALSE	TRUE	FALSE	TRUE
36	-7.95144895831243	112.606445984342	FALSE	TRUE	FALSE	TRUE
37	-7.95171816996128	112.608673739005	FALSE	TRUE	FALSE	TRUE
38	-7.95253447770476	112.609277166974	FALSE	TRUE	FALSE	TRUE
39	-7.95170344110151	112.609781616267	FALSE	TRUE	FALSE	TRUE
40	-7.95048584999391	112.606006534456	FALSE	TRUE	FALSE	TRUE
41	-7.95326240758550	112.607668396086	TRUE	TRUE	TRUE	TRUE
42	-7.95166530302440	112.605238440231	FALSE	TRUE	FALSE	TRUE
43	-7.95366832348267	112.608175295041	TRUE	TRUE	TRUE	TRUE
44	-7.95235900864526	112.608908370269	FALSE	TRUE	FALSE	TRUE
45	-7.95240486553180	112.605529052105	FALSE	TRUE	FALSE	TRUE
46	-7.95033389482012	112.606425119091	TRUE	TRUE	TRUE	TRUE
47	-7.95381819782974	112.608366540515	FALSE	TRUE	FALSE	TRUE
48	-7.95215895812800	112.605852731694	FALSE	TRUE	FALSE	TRUE
49	-7.95309725224246	112.609611816802	FALSE	TRUE	FALSE	TRUE
50	-7.95159020666826	112.606743727783	TRUE	TRUE	TRUE	TRUE
51	-7.95194702976957	112.609331437398	FALSE	TRUE	FALSE	TRUE
52	-7.95378313152430	112.608792001647	FALSE	TRUE	FALSE	TRUE
53	-7.95117345290670	112.606239061001	FALSE	TRUE	FALSE	TRUE
54	-7.94944028519610	112.606796789160	FALSE	TRUE	FALSE	TRUE
55	-7.95206514791918	112.605911473905	FALSE	TRUE	FALSE	TRUE
56	-7.95101664918178	112.606608145537	TRUE	TRUE	TRUE	TRUE
57	-7.95366195236065	112.608723201381	FALSE	TRUE	FALSE	TRUE
58	-7.95280324868908	112.609454058620	FALSE	TRUE	FALSE	TRUE
59	-7.95063299545961	112.608756999390	TRUE	TRUE	TRUE	TRUE
60	-7.95209249741978	112.606341027025	FALSE	TRUE	FALSE	TRUE
61	-7.95037360911958	112.606571664670	TRUE	TRUE	TRUE	TRUE

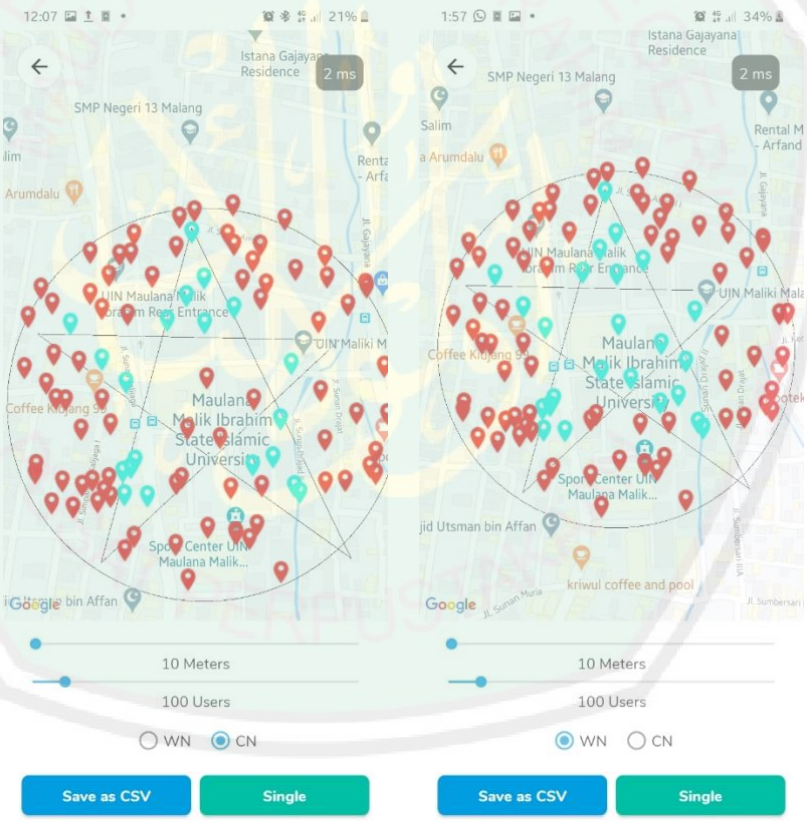
62	-7.94959532175609	112.608844090907	FALSE	TRUE	FALSE	TRUE
63	-7.95292022830041	112.605661911569	FALSE	TRUE	FALSE	TRUE
64	-7.95136613636274	112.609312537514	FALSE	TRUE	FALSE	TRUE
65	-7.95214548341221	112.607268884417	TRUE	TRUE	TRUE	TRUE
66	-7.94914698426974	112.607988933778	FALSE	TRUE	FALSE	TRUE
67	-7.95053184219498	112.606821278775	TRUE	TRUE	TRUE	TRUE
68	-7.95261926437838	112.607325217593	TRUE	TRUE	TRUE	TRUE
69	-7.95124960372870	112.609370643932	FALSE	TRUE	FALSE	TRUE
70	-7.95014867637408	112.606745601499	TRUE	TRUE	TRUE	TRUE
71	-7.95315159193467	112.608600546258	FALSE	TRUE	FALSE	TRUE
72	-7.95346966554542	112.607485917595	TRUE	TRUE	TRUE	TRUE
73	-7.95235751221792	112.608789800328	FALSE	TRUE	FALSE	TRUE
74	-7.95175749957063	112.607630197500	TRUE	TRUE	TRUE	TRUE
75	-7.94986835894403	112.606326584771	FALSE	TRUE	FALSE	TRUE
76	-7.95356235741270	112.606434190807	FALSE	TRUE	FALSE	TRUE
77	-7.95249427105818	112.607438454434	TRUE	TRUE	TRUE	TRUE
78	-7.95335414640017	112.605812441795	FALSE	TRUE	FALSE	TRUE
79	-7.95029563207766	112.605990313907	FALSE	TRUE	FALSE	TRUE
80	-7.95265781131906	112.605429645186	FALSE	TRUE	FALSE	TRUE
81	-7.94971926835369	112.606473257208	FALSE	TRUE	FALSE	TRUE
82	-7.95079453533519	112.606883929880	TRUE	TRUE	TRUE	TRUE
83	-7.95023313743699	112.606965694966	TRUE	TRUE	TRUE	TRUE
84	-7.95157464339469	112.608906080806	FALSE	TRUE	FALSE	TRUE
85	-7.95244962689727	112.608497044614	FALSE	TRUE	FALSE	TRUE
86	-7.95204689874862	112.607367698083	TRUE	TRUE	TRUE	TRUE
87	-7.95306068241530	112.608601878521	FALSE	TRUE	FALSE	TRUE
88	-7.95361886244914	112.608225899869	TRUE	TRUE	TRUE	TRUE
89	-7.95242355907838	112.608384310680	FALSE	TRUE	FALSE	TRUE
90	-7.95060340634808	112.606016210637	FALSE	TRUE	FALSE	TRUE
91	-7.95275340381472	112.607344385998	TRUE	TRUE	TRUE	TRUE
92	-7.94996198845931	112.607874968353	FALSE	TRUE	FALSE	TRUE
93	-7.94983322678120	112.608701679714	FALSE	TRUE	FALSE	TRUE
94	-7.95277127747155	112.607650033365	TRUE	TRUE	TRUE	TRUE
95	-7.95246247248727	112.606212723854	FALSE	TRUE	FALSE	TRUE
96	-7.95181040135053	112.608469192542	FALSE	TRUE	FALSE	TRUE
97	-7.95072552123838	112.608474429138	TRUE	TRUE	TRUE	TRUE
98	-7.95260092169461	112.607478836001	TRUE	TRUE	TRUE	TRUE
99	-7.95006350563011	112.609006370427	FALSE	TRUE	FALSE	TRUE
100	-7.94960489072997	112.607647821714	FALSE	TRUE	FALSE	TRUE
Total Correct			100		100	

3. Pengujian Algoritma pada Poligon Convoluted

a. Tabel titik poligon convoluted yang digunakan

No	Latitude	Longitude
1	-7.94982851259855	112.607143856585
2	-7.95305008761215	112.608712278306
3	-7.95086119916042	112.605627402663
4	-7.95092229693343	112.608594261109
5	-7.95309491454424	112.606251351535
6	-7.94982851259855	112.607143856585

b. Hasil Visual Pengujian Algoritma pada poligon convoluted



c. Tabel hasil pengujian pada poligon convoluted

No	Latitude	Longitude	CN		WN	
			Inside	Correct	Inside	Correct
1	-7.95010238490844	112.607556415272	FALSE	TRUE	FALSE	TRUE
2	-7.95145815641015	112.605712712515	FALSE	TRUE	FALSE	TRUE
3	-7.95245679860857	112.608151700893	TRUE	TRUE	TRUE	TRUE
4	-7.95026152403229	112.607742393479	FALSE	TRUE	FALSE	TRUE
5	-7.95261116147574	112.606217348752	FALSE	TRUE	FALSE	TRUE
6	-7.95283736402282	112.607786700681	FALSE	TRUE	FALSE	TRUE
7	-7.95233051280350	112.605599552375	FALSE	TRUE	FALSE	TRUE
8	-7.95308328842499	112.606483684937	FALSE	TRUE	FALSE	TRUE
9	-7.95237775771314	112.607044067646	FALSE	TRUE	FALSE	TRUE
10	-7.95255974204399	112.606341490106	FALSE	TRUE	FALSE	TRUE
11	-7.95292418052246	112.607581748695	FALSE	TRUE	FALSE	TRUE
12	-7.95161208453314	112.607751336053	FALSE	FALSE	TRUE	TRUE
13	-7.95051891626422	112.608863217950	FALSE	TRUE	FALSE	TRUE
14	-7.95240588166429	112.606164786517	FALSE	TRUE	FALSE	TRUE
15	-7.95086470962925	112.607271549771	TRUE	TRUE	TRUE	TRUE
16	-7.95247671172732	112.607525709555	FALSE	TRUE	FALSE	TRUE
17	-7.95010782826912	112.607818067085	FALSE	TRUE	FALSE	TRUE
18	-7.95129178849715	112.609034730881	FALSE	TRUE	FALSE	TRUE
19	-7.94971835257844	112.607547427617	FALSE	TRUE	FALSE	TRUE
20	-7.95252014872074	112.608204810710	TRUE	TRUE	TRUE	TRUE
21	-7.95084380174582	112.606901247504	TRUE	TRUE	TRUE	TRUE
22	-7.95085719613605	112.608389642461	FALSE	TRUE	FALSE	TRUE
23	-7.95262359503954	112.605762035276	FALSE	TRUE	FALSE	TRUE
24	-7.95286529090290	112.607300524899	FALSE	TRUE	FALSE	TRUE
25	-7.95021257988206	112.608453421027	FALSE	TRUE	FALSE	TRUE
26	-7.95117627157278	112.605787318680	FALSE	TRUE	FALSE	TRUE
27	-7.95000332561166	112.606577045748	FALSE	TRUE	FALSE	TRUE
28	-7.95019906890554	112.606432420977	FALSE	TRUE	FALSE	TRUE
29	-7.95128739913036	112.609120877708	FALSE	TRUE	FALSE	TRUE
30	-7.95225318916311	112.607742355838	TRUE	TRUE	TRUE	TRUE
31	-7.95126189986196	112.608121736328	TRUE	TRUE	TRUE	TRUE
32	-7.95140568670663	112.607290893153	FALSE	FALSE	TRUE	TRUE
33	-7.95190576530878	112.606055269229	FALSE	FALSE	TRUE	TRUE
34	-7.95266423108557	112.605975150963	FALSE	TRUE	FALSE	TRUE
35	-7.95041799718301	112.608476347026	FALSE	TRUE	FALSE	TRUE
36	-7.95233651602457	112.608953338988	FALSE	TRUE	FALSE	TRUE
37	-7.95200611002155	112.608454634453	FALSE	TRUE	FALSE	TRUE
38	-7.95302842361062	112.606985233583	FALSE	TRUE	FALSE	TRUE
39	-7.95175456845007	112.609068448861	FALSE	TRUE	FALSE	TRUE

40	-7.95072582467983	112.606260496893	FALSE	TRUE	FALSE	TRUE
41	-7.95042793409198	112.607261893828	TRUE	TRUE	TRUE	TRUE
42	-7.95067634880726	112.605772255684	FALSE	TRUE	FALSE	TRUE
43	-7.95241815547260	112.608456587567	FALSE	TRUE	FALSE	TRUE
44	-7.95041171151310	112.606753989688	FALSE	TRUE	FALSE	TRUE
45	-7.95243631548980	112.606989130140	FALSE	TRUE	FALSE	TRUE
46	-7.95161972416470	112.605899600162	FALSE	TRUE	FALSE	TRUE
47	-7.95185269031752	112.606345039195	FALSE	TRUE	FALSE	TRUE
48	-7.95017927739050	112.606141812526	FALSE	TRUE	FALSE	TRUE
49	-7.95018308561748	112.606542789817	FALSE	TRUE	FALSE	TRUE
50	-7.94985897814829	112.608060936159	FALSE	TRUE	FALSE	TRUE
51	-7.95295065156343	112.606574790590	FALSE	TRUE	FALSE	TRUE
52	-7.95034459134608	112.608163735341	FALSE	TRUE	FALSE	TRUE
53	-7.95174818608507	112.608761884107	FALSE	TRUE	FALSE	TRUE
54	-7.95238115227456	112.606327589301	FALSE	TRUE	FALSE	TRUE
55	-7.95069042446116	112.608680512689	FALSE	TRUE	FALSE	TRUE
56	-7.95227941606473	112.606545069135	TRUE	TRUE	TRUE	TRUE
57	-7.95044711352072	112.607643568669	FALSE	TRUE	FALSE	TRUE
58	-7.95227024565119	112.608893197337	FALSE	TRUE	FALSE	TRUE
59	-7.95224968165511	112.605611858304	FALSE	TRUE	FALSE	TRUE
60	-7.95060147599792	112.607088620055	TRUE	TRUE	TRUE	TRUE
61	-7.95231516942833	112.606466750583	TRUE	TRUE	TRUE	TRUE
62	-7.95071698672884	112.607556900475	TRUE	TRUE	TRUE	TRUE
63	-7.95160983614955	112.606256639548	FALSE	TRUE	FALSE	TRUE
64	-7.95063035916049	112.607820429929	FALSE	TRUE	FALSE	TRUE
65	-7.95217368497542	112.607940725327	TRUE	TRUE	TRUE	TRUE
66	-7.95144369706762	112.606496098652	TRUE	TRUE	TRUE	TRUE
67	-7.95058062810797	112.606143994615	FALSE	TRUE	FALSE	TRUE
68	-7.95245784617719	112.607832912861	FALSE	TRUE	FALSE	TRUE
69	-7.95048632595741	112.607872670474	FALSE	TRUE	FALSE	TRUE
70	-7.94998185045635	112.607144785527	TRUE	TRUE	TRUE	TRUE
71	-7.94983722571790	112.607020952892	FALSE	TRUE	FALSE	TRUE
72	-7.95245034821347	112.606065297493	FALSE	TRUE	FALSE	TRUE
73	-7.95240661310456	112.608654109924	FALSE	TRUE	FALSE	TRUE
74	-7.95255383609913	112.606707456591	TRUE	TRUE	TRUE	TRUE
75	-7.95048360920396	112.608854308665	FALSE	TRUE	FALSE	TRUE
76	-7.95254779771816	112.606479004877	TRUE	TRUE	TRUE	TRUE
77	-7.95012232739883	112.606989749241	FALSE	TRUE	FALSE	TRUE
78	-7.94977122678159	112.607166836520	FALSE	TRUE	FALSE	TRUE
79	-7.95040127905142	112.606328577280	FALSE	TRUE	FALSE	TRUE
80	-7.95152733525725	112.608411542639	FALSE	TRUE	FALSE	TRUE
81	-7.95052691680743	112.605661980677	FALSE	TRUE	FALSE	TRUE
82	-7.95087522401208	112.605950707714	TRUE	TRUE	TRUE	TRUE

83	-7.95121958223617	112.606257883621	TRUE	TRUE	TRUE	TRUE
84	-7.95124515415466	112.606030770763	FALSE	TRUE	FALSE	TRUE
85	-7.95241001953236	112.605871097388	FALSE	TRUE	FALSE	TRUE
86	-7.95250458813320	112.606282303155	FALSE	TRUE	FALSE	TRUE
87	-7.95297467449032	112.607709568815	FALSE	TRUE	FALSE	TRUE
88	-7.95336759947352	112.607105242978	FALSE	TRUE	FALSE	TRUE
89	-7.95329671223758	112.608020339496	FALSE	TRUE	FALSE	TRUE
90	-7.95212526958470	112.608954044343	FALSE	TRUE	FALSE	TRUE
91	-7.95160950997225	112.605809307803	FALSE	TRUE	FALSE	TRUE
92	-7.95220775726728	112.606587893716	TRUE	TRUE	TRUE	TRUE
93	-7.95080572148346	112.605533775466	FALSE	TRUE	FALSE	TRUE
94	-7.95198872384904	112.607422852284	FALSE	FALSE	TRUE	TRUE
95	-7.95000425357916	112.607493811010	FALSE	TRUE	FALSE	TRUE
96	-7.95180285176662	112.608017530981	TRUE	TRUE	TRUE	TRUE
97	-7.95290645781629	112.607573492716	FALSE	TRUE	FALSE	TRUE
98	-7.95190127465151	112.607102280465	FALSE	TRUE	FALSE	TRUE
99	-7.95078537447399	112.606511472118	FALSE	TRUE	FALSE	TRUE
100	-7.95130729411442	112.605497123233	FALSE	TRUE	FALSE	TRUE
Total correct			96		100	

4. Hasil pengujian kecepatan *winding number* dan *crossing number*

a. Output pengujian

```
//1
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 55 ms
2. WN total time = 30 ms

//2
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 52 ms
2. WN total time = 29 ms

//3
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 55 ms
2. WN total time = 31 ms

//4
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 44 ms
2. WN total time = 25 ms

//5
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 73 ms
2. WN total time = 52 ms

//6
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 46 ms
2. WN total time = 33 ms

//7
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 43 ms
```

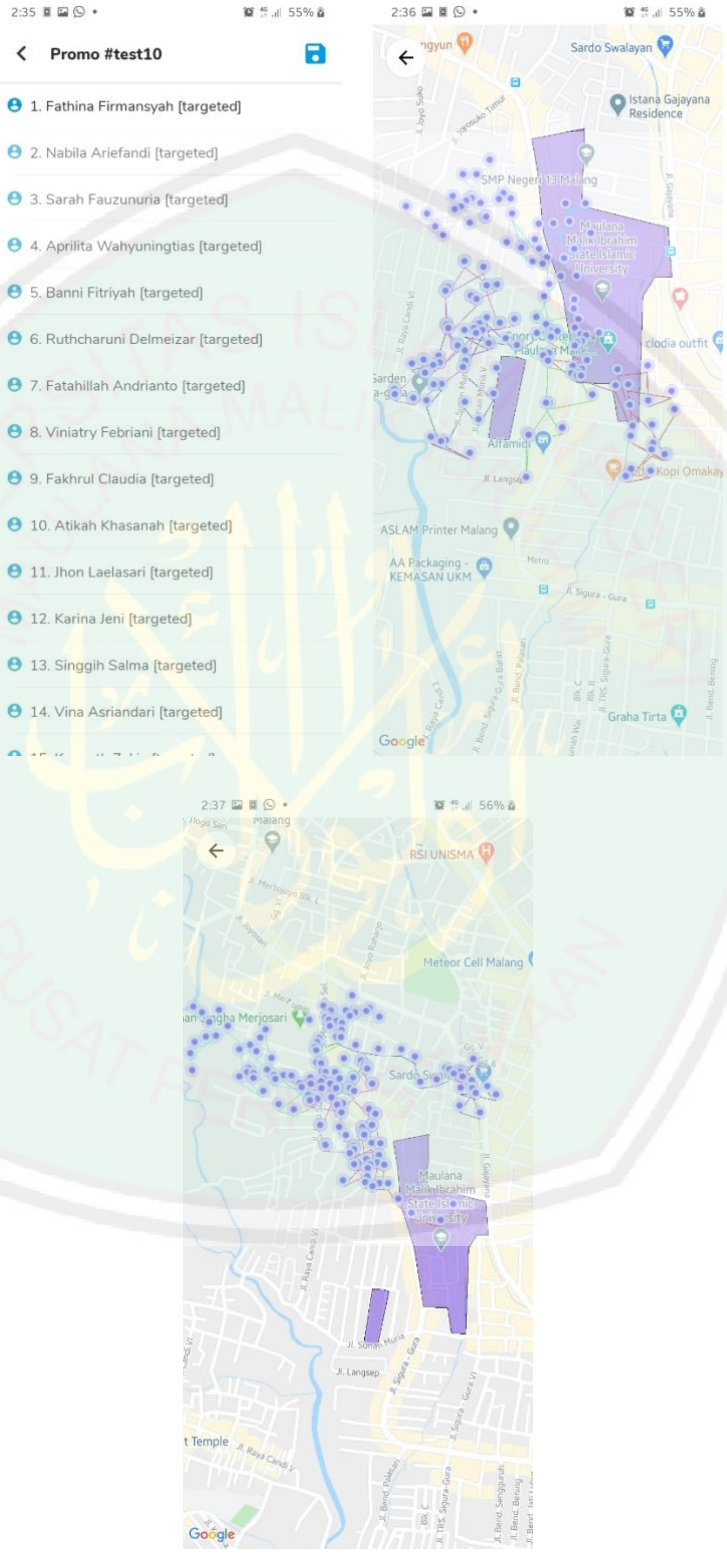
```
2. WN total time = 23 ms

//8
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 41 ms
2. WN total time = 25 ms

//9
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 58 ms
2. WN total time = 32 ms

//10
Testing Speed on WN & CN Algorithm with 100k data
1. CN total time = 44 ms
2. WN total time = 28 ms
```

b. Tabel Data pengujian kecepatan

Percobaan Ke-	Waktu (ms)	
	CN	WN
1	55	30
2	52	29
3	55	31
4	44	25
5	73	52
6	46	33
7	43	23
8	41	25
9	58	32
10	44	28
Rata-rata	51.1	30.8

LAMPIRAN II

1.Visualisasi Pengujian Sistem



2.Data Hasil Pengujian Sistem

No.	UserName	N Inside	Targetted	Correct
1	Fikri Juliyanti	0	FALSE	TRUE
2	Bimo Noordien	0	FALSE	TRUE
3	Syarief Rizky	38	TRUE	TRUE
4	Akbar Prayogi	1	FALSE	TRUE
5	Alvino Kemal Zulfianna	0	FALSE	TRUE
6	Yandra Irwanto	0	FALSE	TRUE
7	Banni Fitriyah	19	TRUE	TRUE
8	Hizkia Kasta	0	FALSE	TRUE
9	Yutama Narendra	0	FALSE	TRUE
10	Anugrah Prasojo	0	FALSE	TRUE
11	Khaznan Purwanto	0	FALSE	TRUE
12	Arfan Susfa	0	FALSE	TRUE
13	Andhika Tursia	0	FALSE	TRUE
14	Sofyan Fadhilah	0	FALSE	TRUE
15	Kemal Verev	0	FALSE	TRUE
16	Yuda Bimo Wirawan	0	FALSE	TRUE
17	Maruto Fauzi Purwahyuningrum	0	FALSE	TRUE
18	Rheza Agus Tio	5	FALSE	TRUE
19	Fatahillah Andrianto	40	TRUE	TRUE
20	Faishal Yuliasti	0	FALSE	TRUE
21	Fakhrul Claudia	49	TRUE	TRUE
22	Rian Fachrully	3	FALSE	TRUE
23	Revi Lukman Nufus	0	FALSE	TRUE
24	Mustafid Pratiwi	0	FALSE	TRUE
25	Herdaru Zain	0	FALSE	TRUE
26	Adam Kahfi	0	FALSE	TRUE
27	Kenneth Zakia	16	TRUE	TRUE
28	Chaerul Villaransi	0	FALSE	TRUE
29	Fariz Anindita	0	FALSE	TRUE
30	Satrya Mustikawati	0	FALSE	TRUE
31	Daniel Idayu	8	FALSE	TRUE
32	Mirza Larassati	1	FALSE	TRUE
33	Ferdiansyah Hermawaty	0	FALSE	TRUE
34	Aliriza Lomo	0	FALSE	TRUE
35	Arrivaldi Widian	0	FALSE	TRUE
36	Rendy Chaerunnisa	0	FALSE	TRUE
37	Singgih Salma	32	TRUE	TRUE
38	Rifqy Damayanti	0	FALSE	TRUE

39	Sumandi Puspitasari	0	FALSE	TRUE
40	I Aldian	0	FALSE	TRUE
41	Fajar Hendrika	0	FALSE	TRUE
42	Andrilla Butar-butar	0	FALSE	TRUE
43	Indra Widyatmo	0	FALSE	TRUE
44	Aburachman Fikri	0	FALSE	TRUE
45	Yosua Dinanti	3	FALSE	TRUE
46	Jhon Laelasari	15	TRUE	TRUE
47	Syahrul Julianto	4	FALSE	TRUE
48	Avicenna Alghifari	0	FALSE	TRUE
49	Devito Wahyudi	19	TRUE	TRUE
50	Andi Saputri	0	FALSE	TRUE
51	Dian Arfianti	0	FALSE	TRUE
52	Ulfi Ramadhansyah	0	FALSE	TRUE
53	Intan Abelardo	0	FALSE	TRUE
54	Nabila Ariefandi	56	TRUE	TRUE
55	Fitria Deviyanto	0	FALSE	TRUE
56	Lavinta Gardito	0	FALSE	TRUE
57	Rarahayu Riahdita	0	FALSE	TRUE
58	Syavira Rizal	2	FALSE	TRUE
59	Annisa Hermawaty	9	FALSE	TRUE
60	Ariesta Ahugrah	0	FALSE	TRUE
61	Renita Idayu	0	FALSE	TRUE
62	Mardhiyah Ivangkia	0	FALSE	TRUE
63	Tifany Journalisanda	3	FALSE	TRUE
64	Bunga Defara Mustikawati	0	FALSE	TRUE
65	Vina Asriandari	10	TRUE	TRUE
66	Jesyca Prihatiwi	0	FALSE	TRUE
67	Karlina Farishy	0	FALSE	TRUE
68	Karina Jeni	17	TRUE	TRUE
69	Imroatun Yuniara	0	FALSE	TRUE
70	Laras Ningsih	0	FALSE	TRUE
71	Septania Satrio	3	FALSE	TRUE
72	Atikah Khasanah	10	TRUE	TRUE
73	Nadiya Rohani	2	FALSE	TRUE
74	Kikhmah Evi Khairunisa	8	FALSE	TRUE
75	Mayang Dewanta	2	FALSE	TRUE
76	Permata Sabri	0	FALSE	TRUE
77	Hilary Wildani	0	FALSE	TRUE
78	Fathina Firmansyah	34	TRUE	TRUE
79	Abi Widyatama	0	FALSE	TRUE
80	Athirah Nezarani	0	FALSE	TRUE
81	Deyuri Daniela	68	TRUE	TRUE
82	Carol June Atarita	0	FALSE	TRUE

83	Aprilita Wahyuningtias	20	TRUE	TRUE
84	Andamari Suryo	0	FALSE	TRUE
85	Eti Obara	0	FALSE	TRUE
86	Hanindita Galih Rahmawati	1	FALSE	TRUE
87	Viniatry Febriani	15	TRUE	TRUE
88	Nella Verev	0	FALSE	TRUE
89	Rahmah Utami	0	FALSE	TRUE
90	Silmi Widiyasari	0	FALSE	TRUE
91	Novia Widihastuti	0	FALSE	TRUE
92	Dea Saury	0	FALSE	TRUE
93	Sarah Fauzunuria	33	TRUE	TRUE
94	Anindyanti Ossi Sobirin	1	FALSE	TRUE
95	Ruthcharuni Delmeizar	30	TRUE	TRUE
96	Wan Pradana	0	FALSE	TRUE
97	Clara Rabani	0	FALSE	TRUE
98	Desyandi Shabrina	0	FALSE	TRUE
99	Aisyah Nugroho	0	FALSE	TRUE
100	Agnes Soleha	0	FALSE	TRUE
Total Target				18
Total Correct				100