



ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - K141502

RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER

OTNIEL YEHEZKIEL BORNOK HUTABARAT
NRP 5112100212

Dosen Pembimbing
Fajar Baskoro, S.Kom., MT.
Rizky Januar Akbar, S.Kom, M.Eng

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]



TUGAS AKHIR - K141502

RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER

**OTNIEL YEHEZKIEL BORNOK HUTABARAT
NRP 5112100212**

**Dosen Pembimbing
Fajar Baskoro, S.Kom, MT.
Rizky Januar Akbar, S.Kom, M.Eng**

**JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016**

[Halaman ini sengaja dikosongkan]



FINAL PROJECT - K141502

***DEVELOPMENT OF BUMP DETECTION SYSTEM
USING ANDROID SMARTPHONE WITH
ACCELEROMETER***

**OTNIEL YEHEZKIEL BORNOK HUTABARAT
NRP 5112100212**

**Dosen Pembimbing
Fajar Baskoro, S.Kom, MT.
Rizky Januar Akbar, S.Kom, M.Eng**

**JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016**

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma dan Pemrograman
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

OTNIEL YEHEZKIEL BORNOK HUTABARAT
NRP : 5112 100 212

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Fajar Baskoro, S.Kom., M.T.

NIP: 19740403 199903 1 002

Rizky Januar Akbar, S.Kom., M.Eng.

NIP: 19870103 201404 1 001



SURABAYA
JUNI 2016

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER

Nama Mahasiswa : Otniel Yehezkiel Bornok Hutabarat
NRP : 5112 100 212
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing 1 : Fajar Baskoro, S.Kom, MT.
Dosen Pembimbing 2 : Rizky Januar Akbar, S.Kom, M.Eng

ABSTRAK

Seiring semakin meningkatnya jumlah polisi tidur yang ilegal atau tanpa izin, maka hal ini dapat menyebabkan ketidaknyamanan oleh pengguna jalan. Bahkan pemerintah telah membuat sanksi yang ditulis dalam Peraturan Daerah terhadap pembuat pita pengganggu atau polisi tidur yang tidak memiliki izin dari Kepala Dinas Perhubungan.

Untuk mengatasi masalah tersebut, pada tugas akhir ini dibangun sebuah sistem pendeteksi bump yang terdiri dari aplikasi Android dan peta digital. Pada aplikasi Android, sistem mengumpulkan data dengan mendeteksi guncangan apabila pengguna melewati polisi tidur, lalu mengirim lokasi tersebut ke server. Deteksi guncangan ini memanfaatkan sensor akselerometer pada Android dan pengiriman lokasi menggunakan GPS. Kemudian pada server data diolah dan ditampilkan dalam bentuk peta digital.

Setelah melakukan pengujian, data yang dihasilkan adalah lokasi guncangan (disebabkan oleh polisi tidur atau jalan rusak) dan ditampilkan dalam bentuk peta digital.

Dengan adanya sistem ini, diharapkan informasi lokasi guncangan (polisi tidur maupun kerusakan jalan yang terdeteksi) dapat diperoleh secepat mungkin sehingga pemerintah dapat bertindak dengan lebih cepat dan efektif.

Kata kunci: Deteksi bump, Sensor Akselerometer, Perangkat Bergerak, Monitor jalan.

DEVELOPMENT OF BUMP DETECTION SYSTEM USING ANDROID SMARTPHONE WITH ACCELEROMETER

Student Name : Otniel Yehezkiel Bornok Hutabarat
Student ID : 5112 100 212
Major : Informatics Department FTIf-ITS
Advisor 1 : Fajar Baskoro, S.Kom, MT.
Advisor 2 : Rizky Januar Akbar, S.Kom, M.Eng

ABSTRACT

As an increasing number of bumps that are illegal can cause discomfort to road users. Even the government has made a written local regulation sanction against the makers of speed bumps that do not have the permission of the Head of the Department of Transportation.

To overcome these problems, this final project constructed a mapping system of bump using Android apps and digital maps. On Android, the system collects data by detecting vibration which caused when the user passes through bumps, and then send its event data including location to the server. This detection utilizes the accelerometer sensor on Android and send the location using GPS. At the end, data event is processed and displayed to the digital maps.

After examination, the location of the bump (caused by a bump or damaged roads) displayed in the form of digital maps.

With this system, we hoped the information of bump location (bumps or road damage is detected) can be obtained as soon as possible so that the government can act more quickly and effectively.

Keywords: Bump Detection, Accelerometer sensor, Mobile Sensing, Road Monitoring.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji syukur bagi Tuhan Yesus Kristus, yang oleh karena anugrahNya penulis dapat menyelesaikan tugas akhir yang berjudul “RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER”.

Pengerjaan tugas akhir ini menjadi suatu pengalaman yang baik bagi penulis. Penulis dapat memperoleh banyak pengalaman yang berharga dalam memperdalam dan meningkatkan keilmuan dalam bidang informatika selama perkuliahan di Teknik Informatika ITS.

Tugas akhir ini selesai karena tidak lepas dari bantuan dan dukungan berbagai pihak. Pada kesempatan kali ini penulis ingin mengucapkan terima kasih kepada:

1. Tuhan Yesus. Atas segala kebaikan dan berkatNya bagi penulis.
2. Mama dan Adik penulis, Kezia, Ruth dan Caleb yang memberikan doa, dukungan dan semangat untuk penulis
3. Bapak Fajar Baskoro selaku dosen pembimbing 1 yang telah membantu dan membimbing penulis dalam menyelesaikan tugas akhir ini.
4. Bapak Rizky Januar Akbar selaku dosen pembimbing 2 yang telah membantu dan membimbing penulis dalam menyelesaikan tugas akhir ini.
5. Anak-anak grup *Illuminati* yang selalu menjadi penghibur dan teman nongkrong saat jenuh mengerjakan tugas akhir.
6. Teman-teman admin lab yang membantu dan menemani selama pembuatan tugas akhir ini.
7. Teman-teman angkatan 2012 yang menjadi semangat bagi penulis selama masa perkuliahan di Teknik Informatika ITS.
8. Romasta Hutagaol, yang tidak bosan-bosannya memberi semangat kepada penulis untuk pengerjaan tugas akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juni 2016

Otniel Yehezkiel Bornok Hutabarat

DAFTAR ISI

LEMBAR PENGESAHAN.....	vii
ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR.....	xiii
DAFTAR ISI.....	xv
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxi
DAFTAR KODE SUMBER	xxiii
1 BAB I PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Tujuan.....	3
1.3. Rumusan Permasalahan.....	4
1.4. Batasan Permasalahan	4
1.5. Metodologi	4
1.6. Sistematika Penulisan.....	6
2 BAB II DASAR TEORI.....	9
2.1. Jalan.....	9
2.1.1. Pengelompokan Jalan	9
2.1.2. Tipe Jalan Berdasarkan Perkerasan Jalan.....	11
2.1.3. Jenis Kerusakan Jalan.....	12
2.2. Alat Pembatas Kecepatan.....	12
2.3. Android.....	14
2.3.1. Komponen Aplikasi pada Android.....	15
2.3.2. Sensor pada Android	17
2.4. PostgreSQL	18
2.5. GPS (<i>Global Positioning System</i>)	19
2.6. PHP (Hypertext Preprocessor).....	21
2.7. Python.....	21
2.8. Google Maps API.....	22
2.9. CodeIgniter.....	23

2.10. Slim	23
2.11. <i>Decision tree</i>	24
2.12. Algoritma <i>Z-Thresh</i>	24
2.13. BIRCH.....	25
3 BAB III ANALISIS DAN PERANCANGAN.....	27
3.1. Deskripsi Umum Sistem.....	27
3.1.1. Analisis Masalah	28
3.1.2. Analisis Kebutuhan Sistem	28
3.2. Perancangan Sistem.....	35
3.2.1. Perancangan Basis Data	35
3.2.2. Perancangan <i>Web Service</i>	39
3.2.3. Perancangan Antarmuka.....	41
3.2.4. Perancangan Proses Data.....	46
4 BAB IV IMPLEMENTASI.....	53
4.1. Lingkungan Implementasi	53
4.1.1. Lingkungan Implementasi Perangkat Keras.....	53
4.1.2. Lingkungan Implementasi Perangkat Lunak.....	53
4.2. Lingkungan Implementasi Antarmuka	54
4.2.1. Antarmuka Peta Digital pada <i>Server</i>	54
4.2.2. Antarmuka Peta pada Android	55
4.2.3. Antarmuka Pengumpulan Data pada Android.....	56
4.2.4. Antarmuka Verifikasi Data.....	57
4.3. Implementasi Basis Data	58
4.3.1. Implementasi Tabel <i>Location</i>	58
4.3.2. Implementasi Tabel <i>Accelerometer</i>	59
4.3.3. Implementasi Tabel Jenis	60
4.3.4. Implementasi Tabel <i>Users</i>	60
4.4. Implementasi <i>Web Service</i>	61
4.4.1. Implementasi <i>Query User</i>	61

4.4.2. Implementasi <i>Query Event Bump</i>	62
4.4.3. Implementasi Koneksi Basis Data	63
4.5. Implementasi Proses Aplikasi	64
4.5.1. Implementasi Inisialisasi Sensor	64
4.5.2. Implementasi Pengiriman Data	64
4.5.3. Implementasi Mengambil Data Sensor	65
4.5.4. Implementasi Reorientasi Akselerometer	65
4.5.5. Implementasi Deteksi <i>Event Bump</i>	66
4.5.6. Implementasi <i>Decision tree</i>	67
4.5.7. Implementasi <i>Clustering</i> pada Peta Digital	68
5 BAB V PENGUJIAN DAN EVALUASI	69
5.1. Lingkungan Uji Coba	69
5.2. Dasar Pengujian	69
5.3. Uji Coba Fungsionalitas	70
5.3.1. Uji Coba Fungsionalitas Melihat Peta Digital	71
5.3.2. Uji Coba Fungsionalitas Mengumpulkan Data	73
5.3.3. Uji Coba Fungsionalitas Verifikasi Data	75
5.4. Analisa Data	76
5.4.1. Analisa Data pada Kondisi Normal	76
5.4.2. Analisa Data pada Kondisi Jalan Rata	78
5.4.3. Analisa Data pada Polisi Tidur	79
5.5. Pengujian Akurasi	81
5.6. Evaluasi Pengujian	87
5.6.1. Evaluasi Pengujian Fungsionalitas	87
5.6.2. Evaluasi Pengujian Akurasi	87
6 BAB VI PENUTUP	91
6.1. Kesimpulan	91
6.2. Saran	92
7 DAFTAR PUSTAKA	93

BIODATA PENULIS.....97

DAFTAR GAMBAR

Gambar 2.1 Jalan Perkerasan Kaku.....	11
Gambar 2.2 Jalan Aspal.....	11
Gambar 2.3 Jalan <i>Paving Block</i>	12
Gambar 2.4 Desain Standar Polisi Tidur.....	13
Gambar 2.5 Sistem Koordinat (Relatif Terhadap Perangkat).....	18
Gambar 2.6 Ide Dasar GPS <i>Positioning</i>	20
Gambar 2.7 Diagram Alir Algoritma <i>Z-Thresh</i>	25
Gambar 2.8 Visualisasi Birch Clustering	26
Gambar 3.1 Alur Umum Sistem.....	28
Gambar 3.2 Diagram Kasus Penggunaan.....	29
Gambar 3.3 Diagram Aktivitas Kasus Penggunaan UC-01	31
Gambar 3.4 Diagram Alir Kasus Penggunaan UC-01.....	31
Gambar 3.5 Diagram Aktivitas Kasus Penggunaan UC-02	32
Gambar 3.6 Diagram Alir Kasus Penggunaan UC-02.....	33
Gambar 3.7 Diagram Aktivitas Kasus Penggunaan UC-03	34
Gambar 3.8 Diagram Alir Kasus Kegunaan UC-03	34
Gambar 3.9 CDM (<i>Conceptual Data Model</i>) pada <i>web service</i> ..	36
Gambar 3.10 PDM (<i>Physical Data Model</i>) pada <i>web service</i>	36
Gambar 3.11 Rancangan Antarmuka pada Peta Digital	42
Gambar 3.12 Rancangan Antarmuka <i>List Bump</i>	42
Gambar 3.13 Rancangan Antarmuka Peta pada Android.....	43
Gambar 3.14 Antarmuka pada Aplikasi	44
Gambar 3.15 Rancangan Antarmuka Verifikasi Data.....	45
Gambar 3.16 Rancangan Halaman Login Verifikasi	46
Gambar 3.17 Diagram Alir Proses Data.....	47
Gambar 3.18 Sistem Koordinat Global	48
Gambar 3.19 Sistem Koordinat Kendaraan.....	49
Gambar 3.20 Contoh Hasil <i>Clustering</i> pada Peta Digital.....	52
Gambar 4.1 Tampilan Peta Digital Hasil Deteksi <i>Bump</i>	54
Gambar 4.2 Tampilan Halaman <i>List Bump</i>	55
Gambar 4.3 Tampilan Peta pada Android	55
Gambar 4.4 Tampilan Aplikasi Utama.....	56
Gambar 4.5 Tampilan Verifikasi Data	57

Gambar 4.6 Halaman Verifikasi Login	58
Gambar 5.1 Tampilan Lokasi <i>Bump</i> pada Peta Digital	72
Gambar 5.2 Tampilan <i>List Bump</i>	72
Gambar 5.3 Tampilan Peta Digital pada Aplikasi Android.....	73
Gambar 5.4 Tampilan Antarmuka Utama Aplikasi Android	74
Gambar 5.5 Tampilan Halaman Verifikasi Data	76
Gambar 5.6 Grafik Data Akselerometer Saat Tidak Bergerak....	77
Gambar 5.7 Grafik Data Akselerometer pada Sumbu X dan Y di Jalan Rata	78
Gambar 5.8 Grafik Data Akselerometer pada Sumbu Z di Jalan Rata.....	79
Gambar 5.9 Grafik Data Akselerometer pada Polisi Tidur	80
Gambar 5.10 Lokasi Polisi Tidur untuk Pengujian	82
Gambar 5.11 Hasil Pengujian P-26 pada Peta Digital.....	87

DAFTAR TABEL

Tabel 3.1 Deskripsi Kasus Penggunaan	30
Tabel 3.2 Rincian Alur Kasus Penggunaan UC-01	30
Tabel 3.3 Rincian Alur Kasus Penggunaan UC-02	32
Tabel 3.4 Rincian Alur Kasus Penggunaan UC-03	33
Tabel 3.5 Atribut Tabel Data Location.....	37
Tabel 3.6 Atribut Tabel Data Accelerometer	38
Tabel 3.7 Atribut Tabel Jenis	38
Tabel 3.8 Atribut Tabel Users	39
Tabel 3.9 Parameter <i>Request</i> Menerima <i>Event Bump</i>	40
Tabel 3.10 Parameter <i>Response</i> Menerima <i>Event Bump</i>	40
Tabel 3.11 Parameter <i>Request</i> Menerima Id <i>User</i>	41
Tabel 3.12 Parameter <i>Response</i> Menerima Id <i>User</i>	41
Tabel 4.1 Lingkungan Implementasi Perangkat Keras.....	53
Tabel 4.2 Lingkungan Implementasi Perangkat Lunak.....	53
Tabel 5.1 Tabel Skenario Uji Coba Melihat Peta Digital.....	71
Tabel 5.2 Tabel Skenario Uji Coba Mengumpulkan Data	74
Tabel 5.3 Tabel Skenario Uji Coba Verifikasi Data.....	75
Tabel 5.4 Tata Cara Pengujian	81
Tabel 5.5 Tabel Skenario Pengujian.....	82
Tabel 5.6 Hasil Uji Coba	85
Tabel 5.7 Rangkuman Hasil Pengujian	87

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

Kode Sumber 4.1 Implementasi Tabel <i>Location</i>	59
Kode Sumber 4.2 Implementasi Tabel <i>Accelerometer</i>	60
Kode Sumber 4.3 Implementasi Tabel Jenis	60
Kode Sumber 4.4 Implementasi Tabel <i>Users</i>	61
Kode Sumber 4.5 Implementasi <i>Query User</i>	62
Kode Sumber 4.6 Implementasi <i>Query Event Bump</i>	63
Kode Sumber 4.7 Implementasi Koneksi Basis Data	63
Kode Sumber 4.8 Implementasi Inisialisasi Sensor	64
Kode Sumber 4.9 Implementasi Fungsi <i>addToJSONArray</i>	65
Kode Sumber 4.10 Implementasi Fungsi <i>postEventData</i>	65
Kode Sumber 4.11 Implementasi Mengambil Data Sensor	65
Kode Sumber 4.12 Implementasi Reorientasi Akslerometer	66
Kode Sumber 4.13 Implementasi Deteksi <i>Event Bump</i>	67
Kode Sumber 4.14 Implementasi Klasifikasi <i>Decision tree</i>	67
Kode Sumber 4.15 Implementasi <i>Clustering</i> pada Peta Digital ..	68

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Polisi tidur adalah bagian jalan yang ditinggikan berupa tambahan aspal atau semen yang dipasang melintang di jalan untuk pertanda meperlambat laju/kecepatan kendaraan. Tujuan utama dari polisi tidur sendiri adalah untuk mencegah terjadinya kecelakaan pada saat aktivitas transportasi. Polisi tidur banyak ditemukan di daerah pemukiman penduduk, perumahan, terminal atau pasar. Namun pada kenyataannya banyak sekali polisi tidur yang dibuat tidak sesuai dengan desain polisi tidur yang diatur berdasarkan Keputusan Menteri Perhubungan No. 3 Tahun 1994 sehingga dapat membahayakan keamanan dan kenyamanan pengguna jalan.

Salah satu pendekatan untuk mengidentifikasi polisi tidur yang memiliki izin adalah adanya laporan dari pihak pembuat polisi tidur dengan pihak yang berwenang. Untuk setiap daerah memiliki peraturan masing-masing untuk menindaklanjuti Keputusan Menteri Perhubungan tersebut. Misalnya untuk daerah jakarta diatur oleh Peraturan Daerah Provinsi DKI yaitu berdasarkan pasal 53 huruf b Perda DKI Jakarta 12/2003, setiap orang tanpa izin dari Kepala Dinas Perhubungan dilarang membuat atau memasang tanggul pengaman jalan dan pita penghaduh (*speed trap*). Pelanggaran terhadap ketentuan tersebut adalah kurungan paling lama 3 bulan atau denda sebanyak-banyaknya Rp5.000.000,00 (lima juta rupiah). Oleh karena itu, untuk mempermudah pemerintah mengetahui polisi tidur yang memiliki izin atau ilegal, pada tugas akhir ini saya menawarkan sistem

pendeteksi *bump* menggunakan *smartphone* Android dengan pendekatan survei otomatis.

Berdasarkan penelitian sebelumnya yang telah dilakukan adalah Pothole Patrol (P2) [1], menggunakan sensor *3-axis accelerometer* dan GPS diletakkan pada kendaraan taksi untuk memonitor keadaan permukaan jalan. Pothole Patrol dapat mendeteksi lubang dan anomali jalan dengan cara mengolah data dari getaran atau *vibrasi* pada akselerometer dan GPS. Proses pengolahan datanya menggunakan filter pengolahan sinyal untuk menolak *event* selain lubang (lubang got, celah jalan, rel kereta api, *expansion joints*). P2 juga menggunakan metode *blacklist* pada lokasi *speedbump* agar tidak terdeteksi sebagai lubang dengan menggunakan peta *speedbump* dari arsip data geografis di daerah penelitian P2. Penelitian ini juga mengklasifikasi data berdasarkan lokasi untuk mengurangi kesalahan dalam. Pada penelitian selanjutnya yaitu CRSM [2] atau disebut juga *crowdsourcing-based road surface monitoring system* mampu mendeteksi lubang dan mengevaluasi tingkat kekasaran permukaan jalan. Namun penelitian ini mengalami kesulitan dalam membedakan *event decelerating belts* atau polisi tidur dengan lubang karena memiliki getaran yang mirip. Sehingga CRSM menggunakan informasi geografis keadaan jalan untuk mengeliminasi tipe *event* seperti lubang got dan polisi tidur berdasarkan informasi GPS.

Penelitian yang menggunakan sensor *smartphone* adalah Wolverine [3]. Wolverine menggunakan sensor akselerometer untuk mengumpulkan data untuk mendeteksi *event bump* dan rem. Perangkatnya direorientasi terlebih dahulu sehingga dapat digunakan pada posisi apapun di dalam kendaraan. Proses reorientasinya menggunakan sensor akselerometer dan magnetometer. Data akselerometer yang dikumpulkan kemudian diolah menggunakan *Support Vector Machine* (SVM) untuk menentukan kondisi jalan mulus atau bergelombang dan kondisi rem atau tidak. Salah satu penelitian yang mendeteksi lubang secara *real time* adalah Mednis et al., [4] mengusulkan sistem yang menggunakan *smartphone* Android beserta sensor

akselerometranya untuk mendeteksi *events* secara *real time*. Sistem ini mengumpulkan data secara *off-line post-processing* dengan menggunakan algoritma *Z-Thresh*, *Z-Diff*, *STDEV(Z)* dan *G-Zero*.

Pendekatan survei otomatis yang diusulkan pada tugas akhir ini dapat dilakukan dengan menggunakan *embedded sensing devices* atau *smartphone*. Pada tugas akhir ini akan berfokus pada proses data akselerometer untuk mendeteksi polisi tidur menggunakan *smartphone* Android Metode untuk mendeteksi *bump* menggunakan kombinasi metode *real-time* dan *nonreal-time*. Metode *real-time* menggunakan algoritma *Z-Thresh* seperti pada penelitian pada Mednis et al namun perbedaannya adalah pada tugas akhir ini juga menggunakan metode klasifikasi *decision tree* pada data yang terdeteksi menggunakan algoritma *Z-Thresh* di *server* untuk menghasilkan deteksi yang lebih akurat.

Data GPS lokasi polisi tidur akan dikirim dari *smartphone* Android ke *server*. Aplikasi ini diharapkan akan digunakan oleh banyak pengguna jalan raya untuk mempercepat proses pemetaan polisi tidur pada sistem. Lalu pada *server* akan menandai lokasi polisi tidur dari data-data lokasi GPS yang dikirimkan dari *smartphone* pengguna.

Data-data GPS lokasi polisi tidur yang dikirim dari para pengguna diolah untuk mengestimasi lokasi polisi tidur dengan lebih akurat dan ditampilkan dalam bentuk peta digital. Dengan sistem yang *crowdsourced* (menggunakan kumpulan data partisipasi yang terdistribusi) dapat meningkatkan skalabilitas melihat banyaknya jumlah pengguna *smartphone* dan terus meningkat. Harapannya adalah informasi tersebut dapat diperoleh sedini mungkin sehingga dapat digunakan pemerintah untuk mempermudah pengawasan pembuatan polisi tidur maupun perbaikan dan pemeliharaan jalan.

1.2. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat sistem *monitoring* yang memetakan lokasi *bump*.

1.3. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini antara lain:

1. Bagaimana mendeteksi *bump* menggunakan *smartphone* Android?
2. Bagaimana menampilkan lokasi *bump* dalam bentuk peta digital?
3. Bagaimana mengolah estimasi lokasi *bump*?

1.4. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, antara lain:

1. Data yang dikirimkan dari Android ke *server* adalah data akselerometer dan lokasi jalan.
2. *Smartphone* yang digunakan memiliki sensor akselerometer dan Android.
3. Uji coba menggunakan kendaraan sepeda motor.
4. Diujikan pada kendaraan dengan estimasi kecepatan 20-40 km/jam.
5. Aplikasi membutuhkan koneksi internet dan tersambung GPS.
6. Aplikasi yang dibangun adalah aplikasi Android dan peta digital.

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini yaitu:

1. Studi literatur

Tahap ini merupakan proses pengumpulan informasi yang dibutuhkan untuk mengerjakan tugas akhir. Informasi yang dibutuhkan dapat berupa literatur dan dokumentasi penggunaan. Studi literatur yang akan dipelajari untuk tugas akhir ini adalah sebagai berikut:

1. Mekanisme sensor akselerometer pada *smartphone* Android.
2. Pengambilan data training alat pembatas jalan dan penentuannya menggunakan algoritma *Z-Thresh* dan *Decision tree*.
3. Mekanisme pengiriman data lokasi dan data akselerasi dari Android.
4. Menampilkan *bump* pada peta digital.

2. Analisis dan Desain Perangkat Lunak

Tahap ini dilakukan analisis terhadap sistem serta perancangan sistem yang akan dibuat. Tujuannya adalah untuk merumuskan solusi dalam pelaksanaan implementasi pada sistem. Secara garis besar, fitur utama yang terdapat pada program ini adalah:

1. Mendeteksi *bump*
2. Menampilkan pada peta digital

3. Implementasi

Untuk implementasi sistem pemetaan *bump*, implementasi pada tugas akhir ini adalah sebagai berikut:

1. Implementasi Peta Digital
Implementasi pada peta digital yang akan menampilkan *bump* yang akan ditandai pada peta. Bahasa pemrograman yang digunakan adalah HTML, PHP, dan CSS.
2. Implementasi *Server*
Pada *server* akan digunakan basis data Postgresql dan bahasa PHP untuk menyimpan dan mengolah data aktivitas dari sensor akselerometer yang dikirim.
3. Implementasi Aplikasi Android
Aplikasi untuk mendeteksi *bump* diimplementasikan pada *smartphone* Android menggunakan bahasa pemrograman Java.

4. Pengujian dan evaluasi

Pengujian untuk tugas akhir ini sebagai berikut:

1. Melakukan uji coba apabila beberapa pengguna mengendarai motor melewati alat pembatas jalan maka pada peta digital akan ditampilkan lokasi *bump*.
2. Menghitung akurasi dari hasil deteksi pada perangkat lunak.
3. Pada peta digital akan menampilkan hasil lokasi-lokasi *bump* yang terdeteksi dan di-*cluster* menggunakan algoritma BIRCH.

5. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan proses dokumentasi dan pembuatan laporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

1.6. Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Metode Pemecahan Masalah

Bab ini membahas mengenai Metode yang digunakan untuk memecahkan masalah yang dipaparkan pada rumusan permasalahan.

Bab IV Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur dan proses.

Bab V Implementasi

Bab ini berisi implementasi dari perancangan dan implementasi dalam bentuk coding. Bab ini berisi proses pembangunan perangkat lunak

Bab VI Pengujian dan Evaluasi

Bab ini membahas tentang pengujian aplikasi berdasarkan skenario yang telah ditentukan. Mengevaluasi hasil uji coba dari perangkat lunak.

Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dari proses pengembangan perangkat lunak dan hasil uji coba.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah atau kode-kode sumber yang penting pada aplikasi

[Halaman ini sengaja dikosongkan]

BAB II

DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir.

2.1. Jalan

Jalan merupakan prasarana transportasi darat yang meliputi segala bagian jalan, termasuk bangunan pelengkap dan perlengkapannya yang diperuntukkan bagi lalu lintas, yang berada pada permukaan tanah, di atas permukaan tanah, di bawah permukaan tanah dan/atau air, serta di atas permukaan air, kecuali jalan kereta api, jalan lori, dan jalan kabel. Jalan sebagai infrastruktur publik yang bertujuan menghubungkan suatu tempat ke tempat lain dalam satu daratan.

2.1.1. Pengelompokan Jalan

Pengelompokan jalan di wilayah Indonesia diatur menurut UU No. 38 Tahun 2004 tentang Jalan, yaitu:

- Klasifikasi berdasarkan sistem jalan adalah sebagai berikut.
 1. Jalan Primer: Distribusi Lingkup Nasional
 2. Jalan Sekunder: Distribusi Lingkup Perkotaan
- Klasifikasi berdasarkan status atau administrasi pemerintahan sebagai berikut.
 1. Jalan Nasional, adalah jalan arteri dan jalan kolektor pada sistem jaringan jalan primer yang menghubungkan antar ibukota provinsi, dan jalan strategis nasional, serta jalan tol.
 2. Jalan Provinsi, merupakan jalan kolektor dan arteri dalam menghubungkan ibukota provinsi dengan ibukota kabupaten/kota, atau antar ibukota kabupaten/kota dan jalan strategis kabupaten.
 3. Jalan Kabupaten, jalan lokal dalam jaringan jalan primer
 4. Jalan Kota, adalah jalan lokal dalam sistem jaringan jalan primer yang menghubungkan antarpusat pelayanan dalam

kota, persil dan antarpersil serta antarpusat pemukiman yang berada di dalam kota.

- Klasifikasi berdasarkan fungsional jalan adalah sebagai berikut.
 1. Jalan Arteri, yaitu jalan umum yang digunakan oleh angkutan utama (jarak jauh) dengan ciri-ciri perjalanan jarak jauh, kecepatan rata-rata tinggi dan jumlah jalan masuk dibatasi.
 2. Jalan Kolektor, merupakan jalan umum yang digunakan oleh angkutan pengumpul atau pembagi (jarak sedang) dengan ciri-ciri perjalanan jarak sedang, kecepatan rata-rata sedang dan jumlah jalan masuk dibatasi.
 3. Jalan Lokal, yaitu jalan umum yang melayani angkutan setempat (jarak dekat) kecepatan rendah dan jalan masuk tidak dibatasi.
 4. Jalan Lingkungan, merupakan jalan umum yang berfungsi untuk pelayanan angkutan lingkungan dengan ciri perjalanan jarak dekat, dan kecepatan rendah.
- Klasifikasi Jalan berdasarkan beban muatan sumbu atau disebut juga kelas jalan berdasarkan UU No 14 Tahun 1992 tentang Lalu Lintas dan Angkutan Jalan adalah sebagai berikut:
 1. Jalan Kelas I, yaitu jalan arteri yang dapat dilalui kendaraan bermotor lebar muatan tidak melebihi 2.500 milimeter dan panjang 18.000 milimeter, dengan sumbu terberat yang diizinkan lebih dari 10 ton.
 2. Jalan Kelas II, yaitu jalan arteri yang dapat dilalui kendaraan bermotor dengan lebar muatan tidak melebihi 2.500 milimeter dan panjang 18.000 milimeter, dengan sumbu terberat yang diizinkan 10 ton, contoh penggunaan jalan ini adalah yang sesuai dengan angkutan peti kemas.
 3. Jalan Kelas III A, yaitu jalan arteri atau kolektor yang dapat dilalui kendaraan bermotor dengan lebar muatan tidak melebihi 2.500 milimeter dan panjang 18.000 milimeter, dengan sumbu terberat yang diizinkan 8 ton.

4. Jalan Kelas III B, yaitu jalan kolektor yang dapat dilalui kendaraan bermotor dengan lebar muatan tidak melebihi 2.500 milimeter dan panjang 12.000 milimeter, dengan sumbu terberat yang diizinkan 8 ton.

2.1.2. Tipe Jalan Berdasarkan Perkerasan Jalan

Tipe jalan berdasarkan perkerasan jalan adalah jenis jalan yang dibedakan berdasarkan proses pembuatan jalan menggunakan beberapa metode yaitu:

1. Perkerasan Kaku (*Rigid Pavement*)

Merupakan jenis jalan yang terbuat dari beton semen, yaitu terdiri dari plat beton semen sebagai lapis pondasi di atas tanah dasar. Contoh penampakan jalan perkerasan kaku dapat dilihat pada Gambar 2.1.



Gambar 2.1 Jalan Perkerasan Kaku

2. Perkerasan Lentur (*Fleksible Pavement*)

Merupakan jenis jalan yang terbuat dari aspal atau hot mix. Contoh penampakan jalan dengan perkerasan lentur atau jalan aspal dapat dilihat pada Gambar 2.2.



Gambar 2.2 Jalan Aspal

3. Perkerasan Blok (Menggunakan *Paving Block*)
Merupakan jenis jalan yang terbuat dari campuran pasir dan semen atau disebut juga blok beton terkunci. Contoh jalan menggunakan perkerasan blok ditunjukkan pada Gambar 2.3.



Gambar 2.3 Jalan *Paving Block*

2.1.3. Jenis Kerusakan Jalan

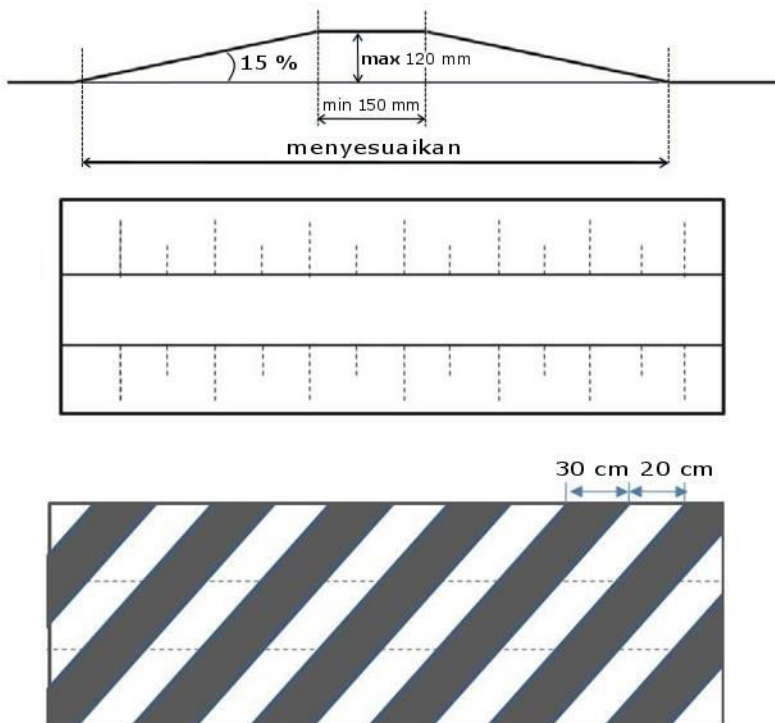
Jenis kerusakan jalan terdapat 2 jenis yaitu kerusakan fungsional dan kerusakan struktural.

1. Kerusakan Fungsional
Kerusakan fungsional merupakan kerusakan pada permukaan jalan yang dapat menyebabkan terganggunya fungsi jalan tersebut.
2. Kerusakan Struktural
Kerusakan pada struktur jalan, sebagian atau seluruhnya yang menyebabkan perkerasan jalan tidak lagi menahan beban yang bekerja di atasnya.

2.2. Alat Pembatas Kecepatan

Polisi tidur (*speed bump*) atau disebut juga sebagai alat pembatas kecepatan adalah bagian jalan yang ditinggikan berupa tambahan aspal atau semen yang dipasang melintang di jalan untuk pertanda memperlambat laju atau kecepatan kendaraan [5]. Polisi tidur umumnya mempunyai ukuran dengan tinggi 7,5 cm sampai 15 cm dan lebar 30-90 cm. Dalam pembuatan polisi tidur terdapat ketentuan yang diatur yaitu ketinggian dan rambu-rambu yang

memberitahu mengenai adanya polisi tidur, khususnya pada saat malam hari, polisi tidur dilengkapi dengan marka jalan dengan garis serong berwarna putih atau kuning kontras sebagai tanda adanya polisi tidur. Hal ini diperlukan untuk meningkatkan keselamatan dan kenyamanan pengguna jalan. Desain standar polisi tidur diatur dalam Keputusan Menteri Perhubungan No. 3 Tahun 1994 tentang Alat Pengendali dan Pengaman Pemakai Jalan. Desain standar polisi tidur di Indonesia dapat dilihat pada Gambar 2.4.



Gambar 2.4 Desain Standar Polisi Tidur

Alat pembatas kecepatan harus diletakkan pada posisi melintang tegak lurus di jalur lalu lintas. Lokasi penempatan pembatas kecepatan adalah sebagai berikut:

- a) Jalan di daerah pemukiman atau Jalan Lingkungan.
- b) Pada daerah jalan yang sedang melakukan pekerjaan konstruksi.
- c) Pada Jalan Lokal yang memiliki Kelas III C.

Pengaturan ketinggian polisi tidur harus diatur agar tidak membahayakan pemakai jalan karena ketinggian dari polisi tidur berkaitan dengan saat melintas maka beban dan berat tubuh bagian atas akan membuat stres signifikan pada struktur tubuh yang rendah dibagian punggung, terutama pada disk antara *lumbalis* kelima dan *vertebra sakral* pertama yang dikenal sebagai L5/S1 *lumbosacral disc* atau pengangkatan beban dengan berat beban tubuh bagian atas yang dapat menyebabkan adanya risiko cedera atau berisiko tinggi bagi para penderita osteoporosis. Ketentuan tentang polisi tidur diatur dalam Keputusan Menteri Perhubungan No: KM. 3 Tahun 1994.

2.3. Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler [6]. Pada tugas akhir ini, aplikasi pendeteksi *bump* akan dibangun pada *smartphone* dengan sistem operasi Android. Aplikasi ini akan dibangun menggunakan Android Studio.

2.3.1. Komponen Aplikasi pada Android

Aplikasi Android menggunakan bahasa pemrograman Java. Android SDK (Software Development Kit) sebagai *compiler* kode aplikasi. Komponen aplikasi adalah bagian penting dalam pengembangan aplikasi Android. Komponen dapat berupa *entry point*, yaitu kontrol penghubung antara sistem operasi dengan program aplikasi. Setiap komponen memiliki ketergantungan dengan komponen yang lain. Berikut ini adalah 4 tipe komponen aplikasi [7].

2.3.1.1. Activities

Activity merepresentasikan sebuah tampilan pada antarmuka pengguna. Misalnya, aplikasi email memiliki satu *activity* yaitu untuk membuat email. Namun pada satu aplikasi dapat terdapat beberapa *activity* yang bekerjasama untuk membuat aplikasi yang dapat digunakan oleh *user* dengan mudah. Sebuah *activity* dapat diimplementasikan menggunakan subclass *Activity*.

2.3.1.2. Services

Komponen *service* merupakan komponen yang bekerja pada latarbelakang untuk menjalankan sebuah operasi. *Service* tidak menyediakan antarmuka pengguna. Contoh penggunaan ini adalah menjalankan musik dilatarbelakang (*background*). Sebuah *service* dapat diimplementasikan menggunakan subclass *Services*.

2.3.1.3. Content Providers

Content provider berguna untuk mengatur seperangkat data aplikasi. Data dapat disimpan dalam sistem, SQLite, dan penyimpanan persisten lainnya. Melalui *content provider*, aplikasi lain dapat mengambil atau memodifikasi data. *Content provider* berguna untuk membaca dan menulis data yang rahasia dan tidak boleh dibagikan, misalnya aplikasi notepad menggunakan *content provider* untuk menyimpan catatan.

2.3.1.4. Broadcast Receivers

Broadcast receiver adalah komponen yang merespon pengumuman *broadcast* ke semua sistem. Contoh *broadcast* yang berasal dari sistem adalah *broadcast* yang muncul ketika layar telah dimatikan, baterai sedikit, dan lain lain. Aplikasi ini juga

dapat memulai *broadcast*, misalnya memberitahu aplikasi lain bahwa data telah diunduh dan siap untuk digunakan. Pada umumnya *broadcast receiver* berguna sebagai pintu gerbang untuk komponen lain dan dimaksudkan untuk melakukan jumlah pekerjaan yang sedikit.

2.3.1.5. SensorEventListener

SensorEventListener merupakan antarmuka publik yang terdapat pada Android yang berfungsi untuk menerima pemberitahuan pembaharuan data dari *sensorManager*. *Public Method* yang digunakan ada 2 yaitu *onAccuracyChanged* dan *onSensorChanged*. *Public Method onAccuracyChanged* adalah *listener* aplikasi untuk perubahan akurasi pada sensor yang sedang digunakan, sedangkan *onSensorChanged* adalah *listener* perubahan nilai sensor.

2.3.1.6. LocationListener

LocationListener [8] adalah penghubung antara aplikasi dengan GPS. *LocationListener* terdiri dari *LocationManager* dan *LocationListener*. *LocationManager* adalah layanan akses lokasi pada Android. Salah satu layanan yang tersedia adalah memperbarui posisi lokasi dimana *smartphone* tersebut sedang digunakan. *Method* ini dipanggil jika *LocationListener* telah terdaftar dengan *LocationManager* menggunakan *requestLocationUpdates()*. Berikut adalah *Public Method* yang digunakan.

1. *onLocationChanged*, fungsi ini dipanggil ketika lokasi berubah. Nilai garis bujur dan garis lintang akan diperbarui ketika fungsi ini dipanggil dalam bentuk objek *LocationListener*.
2. *onProviderDisabled*, digunakan ketika provider dinonaktifkan oleh pengguna.
3. *onProviderEnabled*, digunakan ketika provider diaktifkan oleh pengguna.
4. *onStatusChanged*, dipanggil saat ada perubahan status pada provider.

2.3.1.7. Volley

Volley adalah HTTP *library* yang membuat sistem jaringan aplikasi Android menjadi lebih mudah dan cepat. Volley tersedia pada repositori terbuka AOSP [9]. Volley menawarkan berbagai keuntungan yaitu, penjadwalan otomatis pada permintaan jaringan, mendukung banyak koneksi secara bersamaan, mendukung prioritas permintaan, dapat melakukan pemberhentian permintaan, ringan untuk dimodifikasi sesuai kebutuhan.

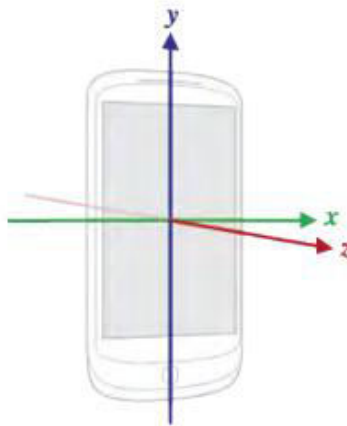
Volley unggul dalam hal operasi bertipe RPC yang digunakan untuk mengisi suatu antarmuka, misalnya mengambil halaman dari suatu pencarian yang terdiri dari data struktur. Volley mudah diintegrasikan dengan berbagai protokol dan keluar dari kotak dengan mendukung rangkaian atau *string* mentah, gambar dan JSON. Dengan menyediakan *built-in* dukungan untuk fitur yang dibutuhkan, Volley membebaskan *developer* untuk menulis kode *boilerplate* dan memperbolehkan *developer* untuk fokus dalam logika pengembangan aplikasi.

2.3.2. Sensor pada Android

Kebanyakan perangkat Android telah dilengkapi oleh sensor yang dapat mengukur gerakan, orientasi, dan berbagai jenis keadaan lingkungan sekitarnya. Sensor ini mampu menyediakan data mentah dengan akurat dan presisi. Sensor yang digunakan pada tugas akhir ini adalah sensor akselerometer dan magnetometer.

2.3.2.1. Sensor Akselerometer

Sensor akselerometer menghitung percepatan yang terjadi pada device termasuk gaya gravitasi. Akselerometer menggunakan standard sensor koordinat sistem sesuai pada Gambar 2.5. Pada tugas akhir ini, sensor akselerometer pada Android akan digunakan untuk memperoleh data akselerasi. Data akselerasi yang digunakan adalah sumbu z yang telah direorientasi dari sumbu koordinat *device* Android menjadi koordinat bumi.



Gambar 2.5 Sistem Koordinat (Relatif Terhadap Perangkat)

2.3.2.2. Sensor Magnetometer

Sensor magnetometer atau disebut juga sensor medan magnet berfungsi untuk melaporkan medan magnet disekitar perangkat *smartphone* yang terukur dalam sensor dengan 3 sumbu. Pengukuran ditampilkan dalam bidang x, y, dan z dalam satuan micro-Tesla (uT). Pada tugas akhir ini sensor magnetometer akan digunakan untuk memperoleh nilai matriks rotasi pada *devices*.

2.4. PostgreSQL

PostgreSQL [10] adalah perangkat lunak *open-source object-relational database management system* dengan penekanan pada ekstensibilitas dan aturan-standar. PostgreSQL menawarkan kombinasi unik fitur-fitur yang lebih baik jika dibandingkan dengan basis data komersial lainnya seperti Sysbase, Oracle dan DB2. PostgreSQL tidak dimiliki oleh perusahaan manapun. PostgreSQL dikembangkan, dipelihara, diperbaiki oleh grup-grup *developer* yang sukarelawan di seluruh dunia.

PostgreSQL menawarkan semua kebutuhan fitur dari basis data relasional ditambah beberapa fitur unik. PostgreSQL menawarkan *inheritance*, sehingga pengembang dapat menambahkan tipe data tertentu sesuai yang diinginkan ke PostgreSQL. PostgreSQL juga mendukung tipe data geometris seperti *point*, *line segment*, *polygon*, *box* and *circle*. PostgreSQL menggunakan sistem indeks struktur yang membuat tipe data geometris cepat diakses.

PostgreSQL bersifat *extended* atau dapat dikembangkan (membuat fungsi baru, tipe data baru, operator baru) sesuai dengan bahasa pemrograman yang diinginkan. PostgreSQL dibangun dengan arsitektur *client/server*. Aplikasi pada sisi *client* dapat dibangun dengan berbagai bahasa termasuk C, C++, Java, Python, Perl, TCL/Tk dan lain-lain. Pada sisi *server*, PostgreSQL mendukung bahasa prosedural yang baik yaitu PL/pgSQL. Pada tugas akhir ini, basis data PostgreSQL digunakan sebagai penyimpanan data *accelerometer*, waktu dan posisi koordinat terjadinya *bump*.

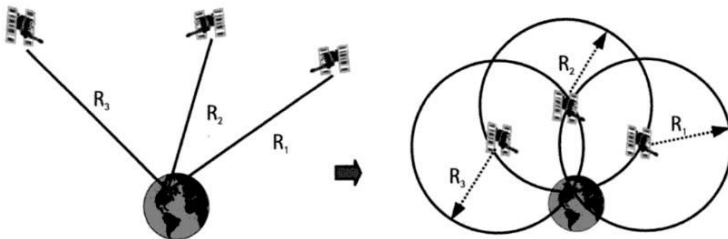
2.5. GPS (*Global Positioning System*)

GPS adalah sistem navigasi berbasis ruang yang menyediakan informasi lokasi dan waktu dalam segala kondisi cuaca, dimanapun di bumi yang tak terhalang pandangan 4 atau lebih satelit GPS. Sistem ini menggunakan 24 satelit yang mengirimkan gelombang mikro ke bumi. Sinyal ini kemudian diterima oleh alat penerima di permukaan bumi. Data yang diterima itu kemudian digunakan untuk menentukan posisi, kecepatan, arah dan waktu. Satelit GPS mengitari bumi secara sirkular (mendekati bentuk elips) dengan inklinasi sebesar 55 derajat [11].

GPS pada awalnya dikelola oleh *US Air Force* (USAF) sebagai operator sistem. GPS yang pada awalnya dikembangkan untuk sistem kekuatan tambahan bagi militer memiliki potensi menguntungkan bagi masyarakat sipil. Pada akhirnya GPS menyediakan dua layanan, yaitu layanan *Precise Positioning Service* (PPS) yang berguna untuk kekuatan militer AS dan layanan

Standard Positioning Service (SPS) yang didesain untuk memberikan kemampuan *positioning*.

GPS memiliki 3 segmen, yaitu segmen angkasa, segmen kontrol, dan segmen pengguna. Segmen angkasa terdiri dari 24-satelit yang telah dijelaskan sebelumnya. Setiap satelit GPS secara terus menerus akan mentransmisikan gelombang sinyal radio mikro yang terdiri dari dua *carriers*, dua kode dan pesan navigasi. Ketika GPS *receiver* diaktifkan, GPS *receiver* akan mengambil sinyal melalui antena penerima. Setelah menerima sinyal GPS, sinyal tersebut akan diproses menggunakan software yang ada didalam perangkat tersebut. Secara teori hanya dibutuhkan informasi tiga jarak antara GPS *receiver* dengan satelit, artinya dibutuhkan sebanyak minimal tiga satelit yang melacak secara bersamaan. Pada kasus ini, penerima berada pada persimpangan dari tiga lingkup; masing-masing memiliki jarak radius seperti ditunjukkan pada Gambar 2.6.



Gambar 2.6 Ide Dasar GPS Positioning

Pada segmen kontrol sistem GPS terdiri dari stasiun pelacak yang tersebar pada jaringan seluruh dunia, dengan kontrol utamanya berada di negara Amerika Serikat di Colorado Springs, Colorado. Tujuan utama segmen kontrol adalah melacak satelit GPS dengan maksud untuk menentukan dan memprediksi lokasi satelit, integritas satelit, perilaku satelit, jam atom, data atmosfer dan lain-lain. Pada segmen pengguna sendiri terdiri dari seluruh bagian militer dan pengguna sipil. Dengan adanya GPS *receiver*

yang tersambung pada antena GPS, pengguna dapat menerima sinyal GPS, yang dapat digunakan untuk menentukan posisi dirinya dimanapun ia berada. GPS sekarang ini tersedia kepada semua pengguna diseluruh dunia dengan gratis.

2.6. PHP (Hypertext Preprocessor)

PHP merupakan bahasa pemrograman *server-side* yang digunakan untuk pengembangan *web* dan bahasa yang digunakan secara luas dalam pengembangan *website*. PHP dikembangkan pertama kali oleh Rasmus Lerdorf pada tahun 1995. Secara resmi dirilis pada tahun 1997. Fungsionalitas dasar pemrograman berorientasi objek ditambahkan pada PHP 3 dan ditingkatkan pada PHP 4. PHP telah digunakan lebih dari 200 juta *website* yang berbeda, dari *website* personal biasa sampai perusahaan raksasa seperti Facebook, Wikipedia, Tumblr, Slack dan Yahoo.

PHP [12] didesain untuk membuat halaman *web* yang dinamis. PHP berguna sebagai bahasa pemrograman scripting yang dikhususkan untuk pengembangan *web server-side*. PHP memiliki peran utama sebagai filter, yaitu mengambil *file* atau *stream* yang mengandung teks dan atau instruksi PHP dan menghasilkan data *stream* lain. PHP dijalankan pada komputer untuk digunakan oleh seorang pengguna, biasanya dijalankan pada *web server* dan diakses oleh banyak pengguna yang menggunakan *web browser*. PHP memiliki kelebihan yaitu menyembunyikan kompleksitas. Pada tugas akhir ini, PHP akan digunakan untuk membangun aplikasi *web service*.

2.7. Python

Python adalah bahasa pemrograman yang digunakan secara luas sebagai bahasa pemrograman yang *high-level*, bertujuan umum, dapat diinterpretasi, dan dinamis [13]. Python didesain dengan filosofi penekanan terhadap kode yang dapat dibaca dan sintaks yang membolehkan pengembang untuk mengeskpresikan konsep hanya dengan sedikit baris kode dibandingkan pada bahasa lain seperti C++ dan Java. Bahasa

pemrograman ini menyediakan gagasan yang memungkinkan untuk pengembangan program dalam skala kecil maupun besar.

Python mendukung berbagai pola pemrograman, termasuk pemrograman berorientasi objek, imperatif, dan pemrograman fungsional atau bergaya prosedural. Python memiliki fitur tipe sistem yang dinamis dan manajemen memori yang otomatis dan memiliki *library* standart yang besar dan komprehensif. Interpreter Python disediakan untuk banyak sistem operasi, sehingga memungkinkan kode Python untuk berjalan dalam sistem yang luas dan bervariasi. Python sangat banyak digunakan untuk pekerjaan kecerdasan buatan. Pada tugas akhir ini, Python digunakan dalam implementasi klasifikasi *decision tree* dan *clustering* BIRCH menggunakan *library sklearn*.

2.8. Google Maps API

Google Maps API [14] adalah layanan oleh Google yang dapat digunakan untuk menampilkan data dalam bentuk peta digital melalui beberapa model fitur. Google Maps API menyediakan *web service* sebagai antarmuka untuk permintaan data Maps API dari layanan luar dan digunakan pada aplikasi Maps pengguna. Layanan *web* ini menggunakan HTTP *request* ke URL yang spesifik, meneruskan parameter URL sebagai argumen ke layanan.

Pada umumnya, layanan ini mengembalikan data dengan HTTP *request* berupa JSON atau XML untuk diuraikan dan atau diproses oleh aplikasi. Fitur *web service* Google Maps API adalah sebagai berikut.

1. *Direction* API
2. *Distance Matrix* API
3. *Elevation* API
4. *Geolocation* API
5. *Road* API
6. *Time Zone* API
7. *Places* API

Fitur yang dapat digunakan pada pengembangan sistem tugas akhir ini adalah menampilkan peta dan menandai peta menggunakan *marker*. *Library* yang digunakan adalah Google Maps V3 API Biostall dan dikembangkan menggunakan kerangka kerja CodeIgniter [15].

2.9. CodeIgniter

CodeIgniter [16] adalah perangkat lunak sumber terbuka untuk pengembangan kerangka kerja *web* secara cepat. CodeIgniter digunakan untuk membangun *web* site yang dinamis menggunakan PHP. CodeIgniter merupakan kerangka kerja yang menggunakan pola pengembangan *model-view-controller* (MVC). CodeIgniter juga dapat diubah menjadi *Hierarchical Model View Controller* (HMVC) yang berguna untuk pemeliharaan grup modular *controller*, *models* dan *view* yang disusun dalam format satu subdirektori.

CodeIgniter tercatat memiliki kecepatan yang lebih baik dibandingkan dengan kerangka kerja PHP lainnya. CodeIgniter dirilis pertama kali oleh EllisLab pada 28 Februari 2006. Pada 6 Oktober 2014, EllisLab mengumumkan bahwa pengembangan CodeIgniter dilanjutkan oleh British Columbia Institute of Technology. Pada tugas akhir ini CodeIgniter digunakan untuk menampilkan pengolahan data dan peta digital.

2.10. Slim

Slim [17] adalah kerangka kerja mikro berbahasa PHP yang membantu dalam pembuatan *web* aplikasi dan APIs dengan cepat. Slim menyediakan fitur-fitur yang lengkap dan menggunakan kode yang sederhana. Slim memiliki 4 fitur utama yaitu *HTTP Router*, *Middleware*, *PSR-7 Support* dan *Dependency Injection*.

HTTP Router adalah fitur Slim yang menyediakan mekanisme *router* yang cepat dan *powerful* yang memetakan rute ke *HTTP Method* yang spesifik dan URIs. Slim juga mendukung parameter dan *pattern matching*. Fitur *Middleware* memungkinkan *developer* untuk membangun aplikasi yang memudahkan penarikan objek

HTTP *request* dan *response*. Fitur PSR-7 adalah Slim mendukung implementasi HTTP *message* sehingga *developer* dapat mengetahui dan memanipulasi HTTP *message method, status, uri, headers, cookies* dan *body*. *Dependency Injection* adalah fitur pada Slim yang berguna untuk mempermudah *developer* dalam mengendalikan alat bantu eksternal dengan menggunakan Container-Interop.

2.11. *Decision tree*

Decision tree adalah metode klasifikasi yang mengubah data menjadi pohon keputusan dan aturan keputusan. *Decision tree* atau pohon keputusan juga memperhatikan faktor-faktor kemungkinan yang mempengaruhi proses pengambilan keputusan tersebut. Pohon keputusan sangat populer karena mudah diinterpretasi oleh manusia. Model prediksi yang digunakan adalah struktur pohon atau struktur hierarki. Algoritma yang digunakan pada *decision tree* ini adalah *CART algorithm*. *CART tree* [18] adalah sebuah pohon keputusan *binary* regresi non parametrik untuk klasifikasi atau yang dibangun dengan membagi suatu *node* induk menjadi 2 *node* anak secara berulang dimulai dari *root node* berdasarkan hasil *learning* data sampel.

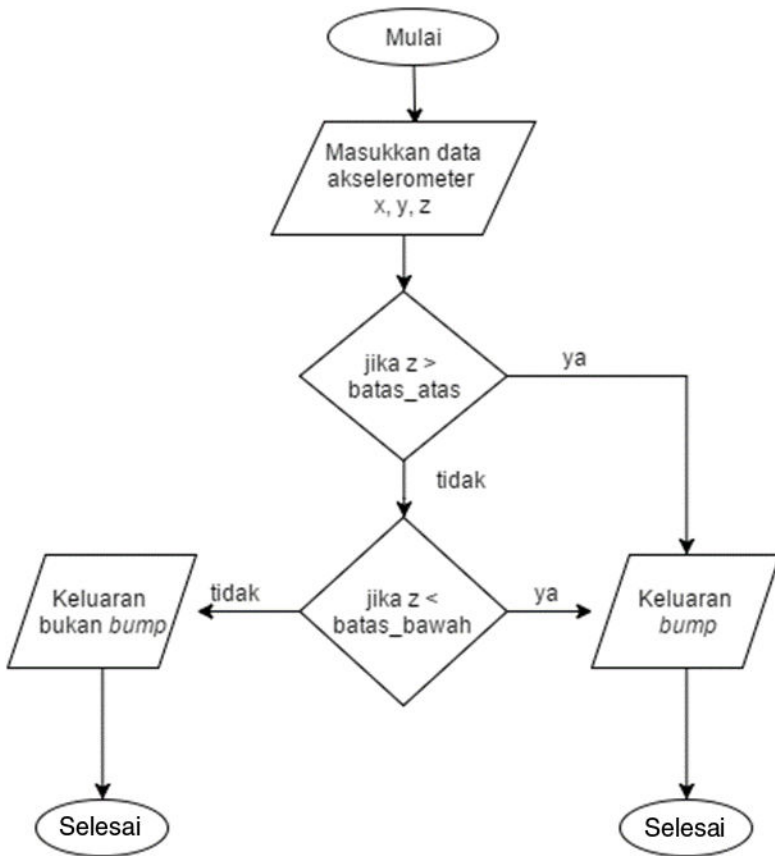
Tahapan yang digunakan dalam membangun pohon keputusan adalah sebagai berikut:

1. Cari pemisah variabel *predictor* terbaik.
2. Cari *node* pemisah terbaik. Dari pemisah terbaik pada langkah pertama, pilih satu yang paling baik sebagai pemisah kategori atau kriteria.
3. Pisahkan *node* menggunakan pemisah terbaik yang ditemukan pada langkah kedua jika hasil belum memuaskan.

2.12. *Algoritma Z-Thresh*

Algoritma *Z-Thresh* merupakan algoritma yang melakukan *thresholding* pada nilai amplitudo akselerometer di sumbu *z*. *Event* direpresentasikan sebagai nilai yang melebihi ambang batas atau *threshold* yang ditentukan. Masukan berupa data akselerometer

pada sumbu x , y dan z . Keluaran berupa informasi apakah terdeteksi *bump* atau bukan *bump*. Diagram alir algoritma *Z-thresh* dapat dilihat pada Gambar 2.7.



Gambar 2.7 Diagram Alir Algoritma *Z-Thresh*

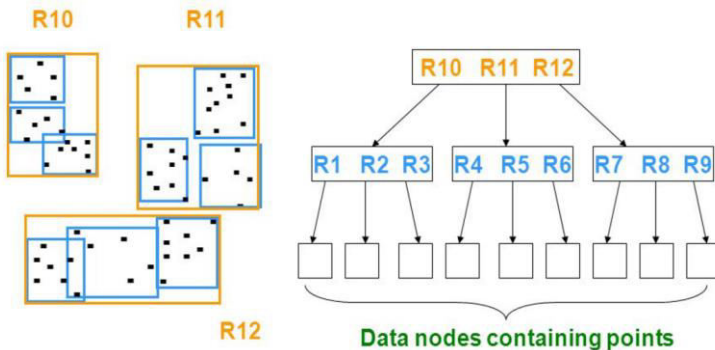
2.13. BIRCH

BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) [19] merupakan algoritma *unsupervised machine learning* yang melakukan *hierarchical clustering* pada dataset

yang besar . Membentuk suatu CF (*Clustering Feature*) tree secara meningkat. CF adalah struktur data hirarkikal pada multifase *cluster*. *Clustering Feature* terdiri dari 3 fitur yaitu N (jumlah data), LS (*Linear Sum of N data*) dan SS (*Square Sum of N data*).

1. Fase 1: Pindai data untuk membuat inisial *CF tree*.
2. Fase 2: Menggunakan algoritma *cluster* yang dinamis untuk melakukan *cluster leaf-nodes* pada *CF tree*.

CF tree memiliki dua parameter yaitu, *branching factor* dan *radius threshold*. *Branching factor* jumlah maksimum *CF subcluster* pada setiap *node*. *Diameter threshold* adalah nilai radius yang dimiliki *subcluster*. Sampel baru dan *subcluster* terdekat dapat digabung menjadi satu *subcluster* apabila jaraknya lebih kecil dari *threshold*, sebaliknya (lebih besar dari *threshold*) akan membentuk *subcluster* yang baru. Pada tugas akhir ini algoritma BIRCH diimplementasikan menggunakan *library scikit-learn* Python. Contoh proses *clustering* dapat dilihat pada Gambar 2.8. Pada gambar ditunjukkan R10, R11 dan R12 adalah *nodes* yang memiliki masing-masing tiga *subcluster* atau *child* yaitu *cluster* R1, R2, dan R3 pada *nodes* R10 dan seterusnya.



Gambar 2.8 Visualisasi Birch Clustering

BAB III

ANALISIS DAN PERANCANGAN

Pada bab ini akan membahas tahap analisis dan perancangan sistem yang akan dibangun pada tugas akhir. Perancangan meliputi perancangan data, perancangan proses, perancangan sistem dan perancangan antarmuka perangkat lunak

3.1. Deskripsi Umum Sistem

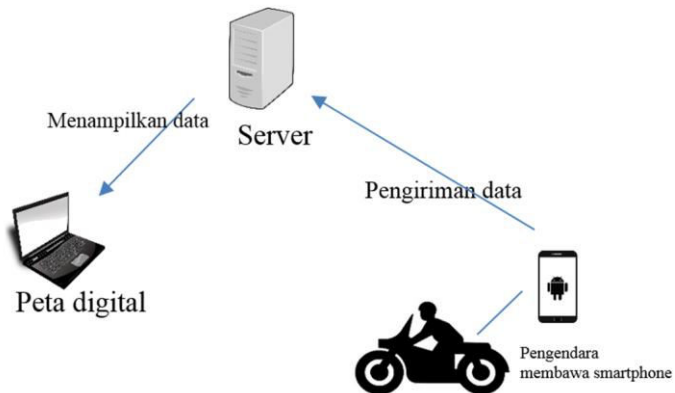
Sistem yang dibangun pada tugas akhir ini merupakan suatu sistem pemetaan *bump* beserta kerusakan jalan yang terdiri dari aplikasi perangkat bergerak berbasis Android sebagai pengirim data dan aplikasi *server* sebagai penerima dan pengolah data lokasi alat pembatas kecepatan lalu menampilkannya pada peta digital.

Aplikasi ini dibangun untuk memetakan lokasi *bump*. Informasi lokasi *bump* ini dapat digunakan oleh pemerintah untuk pertimbangan kebijakan perbaikan dan pemeliharaan jalan. Selain itu dapat digunakan pengguna untuk menghindari jalan yang berbahaya karena jalan rusak.

Tahap awal adalah pengguna akan membawa aplikasi ini kemudian melewati jalan yang terdapat polisi tidur. Alur proses sistem pendeteksi *bump* adalah seperti berikut. Sensor akselerometer pada *smartphone* Android akan diproses untuk mendeteksi *bump* menggunakan algoritma *Z-Thresh*. Apabila terdeteksi sebagai *bump* maka aplikasi Android akan mengirimkan data akselerasi ke *server* beserta lokasi dimana terjadi lonjakan akibat *bump*.

Data mentah berupa data akselerasi, *timestamp* dan GPS akan diolah lagi pada *server* menggunakan klasifikasi *decision tree* untuk meningkatkan akurasi pada hasil deteksi pada aplikasi Android. Pada *server* data lokasi *bump* akan di-*cluster* pada titik-titik koordinat *bump* yang berdekatan. Kemudian pada *server* akan menampilkan lokasi *bump* yang telah di-*cluster* pada peta digital.

Alur sistem secara keseluruhan ditampilkan pada Gambar 3.1.



Gambar 3.1 Alur Umum Sistem

3.1.1. Analisis Masalah

Dalam memudahkan untuk memetakan *bump* maka diperlukan suatu sistem *monitoring* yang dapat memberi lokasi dimana polisi tidur tersebut berada. Identifikasi polisi tidur dapat menggunakan sensor akselerometer yang terdapat pada *smartphone* Android. Data-data akselerasi pengguna Android saat melewati polisi tidur dapat dianalisis untuk menentukan adanya suatu *bump* atau tidak.

Permasalahan yang diidentifikasi pada sistem ini adalah bagaimana mengidentifikasi suatu polisi tidur dan mengolah serta menampilkan koordinat lokasi deteksi.

3.1.2. Analisis Kebutuhan Sistem

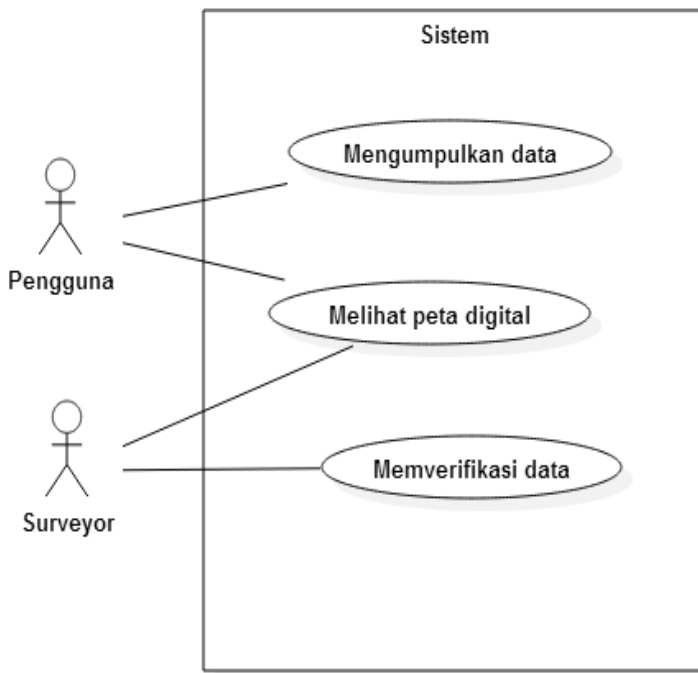
Pada subbab ini akan dibahas secara mendalam tentang kebutuhan sistem fungsional dan kebutuhan sistem non fungsional yang diperlukan pada tugas akhir ini.

3.1.2.1. Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan yang memiliki keterkaitan langsung dengan sistem. Kebutuhan fungsional pada sistem sebagai berikut:

- 1) Kebutuhan pengguna
 - Melihat tampilan peta digital
 - Mengumpulkan data akselerometer saat berkendara
- 2) Kebutuhan surveyor
 - Melakukan verifikasi kesesuaian hasil pemetaan pada peta digital

Kebutuhan fungsional sistem digambarkan dalam diagram kasus penggunaan seperti pada Gambar 3.2.



Gambar 3.2 Diagram Kasus Penggunaan

Penjelasan mengenai kasus penggunaan sistem monitoring ini berada pada Tabel 3.1.

Tabel 3.1 Deskripsi Kasus Penggunaan

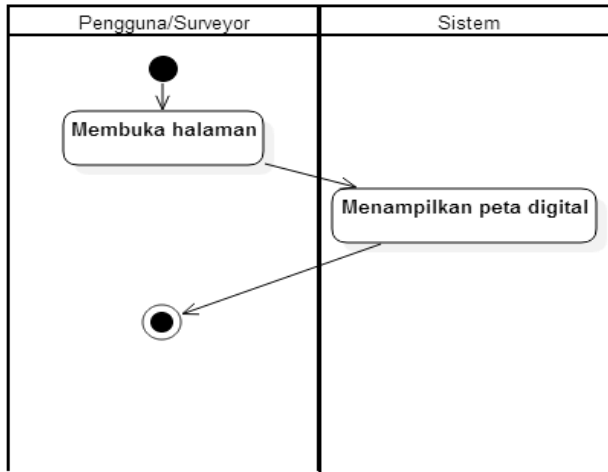
No	Kode	Nama	Keterangan
1	UC-01	Melihat peta digital	Pengguna dan Surveyor dapat melihat informasi pemetaan pada peta digital.
2	UC-02	Mengumpulkan data	Pengguna mengumpulkan data akselerometer menggunakan aplikasi pada Android.
3	UC-03	Memverifikasi data	Surveyor melakukan verifikasi data sesuai pada kondisi di lapangan.

3.1.2.1.1. Deskripsi Kasus Penggunaan UC-01

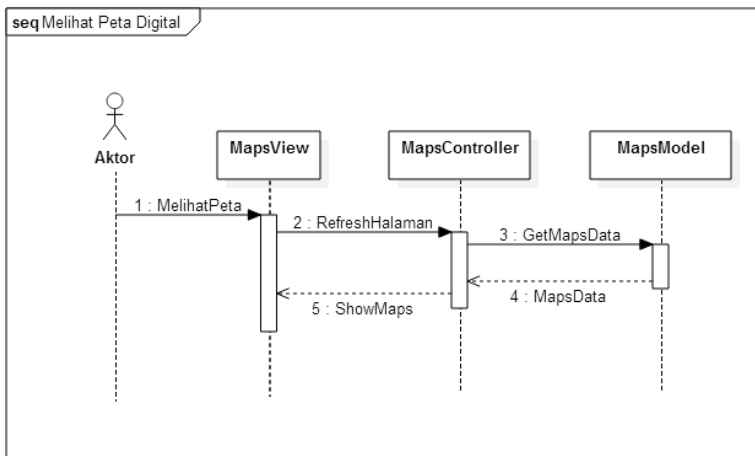
Kasus penggunaan kode UC-01 adalah kasus penggunaan menampilkan informasi pemetaan pada halaman di peta digital. Detail alur kasus melihat peta digital dijelaskan pada Tabel 3.2 beserta diagram aktivitas kasus yang dijelaskan pada Gambar 3.3. Diagram alir dijelaskan pada Gambar 3.4.

Tabel 3.2 Rincian Alur Kasus Penggunaan UC-01

Nama Use Case	Melihat Peta Digital
Nomor	UC-01
Aktor	Pengguna, Surveyor
Kondisi Awal	Peta digital belum ditampilkan
Kondisi Akhir	Peta digital ditampilkan
Alur Normal	<ol style="list-style-type: none"> 1. Pengguna/Surveyor membuka halaman peta digital pada <i>browser</i> atau peta pada Android. 2. Sistem menampilkan peta digital



Gambar 3.3 Diagram Aktivitas Kasus Penggunaan UC-01



Gambar 3.4 Diagram Alir Kasus Penggunaan UC-01

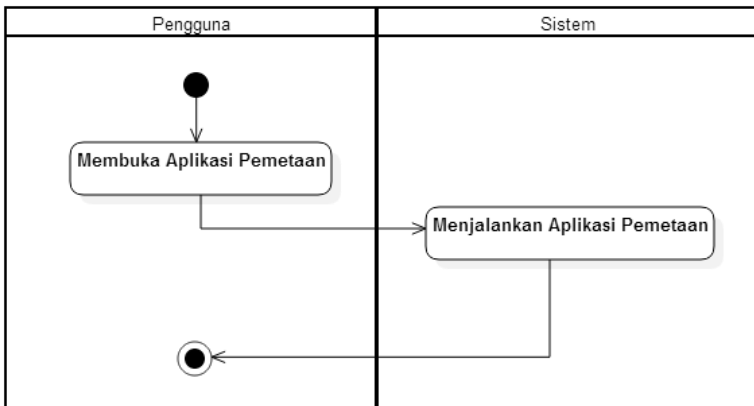
3.1.2.1.2. Deskripsi Kasus Penggunaan UC-02

Kasus penggunaan kode UC-02 merupakan kasus penggunaan mengumpulkan data akselerometer pada perangkat Android. Detail alur kasus melihat peta digital dijelaskan pada Tabel 3.3 beserta

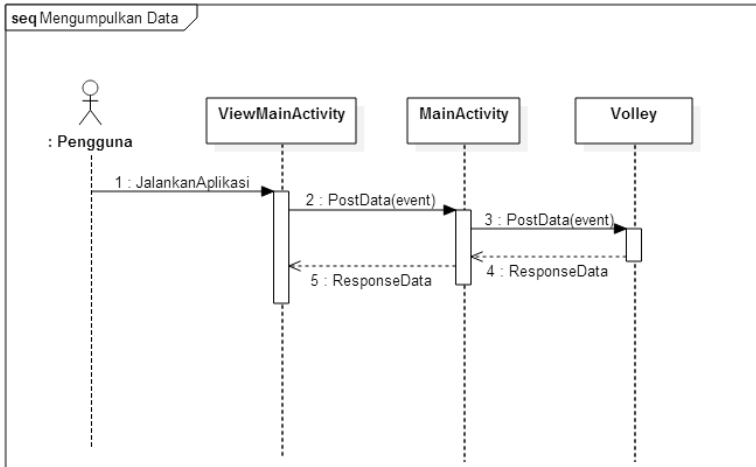
diagram aktivitas kasus yang dijelaskan pada Gambar 3.5. Diagram alir dijelaskan pada Gambar 3.6.

Tabel 3.3 Rincian Alur Kasus Penggunaan UC-02

Nama Use Case	Mengumpulkan Data
Nomor	UC-02
Aktor	Pengguna
Kondisi Awal	Aplikasi pemetaan pada Android belum terbuka
Kondisi Akhir	Aplikasi pemetaan pada Android berjalan
Alur Normal	<ol style="list-style-type: none"> 1. Pengguna membuka aplikasi Android 2. Sistem menjalankan aplikasi dan menampilkan antarmuka



Gambar 3.5 Diagram Aktivitas Kasus Penggunaan UC-02



Gambar 3.6 Diagram Alir Kasus Penggunaan UC-02

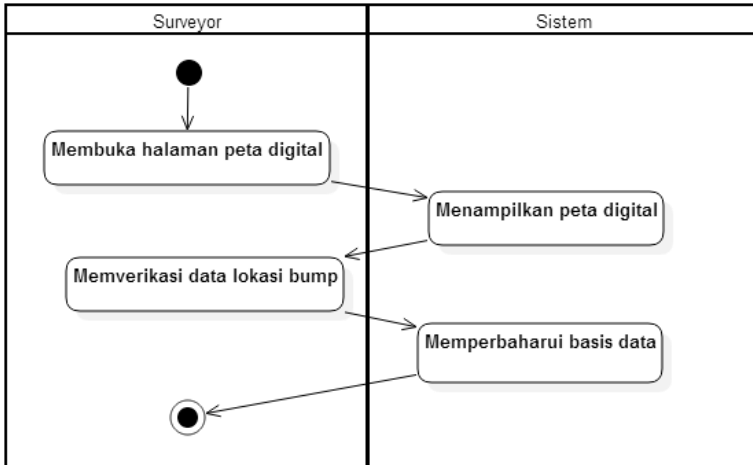
3.1.2.1.3. Deskripsi Kasus Penggunaan UC-03

Kasus penggunaan kode UC-02 merupakan kasus penggunaan melakukan verifikasi data lokasi *bump* yang terdeteksi. Detail alur kasus melihat peta digital dijelaskan pada Tabel 3.4 beserta diagram aktivitas kasus yang dijelaskan pada Gambar 3.7 Diagram alir dijelaskan pada Gambar 3.8.

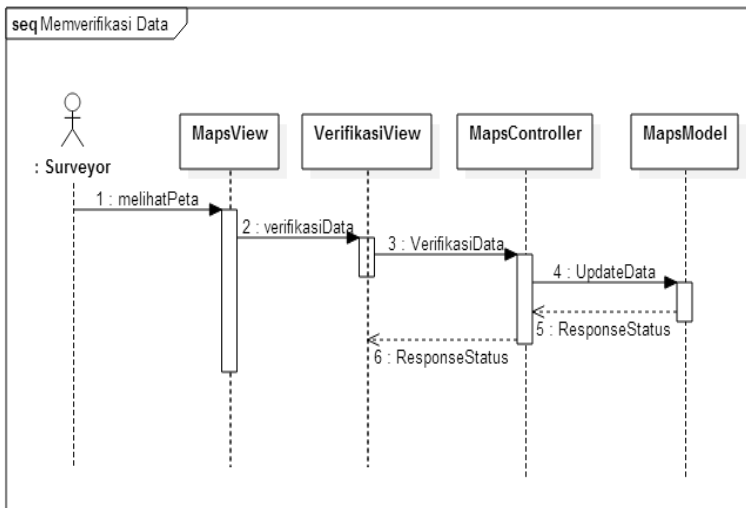
Tabel 3.4 Rincian Alur Kasus Penggunaan UC-03

Nama Use Case	Memverifikasi Data
Nomor	UC-03
Aktor	Surveyor
Kondisi Awal	Lokasi data <i>bump</i> belum terverifikasi
Kondisi Akhir	Data lokasi terjadinya <i>event bump</i> sudah diverifikasi
Alur Normal	1. Surveyor membuka halaman verifikasi
	2. Sistem menampilkan halaman login
	3. Surveyor memverifikasi lokasi yang terdeteksi <i>bump</i>

	4. Sistem memperbarui data yang diverifikasi pada basis data
--	--



Gambar 3.7 Diagram Aktivitas Kasus Penggunaan UC-03



Gambar 3.8 Diagram Alir Kasus Kegunaan UC-03

3.1.2.2. Kebutuhan Non Fungsional

Kebutuhan sistem yang diperlukan dalam membangun sistem *monitoring* ini terdiri dari kebutuhan fungsional dan non-fungsional. Kebutuhan non fungsional yang diperlukan meliputi:

1) Spesifikasi Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk membangun sistem adalah sebagai berikut:

- Sistem Operasi Windows 8.1, Ubuntu 14
- Android 6 Marshmallow
- Android SDK
- Android Studio
- ADT (Android Development Tools)

2) Spesifikasi Perangkat Keras

Pada aplikasi ini, perangkat keras yang digunakan dalam mengembangkan sistem adalah sebagai berikut:

- *Server*: Ubuntu-RAM 512MB - 1CPU
- Evercross Android One X

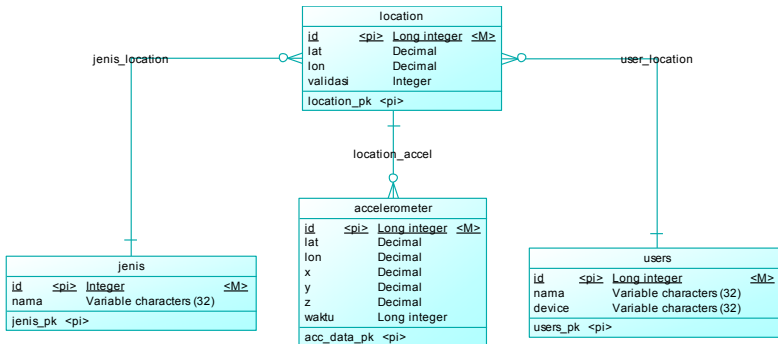
Perangkat keras yang digunakan untuk mengambil data berupa *smartphone* berbasis Android 6.

3.2. Perancangan Sistem

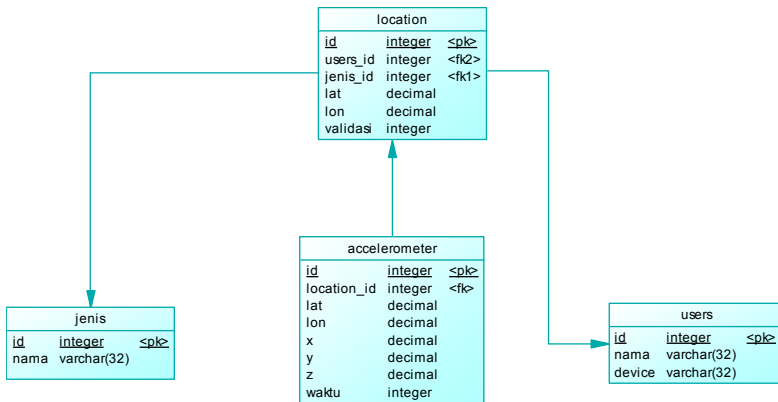
Tahap ini berupa perancangan basis data sistem, analisis proses sistem, analisis antarmuka peta digital. Pembahasan lebih lanjut akan dibahas sebagai berikut.

3.2.1. Perancangan Basis Data

Pada subbab ini akan membahas bagaimana rancangan basis data yang digunakan pada sistem pendeteksi *bump*. Basis data yang digunakan adalah PostgreSQL. PostgreSQL digunakan untuk menyimpan data akselerometer, jenis, *user* dan lokasi *bump*. *Conceptual Data Model* (CDM) dan *Physical Data Model* (PDM) dari basis data dapat dilihat pada Gambar 3.9 dan Gambar 3.10.



Gambar 3.9 CDM (*Conceptual Data Model*) pada web service



Gambar 3.10 PDM (*Physical Data Model*) pada web service

3.2.1.1. Rancangan Tabel *Location*

Tabel *location* digunakan untuk menyimpan data lokasi terjadinya lonjakan atau polisi tidur pada peta. Tabel ini memiliki relasi dengan tabel lainnya sebagai berikut:

1. Tabel *Users*

Hubungan dengan tabel ini adalah menyimpan id *user* setiap kali terjadi *event* lonjakan pada jalan. Kegunaannya adalah

untuk mengetahui *device* yang digunakan saat pengambilan data.

2. Tabel Jenis

Hubungan dengan tabel ini adalah menyimpan id jenis setiap kali melakukan *record* lokasi saat terjadi lonjakan. Manfaatnya adalah untuk mengetahui jenis anomali pada lonjakan.

3. Tabel *Accelerometer*

Hubungan dengan tabel ini adalah id *location* tersimpan pada tabel *accelerometer* setiap kali terjadi lonjakan pada jalan. Manfaatnya adalah untuk mengetahui lokasi data akselerasi.

Detail Atribut Tabel *Location* dijelaskan pada Tabel 3.5.

Tabel 3.5 Atribut Tabel Data *Location*

Nama Kolom	Tipe Data	Keterangan
id	<i>long integer</i>	<i>Primary Key</i> pada tabel <i>Location</i>
users_id	<i>long integer</i>	<i>Foreign Key</i> dari tabel <i>Users</i>
jenis_id	<i>integer</i>	<i>Foreign Key</i> dari tabel <i>Jenis</i>
lat	<i>decimal</i>	<i>Latitude</i> pada Gmaps
Lon	<i>decimal</i>	<i>Longitude</i> pada Gmaps
validasi	<i>integer</i>	Status validasi

3.2.1.2. Rancangan Tabel *Accelerometer*

Tabel *Accelerometer* digunakan untuk menyimpan data akslerasi yang terjadi setiap kali terjadi lonjakan. Tabel ini memiliki relasi dengan tabel lainnya sebagai berikut:

1. Tabel *Location*

Hubungan dengan tabel ini adalah menyimpan informasi lokasi setiap kali terjadi *event* lonjakan. Kegunaannya adalah untuk mengetahui informasi lokasi dimana data akslerometer tersebut diambil.

Detail Atribut Tabel *Accelerometer* dijelaskan pada Tabel 3.6.

Tabel 3.6 Atribut Tabel Data Accelerometer

Nama Kolom	Tipe Data	Keterangan
id	<i>long integer</i>	<i>Primary Key</i> pada tabel Accelerometer
location_id	<i>long integer</i>	<i>Foreign Key</i> dari tabel Location
x	<i>decimal</i>	Data akselerasi yang sudah direorientasi pada sumbu x
y	<i>decimal</i>	Data akselerasi pada sumbu y terhadap koordinat bumi
z	<i>decimal</i>	Data akselerasi pada sumbu z bumi
lat	<i>decimal</i>	<i>Latitude</i> pada Gmaps
lon	<i>decimal</i>	<i>Longitude</i> pada Gmaps
waktu	<i>long integer</i>	<i>Timemillis</i> (waktu dalam <i>milisecond</i>) saat menyimpan data

3.2.1.3. Rancangan Tabel Jenis

Tabel Jenis digunakan untuk menyimpan nama jenis anomali pada jalan. Tabel ini memiliki relasi dengan tabel lainnya yaitu sebagai berikut.

1. Tabel *Location*

Hubungan dengan tabel *location* adalah sebagai *foreign key* untuk menyimpan informasi jenis pada lokasi *bump* atau lonjakan yang terdeteksi. Kegunaannya adalah untuk mengetahui informasi nama jenis anomali jalan yang terdeteksi.

Detail Atribut Tabel Jenis dijelaskan pada Tabel 3.7.

Tabel 3.7 Atribut Tabel Jenis

Nama Kolom	Tipe Data	Keterangan
id	<i>integer</i>	<i>Primary Key</i> pada tabel Jenis
nama	<i>varchar(32)</i>	Nama jenis

3.2.1.4. Rancangan Tabel *Users*

Tabel *Users* digunakan sebagai penyimpanan data *user* dan nama *device* yang digunakan. Tabel ini memiliki relasi dengan tabel lainnya yaitu sebagai berikut.

1. Tabel *Location*

Hubungan dengan tabel ini adalah sebagai *foreign key* untuk menyimpan informasi pengguna yang merekam informasi tersebut. Kegunaannya adalah untuk mengetahui informasi nama *device* yang digunakan saat melakukan pengambilan data.

Detail Atribut Tabel *User* dijelaskan pada Tabel 3.8.

Tabel 3.8 Atribut Tabel *Users*

Nama Kolom	Tipe Data	Keterangan
id	<i>long integer</i>	<i>Primary Key</i> pada tabel <i>Users</i>
nama	<i>varchar(32)</i>	Nama adalah id unik Android yang digunakan
device	<i>varchar(32)</i>	Nama perangkat Android (<i>device</i>)

3.2.2. Perancangan *Web Service*

Pada subbab ini akan membahas desain *web service* yang akan diimplementasikan pada tugas akhir ini. *Web service* yang digunakan adalah RESTful *web service* menggunakan kerangka kerja Slim. Berikut ini adalah fungsi-fungsi yang akan dibangun pada *web service*.

3.2.2.1. *Query Event Bump*

Fungsi *query event bump* adalah fungsi pada *web service* yang berguna untuk menerima data berupa *JSONArray* dari aplikasi Android pada saat terjadi *event bump* dan memasukkan data tersebut ke basis data pada *server*. *Method* yang digunakan pada fungsi ini adalah HTTP POST.

Data *JSONArray* yang dikirim ke *web service* sebagai parameter *request* dijelaskan pada Tabel 3.9.

Tabel 3.9 Parameter Request Menerima Event Bump

Nama Key	Keterangan Value
lat	Koordinat lintang terjadinya <i>event bump</i>
lon	Koordinat bujur terjadinya <i>event bump</i>
x	Nilai akselerometer pada sumbu x
y	Nilai akselerometer pada sumbu y
z	Nilai akselerometer pada sumbu z
waktu	<i>Timestamp</i> terjadinya <i>event bump</i>
location_id	id lokasi terjadinya <i>event bump</i>
jenis_id	Jenis <i>bump</i>
user_id	Id <i>user</i> pengguna aplikasi Android

Parameter *response* yang dikembalikan dari *web service* apabila data telah terkirim dijelaskan pada Tabel 3.10.

Tabel 3.10 Parameter Response Menerima Event Bump

Nama Key	Keterangan Value
status	<ol style="list-style-type: none"> Bernilai “<i>success</i>” jika data yang dikirimkan berhasil ditambahkan pada basis data Bernilai “<i>fail</i>” jika data yang dikirimkan gagal ditambahkan pada basis data

3.2.2.2. Query Id User

Menerima *query id user* adalah fungsi pada *web service* berguna untuk mengirimkan data dari basis data berupa id *user* dari informasi kode unik Android pada *device* pengguna, apabila id *user* belum terdaftar pada basis data, maka *web service* akan memasukkan data kode unik Android dan nama *device* pengguna ke basis data dan *web service* akan mengembalikan nilai id *user* ke aplikasi Android. *Method* yang digunakan adalah HTTP POST.

Data *JSONObject* yang dikirimkan ke *web service* sebagai parameter *request* dijelaskan pada Tabel 3.11.

Tabel 3.11 Parameter *Request* Menerima Id User

Nama Key	Keterangan Value
Android_id	Kode unik Android yang digunakan pengguna
device	Nama merek <i>device</i> yang digunakan pengguna

Parameter *response* yang dikembalikan dari *web service* jika data terkirim dijelaskan pada Tabel 3.12.

Tabel 3.12 Parameter *Response* Menerima Id User

Nama Key	Keterangan Value
id_user	Id pengguna yang akan digunakan pada aplikasi Android

3.2.3. Perancangan Antarmuka

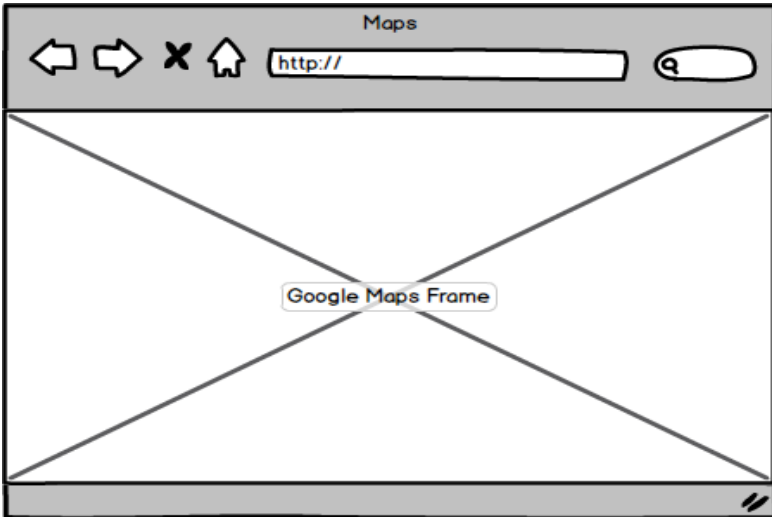
Pada subbab ini akan dibahas dengan terperinci dari rancangan antarmuka sistem. Antarmuka sistem terdiri dari antarmuka pada peta digital dan antarmuka pada aplikasi Android. Berikut ini dibahas antarmuka yang terdapat pada sistem yang dibangun untuk tugas akhir ini.

3.2.3.1. Antarmuka pada Peta Digital

Pada Gambar 3.11 terdapat peta digital beserta *marker* penanda lokasi *bump*. Peta digital ditampilkan menggunakan *library* dari Google Maps API v3 Biostall. Fitur yang akan digunakan Google Maps API adalah sebagai berikut.

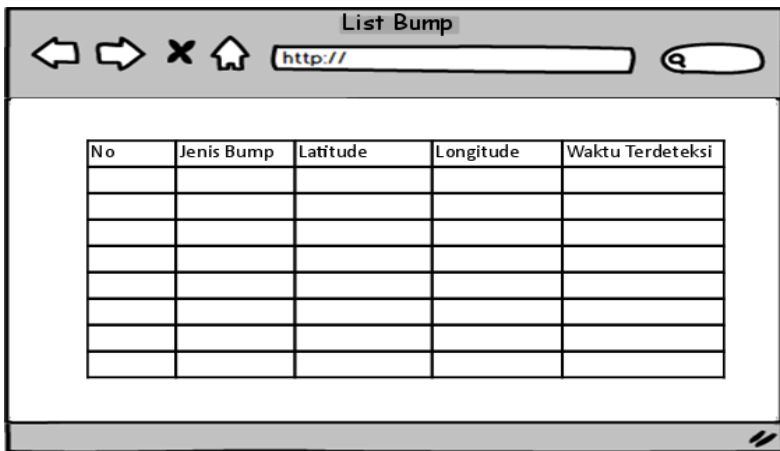
1. Menampilkan peta digital menggunakan Google Maps.
2. Menandai peta menggunakan *marker*.
3. Menentukan posisi dan *zoom* pada peta digital.

Rancangan antarmuka pada halaman peta digital dapat dilihat pada Gambar 3.11.



Gambar 3.11 Rancangan Antarmuka pada Peta Digital

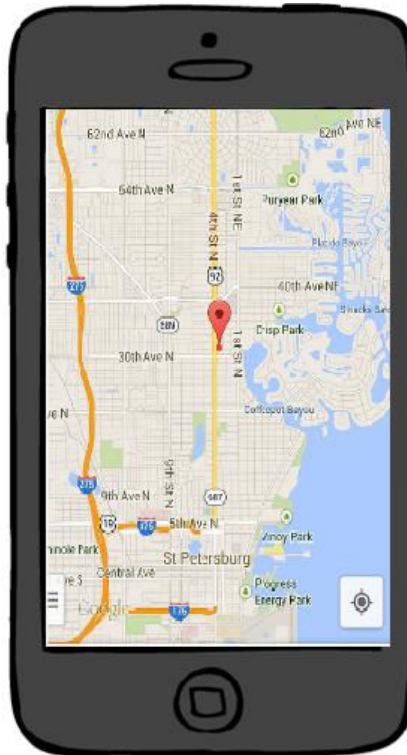
Pada Gambar 3.12 dapat dilihat rancangan antarmuka halaman *list bump* yang berguna untuk menyediakan informasi data-data *bump* yang ada pada peta digital.



Gambar 3.12 Rancangan Antarmuka *List Bump*

3.2.3.2. Antarmuka Peta pada Android

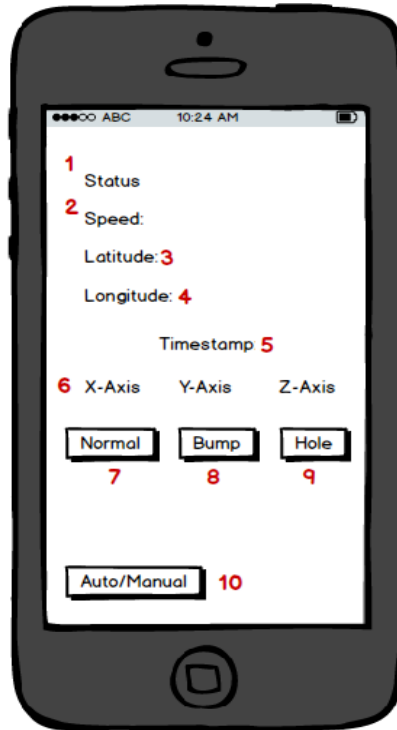
Pada Gambar 3.13 merupakan rancangan antarmuka peta pada aplikasi Android. Pada tampilan peta terdapat *marker* yang akan menunjukkan lokasi terjadinya *bump*.



Gambar 3.13 Rancangan Antarmuka Peta pada Android

3.2.3.3. Antarmuka Pengumpulan Data pada Android

Pada Gambar 3.14 merupakan rancangan antarmuka pada aplikasi Android saat pengumpulan data *training*. Komponen-komponen pada aplikasi Android dijelaskan pada nomor berwarna merah.



Gambar 3.14 Antarmuka pada Aplikasi

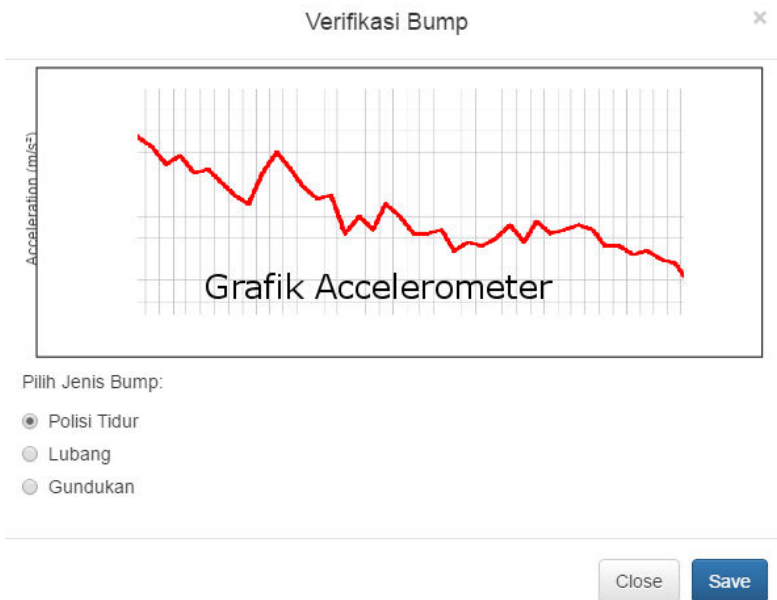
Keterangan pada Gambar 3.14 berdasarkan nomor-nomor di atas sebagai berikut.

1. Label untuk status pengambilan data. Statusnya adalah pengiriman auto/manual, sedang mengirim, *bump*, *hole* atau kondisi normal.
2. Label untuk keterangan kecepatan saat pengambilan data.
3. Label koordinat *latitude*.
4. Label koordinat *longitude*.
5. Label timestamp.
6. Label nilai akselerometer pada X-axis, Y-axis dan Z-axis.
7. Tombol *training* data pada kondisi jalan Normal.

8. Tombol *training* data pada kondisi melonjak.
9. Tombol *training* data pada kondisi lubang.
10. Tombol pengambilan data Auto/Manual.

3.2.3.4. Antarmuka Verifikasi Data

Pada Gambar Gambar 3.15 merupakan rancangan antarmuka verifikasi data pada peta digital. Tampilan verifikasi berupa *modal* Bootstrap 3 yang akan muncul ketika *marker* pada peta digital di-*double click*. Antarmuka login verifikasi terlihat pada Gambar 3.16.



Gambar 3.15 Rancangan Antarmuka Verifikasi Data

Log in

Email

Password

Gambar 3.16 Rancangan Halaman Login Verifikasi

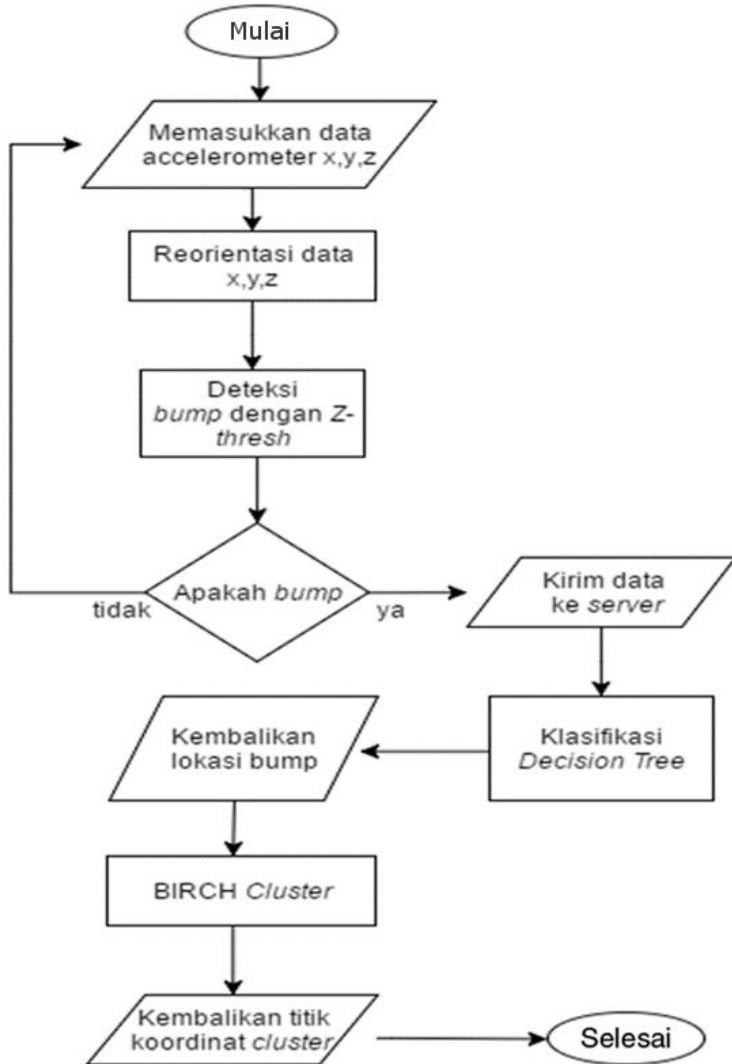
3.2.4. Perancangan Proses Data

Pada subbab ini akan dibahas dengan detail rancangan bagaimana pengiriman dan pengolahan data. Data yang dikirimkan berupa JSON berisi data akselerometer dan lokasi GPS. Pengiriman data JSON dari *smartphone* menuju *server* terjadi setiap kali terjadi *event* yakni ketika pengendara mengalami lonjakan pada jalan.

Data yang dikirim adalah hasil data yang terekam selama 1,5 detik setiap terjadi *event*. Waktu selama 1,5 detik merupakan waktu paling maksimum ketika kendaraan melewati polisi tidur dengan kecepatan yang rendah (kurang dari 10km/jam). Tujuannya adalah pengiriman data yang lebih efisien sehingga tidak perlu menyimpan dan mengirimkan data akselerometer pada jalan normal dan dapat fokus menyimpan data yang menyebabkan *bump*.

Data tersebut kemudian diprediksi menggunakan klasifikasi dengan metode *decision tree*. Apabila benar maka lokasi data tersebut akan ditampilkan pada peta digital. Pada peta digital koordinat lokasi polisi tidur juga akan di-*cluster* untuk titik-titik yang berdekatan (kurang dari 5-9 meter) menggunakan algoritma

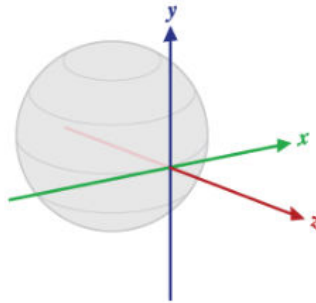
BIRCH. Diagram Alir perancangan proses data ditunjukkan pada Gambar 3.17.



Gambar 3.17 Diagram Alir Proses Data

3.2.4.1. Reorientasi Sistem Koordinat Akselerometer

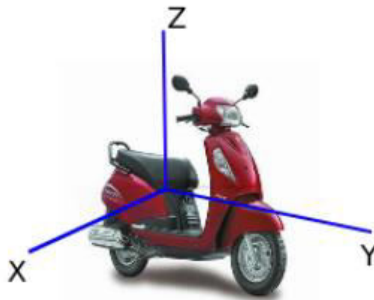
Sensor akselerometer Android memiliki sistem koordinat yang relatif terhadap perangkat Android itu sendiri. Sistem koordinat akselerometer dapat dilihat pada Gambar 2.5. Apabila perangkat tersebut bergerak atau disorientasi maka dapat mempersulit pengukuran data akselerometer. Untuk itu diperlukan suatu mekanisme yang dapat melakukan reorientasi dari sistem koordinat perangkat menjadi sistem koordinat pada motor. Langkah pertama adalah melakukan reorientasi menjadi sistem koordinat global, baru dari sistem koordinat global, direorientasi lagi menjadi sistem koordinat kendaraan. Tujuannya adalah untuk mengetahui *magnetic north*-nya. Hal tersebut dapat dilakukan dengan menggunakan matriks rotasi perangkat Android. Matriks rotasi dapat diperoleh dari API sensor Android yaitu menggunakan fungsi *getRotationMatrix*. Setelah memperoleh matriks rotasinya, selanjutnya adalah melakukan multiplikasi matriks rotasi tersebut dengan nilai akselerometer. Sistem Koordinat Global dapat dilihat pada Gambar 3.18.



Gambar 3.18 Sistem Koordinat Global

Setelah memperoleh nilai akselerasi pada sistem koordinat global, selanjutnya adalah melakukan transformasi menjadi sistem koordinat pada kendaraan (Gambar 3.19). Langkah berikutnya adalah menghitung *bearing* yaitu sudut antara 2 titik *latitude-longitude* dengan *magnetic North*. *Bearing* dapat diperoleh dengan

menggunakan fungsi *getBearing* pada API GPS Location Android. Selanjutnya setelah mendapatkan nilai *bearing* adalah menemukan sudut yang benar antara *magnetic North* dengan arah kendaraan adalah dengan melakukan pengurangan antara *bearing* dengan *magnetic declination*. *Magnetic declination* adalah sudut antara *magnetic north* (arah utara pada kompas, sesuai dengan arah garis medan magnet bumi) dengan *true north* (arah sepanjang meridian terhadap kutub utara). Nilai akselerasi yang sesuai dengan sistem koordinat kendaraan dapat diperoleh dengan cara memultiplikasi nilai akselerasi sistem koordinat global dengan matriks rotasi yang merepresentasikan rotasi sumbu z pada sudut *bearing - declination*.



Gambar 3.19 Sistem Koordinat Kendaraan

3.2.4.2. Pengumpulan Data Training

Proses ini merupakan pengambilan data-data akselerasi pada berbagai kondisi jalan. Kondisi data yang diamati adalah jalan biasa, jalan bergelombang, polisi tidur dan lubang. Data dikirim dari Android dalam bentuk JSON. Proses pengambilan data diambil dalam waktu tertentu saat mengendarai motor. Pengambilan data dimulai ketika pengguna menekan tombol jenis *event* yang ingin direkam dan berhenti ketika pengguna kembali menekan tombol tersebut. Selama melewati jalan, data akan direcord dalam *JSONArray*, yang kemudian akan dikirim ke *server* setelah selesai mengambil data. Dari data-data tersebut dapat

diolah dengan melakukan ekstrak fitur. Fitur ekstrasi yang dipilih adalah standar deviasi, selisih maksimum dengan minimum dan mean data akselerasi pada sumbu z.

Selain mengambil data training dengan cara manual (menekan tombol) aplikasi juga dapat melakukan pengumpulan data training dengan otomatis. Pengumpulan data otomatis menggunakan algoritma *Z-thresh*, yaitu apabila data akselerasi pada sumbu z melewati batas *threshold* yang telah ditentukan maka dianggap telah terjadi lonjakan atau *event bump*.

Aplikasi ini akan merekam aktivitas akselerasi pada saat sebelum dan selama 1,5 detik setelah *event bump* terjadi. Untuk memperoleh data akselerasi sebelum terjadinya *event bump* maka data disimpan pada *list* sebagai satu *window*. Ukuran lebar *window* pada tugas akhir ini adalah 20 buah data akselerometer. Isi *window* akan berubah setiap kali data sensor akselerometer masuk. Data ini kemudian akan dikumpulkan sebagai satu jenis data *bump* dan akan dikirimkan ke *server* beserta data lokasi terjadinya *bump*.

3.2.4.3. Pengolahan Data pada Server

Pada proses ini, data-data yang telah terkirim saat terjadinya *event bump* akan diolah pada *server*. Data-data yang terkirim berupa data akselerometer pada sumbu x, y, z beserta koordinat *latitude* dan *longitude* pada saat terjadi *bump*. Berikut proses data yang terjadi di *server*.

3.2.4.3.1. Prediksi Event dengan Decision tree

Data *event* yang telah dikirimkan ke *server* (data akselerometer) diolah kembali menggunakan algoritma *Decision tree*. Tujuannya adalah untuk mengurangi *false positive bump* yang terdeteksi. Biasanya *false positive* ini diakibatkan permukaan jalan yang kurang rata sehingga menyebabkan guncangan yang dianggap sebagai *bump*. Namun jenis guncangan seperti ini dapat dikenali berdasarkan aktivitas data akselerometer pada sumbu x dan y. Data akselerometer pada sumbu x dan y tersebut kemudian

diekstraksi berdasarkan fitur standar deviasi, rata-rata dan selisih nilai maksimum dengan minimum. Fitur tersebut kemudian dipelajari atau di-*training* menggunakan *decision tree*. Setelah data *training* terkumpul, diharapkan *decision tree* ini dapat memprediksi *bump* dengan akurasi yang lebih baik.

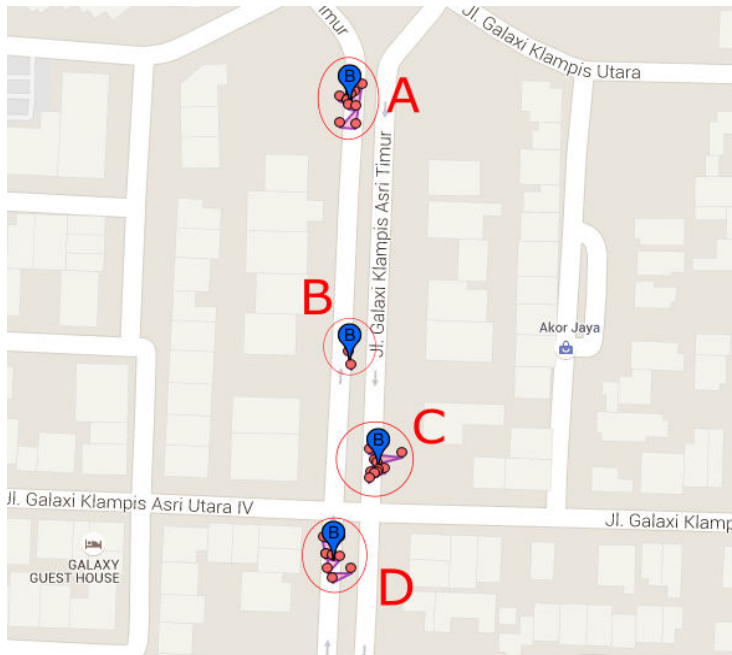
3.2.4.3.2. Clustering Koordinat pada Peta Digital

Data lokasi *bump* yang telah terkirim ke *server* selanjutnya akan dikelompokkan menjadi satu *subcluster* apabila berdekatan. *Subcluster* pada peta digital adalah titik-titik koordinat yang dikelompokkan menjadi satu. Data lokasi berupa titik-titik koordinat lokasi terjadinya *bump* yang terdeteksi. Titik koordinat *latitude* dan *longitude* yang diperoleh dari hasil deteksi *bump* pada aplikasi Android akan diproses pada *server* untuk di-*cluster*. I

Dalam pengujian, apabila pengendara melewati titik yang sama pada lokasi yang terdapat lonjakan pada pengujian sebelumnya, maka sensor GPS akan memberikan estimasi koordinat dimana lonjakan tersebut terjadi. Namun dikarenakan adanya ketidakakuratan pada GPS, maka titik koordinat yang dihasilkan tidak selalu tepat sama pada satu lokasi lonjakan.

Titik-titik koordinat hasil deteksi yang berdekatan dapat dikluster menjadi satu *subcluster* menggunakan algoritma BIRCH. Algoritma BIRCH adalah algoritma algoritma *unsupervised* (tanpa pengawasan) *machine learning* yang melakukan *hierarchical clustering*. Lalu dari titik-titik pada satu *subcluster* itu akan diambil titik *centroid* yang akan menjadi representasi lokasi *bump* di *subcluster* tersebut.

Pada Gambar 3.20 terdapat lingkaran A, B, C dan D yang merupakan *subcluster* dihasilkan dari titik-titik *bump* (pada *marker dot* merah). Masing-masing *subcluster* tersebut diwakilkan oleh titik sentroidnya (pada *marker* warna biru).



Gambar 3.20 Contoh Hasil *Clustering* pada Peta Digital

BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari analisis dan perancangan sistem yang telah dibahas pada Bab III. Namun dalam penerapannya, rancangan tersebut dapat mengalami perubahan minor sewaktu-waktu apabila dibutuhkan.

4.1. Lingkungan Implementasi

Dalam proses perancangan sistem ini digunakan perangkat pendukung sebagai berikut.

4.1.1. Lingkungan Implementasi Perangkat Keras

Tabel 4.1 menjelaskan lingkungan perangkat keras yang digunakan dalam proses pengembangan sistem perangkat lunak.

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

<i>Hardware</i>	Laptop HP, 2.2 Ghz <i>processor</i> Intel Core i3, 4 GB RAM
<i>Server</i>	Digital Ocean: 512 MB RAM, Ubuntu 14.
<i>Smartphone</i>	Android One X Evercoss A65, sistem operasi Android 6 Marshmallow, memori internal 8 GB, 1 GB RAM.

4.1.2. Lingkungan Implementasi Perangkat Lunak

Pada Tabel 4.2 menjelaskan implementasi perangkat lunak yang digunakan dalam proses pengembangan dan implementasi aplikasi ini.

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak

Nama Aplikasi	Kegunaan
Microsoft Windows 8.1	Sebagai sistem operasi pada perangkat keras pengembangan aplikasi
Ubuntu 14	Sebagai sistem operasi pada <i>server</i>
PostgreSQL	Sebagai basis data pada pengembangan aplikasi.

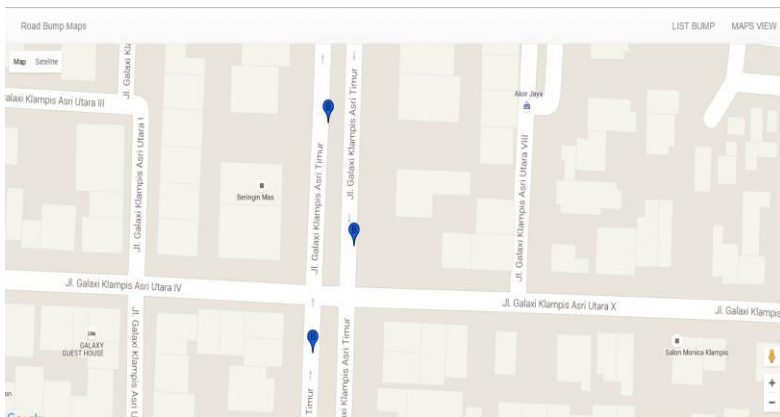
XAMPP	Sebagai <i>web server</i> aplikasi
PhpPgadmin	Sebagai aplikasi manajemen basis data
Power Designer 15	Sebagai aplikasi perancangan basis data
Android Studio	Sebagai IDE untuk implementasi aplikasi
Sublime 3	Sebagai teks editor untuk penulisan kode sumber
Putty dan WinSCP	Untuk instalasi dan integrasi <i>web service</i> pada <i>server</i>

4.2. Lingkungan Implementasi Antarmuka

Pada subbab ini dibahas bagaimana implementasi antarmuka berdasarkan rancangan antar muka yang telah dibahas pada Bab 3.

4.2.1. Antarmuka Peta Digital pada *Server*

Implementasi tampilan peta digital dapat dilihat pada Gambar 4.1. *Marker* berwarna merah menunjukkan lokasi polisi tidur hasil observasi langsung sedangkan *marker* berwarna biru menunjukkan *bump* hasil deteksi dari aplikasi.



Gambar 4.1 Tampilan Peta Digital Hasil Deteksi *Bump*

Implementasi informasi *bump* juga dapat dilihat pada Gambar 4.2. Daftar *bump* ditampilkan dalam bentuk tabel dengan informasi *latitude*, *longitude*, jenis, dan waktu terdeteksinya.

Road Bump Maps LIST BUMP MAPS VIEW

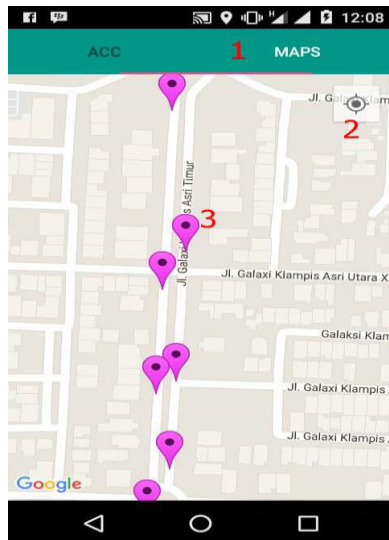
No	Jenis Bump	Latitude	Longitude	Waktu Terdeteksi
1	Polisi Tidur	-7.29322	112.78612	17/05/2016 07:54:25
2	Polisi Tidur	-7.2946234	112.78606	17/05/2016 07:55:08
3	Polisi Tidur	-7.2946234	112.78606	17/05/2016 07:55:08
4	Polisi Tidur	-7.295595	112.786026	17/05/2016 07:55:49
5	Polisi Tidur	-7.297515	112.785934	17/05/2016 07:56:45
6	Polisi Tidur	-7.3007317	112.78582	17/05/2016 07:58:43
7	Polisi Tidur	-7.299327	112.78562	17/05/2016 07:59:45
8	Polisi Tidur	-7.2978034	112.78573	17/05/2016 08:00:14
9	Polisi Tidur	-7.2963867	112.785774	17/05/2016 08:00:39
10	Polisi Tidur	-7.29467	112.785835	17/05/2016 08:01:09

Previous 1 2 3 4 5 ... 37 Next

Gambar 4.2 Tampilan Halaman *List Bump*

4.2.2. Antarmuka Peta pada Android

Implementasi peta pada aplikasi Android ditunjukkan pada Gambar 4.3.

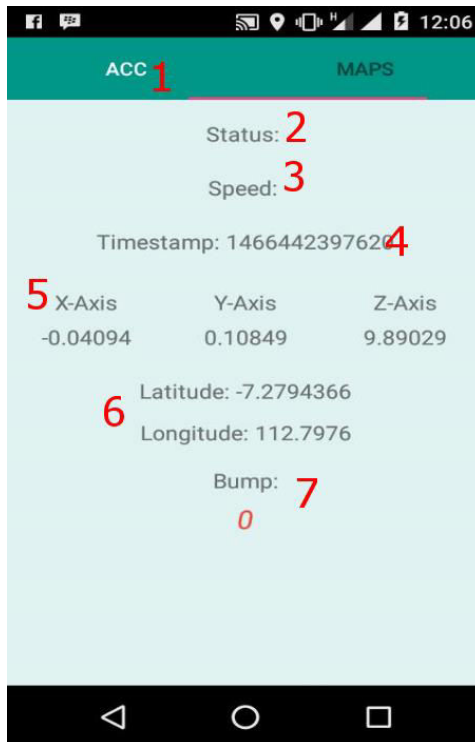


Gambar 4.3 Tampilan Peta pada Android

Keterangan pada peta adalah sebagai berikut. Nomor 1 menunjukkan judul *fragment* yang sedang aktif atau dibuka. Pada nomor 2 merupakan tombol untuk memperoleh lokasi pengguna dan pada nomor 3 merupakan *marker* yang menunjukkan lokasi terjadinya *bump*.

4.2.3. Antarmuka Pengumpulan Data pada Android

Implementasi aplikasi Android ditunjukkan pada Gambar 4.4. Aplikasi setelah dijalankan dapat langsung digunakan untuk melakukan pengumpulan data.



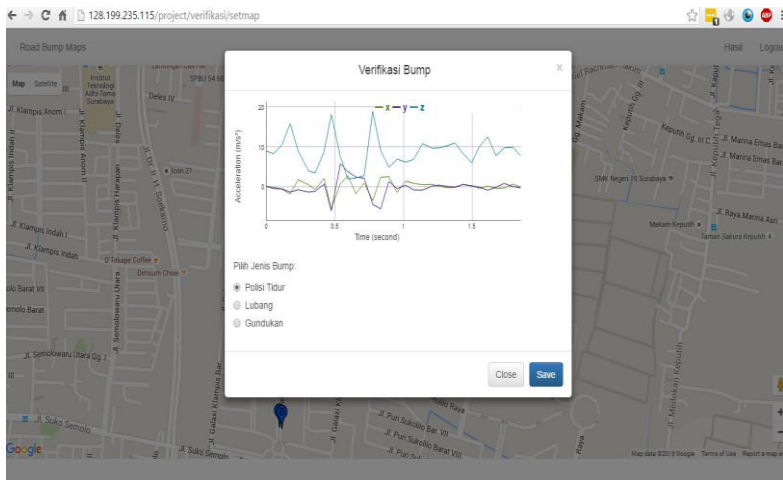
Gambar 4.4 Tampilan Aplikasi Utama

Keterangan pada Gambar 4.4 berdasarkan nomor-nomor di atas sebagai berikut.

1. Label untuk menunjukkan *fragment* yang aktif.
2. Label untuk status pengiriman data. Statusnya adalah *bump* atau normal.
3. Label untuk keterangan kecepatan saat pengambilan data.
4. Label untuk menunjukkan nilai *timestamp*.
5. Label nilai akselerometer pada X-axis, Y-axis dan Z-axis.
6. Label koordinat *latitude* dan *longitude*
7. Label jumlah *bump* yang terdeteksi.

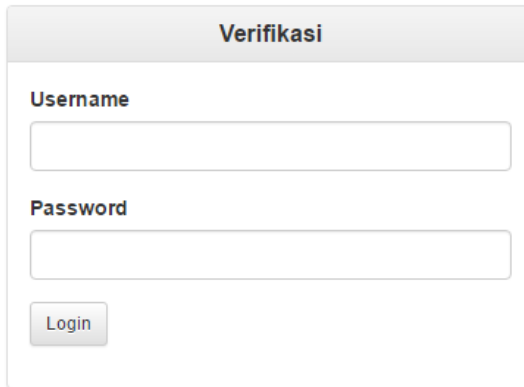
4.2.4. Antarmuka Verifikasi Data

Implementasi verifikasi data ditunjukkan pada dialog *modal* Bootstrap dapat dilihat pada Gambar 4.5. *Modal* berisi grafik nilai akselerometer pada *bump* dan pilihan jenis *bump* berupa radio *button*..



Gambar 4.5 Tampilan Verifikasi Data

Halaman login verifikasi ditunjukkan pada Gambar 4.6. Login form berupa username dan password yang akan dikirimkan ke *server*.



The image shows a web form titled "Verifikasi". It has a light gray header with the title. Below the header, there are two input fields. The first is labeled "Username" and the second is labeled "Password". Below the password field, there is a button labeled "Login".

Gambar 4.6 Halaman Verifikasi Login

4.3. Implementasi Basis Data

Pada subbab ini akan dibahas secara mendetail rancangan basis data yang telah dipaparkan pada Bab 3. Pada implementasi ini dijelaskan mengenai pembuatan tabel-tabel yang digunakan pada sistem. Implementasi tabel basis data menggunakan sintaks PostgreSQL.

4.3.1. Implementasi Tabel *Location*

Tabel *location* adalah tabel yang digunakan untuk menyimpan data lokasi terjadinya lonjakan atau polisi tidur pada peta. Implementasi tabel *location* dapat dilihat pada Kode Sumber 4.1.

Tabel *location* memiliki atribut yaitu *latitude* dan *longitude* sebagai koordinat lokasi terjadinya *bump*. *Jenis_id* dan *user_id* sebagai *foreign key*. Validasi sebagai status data *location* beserta id tabel sebagai primary key.

```

CREATE TABLE location (
    lat double precision NOT NULL,
    lon double precision NOT NULL,
    jenis_id integer,
    id integer NOT NULL,
    user_id integer,
    validasi integer DEFAULT 0
);
CREATE SEQUENCE location_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE ONLY location ALTER COLUMN id SET DEFAULT
    nextval('location_id_seq'::regclass);
ALTER TABLE ONLY location
    ADD CONSTRAINT location_id_jenis_fkey FOREIGN KEY
    (jenis_id) REFERENCES jenis(id);

```

Kode Sumber 4.1 Implementasi Tabel *Location*

4.3.2. Implementasi Tabel *Accelerometer*

Tabel *accelerometer* adalah tabel yang digunakan untuk menyimpan data akslerasi yang terjadi setiap kali terjadi lonjakan. Tabel *accelerometer* memiliki beberapa atribut sebagai berikut: id, latatitute dan *longitude*, nilai akselerometer pada sumbu x, y dan z, beserta waktu dan *location_id* sebagai *foreign key* dari tabel *location*. Implementasi tabel *accelerometer* ditunjukkan pada Kode Sumber 4.2.

```

CREATE TABLE accelerometer(
    id bigint NOT NULL,
    lat double precision NOT NULL,
    lon double precision NOT NULL,
    x double precision,
    y double precision,
    z double precision NOT NULL,
    waktu bigint,
    location_id integer
);

```

```

CREATE SEQUENCE accelerometer_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MINVALUE
  NO MAXVALUE
  CACHE 1;
ALTER TABLE ONLY accelerometer ALTER COLUMN id SET
DEFAULT nextval('
  accelerometer_id_seq'::regclass);
ALTER TABLE ONLY accelerometer
  ADD CONSTRAINT accelerometer_pkey PRIMARY KEY
(id);

```

Kode Sumber 4.2 Implementasi Tabel *Accelerometer*

4.3.3. Implementasi Tabel Jenis

Tabel *Jenis* merupakan tabel yang digunakan untuk menyimpan nama jenis anomali pada jalan. Data yang disimpan berupa data nama dan id jenis guncangan atau anomali jalan. Atribut pada tabel ini adalah id dan nama jenis. Implementasi ditunjukkan pada Kode Sumber 4.3.

```

CREATE TABLE jenis (
  id integer NOT NULL,
  nama character varying(50)
);
ALTER TABLE ONLY jenis
  ADD CONSTRAINT jenis_pkey PRIMARY KEY (id);

```

Kode Sumber 4.3 Implementasi Tabel Jenis

4.3.4. Implementasi Tabel *Users*

Tabel *users* merupakan tabel yang digunakan sebagai penyimpanan data *user* dan nama *device* yang digunakan. Atribut pada tabel *users* adalah id, nama dan *device*. Implementasi tabel ini dapat dilihat pada Kode Sumber 4.4.

```

CREATE TABLE users (
  id integer NOT NULL,
  nama character varying(50),
  device character varying(50)
);
CREATE SEQUENCE users_id_seq
  START WITH 1
  INCREMENT BY 1

```

```

NO MINVALUE
NO MAXVALUE
CACHE 1;
ALTER TABLE ONLY users
  ALTER COLUMN id SET DEFAULT
    nextval('users_id_seq'::regclass);
ALTER TABLE ONLY users
  ADD CONSTRAINT users_pkey PRIMARY KEY (id);

```

Kode Sumber 4.4 Implementasi Tabel *Users*

4.4. Implementasi *Web Service*

Pada subbab ini akan dibahas bagaimana implementasi *web service* dilakukan. *Web service* yang digunakan pada sistem ini menggunakan metode *representational state transfer* (REST) dengan *framework* Slim. Format data yang digunakan dalam proses pertukaran data adalah *JavaScript Object Notation* (JSON). Pada subbab ini akan dibahas implementasi dari proses *query* pada *web service*.

4.4.1. Implementasi *Query User*

Query User merupakan *query* yang dipanggil saat aplikasi pada Android pertama kali dijalankan. *Query* ini bertujuan untuk mengambil id *user* dari tabel *users* berdasarkan nama kode unik Android pada *device* tersebut. Kode unik Android adalah id berupa *string* yang unik dan dimiliki setiap perangkat Android. Apabila kode unik Android belum ada pada tabel *users*, maka kode unik tersebut akan disimpan di kolom nama pada tabel *users* sehingga *device* tersebut memiliki id *user*. Implementasi *query* memperoleh id *user* dapat dilihat pada Kode Sumber 4.5.

```

function getUserId(data){
1.  OPEN db_connection
2.  SET data to json_decode(data)
3.  SET sql_query to "SELECT id FROM users where nama =
   ". data->nama
4.  EXECUTE sql_query
5.  SET result to fetch(sql_query)
6.
7.  IF result is NULL

```

```

8.      THEN
9.      SET sql_statement to "INSERT INTO users
      (nama,device) VALUES (:nama,:device)
      RETURNING id "
10.     PREPARE sql_statement
11.     ADD data->nama to bindParam(:nama)
12.     ADD data->device to bindParam(:device)
13.     EXECUTE sql_statement
14.     SET result to fetch(sql_statement)
15.     ENDIF
16.
17.     CLOSE db_connection
18.     RETURN result->id_user

```

Kode Sumber 4.5 Implementasi *Query User*

4.4.2. Implementasi *Query Event Bump*

Query Event adalah *insert query* yang dipanggil ketika terjadi *event* guncangan atau *bump* pada *device* ketika aplikasi berjalan. *Query* ini akan menerima data berupa *JSONArray* dari *smartphone* lalu memasukkan data tersebut ke dalam tabel *location* dan *accelerometer*. Pada fungsi *insertEvent* terdapat 2 *query* yaitu *query* ketika *insert* ke tabel *location* dan *insert* ke tabel *accelerometer*. *Query* pertama setelah *insert* ke tabel *location* akan mengembalikan *location_id* yang akan digunakan sebagai *foreign key* saat *query insert* data ke tabel akselerometer. *Pseudocode* implementasi *query event* dapat dilihat pada Kode Sumber 4.6.

```

function insertEvent(location,accelerometer){
1.  OPEN db_connection
2.  SET data to json_decode(location)
3.
4.  SET sql_query to "INSERT INTO location (lat, lon,
      jenis_id,user_id) VALUES (:lat, :lon,
      :jenis_id,:user_id) RETURNING id"
5.  PREPARE sql_query
6.  ADD data->lat to bindParam(:lat)
7.  ADD data->lon to bindParam(:lon)
8.  ADD data->jenis_id to bindParam(:jenis_id)
9.  ADD data->user_id to bindParam(:user_id)
10. EXECUTE sql_query
11.
12. SET id_location to fetch(sql_query)
13. SET acc to json_decode(accelerometer)

```

```

14.
15. FOR all items IN acc
16.     DO
17.         SET sql_query2 to "INSERT INTO acc_data (lat,
lon, x, y, z, waktu, location_id) VALUES (:lat,
:lon, :x, :y, :z, :waktu, :location_id)"
18.         PREPARE sql_query2
19.         ADD acc->lat to bindParam(:lat)
20.         ADD acc->lon to bindParam(:lon)
21.         ADD acc->jenis_id to
bindParam(:jenis_id)
22.         ADD acc->x to bindParam(:x)
23.         ADD acc->y to bindParam(:y)
24.         ADD acc->z to bindParam(:z)
25.         ADD acc->user_id to bindParam(:waktu)
26.         ADD location_id to
bindParam(:location_id)
27.         EXECUTE sql_query2
28.     ENDFOR
29.
30. CLOSE db_connection
31. RETURN STATUS_SUCCESS

```

Kode Sumber 4.6 Implementasi *Query Event Bump*

4.4.3. Implementasi Koneksi Basis Data

Koneksi basis data yang digunakan sebagai penghubung antara *web* dengan basis data. Koneksi ini diperlukan untuk proses pengiriman dan penerimaan data yang dibutuhkan antara *server*, peta digital dan aplikasi *smartphone*. Implementasi koneksi basis data dapat dilihat pada Kode Sumber 4.7.

```

getDb()
1. SET db_host
2. SET db_user
3. SET db_pass
4. SET db_name
5. Add to db_connection
6. CONNECT db_connection
7. IF db_connection NOT NULL
8.     THEN RETURN db_connection
9. ELSE RETURN STATUS_FAILED
10. ENDF

```

Kode Sumber 4.7 Implementasi Koneksi Basis Data

4.5. Implementasi Proses Aplikasi

Subbab ini menjelaskan tentang proses-proses yang terjadi dalam aplikasi. Proses-proses yang terjadi mulai dari inialisasi sensor, reorientasi sistem koordinat akselerometer hingga pengiriman data dari *smartphone* ke *server*.

4.5.1. Implementasi Inialisasi Sensor

Pada Kode Sumber 4.8 adalah kode sumber implementasi inialisasi *SensorEventListener* untuk sensor *accelerometer* dan *magnetic field*.

```

11. INITIALIZE SensorManager
12. GET SENSOR_SERVICE
13. INITIALIZE sensor TYPE_ACCELEROMETER
14. INITIALIZE sensor TYPE_MAGNETIC_FIELD
15. REGISTER sensor TYPE_ACCELEROMETER listener
16. REGISTER sensor TYPE_MAGNETIC_FIELD listener

```

Kode Sumber 4.8 Implementasi Inialisasi Sensor

4.5.2. Implementasi Pengiriman Data

Pengiriman data *event* dari *devices* ke *server* menggunakan *library* Volley. Volley adalah HTTP *library* yang biasa digunakan untuk pengiriman jaringan data. Pada Kode Sumber 4.10 dapat dilihat *pseudocode* implementasi pengiriman data *event bump* berupa data *location* dan *accelerometer* setiap terjadi peristiwa lonjakan pada *devices* menggunakan HTTP POST *method* yang diambil dari *library* Volley. Nama fungsi pengirimannya adalah *postEventData*. Data yang dikirimkan dalam bentuk format *JSONArray*. Pada Kode Sumber 4.9 adalah implementasi fungsi *addToJSONArray* yaitu fungsi untuk memasukkan data-data yang mau dikirimkan berupa *JSONObject* kedalam *JSONArray*. Variabel *dataJSONArray* adalah variabel global dengan tipe data *JSONArray* di dalam aplikasi ini.

```

addToJSONArray(data)
1. INITIALIZE json_object
2. FOR all items in data

```



```

3.      DO
4.      PUT all_items to json_object
5.  ENDFOR
6.  PUT json object to dataJsonArray

```

Kode Sumber 4.9 Implementasi Fungsi *addToJSONArray*

```

postEventData(data_array)
1.  INITIALIZE jsonArrayPost
2.
3.  ADD PARAM RequestMethod to POST
4.  ADD PARAM URL
5.  ADD PARAM Data to data_array
6.  ADD PARAM ResponseListener
7.  ADD PARAM ResponseErrorListener
8.
9.  SET PARAM to jsonArrayPost
10. SEND jsonArrayPost

```

Kode Sumber 4.10 Implementasi Fungsi *postEventData*

4.5.3. Implementasi Mengambil Data Sensor

Kode Sumber 4.11 merupakan *pseudocode* untuk mengambil data sensor yaitu sensor akselerometer dan magnetometer setiap kali sensor mengalami perubahan. Fungsi *onSensorChanged* adalah fungsi yang dijalankan ketika sensor mengalami perubahan nilai.

```

onSensorChanged()
1.  IF sensor TYPE_ACCELEROMETER
2.      THEN
3.      get values to accelerometers_array
4.  ENDIF
5.  IF sensor TYPE_MAGNETIC_FIELD
6.      THEN
7.      get values to magnetometers_array
8.  ENDIF

```

Kode Sumber 4.11 Implementasi Mengambil Data Sensor

4.5.4. Implementasi Reorientasi Akselerometer

Reorientasi data akselerometer dari koordinat *devices* menjadi koordinat kendaraan dapat dilihat pada *pseudocode* Kode Sumber 4.12. Fungsi *reorientation* dijalankan dalam fungsi *onSensorChanged*. Apabila terjadi perubahan pada sensor akselerometer maka nilai akselerasi akan direorientasi. Metode

yang digunakan untuk mengimplementasi program sesuai dengan metode reorientasi yang digunakan pada penelitian aplikasi Wolverine [3]. Fungsi untuk mendapatkan *RotationMatrix* sistem koordinat bumi adalah *getRotationMatrix*. *RotationMatrix* harus *ditranspose* agar dapat dimultiplikasi dengan matriks vektor akselerometer [20].

```

reorientation()
1.  INITIALIZE RotationMatrix
2.  INITIALIZE global_accelerometers
3.  INITIALIZE vehicle_accelerometers
4.  GET gravity[] values with HPF(acc)
5.  RotationMatrix :=
    getRotationMatrix(gravity,magnometers)
6.  TRANSPOSE RotationMatrix
7.  global_accelerometers := MULTIPLY accelerometers with
    RotationMatrix
8.  GET bearing FROM LocationManager
9.  GET declination FROM GeomagneticField
10. INITIALIZE HeadingMatrix
11. SET Heading := bearing - declination
12. GET HeadingMatrix from angle Heading in Z-axis
13. vehicle_accelerometers:= MULTIPLY
    global_accelerometers with HeadingMatrix
14. RETURN vehicle_acclerometers

```

Kode Sumber 4.12 Implementasi Reorientasi Akslerometer

4.5.5. Implementasi Deteksi *Event Bump*

Proses pendeteksian *event bump* atau guncangan akibat lonjakan menggunakan algoritma *Z-Thresh* [4]. Apabila data akselerasi pada sumbu z melewati nilai *threshold* maka aplikasi akan mendeteksi bahwa *event bump* sedang terjadi. Setelah berhasil dideteksi maka selanjutnya aplikasi akan merekam aktivitas data pada sensor akselerometer selama 1,5 detik. Kumpulan data akselerometer yang terekam selama 1,5 itu kemudian dikirimkan maka data lokasi dan akselerometer akan dikirimkan ke *server*. Nilai *threshold* terdiri dari nilai batas atas dan batas bawah. Implementasi deteksi *event bump* dapat dilihat pada Kode Sumber 4.13.

```

1.  SET x_acc

```

```

2.  SET y_acc
3.  SET z_acc
4.
5.  IF z_acc > threshold_top
6.      OR z_acc < threshold_bottom
7.      THEN
8.          WHILE 1.5s DO
9.              ADD location to data
10.             ADD accelerometer to data
11.          ENDWHILE
12.          postEventData(data)
13.  ENDIF

```

Kode Sumber 4.13 Implementasi Deteksi *Event Bump*

4.5.6. Implementasi *Decision tree*

Kode Sumber 4.14 merupakan *pseudocode* implementasi untuk klasifikasi *bump* atau normal menggunakan algoritma *decision tree*. Pada setiap data *event*, data akan diklasifikasi berdasarkan fitur yang diekstraksi yaitu simpangan baku, rata-rata dan *diff* (selisih nilai maksimum dan nilai minimum). Fungsi *DecisionTreeClassifier* yang digunakan adalah fungsi klasifikasi dari library *sklearn.tree* Python [21].

```

1.  INITIALIZE data_train
2.  STORE stdZ, diffZ, meanZ, stdY, diffY, meanY to
    data_train
3.  INITIALIZE treeClassifier
4.  SET treeClassifier to
    DecisionTreeClassifier(data_train)
5.  INITIALIZE event_bump
6.  STORE stdZ, diffZ, meanZ, stdY, diffY, meanY to
    event_bump
7.  PREDICT event_bump using treeClassifier
8.  GET predict_result
9.  OPEN dbConnection
10. UPDATE table location SET jenis_id to predict_result
11. CLOSE dbConnection

```

Kode Sumber 4.14 Implementasi Klasifikasi *Decision tree*

4.5.7. Implementasi *Clustering* pada Peta Digital

Pada Kode Sumber 4.15 dapat dilihat *pseudocode* implementasi *clustering* yang digunakan untuk mengelompokkan titik-titik koordinat *bump* yang berdekatan pada peta digital. Fungsi BIRCH merupakan fungsi yang diambil dari *library sklearn.cluster.birch* Python. Parameter *threshold* pada fungsi Birch merupakan jarak maksimum antar koordinat yang boleh dijadikan satu *subcluster*. Data koordinat diperoleh dari database menggunakan fungsi *getBumpLocation*. Data tersebut kemudian *dicluster* menggunakan BIRCH menjadi beberapa kelompok *subcluster*. Setiap *subcluster* memiliki satu titik pusat luasan atau *centroid* yang mewakili titik-titik pada *subcluster* tersebut. Titik-titik tersebut akan ditampilkan pada peta digital.

```

1. INITIALIZE location
2. SET location to getBumpLocation()
3. INITIALIZE map_cluster
4. INITIALIZE Birch, label
5. CLUSTER map_cluster with Birch(threshold = 9 m)
6. SET label to CLUSTER.result
7. INITIALIZE marker, map
8. FOR all items in label
9.     DO
10.        SET marker to centroid(label->position)
11.    ENDFOR
12. ADD marker to map
13. LOAD map

```

Kode Sumber 4.15 Implementasi *Clustering* pada Peta Digital

BAB VI PENUTUP

Pada bab ini akan dibahas kesimpulan yang diperoleh berdasarkan hasil uji coba aplikasi untuk menjawab rumusan masalah dan saran mengenai pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, maka dapat disimpulkan sebagai berikut.

1. Sensor pada *smartphone* Android dapat digunakan untuk mendeteksi *bump*.
2. Peta digital dapat menampilkan lokasi *bump* dan mengestimasi posisinya menggunakan metode *clustering* pada titik-titik koordinat yang berdekatan.
3. Berdasarkan hasil uji coba maka dapat diambil kesimpulan bahwa peletakan posisi *smartphone* didalam kantong atau tas memiliki akurasi yang lebih baik dibandingkan jika *smartphone* dipegang.
4. Kecepatan dengan akurasi yang baik berdasarkan uji coba adalah pada kecepatan sedang atau cepat.
5. Pada kecepatan rendah banyak *false negative* atau *bump* yang tidak terdeteksi.
6. Algoritma pendeteksi *bump* yang digunakan adalah *Z-Threshold* dengan nilai *threshold* dengan akurasi terbaik sebesar 17 pada batas atas dan 3 pada batas bawah. Untuk mengurangi *false positives* yaitu deteksi *bump* akibat anomali jalan (jalan sedikit bergelombang, tidak rata, lubang kecil, pembatas jalan) menggunakan klasifikasi *decision tree*. Dari kombinasi kedua metode tersebut diperoleh nilai akurasi rata-rata sebesar 89.48%.

6.2. Saran

Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan. Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang.

1. Pada penelitian selanjutnya dapat dilakukan pada kendaraan mobil dan pada *range* kecepatan yang lebih luas.
2. Aplikasi ini juga dapat digunakan sebagai *services* pada aplikasi (*crowdsourcing*) seperti Waze atau Google Maps dalam pemetaan *bump* sehingga data yang dihasilkan lebih banyak dan cepat.

DAFTAR PUSTAKA

- [1] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden and H. Balakrishnan, "The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring," *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008)*, pp. 29-39, 2008.
- [2] K. Chen, G. Tan, M. Lu and J. Wu, "CRSM: a practical crowdsourcing-based road surface monitoring system," *Wireless Networks*, pp. 765-779, 2016.
- [3] R. Bhoraskar, N. Vankadhara, B. Raman and P. Kulkarni, "Wolverine: Traffic and road condition estimation using smartphone sensors," *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pp. 1 - 6, 2012.
- [4] A. Mednis, G. Strazdins, R. Zviedris and G. Kanonirs, "Real time pothole detection using Android smartphones with accelerometers," *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1-6, 2011.
- [5] "Polisi Tidur," [Online]. Available: https://id.wikipedia.org/wiki/Polisi_tidur. [Accessed 16 April 2016].
- [6] "Android_(operating_system)," Android, [Online]. Available: [https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system)). [Accessed 2 May 2016].
- [7] "Android_fundamental," Google, [Online]. Available: <https://developer.android.com/guide/components/fundamentals.html>. [Accessed 12 May 2016].

- [8] "LocationListener," Google, [Online]. Available: <https://developer.android.com/reference/android/location/LocationListener.html>. [Accessed 20 May 2016].
- [9] "Volley Library," Google, [Online]. Available: <https://developer.android.com/training/volley/index.html>. [Accessed 14 May 2016].
- [10] K. Douglas and S. Douglas, "PostgreSQL, Second Edition," in *PostgreSQL*, Sams, 2005.
- [11] A. El-Rabanny, "Introduction to GPS The Global Positioning System," Boston, Artech House, 2002.
- [12] "PHP," PHP, [Online]. Available: <https://en.wikipedia.org/wiki/PHP>. [Accessed 17 May 2016].
- [13] [Online]. Available: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)). [Accessed 13 May 2016].
- [14] "Google Maps API," Google, [Online]. Available: <https://developers.google.com/maps/web-services/overview#WebServices>. [Accessed 30 May 2016].
- [15] "BIOSTALL," Biostall, [Online]. Available: <http://biostall.com/demos/google-maps-v3-api-codeigniter-library/>. [Accessed 12 May 2016].
- [16] "CodeIgniter," EllisLab, [Online]. Available: <https://en.wikipedia.org/wiki/CodeIgniter>. [Accessed 12 May 2016].
- [17] "Slim," Slim Framework Team, [Online]. Available: <http://www.slimframework.com/>. [Accessed 12 May 2016].
- [18] . L. Breiman, J. Friedman, R. Olshen and C. J. Stone, *Classification and Regression Tree*, Pacific California: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.
- [19] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large

Databases," *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pp. 103-114, 1996 .

- [20] "Rotation Matrix," [Online]. Available: [https://developer.android.com/reference/android/hardware/SensorManager.html#getRotationMatrix\(float\[\], float\[\], float\[\], float\[\]\)](https://developer.android.com/reference/android/hardware/SensorManager.html#getRotationMatrix(float[], float[], float[], float[])). [Accessed 19 April 2016].
- [21] "DecisionTreeClassifier," scikit-learn, [Online]. Available: <http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>. [Accessed 30 May 2016].
- [22] "Volley," [Online]. Available: <https://developer.android.com/training/volley/index.html>. [Accessed 24 May 2016].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS



Penulis, **Otniel Yehezkiel Bornok Hutabarat**, lahir di Batam, 12 April 1994. Penulis menempuh pendidikan sekolah dasar di SD Kristen Kalam Kudus, Batam. Melanjutkan pendidikan sekolah menengah pertama di SMP Kristen Kalam Kudus Batam dan selanjutnya di SMA Katolik Yos Sudarso Batam. Selanjutnya penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika, Fakultas Teknologi dan Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif menjadi administrator Laboratorium Pemrograman Teknik Informatika, asisten praktikum Dasar Pemrograman, asisten PIKTI dan aktif dalam organisasi tingkat jurusan dan UKM Musik.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Algoritma Pemrograman (AP) dan memiliki ketertarikan dalam bidang *Web and Mobile Application Development*, *Music* dan *Startup*. Penulis dapat dihubungi melalui email: braincreativelife@gmail.com