SKRIPSI

VISUALISASI DATA HISTORI KIRI PADA GOOGLE MAPS

Jonathan Laksamana Purnomo

NPM:

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2020

UNDERGRADUATE THESIS

VISUALIZATION OF KIRI HISTORICAL DATA ON GOOGLE MAPS

Jonathan Laksamana Purnomo

NPM:

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2020

LEMBAR PENGESAHAN

VISUALISASI DATA HISTORI KIRI PADA GOOGLE MAPS

Jonathan Laksamana Purnomo

NPM:

Bandung, 3 Desember 2020

Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
Pascal Alfadian, M.Comp.	

Ketua Tim Penguji	Anggota Tim Penguji
«penguji 1»	**«penguji 2»**

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

VISUALISASI DATA HISTORI KIRI PADA GOOGLE MAPS

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 3 Desember 2020

Meterai
Rp. 6000

Jonathan Laksamana Purnomo
NPM:

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Kata-kata kunci: KIRI, *Navigation System, Data History, Marker Clustering, Heat Map,* Google Javascript API

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Keywords: KIRI, *Navigation System, Data History, Marker Clustering, Heat Map,* Google Javascript API

Dipersembahkan untuk Tuhan YME, keluarga, para dosen, teman-teman yang telah memberi dukungan dalam pembuatan skripsi ini, serta diri sendiri.

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini . . . »
Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, Desember 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi memudahkan manusia untuk mencari berbagai macam informasi. Salah satu informasi yang dapat diperoleh adalah informasi tentang navigasi transportasi publik. KIRI adalah perangkat lunak yang berguna sebagai navigasi antar kota menggunakan transportasi publik dengan menggunakan perangkat peta digital dan informasi posisi dengan menggunakan satelit GPS.[1].

Pada perangkat lunak KIRI seluruh aktivitas yang dilakukan oleh user sudah tercatat. Data yang tercatat disebut juga dengan data histori. Tetapi data histori tersebut belum diolah secara maksimal. Data visualisasi adalah metode yang akan digunakan untuk mengolah data histori sehingga dapat ditemukan pola tertentu.

Metode yang akan digunakan dalam memvisualisasikan data adalah *Heat Map* dan *Marker Clustering*. *Heat Map* adalah teknik visualisasi data yang menunjukkan besarnya suatu fenomena sebagai warna dalam dua dimensi. Sedangkan *Marker Clustering* adalah teknik visualisasi data yang mengelompokan *marker* atau *pointer* yang jarak *latitude* dan *longitude* nya saling berdekatan antara suatu *marker* dengan marker yang lainnya.

1.2 Rumusan Masalah

Rumusan masalah dari topik ini adalah sebagai berikut:

- Bagaimana memvisualisasikan data histori KIRI?
- Bagaimana menemukan pola dari data histori KIRI?

1.3 Tujuan

Tujuan dari topik ini adalah sebagai berikut:

- Mempelajari *Google Maps Javascript API*.
- Melakukan observasi data.

1.4 Batasan Masalah

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris.

Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

1.5 Metodologi

Metodologi yang digunakan dalam penelitian ini adalah:

1. Mempelajari *Google Maps Javascript API* khususnya *Heat Map* dan *Marker Clustering*.
2. Analisis masalah perangkat lunak yang akan dibangun.
3. Merancang perangkat lunak yang akan dibangun.
4. Membangun perangkat lunak yang mengimplementasikan *Heat Map* atau *Marker Clustering* dengan memanfaatkan *Google Maps Javascript API*.
5. Menentukan pola dari hasil visualisasi data.
6. Analisis hasil pengujian dan mengambil kesimpulan.

1.6 Sistematika Pembahasan

Laporan penelitian tersusun ke dalam enam bab secara sistematis sebagai berikut.

- Bab 1 Pendahuluan
  Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
- Bab 2 Dasar Teori
  Berisi metode penentuan pola, *library Google Maps* dan bahasa pemograman *Javascript*
- Bab 3 Analisis
  Berisi analisis masalah terkait implementasi *Goole Map*, studi kasus teknik penentuan pola yang diimplementasikan, dan gambaran umum perangkat lunak yang meliputi diagram aktivitas dan diagram kelas.
- Bab 4 Perancangan Perangkat Lunak
  Berisi perancangan perangkat lunak yang akan dibangun, meliputi perancangan antarmuka, diagram kelas lengkap dan masukan perangkat lunak.
- Bab 5 Implementasi dan Pengujian
  Berisi implementasi antarmuka perangkat lunak, pengujian fungsional, pengujian eksperimental serta kesimpulan dari pengujian.
- Bab 6 Kesimpulan dan Saran
  Berisi kesimpulan dari awal hingga akhir penelitian dan saran untuk penelitian berikutnya.

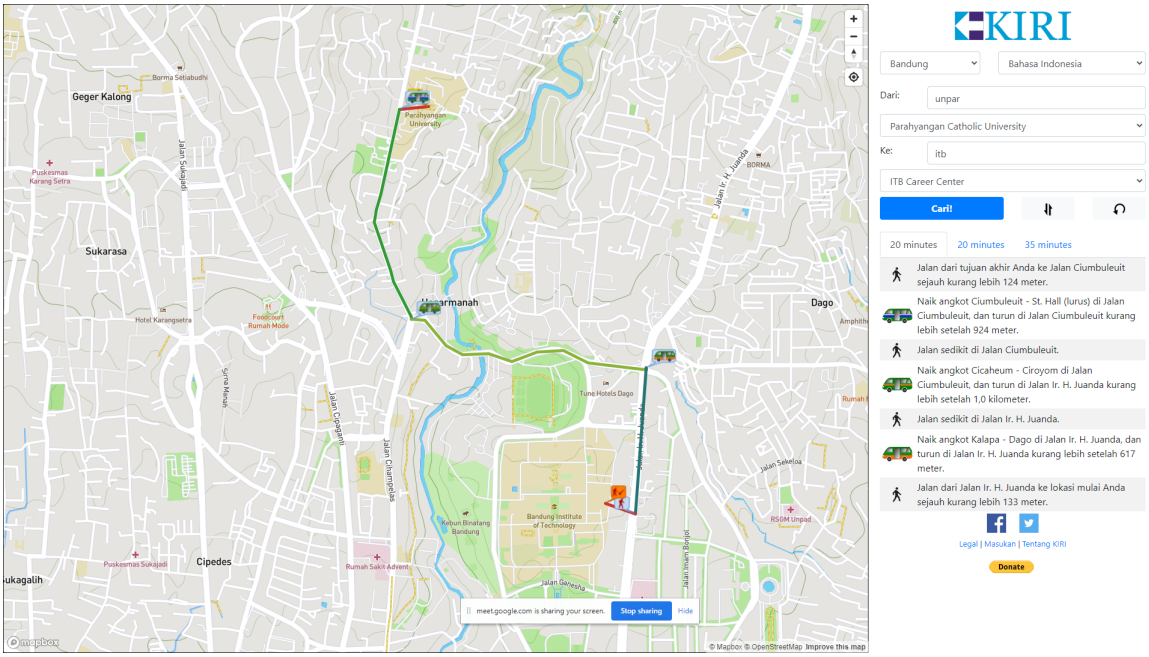
BAB 2

LANDASAN TEORI

2.1 KIRI Website

KIRI adalah aplikasi navigasi angkutan umum berbasis web yang melayani Bandung dan kota-kota lain di Indonesia.[1]. Pada awal pembuatannya KIRI dibuat untuk tujuan komersial. Namun karena dinilai kurang sukses project KIRI sekarang menjadi open source project yang dapat di akses. Aplikasi KIRI memiliki beberapa fitur sebagai berikut:

- Pemilihan rute tercepat menggunakan angkutan kota.
- Menampilkan tempat pergantian angkutan kota.
- Memiliki fitur multi bahasa.
- Memiliki fitur pemilihan lokasi.
- Dapat menampilkan beberapa pilihan rute.
- Dapat menampilkan instruksi lengkap mencapai tujuan.



Gambar 2.1: Tampilan utama website KIRI

2.2 JSON

JSON (*JavaScript Object Notation*) adalah format teks untuk serialisasi data terstruktur. JSON dapat terdiri dari empat tipe primitif (*character*, *number*, *boolean*,dan *null*) dan tiga tipe *refrence* (string, objek dan array).[2] *Object* adalah kumpulan dari nama / nilai yang berpasangan pasangan, di mana nama (*attribute*) adalah string dan nilai (*value*) bisa berupa string, angka, *boolean*, *null*, *object*, atau *array*. Menurut *Internet Assigned Numbers Authority (IANA)* JSON memiliki *meta data* seperti:[3]

- *MIME media type* untuk JSON adalah *text is application/json*.
- *Type name: application*.
- *Subtype name: JSON*
- JSON merupakan suatu format yang digunakan untuk melakukan proses bertukar data yang di support oleh semua bahasa pemrogaman: *ActionScript*,*C* , *C* , *Clojure*, *ColdFusion*, *PHP*, *Python*.

2.2.1 JSON *Data Structure Types*

JSON dapat menerima berbagai macam format tipe data seperti[4]:

- *Character* adalah huruf, angka, spasi, tanda baca, atau simbol yang dapat kenali oleh komputer.
- *number*.
- *boolean* adalah tipe data yang memiliki dua buah kemungkinan (*ture* atau *false*).
- *string* adalah kumpulan dari beberapa *character*.
- *object* adalah tipe data pada *computer* yang memiliki nama dan nilai yang saling terikat.
- *array* adalah sebuah tipe data yang dapat menampung tipe data lainnya.

2.2.2 *JSON Grammar*

pada dasarnya *syntax* pada *JSON* dianggap sebagai bagian dari *syntax javascript* terdiri dari beberapa bagian[2]:

- data direpresentasikan dalam pasangan nama (*atribute*) / nilai (*value*).
- menggunakan kurung kurawal untuk menampung tipe data *object* dan diikuti dengan ':'(*colon*), sebagai penanda batas dari nama dan value dan dipisahkan dengan , (Koma).
- untuk merepresentasikan array dapat menggunakan kurung siku / *square brackets* yang di pisahkan dengan , (koma) untuk setiap elemen.

Listing 2.1: Contoh Struktur JSON

```
{
    "book": [

        {
            "id": "01",
            "language": "Java",
            "edition": "third",
```

```
            "author": "Herbert Schildt"
        },

        {
            "id": "07",
            "language": "C++",
            "edition": "second",
            "author": "E.Balagurusamy"
        }

    ]
}
```

2.2.3 *JSON Parser*

Parsing adalah proses menganalisis teks yang terbuat dari urutan token untuk menentukan struktur tata bahasanya sehubungan dengan tata bahasa formal yang diberikan. dalam melaukan proses *parsing* pada *JSON parser* harus dapat mengubah semua *value* yang terdapat pada *data* sesuai dengan ketentuan pada *JSON Grammar*.

Gambar 2.2: Contoh JSON parser

2.2.4 *Security Considerations*

pada umumnya setiap *scripting languages* memiliki masalah keamanan.JSON merupakan bagian dari *javascript* sebaiknya setiap akan melakukan pertuakaran data menggunakan JSON harus melakukan validasi terlebih dahulu agar dapat memastikan kredebilitas dari data yang diterima maupun dikirim.

2.3 CSV

CSV (*Comma Separated Values*) adalah suatu format data dalam basis data di mana setiap nilai *atribute* dipisahkan dengan tanda koma (,) dan setiap baris data ditandai dengan baris baru.[5] CSV digunakan untuk bertukar data dan mengonversi data dari sebuah program *spreadsheet* ke program *spreadsheet* lainnya. Contoh CSV dapat dilihat pada Listing 2.2.

Listing 2.2: Contoh CSV

```
logId,APIKey,Timestamp (UTC),Action,AdditionalData
113909,E5D9904F0A8B4F99,2/1/2014 0:07,PAGELOAD,/5.10.83.30/
113910,E5D9904F0A8B4F99,2/1/2014 0:07,PAGELOAD,/5.10.83.49/
```

2.3.1 *CSV Grammar*

CSV dapat dituliskan dalam banyak format penulisan. belum ada aturan resmi tentang bagaimana cara penulisan format *CSV*. beberapa aturan dalam penulisan *CSV* yang paling banyak dilakukan adalah:

- setiap *record* dipisahkan perbaris , dan dibatasi oleh *line break / (CLRF)* contoh:

```
aaa,bbb,ccc CRLF
zzz,yyy,xxx CRLF
```

- *record* pada baris terakhir,tidak memerlukan *line break /(CLRF)* contoh:

```
aaa,bbb,ccc CRLF
zzz,yyy,xxx
```

- dapat menambahkan *header*, header dapat diletakan pada baris pertama *csv* contoh:

```
field_name,field_name,field_name CRLF
aaa,bbb,ccc CRLF
zzz,yyy,xxx
```

- setiap *record* harus memiliki jumlah nilai yang sama, spasi meruapakan bagian dari nilai dan tidak boleh di abaikan contoh:

```
aaa,bbb,ccc CRLF
zzz,  yyy  ,xxx
```

- penggunaan *double quote ("")* tidak dilarang namun jika ada record yang menggunakan *double quote* sebaiknya harus diimplementasikan untuk seluruh record contoh:

```
"aaa","bbb","ccc" CRLF
"zzz","yyy","xxx"
```

2.3.2 *CSV mime type*

menurut *Internet Assigned Numbers Authority (IANA) CSV* memiliki mime type:

- *MIME media type name: text*
- *MIME subtype name: csv*
- *Required parameters: none Optional parameters: charset,header*

2.3.3 *Security considerations*

CSV hanya akan berisi *text* sehingga tidak ada potensi untuk mengakibatkan kerugian. Namun secara *theoretical* memungkinkan untuk meletakan *malicious binary data* untuk melakukan *exploit.*

2.4 *Nodejs*

Nodejs adalah *JavaScript runtime environment* dimana dengan menggunakan *nodejs* memungkinkan untuk menjalankan perintah *JavaScript* tanpa harus menggunakan *web browser.Node JS* memungkinkan *developer* untuk melakukan *server-side scripting* dan menghasilkan *dynamic website.* [6]

2.4.1 *package management*

Package manager atau *package-management system* aplikasi yang mengautomatisasi proses *installing*, *upgrade*, dan *delete* suatu aplikasi.[7] *NodeJs* dilengkapi dengan sistem *package management* yang bernama *Node Package Module / NPM* yang berguna untuk mengatur dan mengelola *thrid party aplication.*[8]

2.4.2 *Architecture*

Nodejs merupakan *event-driven programming* yang memungkinkan *developer* membuat *web server* yang *scalable* tanpa harus menggunakan sistem *threading.Nodejs* akan menggunakan *callbacks* sebagai penanda sebuah *event* telah selesai dikerjakan.*Nodejs* dibuat diatas *Google V8 JavaScript engine* dan memiliki lisensi *BSD license.* untuk lisensi *open source*[9].

2.5 *Expressjs*

Expressjs adalah *backend framework* untuk *nodejs. ExpressJs* dibuat oleh TJ Holowaychuk.*Expressjs* yang menyediakan *web application* yang *minimalistic* dan *robust.*[10]

2.5.1 *Routing*

Routing adalah proses untuk menentukan *path* sehingga aplikasi dapat menentukan perintah yang akan dijalankan berdasarkan path yang sudah ditentukan.[11] Pada *expressjs* untuk membuat *routing developer* perlu menentukan tiga hal[12]:

- *http Method* yang akan digunakan seperti (*post,get,put*).
- *path* yang akan digunakan sebagai alamat route.
- *parmeters* nilai yang akan diberikan.
- *callback* fungsi yang akan dipanggil begitu proses path route dijalankan.

contoh *syntax routing* pada *expressjs*:

```
// GET method route
        app.get('/', function (req, res) {
          res.send('GET request to the homepage')
        })

        // POST method route
        app.post('/', function (req, res) {
          res.send('POST request to the homepage')
        })
```

2.5.2 *Security Considerations*

ExpressJs telah mendukung fitur *middleware* [13] sehingga *developer* dapat melakukan authentikasi untuk setiap route yang ingin dijalankan.*Middleware* adalah fungsi yang akan dieksekusi sebelum fungsi utama dijalankan.Contoh *syntax middleware*:

```
const myMiddleware = function (req, res, next) {
  console.log('LOGGED')
  next()
}
```

```
app.use(myMiddleware)

app.get('/', function (req, res) {
  res.send('Hello World!')
})
```

Dengan mengimplentasi *code* diatas maka setiap kali *app* menerima even akan mengeluarkan tulisan "LOGGED" pada terminal.

2.5.3 *Express file structure*

Tidak ada aturan resmi untuk menentukan file structure dalam *expressjs*. Namun ada tiga jenis *file structure* yang paling sering digunakan yakni[14]:

- *Route Listing* dengan menuliskan seluruh *route* pada kelas utama. Contoh:

```
app.get('/', site.index);

// User

app.get('/users', user.list);
app.all('/user/:id/:op?', user.load);
app.get('/user/:id', user.view);
app.get('/user/:id/view', user.view);
app.get('/user/:id/edit', user.edit);
app.put('/user/:id/edit', user.update);

// Posts

app.get('/posts', post.list);
```

- *Route Map* dengan memanfaatkan fungsi *Map* yang disediakan oleh *expressjs*. Contoh:

```
app.map({
  '/users': {
    get: users.list,
    delete: users.delete,
    '/:uid': {
      get: users.get,
      '/pets': {
        get: pets.list,
        '/:pid': {
          delete: pets.delete
        }
      }
    }
  }
});
```

- *MVC* memisahkan module *routing* pada *expressjs* menjadi *Model , View ,Controller*

2.6 Google Maps Javascript API

Google Maps adalah layanan pemetaan web yang dikembangkan oleh Google. Menawarkan citra satelit, foto udara, dan peta jalan yang interaktif, kondisi lalu lintas secara *real time* [15].

Gambar 2.3: Contoh Heat Map

Google Maps dalam service nya telah menyediakan *API (pplication programming interface)* yang dapat di gunakan untuk public. aplication programming interface adalah *computer interface* yang mengatur komunikasi antar perangkat lunak [16]. Google Maps telah menyediakan beberapa teknik pemetaan data yaitu :

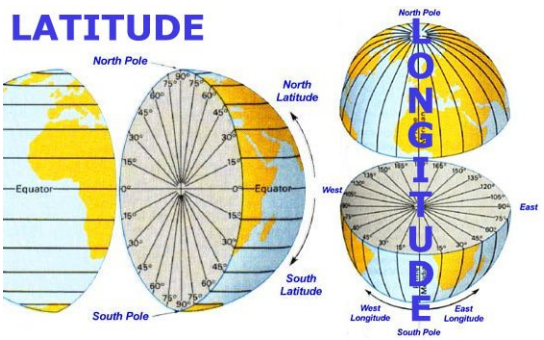
- *Heat Map*
- *Marker Clustering*

terdapat beberapa parmeter untuk menginisialisasi *Google Maps Javascript API*

- *Google Maps Object*
- *Position*
- *Zoom Level*

2.6.1 *Position*

Google Maps menggunakan *Latitude* dan *Longitude* sebagai *atribute* untuk menyatakan sebuah posisi.[17] *Latitude* adalah garis yang horisontal / mendatar. Titik 0 adalah sudut ekuator, tanda + menunjukan arah ke atas menuju kutub utara, sedangkan tanda minus di koordinat Latitude menuju ke kutub selatan. *Longitude* adalah garis lintang . Angka dari sudut bundar bumi horisontal. Titik diawali dari 0 ke 180 derajat, dan 0 ke-180 ke arah sebaliknya.



Gambar 2.4: Contoh Latitude dan Longtitude

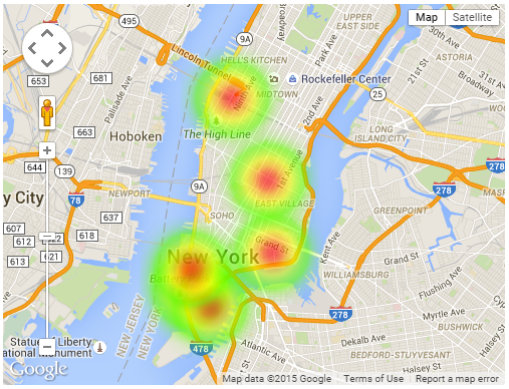
2.6.2 *Google Maps Object*

Maps Object adalah Kelas *JavaScript* yang mewakili peta adalah kelas Peta. Objek kelas ini mendefinisikan satu peta di halaman[18]. (Anda dapat membuat lebih dari satu *instance* kelas ini - setiap objek akan menentukan peta terpisah pada halaman.) Kami membuat *instance* baru dari kelas ini menggunakan *operator* baru *JavaScript*.

```
map = new google.maps.Map(document.getElementById("map"), {...});
```

2.6.3 *Heat Map*

Heat Map adalah sebuah teknik visualisasi data dimana data digambarkan sebagai warna. Saat Lapisan *heat map* diaktifkan, hamparan berwarna akan muncul di atas map. Secara *default*, area dengan intensitas lebih tinggi akan diwarnai merah, dan area dengan intensitas lebih rendah akan tampak hijau.[19]



Gambar 2.5: Contoh Heat Map

Beberapa Jenis *Heat Map*:

- *Click-tracking map*
- *Scroll map*
- *Hover/Mouse tracking map*
- *Eye tracking map*

Untuk menampilkan *heat map* menggunakan *Google Javascript API* dapat menggunakan syntax.

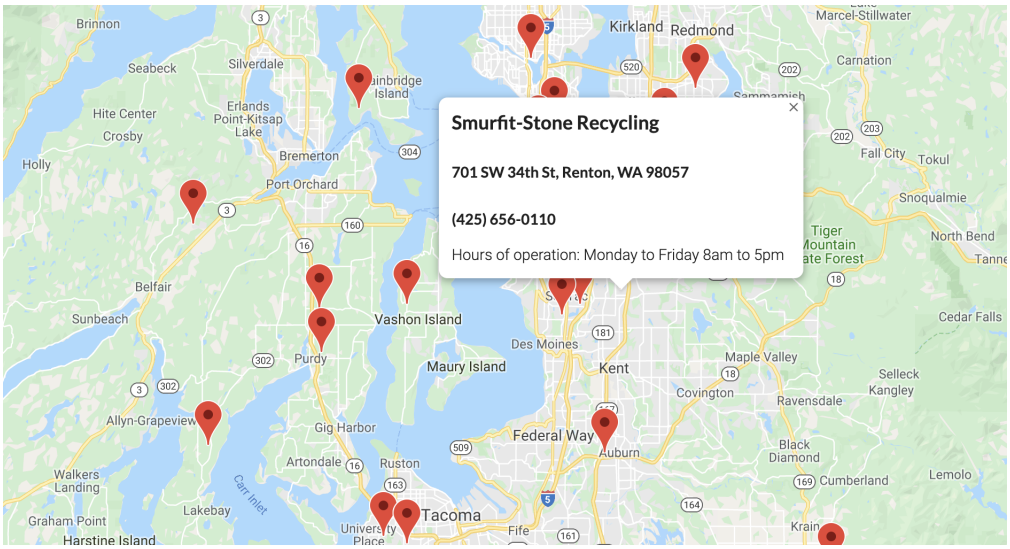
```
var heatmap = new google.maps.visualization.HeatmapLayer({
  data: heatMapData
});
heatmap.setMap(map);
```

2.6.4 *Marker*

Marker adalah suatu tanda yang di pasang di dalam sebuah maps atau peta sebagai petanda lokasi atau tempat. Marker dapat menampilkan gambar khusus, dalam hal ini biasanya disebut sebagai *icon.*[20]. Untuk Menambahkan Marker.Untuk menambahkan *Marker* pada *Google Map Javascript Api* membutuhkan beberapa parameter

- *Position* menggunakan latlng untuk mengidentifikasi letak suatu tempat

- *Map* menggunakan object map dari google javascript api



Gambar 2.6: Contoh Marker

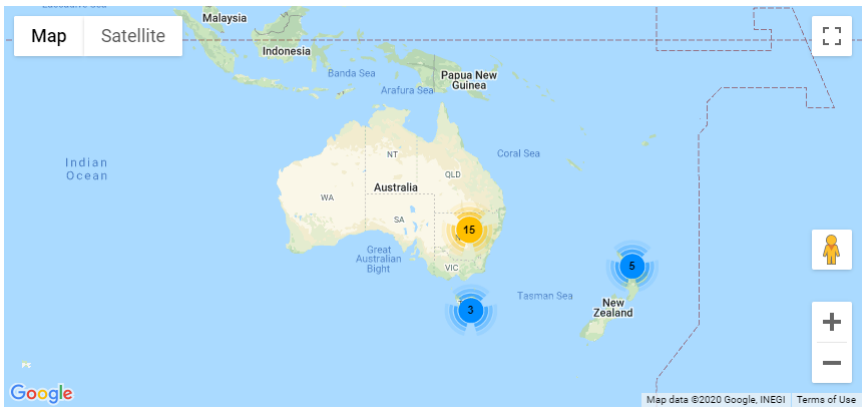
Untuk menambahkan *Marker* dapat menggunakan *syntax* seperti:

```
var heatmap = new google.maps.visualization.HeatmapLayer({
  data: heatmapData
});
heatmap.setMap(map);
```

Gambar 2.7: Add Marker

2.7 *Marker Clustering*

Marker Clustering adalah teknik visualisasi data dimana data akan di representasikan sebagai tanda / *Mark* pada suatu space 2 dimensi, semakin banyak data yang terdapat pada suatu tempat maka akan semakin banyak quantitas penanda / *Mark* yang di berikan. [21]



Gambar 2.8: Contoh Marker Clustering

Untuk menggunakan *Marker Clustering* pada *Google Maps Javascript API* dapat menggunakan Object *Marker Clusterer* yang telah di sediakan oleh *Google Maps*

```
  new MarkerClusterer(map, markers, {
    imagePath:
      "https://developers.google.com/maps/documentation/javascript/examples/marker
  });
}
```

Marker Clustering akan menggunakan *grid-based clustering technique* yang membagi peta menjadi kotak dengan ukuran tertentu (ukuran berubah di setiap tingkat zoom), dan mengelompokkan penanda ke dalam setiap kisi persegi. Ini membuat *cluster* di *marker* tertentu, dan menambahkan marker yang berada dalam batas-batasnya ke *cluster.Marker Clustering* merupakan sebuah teknik yang menggunakan *Marker* sehingga untuk dapat memunculkan *Marker Clustering* diperlukan object *Marker* berikut ini contoh pengimplenentasian *Marker Clustering*[21]

```
function initMap() {
  const map = new google.maps.Map(document.getElementById("map"), {
    zoom: 3,
    center: { lat: -28.024, lng: 140.887 },
  });
  const labels = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  const markers = locations.map((location, i) => {
    return new google.maps.Marker({
      position: location,
      label: labels[i % labels.length],
    });
  });
  new MarkerClusterer(map, markers, {
    imagePath:"",
  });
}
const locations = [
  { lat: -31.56391, lng: 147.154312 },
  { lat: -33.718234, lng: 150.363181 },
  { lat: -33.727111, lng: 150.371124 },
];
```

DAFTAR REFERENSI

[1] Nugroho, P. dan Natali, V. (2017) Open sourcing proprietary application case study: Kiri website. *International Journal of New Media Technology*, **4**, 82–86.

[2] Bray, T. dan Google (2014) he javascript object notation (json) data interchange format. Technical Report 8259.

[3] Crockford, D. Json iana. https://www.iana.org/assignments/media-types/application/json.

[4] Team, W. Ecma-404 the json data interchange standard. https://www.json.org/json-en.html.

[5] Shafranovich, Y. (2005) Common format and mime type for comma-separated values (csv) files. Technical Report 7111.

[6] Dahl, R. nodejs. https://nodejs.org/en/about/.

[7] web archive package manager. https://web.archive.org/web/20171017151526/http://aptitude.alioth.debian.org/doc/en/pr01s0.html.

[8] npmjs docs Npm. https://docs.npmjs.com/about-npm.

[9] Team, J. Nodejs platform architecture. https://en.wikipedia.org/wiki/Node.js#Platform_architecture.

[10] web docs, M. Express. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction.

[11] Rekhter, Y. (1992) A unified approach to inter-domain routing. Technical Report 8259.

[12] Team, E. Express routing. https://expressjs.com/en/guide/routing.html.

[13] js team, E. Expressjs middleware. https://expressjs.com/en/guide/using-middleware.html.

[14] Team, E. Express file structure. https://expressjs.com/en/starter/faq.html.

[15] Mehta, H., Kanani, P., dan Lande, P. (2019) Google maps. *International Journal of Computer Applications*, **178**, 41–46.

[16] Libby, A. (2020) Working with the API.

[17] Developers, G. Coordinates. https://developers.google.com/maps/documentation/javascript/reference/coordinates.

[18] Developers, G. Maps object. https://developers.google.com/maps/documentation/javascript/reference/map.

[19] Developers, G. Coordinates. https://developers.google.com/maps/documentation/javascript/reference/visualization.

[20] Developers, G. Marker. https://developers.google.com/maps/documentation/javascript/reference/marker.

[21] Developers, G. Marker clustering. https://developers.google.com/maps/documentation/javascript/marker-clustering.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

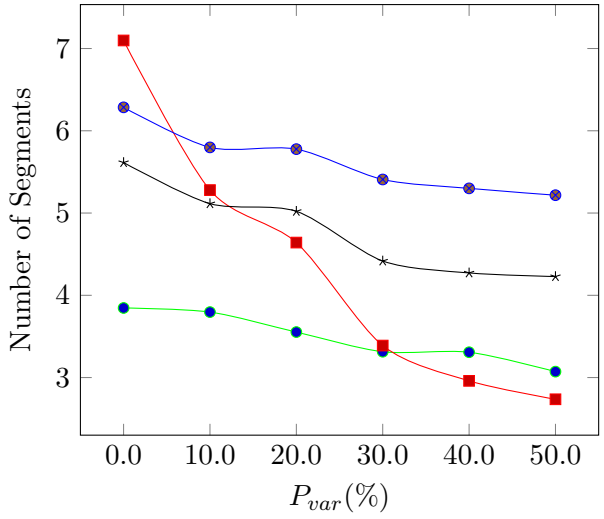
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

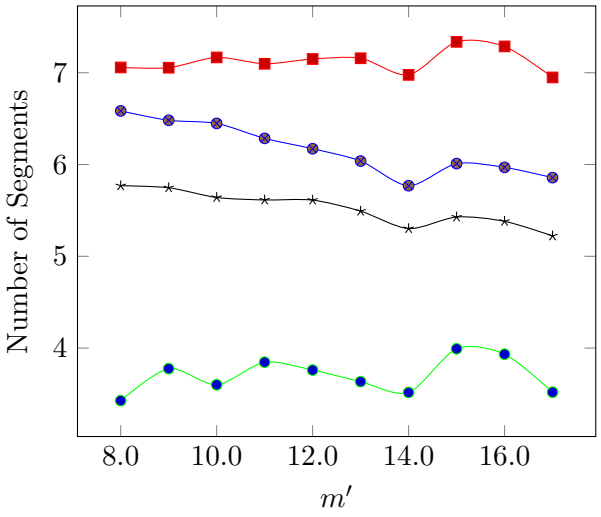
LAMPIRAN B

HASIL EKSPERIMEN

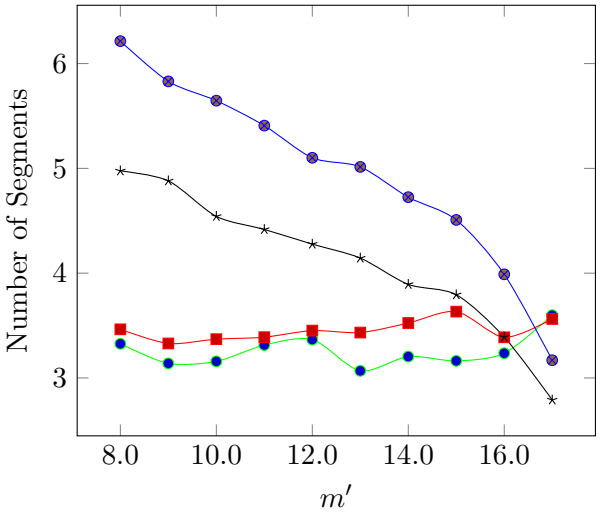
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



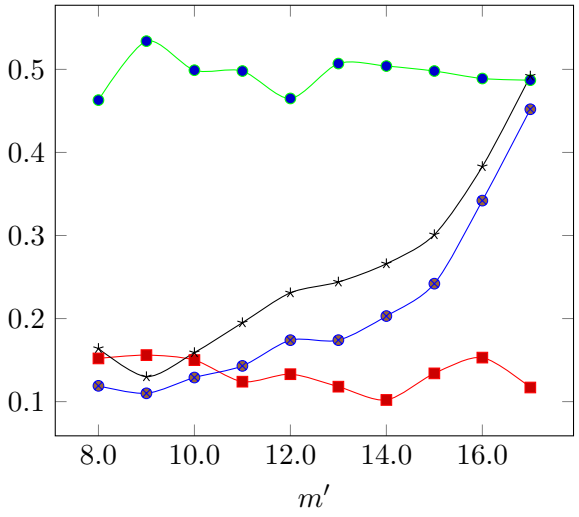
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4