

**SKRIPSI**

**ANALISI WAKTU TEMPUH KOTA BANDUNG**



**FRASETIAWAN HIDAYAT**

**NPM: 2010730121**

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



**UNDERGRADUATE THESIS**

**ANALYSIS OF TRAVEL TIME BANDUNG CITY**



**FRASETIAWAN HIDAYAT**

**NPM: 2010730121**

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



## ABSTRAK

Dalam melakukan suatu perjalanan, manusia melalui suatu jalur yang relatif konstan dimana jalur tersebut akan menjadi rutinitas yang akan dilalui. Dari jalur tersebut sering kali terjadi kemacetan dan biasanya kemacetan itu terjadi pada jam-jam tertentu.

Pada kota-kota besar sering terjadi kemacetan. Efeknya adalah keterlambatan yang akan mempengaruhi seluruh rangkaian kegiatan yang telah direncanakan. Bandung adalah salah satunya dari kota besar yang sering mengalami kemacetan ini dan terkadang kemacetan sendiri tidak dapat diprediksi. Kemacetan ini sendiri bisa dianalisis dengan menentukan pada pukul berapa sajakah terjadi kemacetan pada jalur yang ditempuh.

Dengan memanfaatkan Google Direction yang dimana Google Direction itu sendiri adalah suatu layanan web untuk menghitung arah antar lokasi. Dengan layanan web ini, pengguna bisa mendapatkan data waktu tempuh dari lokasi awal sampai lokasi tujuan. Cara mendapatkan data waktu tempuh tersebut adalah dengan input berupa URL beserta dengan parameter wajib dan beberapa parameter opsional. Parameter wajib yang dimasukkan kedalam URL adalah *origin* yang berupa suatu titik *longitude* dan *latitude* dari tempat asal keberangkatan, *destination* yang berupa suatu titik *longitude* dan *latitude* dari tempat tujuan, dan *keys* yang didapatkan dari Google Console. Pengguna menyematkan parameter-parameter tersebut kedalam URL dan akan menghasilkan suatu *output* dengan menggunakan suatu format. Salah satu dari format itu adalah format JSON.

Aplikasi sederhana yang akan dibangun bertujuan untuk mengekstraksi data waktu tempuh dari input request dalam satu hari selama satu minggu. Aplikasi tersebut berbasis Java dengan memanfaatkan *library* jsoup untuk bisa melakukan *request* ke layanan Google Direction dan *library* JSON untuk melakukan ekstraksi data waktu tempuh. Pengujian dari aplikasi sederhana ini dilakukan dengan menggunakan *test case* dengan melakukan permintaan pada suatu hari. Berdasarkan hasil pengujian, aplikasi dapat berjalan dengan baik dan memberikan keluaran file .csv yang akan dianalisis untuk memberikan waktu terbaik dalam melakukan perjalanan dengan bantuan aplikasi Microsoft Excel. Hasil pengujian aplikasi sederhana ini membuktikan bahwa Google Direction API dapat dimanfaatkan untuk menganalisis waktu tempuh antar titik agar mendapatkan waktu tempuh yang optimal.

**Kata-kata kunci:** Kemacetan, Kota Bandung, Google Direction, JSON, Java, jsoup, Microsoft Excel.



## ABSTRACT

In doing a travel, man through a relatively constant path where the path will be a routine to be traversed. From this path there is often a traffic jam and usually the jam occurs at certain hours.

In big cities there are frequent congestion. The effect is the delay that will affect the whole set of planned activities. Bandung is one of the big cities that often experience this bottleneck and sometimes congestion itself can not be predicted. The congestion itself can be analyzed by determining at what time there are congestion on the path taken.

By using Google Direction which Google Direction itself is a web service to calculate the direction between locations. With this web service, users can get data travel time from start location to destination location. How to get the data travel time is with the input of the URL along with mandatory parameters and some optional parameters. The mandatory parameter entered into the URL is the origin in the form of a longitude and latitude point from the place of origin of departure, destination which is a point of longitude and latitude of the destination, and keys obtained from Google Console. The user embeds these parameters into the URL and will generate an output using a format. One of those formats is the JSON format.

A simple application to be built aims to extract data travel time from input request in one day for one week. The application is based on Java by using jsoup library to be able to request to service Google Direction and JSON library to extraction time travel data. Testing of this simple application is done by using a test case by making a request on one day. Based on the test results, the application can run well and provide output .csv files to be analyzed to provide the best time to travel with the help of Microsoft Excel applications. The results of testing this simple application proves that Google Direction API can be utilized to analyze the travel time between points in order to get the optimal travel time.

**Keywords:** Congestion, Bandung City, Google Direction, JSON, Java, jsoup, Microsoft Excel.





# DAFTAR ISI

DAFTAR ISI	ix
DAFTAR GAMBAR	xi
DAFTAR TABEL	xii
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi	3
1.6 Sistematika Pembahasan	3
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 Protokol HTTP	5
2.1.1 Transaksi HTTP	5
2.1.2 Kode Status	7
2.1.3 <i>Request method</i>	9
2.1.4 <i>Response Headers</i>	9
2.2 <i>Library</i> jsoup	10
2.2.1 Fungsi jsoup	11
2.2.2 Kelas- kelas jsoup	11
2.3 <i>JavaScript Object Notation (JSON)</i>	12
2.3.1 Struktur JSON	12
2.3.2 Bentuk-Bentuk JSON	12
2.3.3 Value JSON	13
2.3.4 kelas-kelas pada <i>Library</i> JSON	14
2.4 Google Direction	15
2.4.1 Permintaan Arah	16
2.4.2 Parameter Permintaan	16
2.4.3 <i>Response</i> Arah	18
2.4.4 Elemen <i>Response</i> Arah	19
<b>3 ANALISIS</b>	<b>23</b>
3.1 Flow Chart Alur Layanan Google Direction	23
3.2 Analisis permintaan ke layanan Google Direction	24
3.2.1 Parameter yang digunakan	24
3.3 Analisis response dari layanan Google Directions	24
3.4 Gambaran Umum Perangkat Lunak	25
3.5 Analisis Perangkat Lunak	25
3.6 Analisis <i>Use Case</i>	25
3.6.1 Diagram <i>Use Case</i>	25

3.6.2	Skenario <i>Use Case</i> . . . . .	26
3.7	Analisis Kelas . . . . .	27
<b>4</b>	<b>PERANCANGAN</b>	<b>29</b>
4.1	Kebutuhan Masukan dan Keluaran . . . . .	29
4.1.1	Masukan . . . . .	29
4.1.2	Keluaran . . . . .	29
4.2	Parameter <i>request</i> ke layanan Google Direction . . . . .	29
4.3	Rancangan file keluaran . . . . .	30
4.4	Diagram Kelas Rinci . . . . .	30
4.5	Perancangan Antarmuka . . . . .	38
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>41</b>
5.1	Implementasi . . . . .	41
5.1.1	Lingkungan Implementasi . . . . .	41
5.1.2	Implementasi Kode Program . . . . .	41
5.1.3	Tampilan antarmuka . . . . .	42
5.2	Pengujian . . . . .	42
5.2.1	Pengujian Fungsional . . . . .	42
5.2.2	Pengujian Eksperimental . . . . .	45
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>47</b>
6.1	Kesimpulan . . . . .	47
6.2	Saran . . . . .	47
	<b>DAFTAR REFERENSI</b>	<b>49</b>
<b>A</b>	<b>KODE PROGRAM PADA <i>package Module</i></b>	<b>51</b>
<b>B</b>	<b>KODE PROGRAM PADA <i>package Controller</i></b>	<b>55</b>
<b>C</b>	<b>KODE PROGRAM PADA <i>package View</i></b>	<b>57</b>
<b>D</b>	<b>DATA HASIL PENGUJIAN</b>	<b>63</b>
<b>E</b>	<b>HASIL PENGUJIAN EKSPERIMENTAL</b>	<b>81</b>

## DAFTAR GAMBAR

2.1	HTTP Request . . . . .	6
2.2	HTTP Respond . . . . .	6
2.3	Transaksi sederhana . . . . .	7
2.4	JSON Object . . . . .	13
2.5	JSON Object . . . . .	13
2.6	Value . . . . .	13
2.7	String . . . . .	14
2.8	Angka . . . . .	14
3.1	Flow Chart Alur Layanan Google Direction . . . . .	23
3.2	Diagram <i>Use Case</i> Perangkat Lunak . . . . .	26
3.3	Diagram Kelas untuk Perangkat Lunak . . . . .	27
4.1	Kelas Diagram Rinci . . . . .	38
4.2	Antarmuka Utama . . . . .	39
4.3	Antarmuka <i>File Chooser</i> . . . . .	39
5.1	Implementasi Antarmuka Utama . . . . .	42
5.2	Implementasi <i>file chooser</i> . . . . .	42
E.1	Hasil Pengujian Eksperimental . . . . .	84
E.2	Hasil Pengujian Eksperimental . . . . .	87
E.3	Hasil Pengujian Eksperimental . . . . .	90
E.4	Hasil Pengujian Eksperimental . . . . .	93

## DAFTAR TABEL

2.1	Tabel Kode Status . . . . .	8
2.2	Tabel Request Method . . . . .	9
2.3	Tabel Response Headers . . . . .	10
5.1	Tabel Hasil Pengujian Fungsional . . . . .	44
D.1	Data sampel 1 pada tanggal 8 Mei 2017 - 14 Mei 2017 . . . . .	67
D.2	Data sampel 1 pada tanggal 15 Mei 2017 - 21 Mei 2017 . . . . .	71
D.3	Data sampel 2 pada tanggal 15 Mei 2017 - 21 Mei 2017 . . . . .	75
D.4	Data sampel 2 pada tanggal 8 Mei 2017 - 14 Mei 2017 . . . . .	79

# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Dalam melakukan kegiatan dan rutinitas, manusia akan melakukan perpindahan tempat dari suatu tempat ke tempat lain. Salah satu contohnya adalah melakukan kegiatan perkuliahan. Dalam melakukan kegiatan tersebut, mahasiswa harus berpindah dari rumah ke tempat perkuliahan diselenggarakan. Dalam melakukan suatu perpindahan itu, kita melalui suatu jalur yang relatif konstan dimana jalur tersebut akan menjadi rutinitas yang akan dilalui. Dari jalur tersebut sering kali terjadi kemacetan dan biasanya kemacetan itu terjadi pada jam-jam tertentu.

Pada kota-kota besar sering terjadi kemacetan. Efeknya adalah keterlambatan yang akan mempengaruhi seluruh rangkaian kegiatan yang telah direncanakan. Bandung adalah salah satunya dari kota besar yang sering mengalami kemacetan ini dan terkadang kemacetan sendiri tidak dapat diprediksi.

Dengan demikian, untuk merencanakan segalanya agar berjalan sesuai dengan rencana, perlu untuk mengetahui waktu tempuh yang paling cepat dari jalur yang relatif konstan agar tidak terjebak dalam kemacetan. Kemacetan ini sendiri bisa dianalisis dengan menentukan pada pukul berapa sajakah terjadi kemacetan pada jalur yang ditempuh.

Salah satu teknologi yang telah ada, *Google Direction* adalah suatu layanan web untuk menghitung arah antar lokasi. Layanan web ini didesain menghitung arah alamat statis untuk penempatan konten aplikasi pada peta (*Google Maps*). Dengan layanan web ini juga kita bisa mendapatkan data waktu tempuh dari lokasi awal sampai lokasi tujuan dengan input berupa URL beserta dengan parameter wajib dan beberapa parameter opsional yang bisa disesuaikan dengan kebutuhan seperti waktu keberangkatan dan model lalu lintas apakah optimis atau pesimis yang akan mempengaruhi waktu tempuh. Pesimis adalah model lalu lintas dengan memperhitungkan kemacetan dan optimis adalah model lalu lintas yang tidak memperhitungkan kemacetan. Salah satu format output *Google Direction* yang dikeluarkan berupa JSON (*JavaScript Object Notation*).

JSON sendiri adalah suatu format pertukaran data yang ringan agar bisa dibaca dan dibuat oleh komputer. JSON juga standar dipakai oleh manusia untuk dapat berkomunikasi dengan tidak terikat pada satu sistem operasi atau bahasa pemrograman agar bisa berkomunikasi dengan komputer dan bisa diakses oleh aplikasi lain.

*Google Direction* sendiri menggunakan protokol HTTP untuk bisa saling berkomunikasi dengan aplikasi. Protokol HTTP (*HyperText Transfer Protocol*) merupakan protokol yang berjalan diatas protokol TCP (*Transmission Control Protocol*) pada port 80 yang digunak-

an untuk mengirim dokumen atau halaman. Pesan protokol http diformat untuk dapat ditampilkan pada aplikasi.

Dalam penelitian ini, akan dibuat sebuah perangkat lunak yang dapat menampilkan hasil analisis dari data yang didapatkan dari Google Direction API. tujuan aplikasi ini adalah untuk membantu mengambil keputusan pada jam berapakah harus melakukan perjalanan dengan waktu tempuh yang tercepat dengan data-data yang telah ada dalam kurun waktu 7 hari. Aplikasi ini memanfaatkan layanan dari *Google* yaitu *Google Direction* untuk mendapatkan data-data waktu tempuh dari suatu jalur. Pada penelitian ini menggunakan 2 sampel yaitu : menghitung waktu tempuh dari Universitas Katolik Parahyangan dengan alamat Jln. Ciumbuleuit No.94 dan Komplek Amaya Residence, menghitung waktu tempuh dari Universitas Katolik Parahyangan dengan alamat Jln. Ciumbuleuit No.94 dan Komplek Taman Puspa Indah.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan, rumusan masalah pada penelitian ini adalah:

- Bagaimana cara menggunakan Google Direction API dalam bahasa Java?
- Bagaimana memanfaatkan layanan Google Direction API untuk memberikan kesimpulan waktu perjalanan terbaik?
- Kapan waktu terbaik untuk berangkat/pulang untuk dua sampel tempat yang dimaksud?

## 1.3 Tujuan

Berdasarkan rumusan masalah di atas, maka tujuan dari penelitian ini adalah:

- memahami cara menggunakan Google Direction API.
- memahami layanan Google Direction API untuk memberikan kesimpulan waktu perjalanan terbaik.
- memutuskan kapan waktu terbaik untuk berangkat/pulang untuk dua sampel yang dimaksud.

## 1.4 Batasan Masalah

Batasan masalah yang akan digunakan untuk penelitian ini adalah:

1. Output dari permintaan komunikasi menggunakan format JSON.
2. Cakupan wilayah yang akan dihitung waktu tempuhnya adalah kota Bandung.
3. Waktu tempuh dihitung setiap jam dalam satu hari.
4. Waktu tempuh dihitung setiap hari dalam seminggu.

- 1 5. Menghitung Waktu tempuh dengan sampel yang beralamat Jln. Ciumbuleuit No.94,  
2 Komplek Amaya Residence dan Komplek Taman Puspa Indah.
- 3 6. Program dijalankan selalu dari hari Senin.

## 4 1.5 Metodologi

5 Dalam penyusunan skripsi ini mengikuti langkah-langkah metodologi penelitian sebagai  
6 berikut :

- 7 1. Melakukan studi pustaka untuk dijadikan referensi dalam melakukan pembangunan  
8 aplikasi Analisis waktu tempuh kota Bandung,
- 9 2. Melakukan analisis *Google Direction* untuk mendapatkan hasil waktu tempuh dari  
10 tujuan asal ke tujuan akhir,
- 11 3. Melakukan perancangan perangkat lunak,
- 12 4. Melakukan uji coba sesuai dengan sampel,
- 13 5. Melakukan penarikan kesimpulan dan saran pada hasil analisis tersebut.

## 14 1.6 Sistematika Pembahasan

15 Sistematika penulisan laporan pada skripsi ini adalah sebagai berikut :

- 16 1. Bab Pendahuluan  
17 Bab 1 berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi  
18 penelitian, dan sistematika pembahasan dalam pelaksanaan penelitian ini.
- 19 2. Bab Dasar Teori  
20 Bab 2 berisi tentang definisi-definisi dasar teori tentang *Google direction* beserta teori  
21 pendukung lainnya.
- 22 3. Bab Analisis  
23 Bab 3 berisi analisis *Google Direction*, analisis teori pendukung lainnya dan analisis  
24 perangkat lunak.
- 25 4. Bab Perancangan  
26 Bab 4 berisi tentang pembahasan mengenai perancangan perangkat lunak.
- 27 5. Bab Implemntasi dan Pengujian  
28 Bab 5 berisi tentang pengimplementasian perangkat lunak.
- 29 6. Bab Kesimpulan dan Saran  
30 Bab 6 berisi penarikan kesimpulan selama menyelesaikan skripsi dan saran yang diu-  
31 sulkan untuk penelitian berikutnya.





## BAB 2

### LANDASAN TEORI

Pada bab ini akan diuraikan teori-teori yang akan digunakan untuk pembangunan aplikasi ke analisis kota Bandung. Teori-teori tersebut adalah tentang protokol HTTP, *library* Jsoup meliputi kelas jsoup dan Connection. Selain itu akan dibahas juga mengenai *JavaScript Object Notation (JSON)* meliputi kelas pada *library* JSON : JSONObject dan *Google Direction API*.

#### 2.1 Protokol HTTP

HTTP(*HyperText Transfer Protocol*) adalah protokol di balik World Wide Web. Dengan setiap transaksi web, HTTP dipanggil. HTTP adalah di balik setiap permintaan dokumen web atau grafis, setiap klik link hypertext, dan setiap penyerahan formulir. Web adalah tentang penyebaran informasi melalui Internet, dan HTTP adalah protokol yang digunakan untuk melakukannya.

##### 2.1.1 Transaksi HTTP

Berikut akan diilustrasikan transaksi web umum, menunjukkan HTTP yang dipertukarkan antara program *client* dan *program* server. [1]:

- berikut diberikan sebuah url : `http://hypothetical.ora.com:80/`.
- Browser akan menginterpretasikan URL tersebut sebagai berikut :
  - `http : //` : menggunakan protokol HTTP.
  - `hypothetical.ora.com` : menghubungi komputer melalui jaringan dengan hostname `hypothetical.ora.com`.
  - `: 80` : Terhubung ke komputer di port 80. Nomor port IP nomor dari 1 sampai 65535. Jika titik dua dan nomor port dihilangkan, nomor port diasumsikan nomor port *default* HTTP, yang merupakan 80.
  - `:` : Apapun setelah nama host dan nomor port opsional dianggap sebagai jalan dokumen. Dalam ilustrasi ini, jalan dokumen adalah .
- Pada ilustrasi ini browser menghubungkan ke `hypothetical.ora.com` pada port 80 menggunakan protokol HTTP. Pesan bahwa browser mengirimkan ke server adalah sebagai berikut:

```

GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/
    jpeg, image/png, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE
    5.01; Windows NT)
Host: hypothetical.ora.com
Connection: Keep-Alive

```

Gambar 2.1: HTTP Request[1]

- 1 • Pada baris pertama pada request (Gambar 2.1) disebut dengan request line dan diawa-
- 2 li dengan *request method*(metode permintaan), dalam gambar tersebut adalah GET.
- 3 *Request method* diikuti dengan *resource* yang diinginkan, dalam gambar tersebut ada-
- 4 lah /. *Request line* diakhiri dengan versi protokol yang digunakan dalam gambar
- 5 diatas adalah HTTP/1.1.
- 6 • baris kedua dan baris-baris berikutnya sampai ditemukan baris kosong, berisi request
- 7 headers dalam format *nama-header:nilai-header*. pada gambar 2.1 terdapat header
- 8 host yang menandakan bahwa browser ingin mengakses situs dari nilai yang ada di
- 9 header host.
- 10 • Dibawah header-header pada gambar 2.1 terdapat baris kosong di akhir *request*. pada
- 11 *request*, baris kosong memisahkan antara *request headers* dengan *request body*(tubuh
- 12 permintaan).
- 13 Setelah *client* memberikan *request* server memberikan *response*. Dari kasus diatas ber-
- 14 ikut adalah sebagai berikut :

```

HTTP/1.1 200 OK
Date: Mon, 06 Dec 1999 20:54:26 GMT
Server: Apache/1.3.6 (Unix)
Last-Modified: Fri, 04 Oct 1996 14:06:11 GMT
ETag: "2f5cd-964-381e1bd6"
Accept-Ranges: bytes
Content-length: 327
Connection: close
Content-type: text/html

<title>Sample Homepage</title>

<h1>Welcome</h1>
Hi there, this is a simple web page.  Granted,
it may not be as elegant as some other web
pages you've seen on the net, but there are
some common qualities:

<ul>
  <li> An image,
  <li> Text,
  <li> and a <a href="/example2.html"> hyperlink. </a>
</ul>

```

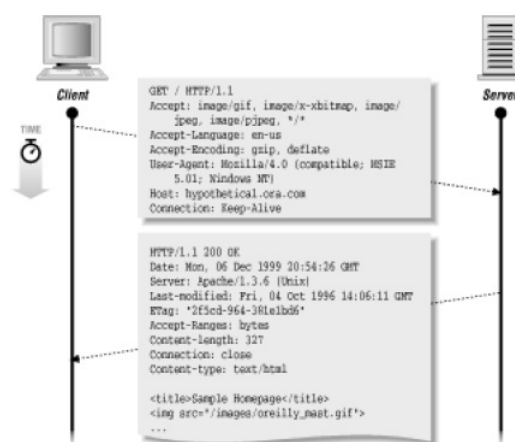
Gambar 2.2: HTTP Respond[1]

- 15 • Pada baris pertama pada respon (Gambar 2.2) disebut *status line*, dan diawali dengan
- 16 versi protokol yang digunakan, dalam kasus ini HTTP/1.1. *Status line* diikuti dengan

3 digit kode status, dalam kasus ini 200. *Status line* diakhiri dengan representasi tekstual dari status tersebut dalam kasus ini *OK*.

- Baris kedua dan baris-baris berikutnya sampai ditemukan baris kosong, berisi request headers dalam format *nama-header:nilai-header*. pada gambar 2.2 terdapat header server yang menandakan bahwa server yang digunakan untuk melayani request.
- Setelah baris kosong adaah *body* dari *response*, gambar 2.2 berupa teks HTML.
- Pada gambar 2.2 ada kebutuhan akan *file* oreilly\_mast.gif di HTML ini. *File* tersebut akan diunduh secara terpisah, tetapi juga dengan protokol HTTP.

Setelah semua terjadi dan dibaca dengan baik, maka baris kosong dan teks dokumen muncul. dengan demikian transaksi yang terjadi adalah sebagai berikut :



Gambar 2.3: Transaksi Sederhana[1]

Berdasarkan gambar 2.3 terjadi transaksi data antara *client* dan *server* berikut adalah penjelasannya.

1. *Client* melakukan *request* dengan *header get*. Selain itu juga terdapat *header hostname* *hypotetical.ora.com* yang menjelaskan hostname yang dituju. Pada *request* tersebut terdapat header protokol HTTP dan juga versi berapa yang digunakan.
2. Diterima oleh *server* dan *server* kemudian membalas dengan *header ok*. Arti *header* tersebut adalah *request* yang diterima telah tersampaikan dengan baik dan mengembalikan HTML yang diminta ke *client*. Dari *respond* tersebut terdapat header : tanggal detail meliputi waktu pada saat melakukan *respond*, protokol HTTP dan versi yang digunakan. Selain itu *respond* ini juga mengandung *body* dari HTML yang diminta.

### 2.1.2 Kode Status

Kode status adalah bilangan bulat tiga digit yang menyatakan status dari pemrosesan permintaan yang dikirimkan. Berikut adalah beberapa kode status yang umum ditemui :

Kode Status	Status	Deskripsi
200	OK	Request berhasil diproses dengan baik.
301	Moved Permanently	Resource yang diminta sudah berpindah ke URI yang lain secara permanen.
302	Found	Resource yang diminta untuk sementara berpindah pada URL yang lain. Untuk alasan historis, client diperkenankan untuk mengubah metode permintaan dan POST menjadi GET.
307	Temporary Redirect	Resource yang diminta untuk sementara berpindah pada URL yang lain. Mirip dengan status 302 namun client tidak diperkenankan mengubah metode permintaan dari POST menjadi GET.
400	Bad Request	Server tidak dapat memproses permintaan karena ada kesalahan dari client
401	Unauthorized	Server tidak dapat memproses permintaan karena kredensial diperlukan dan client tidak menyediakannya.
404	Not Found	Resource yang diminta tidak tersedia pada server.
500	Internal Server Error	Server mengalami masalah internal, sehingga tidak dapat memproses permintaan yang dikirimkan.
501	Not Implemented	Server belum atau tidak mendukung fungsionalitas yang diminta oleh client.
503	Service Unavailable	Server tidak dapat menjawab permintaan client, karena terlalu sibuk atau perawatan. Status ini mengindikasikan client dapat mencoba lagi setelah jangka waktu tertentu.

Tabel 2.1: Tabel Kode Status

1 kode status yang tersedia dikelompokkan menjadi lima, diindikasikan oleh digit pertama  
2 dari kode tersebut:

- 3 • 1xx(informational): Request diterima, dan proses dilanjutkan.
- 4 • 2xx(Successfull): Request diterima, dan dimengertian dengan baik.
- 5 • 3xx(Redirection): Aksi tambahan diperlukan untuk menyelesaikan permintaan.

- 1     • 4xx(Client Error): Terjadi kesalahan dan client harus memperbaikinya
- 2     • 5xx(Server Error): Terjadi kesalahan pada sisi server.

### 3   2.1.3   *Request method*

4   *Request method* menentukan karakteristik dari permintaan yang dikirimkan. Ada 2 *method*  
5 yang sudah dikenal umum yaitu GET dan POST. Selain kedua *method* tersebut, ada bebe-  
6 rapa *method-method* lain yang dapat juga digunakan pada protokol HTTP seperti dijelaskan  
7 pada tabel berikut:

Metode	Deskripsi
GET	Metode yang paling umum digunak- an, dan digunakan untuk menda- patkan konten dari resource yang di- tentukan pada request.
POST	Metode ini digunakan untuk me- minta server memproses data yang dikirimkan. Pada umumnya, me- tode POST diikuti dengan requ- est body, yang berisi parameter- parameter yang dikirimkan
HEAD	Metode HEAD mirip dengan me- tode GET, tetapi bedanya di sini server tidak mengembalikan konten body, melainkan hanya sampai res- ponse headers saja.
PUT	Metode ini digunakan untuk mem- buat atau menggantikan resource yang ditentukan pada request.
DELETE	Metode ini digunakan untuk meng- hapus resource dari server.

Tabel 2.2: Tabel Request Method

### 8   2.1.4   *Response Headers*

9   *Response Headers* digunakan untuk memberikan informasi-informasi tambahan pada sebuah  
10 jawaban. Sama seperti *request header*, setiap header terdiri dari nama dan nilai, dan terpisah  
11 oleh titik dua dan spasi(: ). Tabel berikut menjelaskan beberapa header yang umum  
12 dipakai:

Header	Deskripsi
Content-Type	Header ini menunjukkan tipe media dari konten yang akan diberikan. Pada bentuk sederhana, nilai dari header ini berisi dari kode tipe MIME(Multipurpose Internet Mail Extension). Beberapa kode tipe MIME yang umum antara lain: text/plain untuk teks, text/html untuk halaman HTML; image/gif, image/jpg, image/png untuk gambar berformat GIF, JPEG, PNG; dan application/json untuk data JSON.
Cache-control	Header ini mengatur bagaimana konten yang dikirimkan dapat dikirimkan sementara di client. Pada konten-konten statis seperti gambar, secara default konten akan disimpan pada client dalam jangka waktu tertentu, sehingga jika dibutuhkan dalam waktu dekat di masa depan, tidak perlu mengirimkan permintaan lagi ke server. jika secara eksplisit diinginkan konten diminta lagi setiap kali diperlukan, dapat mengisi header ini dengan nilai no-cache.
Location	Header ini digunakan untuk beberapa jenis jawaban untuk menunjukkan lokasi sumberdaya dalam bentuk URI. Pada jawaban dengan kode 3xx, nilai dari header ini menunjukkan lokasi baru yang harus dituju.

Tabel 2.3: Tabel Response Headers

## 1 2.2 *Library* jsoup

2 Jsoup adalah sebuah *library* java untuk bekerja dengan HTML dunia nyata. Jsoup me-  
 3 nyediakan API yang sangat nyaman untuk mengekstrak dan memanipulasi data, meng-  
 4 gunakan DOM(*Document Object Model*), CSS(*Cascading Style Sheets*), dan *method* yang  
 5 mirip dengan jquery. Jsoup mengimplementasikan spesifikasi standar WHATWG(*Web*  
 6 *Hypertext Application Technology Working Group*) *HTML5* dan mengurai HTML menjadi  
 7 DOM(*Document Object Model*) yang sama dengan peramban modern lakukan. Jsoup sen-  
 8 diri dirancang untuk menangani semua jenis HTML yang biasa ditemukan dengan membuat  
 9 *parsing tree* yang dapat dimengerti.

10 Dalam subbab berikut akan dijelaskan fungsi dan beberapa kelas dari jsoup[2].

### 2.2.1 Fungsi jsoup

berikut adalah fungsi dari jsoup :

- menghimpun dan mengurai HTML dari URL, file, atau *string*.
- mencari dan mengambil data, menggunakan *DOM traversal* atau *CSS selectors*.
- memanipulasi elemen HTML, atribut, dan teks.
- membersihkan konten yang dikirimkan pengguna terhadap daftar putih yang aman, untuk mencegah serangan XSS.
- memberi *output* HTML yang rapi.

### 2.2.2 Kelas- kelas jsoup

#### Jsoup

Kelas ini merupakan inti untuk mengakses fungsi jsoup. Seluruh method dalam kelas ini merupakan *static method* sehingga kelas ini tidak perlu dikonstruksi. Salah satu method yang dimiliki kelas ini adalah sebagai berikut :

- **public static Connection connect(String url)**

Berfungsi untuk membuat koneksi baru dengan suatu situs web.

Parameter:

- **url**: URL situs web dengan protokol HTTP.

**Kembalian**: koneksi dengan situs web.

#### Connection

Kelas ini merupakan interface yang menyediakan pengambilan data dari situs web. Beberapa method yang dimiliki kelas ini adalah sebagai berikut:

- **Connection data(String key, String value)**

Berfungsi untuk menambahkan parameter data yang bisa dikirim melalui metode HTTP GET atau POST.

Parameter:

- **key**: kunci data.
- **value**: nilai data.

**Kembalian**: koneksi yang sama tetapi sudah diubah.

- **Connection ignoreContentType(boolean ignoreContentType)**

Berfungsi untuk Mengabaikan tipe konten dokumen saat *parsing* respon.

Parameter:

- 1       – **ignoreContentType**: set true jika ingin jenis konten diabaikan pada *parsing*
- 2       respon dalam dokumen.

3       **Kembalian**: koneksi pada situs web.

- 4       • **Connection.Response execute() throws IOException**

5       Berfungsi untuk mengeksekusi **request** dari **Connection**.

6       **Kembalian**: objek respon.

- 7       • **String body()**

8       Berfungsi untuk mendapatkan *body* respon sebagai string biasa.

9       **Kembalian**: *string* dari *body*.

## 10   2.3   *JavaScript Object Notation (JSON)*

11   JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah  
12   dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh  
13   komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript,  
14   Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak  
15   bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang  
16   umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript,  
17   Perl, Python dll[3].

### 18   2.3.1   Struktur JSON

19   JSON terbuat dari dua struktur :

- 20       • Kumpulan pasangan nama/nilai.
- 21       • Daftar nilai terurutkan (an ordered list of values).

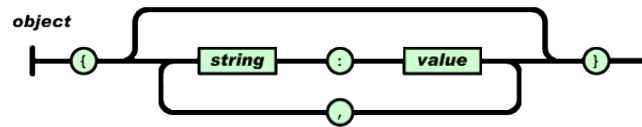
22       Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, se-  
23   mula bahasa pemrograman moderen mendukung struktur data ini dalam bentuk yang sama  
24   maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertu-  
25   karkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data  
26   ini.

### 27   2.3.2   Bentuk-Bentuk JSON

- 28       • Objek

29       Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan  
30       (kurung kurawal buka) dan diakhiri dengan (kurung kurawal tutup). Setiap nama  
31       diikuti dengan : (titik dua) dan setiap pasangan nama atau nilai dipisahkan oleh ,  
32       (koma).

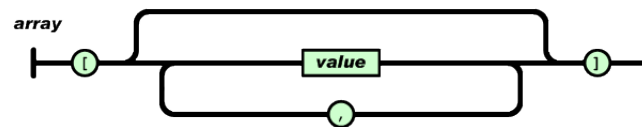




Gambar 2.4: JSON Object

### 1 • Array

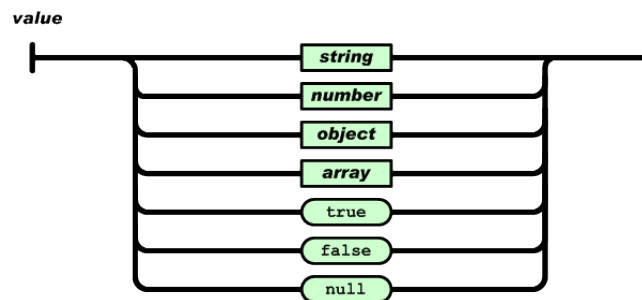
2 *Array* adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [ (kurung kotak  
3 buka) dan diakhiri dengan ] (kurung kotak tutup). Setiap nilai dipisahkan oleh ,  
4 (koma).



Gambar 2.5: JSON Array

### 5 2.3.3 Value JSON

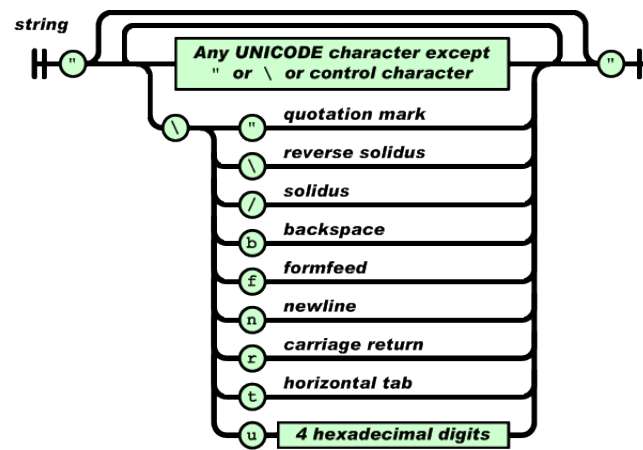
6 Nilai(*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true  
7 atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat  
8 disusun bertingkat.



Gambar 2.6: Value

### 9 • String

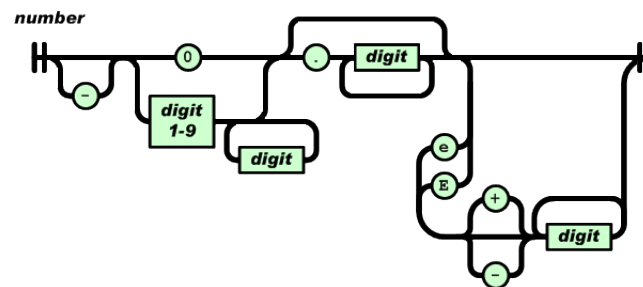
10 String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus de-  
11 ngan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes " untuk  
12 membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string.  
13 String sangat mirip dengan string C atau Java.



Gambar 2.7: String

- Angka

Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.



Gambar 2.8: Angka

#### 2.3.4 kelas-kelas pada *Library* JSON

Subbab-subbab berikut menjelaskan beberapa kelas dari *library* JSON<sup>1</sup>.

##### JSONObject

Kelas ini merepresentasikan sebuah objek JSON yang merupakan koleksi yang tak terurut dari pasangan nama dan nilai. Bentuk eksternal objek JSON adalah sebuah string dibungkus dalam kurung kurawal dengan titik dua antara nama dan nilai-nilai, dan koma antara nilai-nilai dan nama. Nilai-nilai dapat salah satu dari jenis: Boolean, JSONArray, JSONObject, Nomor, String, atau benda JSONObject.NULL. beberapa *method* dan *constructor* yang dimiliki kelas ini adalah sebagai berikut:

- **public JSONObject(String source) throws JSONException**

Berfungsi untuk membangun JSONObject dari sumber JSON string teks.

Parameter:

- **source:** Sebuah string dimulai dengan {(kurung kurawal kiri) dan berakhir dengan} (kurung kurawal kanan).

<sup>1</sup><https://stleary.github.io/JSON-java/>

- 1     • **public String getString(String key) throws JSONException**  
2       Berfungsi untuk mendapatkan objek nilai yang terkait dengan kunci.  
3       Parameter:  
4         – **key**: kunci data.  
5       **Kembalian**: Sebuah string yang merupakan nilai.
- 6     • **public String optString(String key)**  
7       Berfungsi untuk mendapatkan string opsional terkait dengan kunci. Ia mengemba-  
8       likan string kosong jika tidak ada kunci yang ditemukan. Jika nilai tidak string dan  
9       tidak null, maka dikonversi ke string.  
10      Parameter:  
11      – **key**: kunci data.  
12      **Kembalian**: Sebuah string yang merupakan nilai.
- 13    • **public JSONArray getJSONArray(String key) throws JSONException**  
14      Berfungsi untuk mendapatkan nilai JSONArray terkait dengan kunci.  
15      Parameter:  
16      – **key**: kunci data.  
17      **Kembalian**: Sebuah JSONArray yang merupakan nilai.
- 18    • **public JSONObject getJSONObject(String key) throws JSONException**  
19      Berfungsi untuk mendapatkan nilai JSONObject terkait dengan kunci.  
20      Parameter:  
21      – **key**: kunci data.  
22      **Kembalian**: Sebuah JSONObject yang merupakan nilai.

## 23   2.4   Google Direction

24   Google Maps Directions adalah layanan yang menghitung arah antar lokasi menggunakan  
25   permintaan HTTP[4]. Anda bisa mencari arah untuk beberapa moda transportasi, terma-  
26   suk angkutan umum, mengemudi, berjalan atau bersepeda. Arah bisa menetapkan tempat  
27   asal, tujuan dan *waypoint* baik sebagai string teks atau sebagai koordinat garis lintang/ga-  
28   ris bujur. Layanan ini didesain untuk menghitung arah alamat statis (sudah diketahui  
29   sebelumnya) untuk penempatan konten aplikasi pada peta.

### 2.4.1 Permintaan Arah

Permintaan Google Maps Directions mengambil bentuk berikut:

`| https://maps.googleapis.com/maps/api/directions/json?parameters`

Listing 2.1: *Request* Google Directions[4]

HTTP disarankan untuk aplikasi yang berisi data pengguna sensitif, seperti lokasi pengguna, dalam permintaan. URL Google Maps Directions API dibatasi sekitar 2000 karakter, setelah Pengkodean URL. Karena sebagian URL Google Maps Directions API bisa melibatkan banyak lokasi sepanjang lintasan. Pada subbab berikutnya akan dijelaskan parameter apa saja yang digunakan pada permintaan ke layanan ini.

### 2.4.2 Parameter Permintaan

Beberapa parameter tertentu diperlukan sementara yang lainnya bersifat opsional. Sebagaimana standar dalam URL, semua parameter dipisah menggunakan karakter ampersand (&). Daftar parameter dan kemungkinan nilainya disebutkan di bawah ini[4].

#### Parameter yang diperlukan

- **origin** adalah alamat, nilai garis lintang/garis bujur tekstual, atau ID tempat asal yang ingin Anda hitung arahnya. ketentuan dari alamat dari origin adalah sebagai berikut :
  - Jika Anda meneruskan sebuah alamat sebagai string, layanan Directions akan melakukan geocode atas string itu dan mengubahnya menjadi koordinat garis lintang/garis bujur untuk menghitung arah. Koordinat ini mungkin berbeda dengan yang dikembalikan oleh Google Maps Geocoding API, misalnya pintu masuk bangunan dan bukan pusatnya.
  - Jika Anda meneruskan koordinat, itu akan digunakan tanpa diubah untuk menghitung arah. Pastikan tidak ada spasi di antara nilai garis lintang dan garis bujur.
  - ID Tempat harus diawali dengan **place\_id**: ID tempat hanya bisa ditetapkan jika permintaan menyertakan kunci API atau ID klien Google Maps API for Work. Anda bisa mendapatkan ID tempat dari Google Maps Geocoding API dan Google Places API (termasuk Place Autocomplete).
- **destination** adalah alamat, nilai garis lintang/garis bujur tekstual, atau ID tempat tujuan yang ingin Anda hitung arahnya. Opsi untuk parameter destination sama dengan opsi untuk parameter origin yang dijelaskan di atas.
- **key** adalah kunci API aplikasi Anda. Kunci ini mengidentifikasi aplikasi Anda untuk keperluan manajemen kuota.

#### Parameter yang opsional

- **mode** (default-nya adalah driving) adalah menetapkan moda transportasi yang akan digunakan saat menghitung arah.

- 1     • **waypoint** adalah menetapkan larik *waypoint*. *Waypoint* mengubah rute dengan  
2       mengarahkannya melalui lokasi yang ditetapkan. *Waypoint* ditetapkan berupa ko-  
3       ordinat garis lintang/garis bujur, ID tempat, atau alamat yang akan di-geocode. ID  
4       Tempat harus diawali dengan **place\_id**. ID tempat hanya bisa ditetapkan jika per-  
5       mintaan menyertakan kunci API atau ID klien Google Maps API for Work. *Waypoint*  
6       hanya didukung untuk arah mengemudi, berjalan dan bersepeda.
- 7     • **alternative** adalah jika diatur ke true, menetapkan bahwa layanan Directions mung-  
8       kin menyediakan lebih dari satu rute alternatif dalam respons. Perhatikan, membe-  
9       rikan alternatif rute bisa meningkatkan waktu respons dari server.
- 10    • **avoid** adalah menunjukkan rute yang dihitung harus menghindari fitur yang ditandai.  
11      Parameter ini mendukung argumen berikut:
  - 12       – **tolls** menunjukkan rute yang dihitung harus menghindari jalan/jembatan tol.
  - 13       – **highways** menunjukkan rute yang dihitung harus menghindari jalan raya.
  - 14       – **ferries** menunjukkan rute yang dihitung harus menghindari penyeberangan feri.
  - 15       – **indoor** menunjukkan rute yang dihitung harus menghindari tangga dalam ru-  
16       angan untuk arah berjalan dan arah angkutan umum. Hanya permintaan yang  
17       menyertakan kunci API atau ID klien Google Maps API for Work yang akan  
18       menerima tangga dalam ruangan secara default.
- 19    • **language** adalah menetapkan bahasa yang digunakan untuk mengembalikan hasil.
- 20    • **unit** adalah menetapkan sistem satuan yang akan digunakan saat menampilkan hasil.
- 21    • **region** adalah menetapkan kode wilayah, ditetapkan sebagai nilai yang berisi dua  
22      karakter ccTLD ("top-level domain").
- 23    • **arrival\_time** adalah menetapkan waktu kedatangan yang diinginkan untuk arah  
24      angkutan umum, dalam detik sejak tengah malam, 1 Januari 1970 UTC. Anda bisa  
25      menetapkan **departure\_time** atau **arrival\_time**, namun tidak boleh duanya.
- 26    • **departure\_time** adalah menetapkan waktu keberangkatan yang diinginkan. Anda  
27      bisa menetapkan waktu berupa integer dalam detik sejak tengah malam 1 Januari 1970  
28      UTC. Atau, Anda bisa menetapkan nilai now, yang mengatur waktu keberangkatan  
29      ke waktu saat ini (dikoreksi ke detik terdekat).
- 30    • **traffic\_model** (default-nya adalah **best\_guess**) adalah menetapkan asumsi yang  
31      akan digunakan saat menghitung waktu dalam lalu lintas. Pengaturan ini memenga-  
32      ruhi nilai yang dikembalikan di bidang **duration\_in\_traffic** dalam respons, yang  
33      berisi prediksi waktu dalam lalu lintas berdasarkan rata-rata historis. Parameter  
34      **traffic\_model** hanya bisa ditetapkan untuk arah mengemudi yang permintaannya  
35      menyertakan **departure\_time**, dan hanya jika permintaan menyertakan kunci API  
36      atau ID klien Google Maps API for Work. Nilai yang tersedia untuk parameter ini  
37      adalah:

- 1       – **best\_guess** (default) menunjukkan **duration\_in\_traffic** yang dikembalikan  
2       harus berupa perkiraan waktu tempuh terbaik berdasarkan informasi riwayat  
3       kondisi lalu lintas dan lalu lintas saat ini. Lalu lintas saat ini menjadi kian  
4       penting bila **departure\_time** semakin dekat ke waktu sekarang.
- 5       – **pessimistic** menunjukkan **duration\_in\_traffic** yang dikembalikan lebih lama  
6       dari waktu tempuh sesungguhnya di hari-hari biasa, meskipun hari-hari tertentu  
7       dengan kondisi lalu lintas yang buruk mungkin melebihi nilai ini.
- 8       – **optimistic** menunjukkan **duration\_in\_traffic** yang dikembalikan harus le-  
9       bih singkat dari waktu tempuh sesungguhnya di hari biasa, meskipun hari-hari  
10      tertentu dengan kondisi lalu lintas yang baik bisa lebih cepat dari nilai ini.
- 11   • **transit\_mode** adalah menetapkan satu atau beberapa mode angkutan umum yang  
12   disukai. Parameter ini hanya bisa ditetapkan untuk arah angkutan umum, dan hanya  
13   jika permintaan menyertakan kunci API atau ID klien Google Maps API for Work.  
14   Parameter ini mendukung argumen berikut:
  - 15      – **bus** menunjukkan rute yang sudah dihitung akan mengutamakan perjalanan  
16      dengan bus.
  - 17      – **subway** menunjukkan rute yang sudah dihitung akan mengutamakan perjalanan  
18      dengan kereta bawah tanah.
  - 19      – **train** menunjukkan rute yang sudah dihitung akan mengutamakan perjalanan  
20      dengan kereta api.
  - 21      – **tram** menunjukkan rute yang sudah dihitung akan mengutamakan perjalanan  
22      dengan trem dan kereta ringan.
  - 23      – **rail** menunjukkan rute yang sudah dihitung akan mengutamakan perjalanan  
24      dengan kereta api, trem, kereta ringan, dan kereta bawah tanah. Ini sama dengan  
25      **transit\_mode=train|tram|subway**.
- 26   • **transit\_routing\_preference** adalah menetapkan preferensi untuk rute angkutan  
27   umum. Dengan parameter ini, Anda bisa mencondongkan opsi yang dikembalikan,  
28   bukannya menerima rute default terbaik yang dipilih oleh API. Parameter ini hanya  
29   bisa ditetapkan untuk arah angkutan umum, dan hanya jika permintaan menyertakan  
30   kunci API atau ID klien Google Maps API for Work. Parameter ini mendukung  
31   argumen berikut:
  - 32      – **less\_walking** menunjukkan rute yang sudah dihitung akan mengutamakan  
33      jumlah berjalan kaki yang terbatas.
  - 34      – **fewer\_transfers** menunjukkan rute yang sudah dihitung akan mengutamakan  
35      jumlah ganti angkutan yang terbatas.

### 36   2.4.3   *Response Arah*

37   Response Arah dikembalikan dalam format yang ditunjukkan oleh flag output dalam jalur  
38   permintaan URL. Hasil *response* yang dikeluarkan adalah jalur yang dilalui menggunakan  
39   format JSON yang terdapat elemen-elemen yang menjelaskan jalur yang dilewati. Pada

1 subbab berikutnya akan dijelaskan elemen-elemen yang ada pada *output* yang dihasilkan  
2 dari permintaan arah.

#### 3 2.4.4 Elemen *Response* Arah

4 Berikut adalah penjelasan dari setiap elemen *output* yang dihasilkan dari permintaan arah  
5 :

- 6 • **Status** adalah status *response* dari permintaan yang dikirimkan, isinya dapat berupa  
7 salah satu dari berikut ini :
  - 8 – **OK** jika permintaan berhasil, dan permintaan akan mengandung informasi tam-  
9 bahan terkait hasil pencarian.
  - 10 – **NOT\_FOUND** jika salah satu dari **origin** atau **destination** bukan berupa  
11 *latitude*, *longitude* dan tidak dapat ditemukan.
  - 12 – **ZERO\_RESULTS** jika Google tidak berhasil menemukan rute yang diminta.
  - 13 – **INVALID\_REQUEST** jika ada parameter wajib yang tidak diberikan, atau  
14 ada parameter yang tidak valid.
  - 15 – **OVER\_QUERY\_LIMIT** yang berarti jumlah permintaan sudah melebihi  
16 kuota.
  - 17 – **REQUEST\_DENIED** jika permintaan ditolak.
- 18 • **geocoded\_waypoints** adalah hasil *geocoding* dari **origin**, **destination**, maupun  
19 *waypoints* pada permintaan. *Geocoding* pada API ini adalah proses konversi dari  
20 lokasi maupun nama tempat menjadi *place\_id*.
- 21 • **routes** adalah *array* dari objek yang berisi informasi detail setiap alternatif rute yang  
22 ditemukan. elemenn dari **routes** akan dijelaskan pada subsubbab berikutnya.

#### 23 Elemen dari *routes*

24 Setiap elemen dari **routes** adalah objek yang memiliki anggota sebagai berikut :

- 25 • **summary** adalah ringkasan dari alternatif rute ini, untuk membedakan dengan rute  
26 alternatif lainnya.
- 27 • **legs** adalah *array* yang berisi objek yang mempresentasikan *leg*. *Leg* adalah subrute  
28 untuk setiap *waypoints* yang diberikan (jika parameter opsional *waypoints* diberikan).  
29 Jika *waypoints* tidak diberikan, *array* ini akan berisi satu elemen saja. Penjelasan  
30 setiap elemen *legs* akan dijelaskan pada subsubbab berikutnya.
- 31 • **waypoint\_order** adalah *array* yang berisi urutan *waypoint* yang baru, jika parame-  
32 ter *waypoints* diawali dengan *optimized:true*.
- 33 • **overview\_polyline** adalah berisi daftar titik-titik yang dilalui oleh rute yang dida-  
34 patkan. Titik-titik rute ini sudah disederhanakan (tidak detail), dan diringkaskan dengan  
35 format *encoded polyline*.

- **bounds** adalah menyatakan kotak yang menyelubungi rute yang diberikan. Kotak ini direpresentasikan dalam sebuah objek yang mengandung dua anggota yaitu : *northeast*(kanan-atas) dan *southwest*(kiri-bawah). Setiap anggota berupa objek lain yang mengandung dua anggota yaitu : *lat* yang merepresentasikan *latitude* dan *lng* yang merepresentasikan *longitude*.
- **copyrights** adalah berisi teks *copyright* yang harus ditampilkan kepada pengguna.
- **warnings** adalah *array string* yang berisi peringatan yang harus ditampilkan kepada pengguna, jika ada.
- **fare** adalah informasi biaya transportasi publik yang harus dikeluarkan, jika parameter *mode* berisi *transit* dan Google memiliki informasi tarif untuk setiap moda yang digunakan. Informasi ini belum tersedia di Indonesia.

## Elemen dari *legs*

Setiap elemen dari **legs** adalah sebagai berikut :

- **steps** adalah *array* yang berisi objek yang menyatakan setiap langkah yang harus diambil. Penjelasan setiap elemen *steps* dijelaskan pada subsubbab berikutnya.
- **distance** adalah menyatakan jarak yang harus ditempuh pada *leg* ini, berupa objek yang berisi dua anggota yaitu *value* yang merepresentasikan angka yang menyatakan jarak dalam meter dan *text* yang merepresentasikan jarak dalam format teks yang dapat dibaca manusia.
- *duration* adalah menyatakan waktu yang dibutuhkan untuk menempuh *leg* ini, berupa objek yang berisi dua anggota yaitu : *value* yang merepresentasikan angka yang menyatakan waktu dalam detik dan *text* yang merepresentasikan waktu yang dibutuhkan dalam format teks yang dapat dibaca manusia.
- **duration\_in\_traffic** adalah menyatakan waktu mirip dengan *duration*. perbedaannya pada elemen ini memperhitungkan faktor kepadatan lalu lintas.
- **arrival\_time** dan **departure\_time** adalah waktu sampai di *destination* dan waktu keberangkatan ke *destination*, jika parameter *mode* berisi *transit*. berupa objek yang mengandung tiga anggota yaitu : *value* yang merepresentasikan waktu sampai sesuai dengan objek *date* pada *javascript*, *text* yang merepresentasi waktu sampai dalam format teks yang dapat dibaca manusia, dan *time\_zone* yang merepresentasikan zona waktu pada lokasi akhir *leg*.
- **start\_location** dan **end\_location** adalah berisi lokasi awal dan akhir dari *leg* ini, berupa objek yang memiliki dua anggota yaitu : *lat* yang merepresentasikan *latitude* dan *lng* yang merepresentasikan *longitude*.
- **start\_address** dan **end\_address** adalah berisi lokasi awal dan akhir dari *leg* ini, dalam format teks yang dapat dibaca manusia.



### 1 Elemen dari *steps*

2 Setiap elemen dari **steps** adalah sebagai berikut :

- 3 • **html\_instructions** adalah berisi instruksi *step* ini, dalam format HTML.
- 4 • **distance** adalah jarak dari *step* ini, dengan format yang sama seperti anggota *duration*  
5 pada elemen *legs* di atas.
- 6 • **start\_location** dan **end\_location** adalah lokasi awal dan akhir dari *step* ini, dengan  
7 format yang sama seperti anggota **start\_location** dan **end\_location** pada elemen  
8 *legs* di atas.
- 9 • **polyline** adalah berisi daftar titik-titik yang dilalui pada *step* ini. titik- titik rute ini  
10 diringkas dengan format *encoded polyline*.
- 11 • **steps** adalah *array* yang berisi *sub-step* dari *step* ini, jika parameter *mode* berisi  
12 *transit*. Formatnya sama dengan elemen *step* ini.
- 13 • **transit\_details** adalah berisi detail transit, jika parameter *mode* berisi *transit*. Pen-  
14 jelasan objek **transit\_details** akan dijelaskan pada subsubbab berikutnya.

### 15 Elemen dari *transit\_details*

16 Setiap elemen dari **transit\_details** adalah sebagai berikut :

- 17 • **name** adalah berisi nama jalur ini.
- 18 • **short\_name** adalah berisi nama jalur yang lebih singkat, biasanya kode jalur.
- 19 • **color** adalah berisi warna yang umum digunakan untuk merepresentasikan jalur ini,  
20 dalam format string heksadesimal.
- 21 • **agencies** adalah *array* yang tiap elemennya berupa objek yang merepresentasikan  
22 penyedia layanan, dan mengandung tiga anggota yaitu : *name* yang merepresentasikan  
23 nama penyedia layanan, *url* yang merepresentasikan alamat situs web, dan *phone* yang  
24 mere[representasikan nomor telepon. Informasi ini wajib ditampilkan ke pengguna.
- 25 • **url** adalah alamat situs web dari jalur ini.
- 26 • **icon** adalah URL untuk mendapatkan gambar yang merepresentasikan jalur ini.
- 27 • **text\_color** adalah berisi warna yang umum digunakan untuk teks yang merepresen-  
28 tasikan jalur ini dalam format string heksadesimal.
- 29 • **vehicle** adalah berisi informasi kendaraan yang digunakan pada jalur ini dalam  
30 bentuk objek yang mengandung empat anggota yaitu : *name* yang merepresentasikan  
31 nama kendaraan, *type* yang merepresentasikan tipe kendaraan, *icon* yang merepresen-  
32 tasikan URL gambar kendaraan, *local\_icon* yang merepresenasikan gambar kendaraan  
33 secara lokal.



## BAB 3

### ANALISIS

Berdasarkan hasil studi pustaka yang telah dilakukan, pada bab ini akan dijelaskan hasil analisis berupa uraian dari perangkat lunak yang akan dibangun, analisis google direction API, diagram use-case beserta dengan skenario dan analisis diagram kelas.

#### 3.1 Flow Chart Alur Layanan Google Direction

Dalam mengakses layanan Google Direction sesuai dengan subbab 2.4 yang berjalan pada protokol HTTP, terjadi transaksi data yang bergerak antara *user* dan *server* Google. Dengan menggunakan diagram *flow chart* akan memudahkan dalam pembangunan perangkat lunak dan mengetahui alur transaksi dari layanan Google Direction. Diagram *flow chart* yang menunjukkan alur transaksi layanan Google Direction dapat dilihat pada Gambar 3.1



Gambar 3.1: Flow Chart Alur Layanan Google Direction

## 3.2 Analisis permintaan ke layanan Google Direction

Sesuai dengan subbab 2.4.1 permintaan dari google direction ini menggunakan protokol HTTP. Permintaan tersebut menghubungi hostname `www.google.com` dengan port default untuk port HTTP yaitu 80. Permintaan tersebut disertai dengan parameter-parameter opsional lainnya untuk mendapatkan data yang diinginkan.

### 3.2.1 Parameter yang digunakan

Untuk mendapatkan data waktu tempuh yang beragam untuk menganalisis waktu tempuh dari 2 titik sesuai dengan 2.4.2, parameter opsional yang digunakan adalah : **departure\_time** dan **traffic\_model**. Dari memanipulasi kedua parameter tersebut akan menghasilkan data waktu tempuh yang beragam. Selain itu memanipulasi nilai parameter pada **destination** dan **origin** juga akan mempengaruhi data waktu tempuh yang dihasilkan karena pada perhitungan dari masing-masing **destination** ke **origin** akan menghasilkan waktu tempuh yang berbeda. Dari masing-masing **destination** ke **origin** juga memiliki jam kepadatan tertentu dimana nilai waktu tempuh akan berbeda dengan jam-jam lainnya sesuai dengan **departure\_time**. Parameter **traffic\_model** ini juga mempengaruhi nilai yang waktu tempuh dikeluarkan tergantung model apakah yang digunakan yang telah dibahas pada subbab 2.4.2.

```
18 | https://maps.googleapis.com/maps/api/directions/json?...traffic_model=best_guess
```

Listing 3.1: Traffic\_model : best\_guess

```
19 | https://maps.googleapis.com/maps/api/directions/json?...traffic_model=optimistic
```

Listing 3.2: Traffic\_model : optimistic

```
20 | https://maps.googleapis.com/maps/api/directions/json?...traffic_model=pessimistic
```

Listing 3.3: Traffic\_model : pessimistic

## 3.3 Analisis response dari layanan Google Directions

Pada saat melakukan permintaan, Server akan memberikan *response* dengan format JSON. Response yang diterima adalah hasil perhitungan dari *origin* ke *destination*. Dari response ini terdapat banyak data didalamnya.

Data waktu tempuh pada hasil response permintaan ada pada *duration\_in\_traffic* dimana *duration\_in\_traffic* ini adalah salah satu elemen dari *legs* (subsubbab 2.4.4) yang merupakan sebuah *json array* dan *legs* ini sendiri adalah salah satu elemen dari *routes* yang merupakan elemen dari *response* yang diterima.

```
29 | {
30 |   "geocoded_waypoints" : [
31 |     ...
32 |     "routes" : [
33 |       ...
34 |       "legs" : [
35 |         ...
36 |         "duration_in_traffic" : {
37 |           "text" : "57 menit",
38 |           "value" : 3439
39 |         },
40 |         ...
41 |       "status" : "OK"
```

1 | }

Listing 3.4: Hasil *response* Google Directions

## 2 3.4 Gambaran Umum Perangkat Lunak

3 Perangkat lunak yang akan dibangun adalah perangkat lunak untuk menghitung waktu  
 4 tempuh dari 2 titik yang ditentukan. Perangkat lunak yang akan dibangun ini bertujuan  
 5 untuk membantu menganalisis pada jam berapakah waktu tempuh paling cepat dalam  
 6 waktu 1 minggu terhitung dari hari senin. Selain itu, perangkat lunak ini bertujuan un-  
 7 tuk membantu pengambilan keputusan pengguna untuk menentukan pada jam berapakah  
 8 pengguna melakukan perjalanan agar tidak terjebak dalam kemacetan. Perangkat lunak  
 9 ini berjalan pada protokol HTTP. Perangkat lunak ini dibangun pada perangkat kom-  
 10 puter(desktop) yang berfungsi sebagai penghitung waktu tempuh dengan memanfaatkan  
 11 Google Direction API. Perangkat lunak mengeluarkan *output* berupa file yang berekstensi  
 12 .csv untuk mencatat seluruh data yang diterima oleh perangkat lunak dari layanan Google  
 13 Direction dan menggunakan aplikasi *Microsoft Excel* untuk membantu menganalisis data  
 14 dengan cara me-*generate* bagan secara manual oleh pengguna. Perangkat lunak ini akan  
 15 diuji coba sesuai dengan sampel sebagai berikut : menghitung waktu tempuh antar loka-  
 16 si yang beralamat Jln. Ciumbuleuit No.94 dan Komplek Amaya Residence; menghitung  
 17 waktu tempuh antar lokasi yang beralamat Jln. Ciumbuleuit No.94 dan Komplek Taman  
 18 Puspa Indah. Penetapan sampel untuk memudahkan mendapatkan waktu tempuh dengan  
 19 alamat yang konstant dan memudahkan untuk output yang dikeluarkan.

## 20 3.5 Analisis Perangkat Lunak

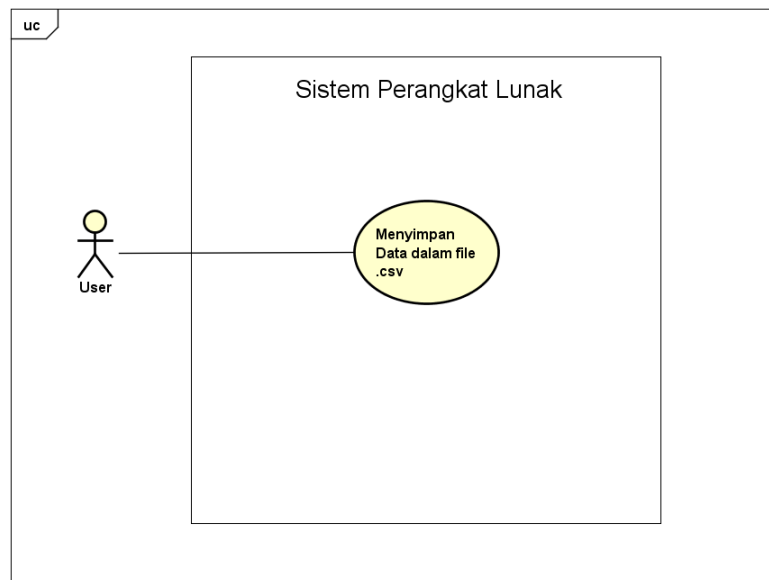
21 Perangkat lunak yang akan dibangun adalah perangkat lunak yang dapat melakukan peng-  
 22 hitungan waktu tempuh tercepat berdasarkan *request-request* yang dikirimkan oleh user  
 23 dalam jangka waktu 1 minggu terhitung dari senin. perangkat lunak dibangun dengan  
 24 menggunakan bahasa pemrograman Java dan membutuhkan *library* jsoup yang akan digu-  
 25 nakan untuk membantu perancangan dan pengimplementasian perangkat lunak yang akan  
 26 dibangun oleh penulis. Berikut adalah fitur-fitur yang akan dibangun pada perangkat lunak:

- 27 1. Mengekstaksi data waktu tempuh dari keluaran *response* Google Direction dan me-  
 28 nampilkan pukul berapa yang memiliki waktu tempuh terbaik dalam kurun waktu 1  
 29 minggu.
- 30 2. Menyimpan data-data waktu tempuh dari keluaran *response* Google Direction pada  
 31 file berekstensi .csv.

## 32 3.6 Analisis *Use Case*

### 33 3.6.1 Diagram *Use Case*

34 Diagram use case pada perangkat lunak yang akan dibangun hanya mengandung satu aktor,  
 35 yaitu User. Diagram use case dapat dilihat pada Gambar 3.2.



Gambar 3.2: Diagram *Use Case* Perangkat Lunak

1 Berdasarkan subbab 3.5. dari dua fitur yang akan dibuat, dibentuk satu *use case* antara  
 2 lain:

- 3 • **Menyimpan Data dalam file .csv**, User dapat menyimpan data dari penghitungan  
 4 antar 2 titik : sampel dan Unpar.

### 5 3.6.2 Skenario *Use Case*

#### 6 1. Menghitung Waktu Tempuh

- 7 • Nama : Menyimpan Data dalam file .csv.
- 8 • Aktor : User.
- 9 • Deskripsi : Menyimpan data dari hasil penghitungan dari tempat asal tempat  
 10 asal ke tempat tujuan.
- 11 • Kondisi awal : User memulai program.
- 12 • Kondisi akhir : User berhasil menyimpan data dan membuka file tersebut dengan  
 13 bantuan excel.
- 14 • Skenario Utama :

15

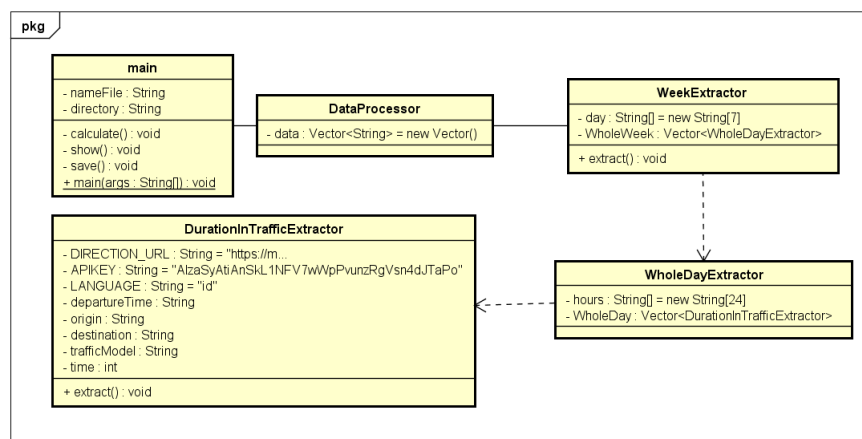
No	Aksi Aktor	Reaksi Sistem
1	User memulai program	Sistem menampilkan GUI dari perangkat lunak
2	User melakukan input	
3	User menekan tombol save	Sistem mengolah data sesuai dengan input dari user
4		Sistem menampilkan layar untuk penyimpanan file
5	User melakukan input untuk penyimpanan file	Sistem melakukan penyimpanan file sesuai input dari user
6		Sistem membuka file tersebut dengan menggunakan aplikasi Microsoft Excel

• Eksepsi :

- tidak ada model traffic yang dipilih.
- user belum memilih sampel mana yang akan dihitung.
- user belum memilih tanggal.
- user memilih tanggal pada masa lampau dan hari ini.

### 3.7 Analisis Kelas

Diagram kelas analisis untuk perangkat lunak ditunjukkan pada Gambar 3.3



Gambar 3.3: Diagram Kelas untuk Perangkat Lunak

Penjelasan dari kelas-kelas lainnya sebagai berikut:

1. **DurationInTrafficExtractor** adalah kelas yang bertugas untuk mengirimkan permintaan ke layanan Google Direction dan men ekstrak data waktu tempuh.
2. **WholeDayExtractor** adalah kelas yang bertugas men ekstrak waktu tempuh dalam 1 hari.

- 1     3. **WholeWeekExtractor** adalah kelas yang bertugas untuk mengekstraksi waktu tem-  
2        puh dalam 1 minggu.
- 3     4. **DataProsesor** adalah kelas yang bertugas sebagai tempat penyimpanan data dan  
4        bertugas untuk proses penyimpanan data ke dalam file.
- 5     5. **main** adalah kelas yang bertugas sebagai tampilan utama pada perangkat lunak ini.



## BAB 4

### PERANCANGAN

Berdasarkan analisis yang telah dilakukan, terdapat beberapa hal yang perlu dirancang untuk pembangunan perangkat lunak sederhana analisis waktu tempuh kota Bandung ini. Pada bab ini akan dijelaskan mengenai perancangan aplikasi yang akan dibangun meliputi perancangan file keluaran, perancangan antarmuka, kelas diagram kelas rinci beserta deskripsi dan fungsinya.

#### 4.1 Kebutuhan Masukan dan Keluaran

Perangkat lunak yang dibangun merupakan perangkat lunak untuk melakukan ekstraksi data dari layanan Google Direction. Pada perancangan perangkat lunak ini menggunakan *library* jsoup untuk melakukan *request* ke layanan Google Direction. Selain itu perangkat lunak ini akan menggunakan *library* JSON untuk mengekstraksi *output* JSON yang merupakan *response* dari *request* yang diminta. Masukan dan keluaran perangkat lunak adalah sebagai berikut :

##### 4.1.1 Masukan

Masukan dari perangkat lunak sederhana ini adalah parameter-parameter yang digunakan untuk melakukan request. Parameter-parameter tersebut adalah : *origin*, *departure\_time* dalam bentuk unix, dan *traffic\_model*. Nilai parameter *origin* adalah nilai *longitude* dan *latitude* yang disatukan menjadi sebuah string. *Longitude* dan *latitude* itu sendiri adalah *longitude* dan *latitude* dari masing-masing sample. Nilai parameter *departure\_time* adalah nilai unix yang di-*parsing* menjadi bentuk string dari sebuah tanggal. Nilai parameter *traffic\_model* adalah sebuah string salah satu dari *best\_guess*, *optimistic*, *pessimistic*.

##### 4.1.2 Keluaran

Keluaran dari perangkat lunak sederhana ini adalah sebuah file dengan ekstensi *Comma Separated Value* (.csv) dimana dalam file ini berisi data hasil ekstraksi dari *request* ke layanan Google Direction.

#### 4.2 Parameter *request* ke layanan Google Direction

Sesuai dengan subbab 3.2, *request* berbentuk sebuah url dengan memasukkan parameter-parameter yang dibutuhkan ke dalam url untuk melakukan suatu *request*. Nilai dari parameter *origin* yang akan digunakan pada perangkat lunak ini dibagi menjadi 2 yaitu : sampel 1

1 dengan nilai "-6.9536001,107.6193958" dan sampel 2 dengan nilai "-6.937021,107.6643817".  
 2 Nilai dari parameter *destination* yang akan digunakan pada perangkat lunak ini adalah  
 3 "-6.8746025,107.6024968" yang merupakan nilai *longitude* dan *latitude* dari Universitas  
 4 Katolik Parahyangan.

### 5 4.3 Rancangan file keluaran

6 Sesuai dengan subsubbab 4.1.2, data pada file ini memiliki format tersusun dari nilai-nilai  
 7 yang dipisahkan oleh koma(.). Menggunakan aplikasi Microsoft Excel, setiap nilai-nilai  
 8 yang dipisahkan oleh koma(.) ini di direpresentasikan dengan masing-masing kolom. Sesuai  
 9 masukan dari pengguna, file keluaran ini sendiri terbagi menjadi tiga tergantung dari *tra-*  
 10 *ffic\_model* yang dipilih. Berikut ini merupakan tiga macam keluaran yang akan dihasilkan  
 11 oleh perangkat lunak sederhana ini :

- 12 • file ekstensi .csv dengan tiga nilai : file ini akan dihasilkan jika pengguna memilih  
 13 satu *traffic\_model* dimana nilai pertama adalah nilai yang merepresentasikan hari,  
 14 nilai kedua adalah nilai merepresentasikan jam dan nilai yang terakhir adalah nilai  
 15 waktu tempuh sesuai dengan yang dipilih oleh pengguna.
- 16 • file ekstensi .csv dengan empat nilai : file ini akan dihasilkan jika pengguna me-  
 17 milih dua *traffic\_model* dimana nilai pertama adalah nilai yang merepresentasikan  
 18 hari, nilai kedua adalah nilai merepresentasikan jam, nilai yang ketiga dan keempat  
 19 adalah nilai waktu tempuh sesuai dengan yang dipilih oleh pengguna. Kombinasi  
 20 pilihan pengguna yang memungkinkan adalah *best\_guess* dan *optimistic*, *best\_guess*  
 21 dan *pessimistic*, *optimistic* dan *pessimistic*.
- 22 • file ekstensi .csv dengan lima nilai : file ini akan dihasilkan jika pengguna memilih  
 23 satu *traffic\_model* dimana nilai pertama adalah nilai yang merepresentasikan hari,  
 24 nilai kedua adalah nilai merepresentasikan jam, nilai ketiga, keempat dan kelima ada-  
 25 lah nilai waktu tempuh sesuai dengan yang dipilih oleh pengguna dengan kombinasi  
 26 pilihan pengguna adalah *best\_guess*, *optimisic* dan *pessimistic*.

### 27 4.4 Diagram Kelas Rinci

28 Diagram kelas rinci diperoleh dari hasil pengembangan diagram kelas analisis pada subbab  
 29 3.7. Diagram kelas rinci dapat dilihat pada Gambar 4.1. Deskripsi kelas beserta fungsi dari  
 30 diagram kelas rinci tersebut adalah sebagai berikut:

#### 31 1. DurationIntrafficExtractor

32 Kelas ini merupakan kelas yang bertugas untuk melakukan *request* ke Google Direction  
 33 API dan mengekstraksi nilai waktu dalam satuan detik dari *response* balasan dari  
 34 *request* yang diminta. Kelas ini melakukan satu *request* dalam satu waktu. Kelas ini  
 35 juga melakukan penghitungan konversi dari nilai waktu yang bersatuan detik menjadi  
 36 menit. Atribut yang dimiliki oleh kelas ini antara lain :

- 37 • **private String DIRECTION\_URL** : merupakan url dasar untuk melakukan  
 38 *request* ke Google Direction API.

- 1       • **private String APIKEY** : merupakan sebuah kunci API yang akan dimasukkan
- 2       kedalam url dasar sebagai salah satu parameter untuk dapat melacak pengguna-
- 3       annya.
- 4       • **private String LANGUAGE** : merupakan salah satu parameter untuk url
- 5       dasar yang berfungsi untuk memberikan *response* dari Google Direction dalam
- 6       suatu bahasa.
- 7       • **private String departureTime** : merupakan salah satu parameter untuk url
- 8       dasar untuk menentukan waktu keberangkatan dari suatu titik ke tempat tujuan.
- 9       • **private String origin** : merupakan parameter wajib untuk url dasar yang
- 10       merepresentasikan suatu titik berupa *longitude* dan *latitude* dari suatu tempat
- 11       yang dijadikan acuan dasar tempat keberangkatan ke tempat tujuan.
- 12       • **private String destination** : merupakan parameter wajib untuk url dasar yang
- 13       merepresentasikan suatu titik berupa *longitude* dan *latitude* dari suatu tempat
- 14       yang dijadikan acuan dasar tempat tujuan.
- 15       • **private int time** : merupakan sebuah atribut yang berisikan waktu dalam hi-
- 16       tungan menit yang didapatkan dari hasil ekstraksi *response duration\_in\_traffic*
- 17       dari suatu *request*.

18       Method-method yang dimiliki kelas ini merupakan action method dengan rincian se-

19       bagai berikut:

- 20       • **public DurationInTrafficExtractor(String unix, String origin, String**
- 21       **destination, String trafficModel)**

22       merupakan konstruktor dari kelas ini. Fungsinya untuk menginisialisasi dari

23       masing-masing atribut yang dimiliki kelas ini.

24       Parameter:

- 25       – **unix**: nilai waktu berbentuk string yang telah dikonversi kedalam bentuk
- 26       unix yang merepresentasikan waktu keberangkatan dari tempat asal ke tem-
- 27       pat tujuan.
- 28       – **origin**: nilai *longitude* dan *latitude* yang disatukan menjadi string yang
- 29       merepresentasikan tempat asal keberangkatan.
- 30       – **destination**: nilai *longitude* dan *latitude* yang disatukan menjadi string
- 31       yang merepresentasikan tempat tujuan dari suatu keberangkatan.
- 32       – **trafficModel**: nilai string yang merepresentasikan model traffic yang akan
- 33       digunakan.

- 34       • **public void setTime(int time)**

35       Berfungsi untuk menetapkan nilai dari atribut *time*.

36       Parameter:

- 37       – **time**: nilai waktu yang akan ditetapkan.

- 38       • **public int getTime()**

39       Berfungsi untuk mendapatkan nilai yang dari atribut *time*.

40       **Kembalian**: Sebuah integer yang merupakan nilai dari atribut *time*.

- **public void extract()**

Berfungsi untuk menetapkan seluruh parameter pada url dasar dan melakukan *request* ke layanan Google dan mendapatkan *response*-nya. Setelah mendapatkan *response*-nya method ini melakukan ekstraksi untuk mendapatkan waktu tempuh pada suatu waktu.

- **public String getDepartureTimeHours()**

Berfungsi untuk mendapatkan nilai jam dalam bentuk string dari atribut *departureTime*.

**Kembalian:** Sebuah string yang merupakan nilai jam dari atribut *departureTime*.

## 2. WholeDayExtractor

Kelas ini merupakan kelas yang bertugas untuk mendapatkan nilai waktu tempuh selama satu hari penuh. Kelas ini terdiri dari *DurationInTrafficExtractor* dimana setiap *DurationInTrafficExtractor* merepresentasikan *request* ke Google Direction API dalam setiap jam. Atribut yang dimiliki oleh kelas ini antara lain :

- **private String[] hours** : merupakan sebuah atribut array yang memiliki ukuran duapuluh empat dimana setiap string dalam atribut ini merepresentasikan setiap jam.
- **private DurationInTrafficExtractor[] wholeDay** : merupakan sebuah atribut array yang memiliki ukuran duapuluh empat dimana setiap *DurationInTraffic* dalam atribut ini merepresentasikan waktu tempuh dalam setiap jamnya yang memiliki nilai yang berbeda.

Method-method yang dimiliki kelas ini merupakan action method dengan rincian sebagai berikut:

- **public void initialize(String unix, String origin, String destination, String trafficModel) throws ParseException**

Berfungsi untuk menetapkan seluruh parameter pada url dasar pada setiap *DurationInTrafficExtractor* dalam array *wholeDay* dengan menentukan parameter-parameter ke setiap *DurationInTrafficExtractor*.

Parameter:

- **unix**: nilai waktu berbentuk string yang telah dikonversi kedalam bentuk unix yang merepresentasikan waktu keberangkatan dari tempat asal ke tempat tujuan.
- **origin**: nilai *longitude* dan *latitude* yang disatukan menjadi string yang merepresentasikan tempat asal keberangkatan.
- **destination**: nilai *longitude* dan *latitude* yang disatukan menjadi string yang merepresentasikan tempat tujuan dari suatu keberangkatan.
- **trafficModel**: nilai string yang merepresentasikan model traffic yang akan digunakan.

1       • **public void extract() throws IOException**

2       Berfungsi untuk melakukan *request* ke layanan Google dan mendapatkan *respon-*  
 3       *se*-nya pada setiap *DurationInTrafficExtractor*. Setelah mendapatkan *response*-  
 4       nya method ini melakukan ekstraksi untuk mendapatkan waktu tempuh pada  
 5       setiap *DurationInTrafficExtractor*.

6       • **public DurationInTrafficExtractor[] getWholeDay()**

7       Berfungsi untuk mendapatkan setiap *DurationInTrafficExtractor* yang ada dida-  
 8       lam array *wholeDay*.

9       **Kembalian:** Sebuah array *DurationInTrafficExtractor*.

10      • **public String[] getHours()**

11      Berfungsi untuk mendapatkan nilai string setiap jam.

12      **Kembalian:** Sebuah array string.

13    3. WeekExtractor

14      Kelas ini merupakan kelas yang bertugas untuk mendapatkan nilai waktu tempuh  
 15      selama tujuh hari. Kelas ini terdiri dari *WholeDayExtractor* dimana setiap *Whole-*  
 16      *DayExtractor* merepresentasikan *request* ke Google Direction API dalam setiap hari.  
 17      Atribut yang dimiliki oleh kelas ini antara lain :

18      • **private String[] day** : merupakan sebuah atribut array yang memiliki ukuran  
 19      tujuh dimana setiap string dalam atribut ini merepresentasikan setiap harinya.

20      • **private WholeDayExtractor[] oneWeek** : merupakan sebuah atribut array  
 21      yang memiliki ukuran tujuh dimana setiap *DurationInTraffic* dalam atribut ini  
 22      merepresentasikan waktu tempuh dalam setiap harinya yang memiliki nilai yang  
 23      berbeda.

24      Method-method yang dimiliki kelas ini merupakan action method dengan rincian se-  
 25      bagai berikut:

26      • **public void initialize(String date, String origin, String destination,**  
 27      **String trafficModel) throws ParseException**

28      Berfungsi untuk menetapkan seluruh parameter pada url dasar pada setiap *Who-*  
 29      *leDayExtractor* dalam array *oneWeek* dengan menentukan parameter-parameter  
 30      ke setiap *WholeDayExtractor*.

31      Parameter:

32      – **unix**: nilai waktu berbentuk string yang telah dikonversi kedalam bentuk  
 33      unix yang merepresentasikan waktu keberangkatan dari tempat asal ke tem-  
 34      pat tujuan.

35      – **origin**: nilai *longitude* dan *latitude* yang disatukan mejadi string yang me-  
 36      representasikan tempat asal keberangkatan.

37      – **destination**: nilai *longitude* dan *latitude* yang disatukan menjadi string  
 38      yang merepresentasikan tempat tujuan dari suatu keberangkatan.

39      – **trafficModel**: nilai string yang merepresentasikan model traffic yang akan  
 40      digunakan.

- **public void extract() throws IOException**

Berfungsi untuk melakukan *request* ke layanan Google dan mendapatkan *response*-nya pada setiap *WholeDayExtractor*. Setelah mendapatkan *response*-nya method ini melakukan ekstraksi untuk mendapatkan waktu tempuh pada setiap *WholeDayExtractor*.

- **public WholeDayExtractor[] getOneWeek()**

Berfungsi untuk mendapatkan setiap *WholeDayExtractor* yang ada didalam array *oneWeek*.

**Kembalian:** Sebuah array *WholeDayExtractor*.

- **public String[] getDay()**

Berfungsi untuk mendapatkan nilai string setiap harinya.

**Kembalian:** Sebuah array string.

#### 4. DataProcessor

Kelas ini merupakan kelas yang bertugas untuk memproses semua data yang didapatkan. Selain itu Kelas ini juga bertugas untuk melakukan penulisan data-data kedalam file dengan ekstensi *.csv*. Atribut yang dimiliki oleh kelas ini antara lain :

- **private String csvSplitBy :** merupakan atribut untuk memisahkan antar data yang didapatkan untuk dituliskan kedalam file.
- **private Vector<String> data :** merupakan atribut *Vector* dimana data yang didapatkan dari hasil ekstraksi disimpan.
- **private Vector<String> trafficModel :** merupakan atribut *Vector* yang merepresentasikan model traffic yang digunakan oleh data yang disimpan dalam *Vector data*.

Method-method yang dimiliki kelas ini merupakan action method dengan rincian sebagai berikut:

- **public void initalize(JFormattedTextField date, String origin, String destination, JCheckBox trafficModel) throws ParseException, IOException**

Berfungsi untuk menetapkan parameter-parameter untuk melakukan *request* ke layanan Google dimana model traffic yang digunakan adalah **satu** model traffic dan mendapatkan hasil ekstraksinya. Hasil-hasil ekstraksi tersebut dimasukan kedalam *Vector data*.

Parameter:

- **date:** merupakan sebuah *JFormattedTextField* yang memiliki nilai tanggal yang merepresentasikan tanggal berapa yang akan dihitung.
- **origin:** nilai *longitude* dan *latitude* yang disatukan menjadi string yang merepresentasikan tempat asal keberangkatan.
- **destination:** nilai *longitude* dan *latitude* yang disatukan menjadi string yang merepresentasikan tempat tujuan dari suatu keberangkatan.

- 1           – **trafficModel1**: merupakan sebuah *JCheckBox* yang dipilih untuk dihitung
- 2           waktu tempuhnya.
- 3           – **trafficModel2**: merupakan sebuah *JCheckBox* yang dipilih untuk dihitung
- 4           waktu tempuhnya.

5       • **public void initalize(JFormattedTextField date, String origin, String**  
 6       **destination, JCheckBox trafficModel1, JCheckBox trafficModel2) thro-**  
 7       **ws ParseException, IOException**

8       Berfungsi untuk menetapkan parameter-parameter untuk melakukan *request* ke  
 9       layanan Google dimana model traffic yang digunakan adalah **dua** model traffic  
 10       dan mendapatkan hasil ekstraksinya. Hasil-hasil ekstraksi tersebut dimasukan  
 11       kedalam *Vector data*.

12       Parameter:

- 13           – **date**: merupakan sebuah *JFormattedTextField* yang memiliki nilai tanggal
- 14           yang merepresentasikan tanggal berapa yang akan dihitung.
- 15           – **origin**: nilai *longitude* dan *latitude* yang disatukan menjadi string yang
- 16           merepresentasikan tempat asal keberangkatan.
- 17           – **destination**: nilai *longitude* dan *latitude* yang disatukan menjadi string
- 18           yang merepresentasikan tempat tujuan dari suatu keberangkatan.
- 19           – **trafficModel1**: merupakan sebuah *JCheckBox* yang dipilih untuk dihitung
- 20           waktu tempuhnya.
- 21           – **trafficModel2**: merupakan sebuah *JCheckBox* yang dipilih untuk dihitung
- 22           waktu tempuhnya.

23       • **public void initalize(JFormattedTextField date, String origin, String**  
 24       **destination, JCheckBox trafficModel1, JCheckBox trafficModel2, JCheck-**  
 25       **Box trafficModel3) throws ParseException, IOException**

26       Berfungsi untuk menetapkan parameter-parameter untuk melakukan *request* ke  
 27       layanan Google dimana model traffic yang digunakan adalah **tiga** model traffic  
 28       dan mendapatkan hasil ekstraksinya. Hasil-hasil ekstraksi tersebut dimasukan  
 29       kedalam *Vector data*.

30       Parameter:

- 31           – **date**: merupakan sebuah *JFormattedTextField* yang memiliki nilai tanggal
- 32           yang merepresentasikan tanggal berapa yang akan dihitung.
- 33           – **origin**: nilai *longitude* dan *latitude* yang disatukan menjadi string yang
- 34           merepresentasikan tempat asal keberangkatan.
- 35           – **destination**: nilai *longitude* dan *latitude* yang disatukan menjadi string
- 36           yang merepresentasikan tempat tujuan dari suatu keberangkatan.
- 37           – **trafficModel1**: merupakan sebuah *JCheckBox* yang dipilih untuk dihitung
- 38           waktu tempuhnya.
- 39           – **trafficModel2**: merupakan sebuah *JCheckBox* yang dipilih untuk dihitung
- 40           waktu tempuhnya.
- 41           – **trafficModel3**: merupakan sebuah *JCheckBox* yang dipilih untuk dihitung
- 42           waktu tempuhnya.

- **public void saveFile(String directory, String fileName) throws IOException**

Berfungsi untuk menuliskan data yang ada dalam *Vector data* kedalam sebuah file berekstensi *.csv*.

Parameter:

- **directory**: merupakan sebuah string yang merepresentasikan *directory* penyimpanan file.
- **fileName**: merupakan sebuah string yang merepresentasikan nama file yang akan disimpan.

## 5. DurationTimeController

Kelas ini merupakan kelas yang bertugas sebagai jembatan penghubung antara *graphical user interface* dengan kelas *DataProcessor*. Atribut yang dimiliki oleh kelas ini antara lain :

- **private DataProcessor processor** : merupakan atribut *DataProcessor* dimana kelas ini bertugas memproses data yang diinput melalui *graphical user interface*.

Method-method yang dimiliki kelas ini merupakan action method dengan rincian sebagai berikut:

- **public void doCalculate(JFormattedTextField date, String origin, String destination, JCheckBox trafficModel1, JCheckBox trafficModel2, JCheckBox trafficModel3) throws ParseException, IOException**

Berfungsi untuk meneruskan parameter-parameter yang diinput melalui *graphical user interface* dan memerintah *processor* untuk melakukan pemrosesan data.

Parameter:

- **date**: merupakan sebuah *JFormattedTextField* yang diinput melalui *graphical user interface*. *JFormattedTextField* tersebut memiliki nilai tanggal yang merepresentasikan tanggal berapa yang akan dihitung.
- **origin**: nilai *longitude* dan *latitude* yang disatukan menjadi string yang merepresentasikan tempat asal keberangkatan.
- **destination**: nilai *longitude* dan *latitude* yang disatukan menjadi string yang merepresentasikan tempat tujuan dari suatu keberangkatan.
- **trafficModel1**: merupakan sebuah *JCheckBox* yang diinput melalui *graphical user interface*. *JCheckBox* tersebut merupakan input yang dipilih untuk dihitung waktu tempuhnya.
- **trafficModel2**: merupakan sebuah *JCheckBox* yang diinput melalui *graphical user interface*. *JCheckBox* tersebut merupakan input yang dipilih untuk dihitung waktu tempuhnya.
- **trafficModel3**: merupakan sebuah *JCheckBox* yang diinput melalui *graphical user interface*. *JCheckBox* tersebut merupakan input yang dipilih untuk dihitung waktu tempuhnya.



- **public void saveData(String dir, String filename) throws IOException**

Berfungsi untuk meneruskan parameter-parameter yang diinput melalui *graphical user interface* dan memerintah *processor* untuk melakukan pemrosesan penyimpanan data.

Parameter:

- **dir**: merupakan sebuah string yang diinput melalui *graphical user interface*. String tersebut memiliki nilai yang merepresentasikan *directory* penyimpanan file.
- **filename**: merupakan sebuah string yang diinput melalui *graphical user interface*. String tersebut memiliki nilai yang merepresentasikan nama file.

## 6. FileTypeFilter

Kelas ini merupakan kelas yang bertugas sebagai *filter* file yang memiliki suatu ekstensi dengan kelas. Dalam kelas ini peneliti mem-*filter* file dengan berekstensi *.csv*. Atribut yang dimiliki oleh kelas ini antara lain :

- **private String extension** : merupakan atribut yang menentukan ekstensi yang di-*filter*.
- **private String description** : merupakan atribut deskripsi dari ekstensi yang di-*filter*.

Method-method yang dimiliki kelas ini merupakan action method dengan rincian sebagai berikut:

- **public boolean accept(File f)**

Berfungsi untuk memeriksa input dari *graphical user interface*.

Parameter:

- **f**: merupakan sebuah file.

**Kembalian:** Sebuah boolean.

- **public String getDescription()**

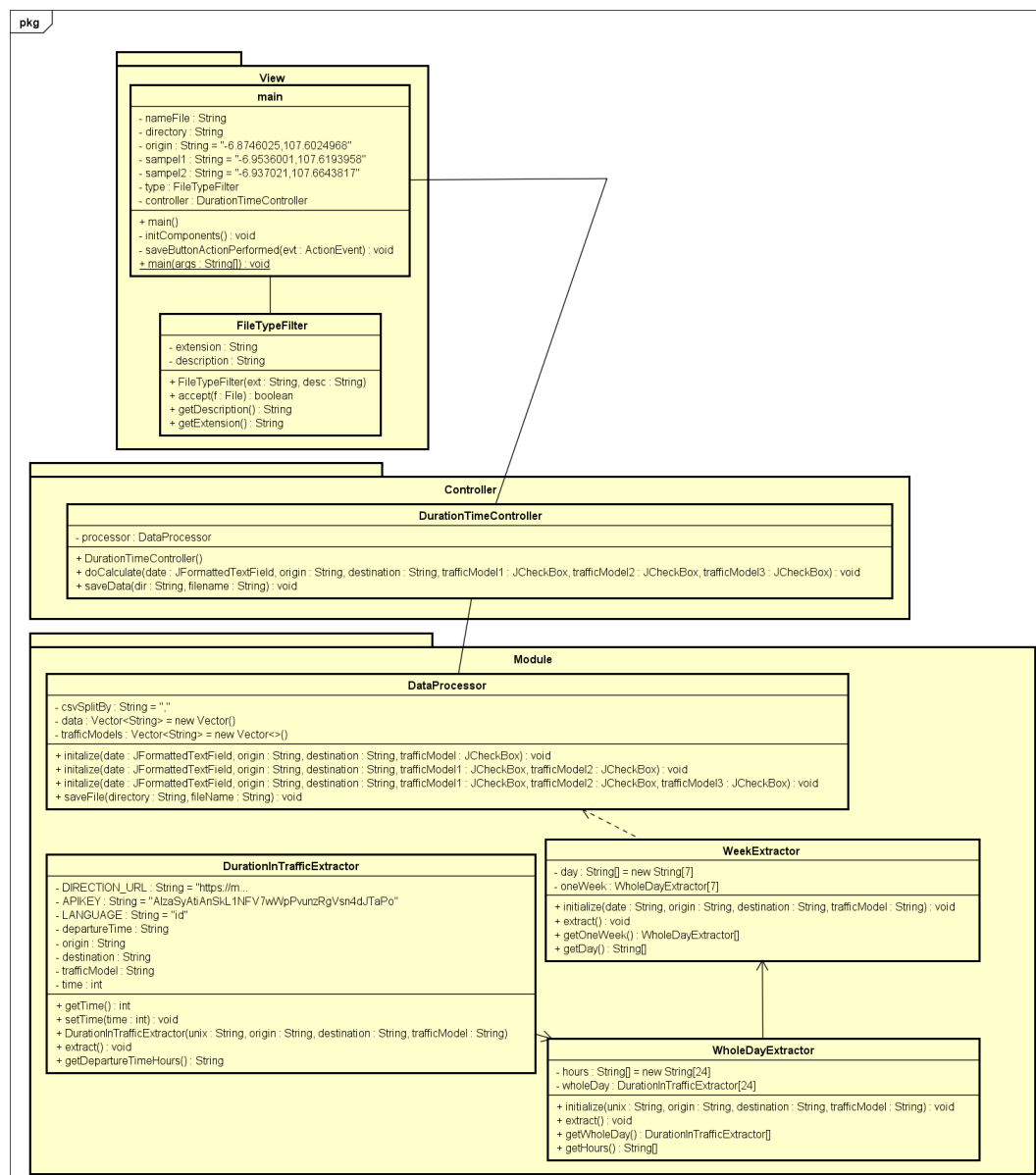
Berfungsi untuk mendapatkan nilai yang dari atribut *description*.

**Kembalian:** Sebuah string yang merupakan nilai dari atribut *description*.

- **public String getExtension()**

Berfungsi untuk mendapatkan nilai yang dari atribut *extension*.

**Kembalian:** Sebuah string yang merupakan nilai dari atribut *extension*.



Gambar 4.1: Kelas Diagram Rinci

## 4.5 Perancangan Antarmuka

Untuk memenuhi kebutuhan interaksi antara pengguna dengan sistem, maka dirancanglah sebuah antarmuka dari perangkat lunak Analisis Waktu Tempuh Kota Bandun. Rancangan antarmuka dibagi menjadi dua antarmuka antara lain:

### 1. Antarmuka utama.

Antarmuka ini adalah antarmuka utama dari perangkat lunak. Komponen antarmuka ini terdiri dari dua buah *radio button*, *date picker*, tiga buah *check box*, dan sebuah tombol *save* seperti yang ditunjukkan pada Gambar 4.2. Untuk dapat menyimpan data yang sudah diolah pengguna perlu melakukan input dan memilih sesuai dengan pilihan yang ada di antarmuka yang sesuai kemudian menekan tombol *save*. Jika berhasil pengguna akan diarahkan ke aplikasi Microsoft Excel yang berisikan data-data yang telah didapat.

**Analisi Waktu Tempuh Kota Bandung**

origin : ☐ Sampel 1 : Amaya Residence  
☐ Sampel 2 : Jl. Puspa Utara

date :

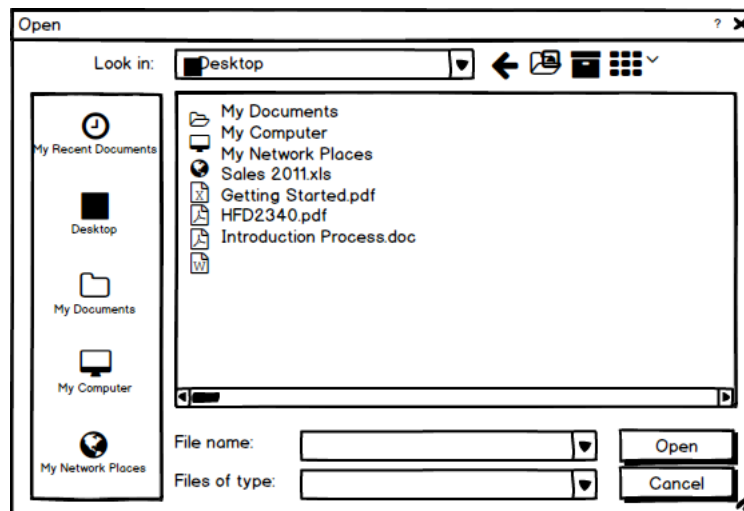
traffic model ☐ best\_guess ☐ optimistic ☐ pessimistic

**Save to .csv & Show**

Gambar 4.2: Antarmuka Utama

1     2. Antarmuka *file chooser*.

2     Antarmuka ini adalah antarmuka untuk menentukan *directory* penyimpanan file dan  
 3     menentukan nama file yang akan disimpan. Komponen antarmuka ini seperti yang  
 4     ditunjukkan pada Gambar 4.3..

Gambar 4.3: Antarmuka *File Chooser*



## BAB 5

### IMPLEMENTASI DAN PENGUJIAN

Bab ini terdiri atas dua bagian, yaitu Implementasi Perangkat Lunak dan Pengujian Perangkat Lunak. Bagian implementasi berisi penjelasan lingkungan pengembangan perangkat lunak dan hasil implementasi. Sedangkan bagian pengujian berisi hasil pengujian perangkat lunak yang telah dibangun.

#### 5.1 Implementasi

##### 5.1.1 Lingkungan Implementasi

Implementasi perangkat lunak ini dilakukan di sebuah komputer peneliti untuk keperluan pengujian dan penarikan kesimpulan. Komputer tersebut memiliki spesifikasi sebagai berikut :

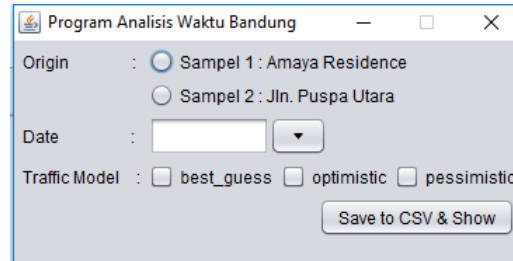
1. Processor : 1.3GHz
2. RAM: 4.00 GB DDR3
3. Sistem Operasi: Windows 10 Home 64-bit
4. Versi Java : 1.8.0\_92
5. Koneksi Internet : *bandwidth up to 1,2MBps*
6. Versi Microsoft Excel : 2016

##### 5.1.2 Implementasi Kode Program

Kode program pada perangkat lunak ditulis dalam bahasa pemrograman Java. Penulisan kode program dibagi menjadi tiga *package* yaitu : *Module*, *Controller* dan *View*. Tujuan penulisan program dibagi menjadi tiga *package* adalah untuk memudahkan proses debugging. Didalam *package Module* merupakan kode-kode program yang menjalankan semua fungsi mulai dari *request* sampai penyimpanan file. Untuk kode program yang ada pada *package Controller* merupakan kode program yang berfungsi untuk menjembatani antara tampilan dengan fungsi-fungsi untuk menjalankan perangkat lunak. Tampilan perangkat lunak ditulis didalam *package View* agar dapat mendukung interaksi antarmuka agar interaksi aplikasi lebih interaktif. Penulisan kode program menggunakan *library* : *jsoup* dengan versi 1.10.1, *JSON* dengan versi 20160810 dan *swingx-all* dengan versi 1.6.4. Untuk kode-kode program tersebut dapat dilihat pada lampiran A, lampiran B dan lampiran C.

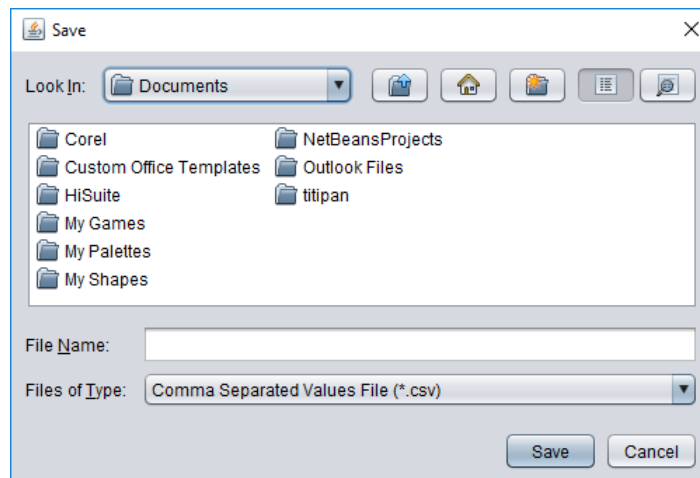
### 5.1.3 Tampilan antarmuka

Berikut ini merupakan hasil implementasi antarmuka untuk perangkat lunak analisis waktu tempuh kota Bandung. Pada Gambar 5.1 merupakan tampilan utama dari perangkat lunak yang memiliki tiga buah input yaitu : *radio button*, *datepicker*, *check box*. Terdapat 1 buah tombol *save* yang digunakan untuk melakukan ekstraksi data dan penyimpanan data.



Gambar 5.1: Implementasi Antarmuka Utama

Pada Gambar 5.2 merupakan tampilan *file chooser* dari perangkat lunak yang memiliki satu buah *window* untuk memilih *directory* penyimpanan file. Terdapat satu buah input untuk memberi nama file. Selain itu tampilan juga terdapat filter file untuk menyimpan data dengan suatu ekstensi tertentu. Terdapat dua buah tombol untuk fitur penyimpanan yaitu : *save* dimana tombol ini berfungsi untuk mengeksekusi penyimpanan file kemudian membuka file tersebut dengan aplikasi Microsoft Excel dan *cancel* untuk membatalkan penyimpanan file.



Gambar 5.2: Implementasi *file chooser*

## 5.2 Pengujian

### 5.2.1 Pengujian Fungsional

Pengujian fungsional perangkat lunak sederhana analisis waktu tempuh kota Bandung dengan memanfaatkan Google Direction API dilakukan untuk mengetahui kesesuaian reaksi perangkat lunak dengan reaksi yang diharapkan berdasarkan aksi pengguna terhadap perangkat lunak. Pengujian ini dilakukan pada lingkungan implementasi sesuai pada subbab 5.1.1. Terdapat 4 tes kasus yang diujikan, detail serta hasilnya dapat dilihat pada Tabel 5.1.

1 Beberapa data hasil ekstraksi pada pengujian ini bisa dilihat pada Lampiran D. Spesifikasi  
 2 pengujian fungsional adalah sebagai berikut :

- 3 1. Pengujian dilakukan 14 kali untuk masing-masing sampel sesuai dengan tes kasus.
- 4 2. Masukan tanggal yang digunakan adalah 8 Mei 2017 dan 15 Mei 2017.
- 5 3. Tes kasus nomor 2 dilakukan sebanyak 2 kali untuk masing-masing sampel : 1 kali  
 6 dengan masukan 8 Mei 2017 dan 1 kali dengan masukan 15 Mei 2017
- 7 4. Tes kasus nomor 3 dilakukan sebanyak 6 kali untuk masing-masing sampel : 1 kali  
 8 dengan masukan 8 Mei 2017 dengan kombinasi model *traffic best\_guess* dan *optimis-*  
 9 *tic*, 1 kali dengan masukan 15 Mei 2017 dengan kombinasi model *traffic best\_guess*  
 10 dan *optimistic*, 1 kali dengan masukan 8 Mei 2017 dengan kombinasi model *traffic*  
 11 *best\_guess* dan *pessimistic*, 1 kali dengan masukan 15 Mei 2017 dengan kombinasi  
 12 model *traffic best\_guess* dan *pessimistic*, 1 kali dengan masukan 8 Mei 2017 dengan  
 13 kombinasi model *traffic optimistic* dan *pessimistic*, dan 1 kali dengan masukan 15 Mei  
 14 2017 dengan kombinasi model *traffic optimistic* dan *pessimistic*
- 15 5. Tes kasus nomor 4 dilakukan sebanyak 6 kali untuk masing-masing sampel : 3 kali  
 16 dengan masukan 8 Mei 2017 dengan masing-masing model *traffic* dan 3 kali dengan  
 17 masukan 15 Mei 2017 dengan masing-masing model *traffic*.

No	Aksi Pengguna	Reaksi yang diharapkan	Reaksi perangkat lunak
1	Pengguna menjalankan program	Antarmuka utama di-tampilkan	sesuai
2	Pengguna memilih salah satu sampel, memilih tanggal, memilih ketiga model <i>traffic</i> dan menekan tombol save	File berhasil disimpan dan ditampilkan dengan aplikasi microsoft excel	sesuai
3	Pengguna memilih salah satu sampel, memilih tanggal, memilih dua diantara tiga model <i>traffic</i> dan menekan tombol save	File berhasil disimpan dan ditampilkan dengan aplikasi microsoft excel	sesuai
4	Pengguna memilih salah satu sampel, memilih tanggal, memilih salah satu model <i>traffic</i> dan menekan tombol save	File berhasil disimpan dan ditampilkan dengan aplikasi microsoft excel	sesuai

		Jika pengguna belum memilih salah satu dari sampel menampilkan pesan "Silahkan pilih sampel yang akan dihitung antara sampel 1 atau sampel 2"	sesuai
		Jika pengguna belum memilih salah satu dari traffic_model menampilkan pesan "Anda harus memilih minimal salah satu dari 3 traffic model yang telah disediakan"	sesuai
		Jika pengguna belum memilih tanggal menampilkan pesan "Anda belum memilih tanggal, silahkan pilih tanggal"	sesuai
		Jika pengguna memilih tanggal yang sudah lampau atau hari ini menampilkan pesan "Tanggal yang anda masukan adalah masa lampau atau hari ini, silahkan pilihlah tanggal yang akan datang"	sesuai
		Jika pengguna memilih tanggal yang bukan hari senin menampilkan pesan "Tanggal yang anda pilih bukan hari senin, silahkan pilih tanggal yang merupakan hari senin"	sesuai

Tabel 5.1: Tabel Hasil Pengujian Fungsional



### 1 5.2.2 Pengujian Eksperimental

2 Pengujian eksperimental dilakukan dengan melakukan eksperimen dari hasil ekstraksi data  
3 yang ada pada Lampiran D dengan cara membuat analisis dari bagan yang datanya berasal  
4 dari hasil ekstraksi data tersebut. Bagan itu sendiri dapat dibuat dengan memanfaatkan  
5 aplikasi Microsoft Excel secara manual. Hasil pengujian eksperimental dapat dilihat pada  
6 Lampiran E yang menunjukkan perbedaan waktu tempuh pada setiap jamnya dari masing-  
7 masing sampel. Hasil pengujian eksperimental dirangkum sebagai berikut.

8 Pada grafik waktu masing-masing model dalam seminggu yang dapat dilihat pada Lam-  
9 piran E, bahwa waktu tempuh untuk setiap hari relatif memiliki waktu tempuh yang sama  
10 setiap jamnya terkecuali pada hari jumat. Pada hari jumat, waktu tempuh cenderung  
11 menurun pada pukul 12, dan lalu menaik kembali setelah itu. Hal tersebut diperkirakan  
12 oleh karena mayoritas warga indonesia beragama muslim dan melaksanakan ibadah shalat  
13 jumat pada pukul 12. Selain itu, waktu tempuh mulai menaik sejak dari pukul 6 hingga  
14 mencapai pukul 18, dan setelah itu akan selalu menurun. Waktu tempuh maksimum pada  
15 setiap harinya berada pada sekitar pukul 18. Hal tersebut diperkirakan terjadi karena pada  
16 jam tersebut merupakan saat sebagian besar orang selesai beraktifitas dan pulang ke rumah  
17 masing-masing. Sedangkan waktu tempuh yang paling minimum ada pada setiap harinya  
18 berada pada sekitar pukul 3. Hal tersebut diperkirakan terjadi karena pada jam tersebut  
19 sebagian orang masih beristirahat dirumah masing-masing.



## BAB 6

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Dari hasil pembangunan perangkat lunak analisis waktu tempuh kota Bandung dengan memanfaatkan layanan Google Direction dan hasil data yang didapatkan, didapatkanlah kesimpulan-kesimpulan sebagai berikut :

1. Telah berhasil mengimplementasi Google Direction API dalam bahasa pemrograman Java menggunakan *library* jsoup. Dengan *library* jsoup, dapat melakukan *request* ke layanan Google Direction menggunakan url yang telah mengandung parameter-parameter yang diperlukan.
2. Telah berhasil mengekstraksi data dengan menggunakan *library* JSON. Dengan *library* JSON, *response* dari *request* ke layanan Google Direction dapat diekstraksi dengan cara mengekstrak nilai dari *duration\_in\_traffic*.
3. Sesuai dengan apa yang telah dipaparkan pada subbab 5.2.2, Waktu terbaik untuk melakukan perjalanan pada sampel 1 adalah **hari rabu pukul 4**. Pada sampel 2 waktu terbaik untuk melakukan perjalanan adalah hari **hari rabu pukul 4**.

#### 6.2 Saran

Dari hasil penelitian termasuk kesimpulan yang didapat, berikut adalah beberapa saran untuk pengembangan:

1. Penelitian ini memanfaatkan Google Direction API dengan mengembalikan *response* dengan format JSON. Oleh karena itu, sebaiknya perangkat lunak yang akan dibangun bisa menangani segala format keluaran/*response* dari layanan ini.
2. Pada parameter *APIKEY* yang terintegrasi dengan akun *Google* dimana parameter tersebut digunakan untuk melakukan *request* ke layanan Google Direction memiliki batas untuk melakukan *request*. Oleh karena itu, sebaiknya perhatikan kuota sebelum pengembang melakukan *request* atau memperbesar kuota *request* dengan membayar sejumlah uang ke pihak Google untuk berjaga-jaga terjadi melebihi batas kuota pada saat melakukan *request*.



## DAFTAR REFERENSI

1

- 2 [1] Wong, C. (2000) *Http pocket reference: Hypertext transfer protocol*. O'Reilly Media.
- 3 [2] Hedley, J. (2016) *jsoup: Java HTML Parser*, 1.10.1 edition.
- 4 [3] International, E. (2013) *Standard ECMA- 404 - The JSON Data Interchange Format*,  
5 1st edition.
- 6 [4] Kotwal, A. (2017) *Google Direction API*.



## LAMPIRAN A

### KODE PROGRAM PADA *PACKAGE MODULE*

Listing A.1: DurationInTrafficExtractor.java

```
3 package Module;
4
5 import java.io.IOException;
6 import java.util.Calendar;
7 import java.util.Date;
8 import java.util.GregorianCalendar;
9 import org.json.JSONArray;
10 import org.json.JSONObject;
11 import org.jsoup.Connection;
12 import org.jsoup.Jsoup;
13
14 public class DurationInTrafficExtractor {
15
16     private final String DIRECTION_URL = "https://maps.googleapis.com/maps/api/directions/json";
17     private final String APIKEY = "AIzaSyAtiAnSkL1NFV7wWpPvunzRgVsn4dJTaPo";
18     private final String LANGUAGE = "id";
19     private final String departureTime;
20     private final String origin;
21     private final String destination;
22     private final String trafficModel;
23     private int time;
24
25     public int getTime() {
26         return time;
27     }
28
29     public void setTime(int time) {
30         this.time = time;
31     }
32
33     public DurationInTrafficExtractor(String unix, String origin, String destination, String
34         trafficModel) {
35         this.departureTime = unix;
36         this.origin = origin;
37         this.destination = destination;
38         this.trafficModel = trafficModel;
39     }
40
41     public void extract() throws IOException {
42         String content;
43         JSONObject jsonResponse;
44         //Connection connection = HttpConnection.connect(this.DIRECTION_URL);
45         Connection connection = Jsoup.connect(this.DIRECTION_URL);
46         connection.timeout(180000);
47         connection.data("key", this.APIKEY);
48         connection.data("origin", this.origin);
49         connection.data("destination", this.destination);
50         connection.data("language", this.LANGUAGE);
51         connection.data("departure_time", this.departureTime);
52         connection.data("traffic_model", this.trafficModel);
53         connection.ignoreContentType(true);
54         content = connection.execute().body();
55         jsonResponse = new JSONObject(content);
56
57         if (jsonResponse.getString("status").equals("OK")) {
58             JSONArray routes = jsonResponse.getJSONArray("routes");
59             JSONObject route = routes.getJSONObject(0);
60             JSONArray legs = route.getJSONArray("legs");
61             JSONObject leg = legs.getJSONObject(0);
62             JSONObject duration = leg.getJSONObject("duration_in_traffic");
63             int durationValue = Integer.parseInt(duration.optString("value"));
64             if (durationValue%60 == 0) {
65                 this.setTime(durationValue/60);
66             } else {

```

```

1      this.setTime((durationValue/60)+1);
2      }
3      } else {
4          System.err.println(jsonResponse.getString("status"));
5      }
6  }
7
8  public String getDepartureTimeHours() {
9      Calendar tempCal = new GregorianCalendar();
10     Long tempLong = new Long(this.departureTime);
11     tempLong = tempLong*1000;
12     Date tempDate = new Date(tempLong);
13     tempCal.setTime(tempDate);
14     return tempCal.get(Calendar.HOUR_OF_DAY) + "";
15 }
16 }

```

Listing A.2: WholeDayExtractor.java

```

17 package Module;
18
19 import java.io.IOException;
20 import java.text.ParseException;
21 public class WholeDayExtractor {
22     private final DurationInTrafficExtractor[] wholeDay = new DurationInTrafficExtractor[24];
23     private final String[] hours = new String[24];
24
25     public void initialize(String unix, String origin, String destination, String trafficModel)
26         throws ParseException{
27         Long unixDate = new Long(unix);//temp.getTimeInMillis()/1000;
28         long counterUnixperHour = 3600;
29         for (int i = 0; i < wholeDay.length; i++) {
30             wholeDay[i] = new DurationInTrafficExtractor(unixDate.toString(), origin, destination
31                 , trafficModel);
32             unixDate += counterUnixperHour;
33         }
34     }
35
36     public void extract() throws IOException{
37         for (int i = 0; i < wholeDay.length; i++) {
38             wholeDay[i].extract();
39             hours[i] = wholeDay[i].getDepartureTimeHours();
40         }
41     }
42
43     public DurationInTrafficExtractor[] getWholeDay() {
44         return wholeDay;
45     }
46
47     public String[] getHours() {
48         return hours;
49     }
50 }

```

Listing A.3: WeekExtractor.java

```

51 package Module;
52
53 import java.io.IOException;
54 import java.text.ParseException;
55 import java.text.SimpleDateFormat;
56 import java.util.Calendar;
57 import java.util.Date;
58 import java.util.GregorianCalendar;
59
60 public class WeekExtractor {
61
62     private final WholeDayExtractor[] oneWeek = new WholeDayExtractor[7];
63     private final String[] day = new String[7];
64
65     public void initialize(String date, String origin, String destination, String trafficModel)
66         throws ParseException{
67         SimpleDateFormat format = new SimpleDateFormat("dd MM yyyy");
68         Calendar temp = new GregorianCalendar();
69         Date tempDate = format.parse(date);
70         temp.setTime(tempDate);
71         temp.set(Calendar.HOUR_OF_DAY, 0);
72         temp.set(Calendar.MINUTE, 0);
73         Long unixDate = temp.getTimeInMillis()/1000;
74         long counterUnixperDay = 86400;
75         for (int i = 0; i < oneWeek.length; i++) {
76             oneWeek[i] = new WholeDayExtractor();
77             oneWeek[i].initialize(unixDate.toString(), origin, destination, trafficModel);
78             unixDate = unixDate + counterUnixperDay;

```



```

1      }
2  }
3
4  public void extract() throws IOException{
5      for (int i = 0; i < oneWeek.length; i++) {
6          oneWeek[i].extract();
7          day[i] = i + 1 + "";
8      }
9  }
10
11 public WholeDayExtractor[] getOneWeek() {
12     return oneWeek;
13 }
14
15 public String[] getDay() {
16     return day;
17 }
18 }

```

Listing A.4: DataProcessor.java

```

19 package Module;
20
21 import java.io.BufferedWriter;
22 import java.io.File;
23 import java.io.FileWriter;
24 import java.io.IOException;
25 import java.text.ParseException;
26 import java.util.Vector;
27 import javax.swing.JCheckBox;
28 import javax.swing.JFormattedTextField;
29
30 public class DataProcessor {
31
32     private final String csvSplitBy = ",";
33     private final Vector<String> data = new Vector();
34     private final Vector<String> trafficModels = new Vector<>();
35
36     public void initialize(JFormattedTextField date, String origin, String destination, JCheckBox
37         trafficModel) throws ParseException, IOException {
38         data.clear();
39         trafficModels.clear();
40         WeekExtractor temp = new WeekExtractor();
41         temp.initialize(date.getText(), origin, destination, trafficModel.getText());
42         temp.extract();
43         trafficModels.add(trafficModel.getText());
44
45         DurationInTrafficExtractor[] tempDuration;
46         String[] tempHours;
47         WholeDayExtractor[] tempWeek = temp.getOneWeek();
48         String[] tempDays = temp.getDay();
49         for (int i = 0; i < tempWeek.length && i < tempDays.length; i++) {
50             tempDuration = tempWeek[i].getWholeDay();
51             tempHours = tempWeek[i].getHours();
52             for (int j = 0; j < tempDuration.length && j < tempHours.length; j++) {
53                 data.add(tempDays[i] + csvSplitBy + tempHours[j] + csvSplitBy + tempDuration[j].
54                     getTime());
55             }
56         }
57     }
58
59     public void initialize(JFormattedTextField date, String origin, String destination, JCheckBox
60         trafficModel1, JCheckBox trafficModel2) throws ParseException, IOException {
61         data.clear();
62         trafficModels.clear();
63         WeekExtractor temp1 = new WeekExtractor();
64         WeekExtractor temp2 = new WeekExtractor();
65
66         temp1.initialize(date.getText(), origin, destination, trafficModel1.getText());
67         temp1.extract();
68         temp2.initialize(date.getText(), origin, destination, trafficModel2.getText());
69         temp2.extract();
70         trafficModels.add(trafficModel1.getText());
71         trafficModels.add(trafficModel2.getText());
72
73         DurationInTrafficExtractor[] tempDuration1;
74         DurationInTrafficExtractor[] tempDuration2;
75         String[] tempHours1;
76         String[] tempHours2;
77         WholeDayExtractor[] tempWeek1 = temp1.getOneWeek();
78         WholeDayExtractor[] tempWeek2 = temp2.getOneWeek();
79         String[] tempDays1 = temp1.getDay();
80
81         for (int i = 0; i < tempWeek1.length && i < tempDays1.length; i++) {
82             tempDuration1 = tempWeek1[i].getWholeDay();

```

```

1      tempDuration2 = tempWeek2[i].getWholeDay();
2      tempHours1 = tempWeek1[i].getHours();
3      tempHours2 = tempWeek2[i].getHours();
4      for (int j = 0; j < tempDuration1.length && j < tempHours1.length; j++) {
5          data.add(tempDays1[i] + csvSplitBy + tempHours1[j] + csvSplitBy + tempDuration1[j]
6              .getTime() + csvSplitBy + tempDuration2[j].getTime());
7      }
8  }
9  }
10
11 public void initialize(JFormattedTextField date, String origin, String destination, JCheckBox
12     trafficModel1, JCheckBox trafficModel2, JCheckBox trafficModel3) throws ParseException,
13     IOException {
14     data.clear();
15     trafficModels.clear();
16     WeekExtractor temp1 = new WeekExtractor();
17     WeekExtractor temp2 = new WeekExtractor();
18     WeekExtractor temp3 = new WeekExtractor();
19
20     temp1.initialize(date.getText(), origin, destination, trafficModel1.getText());
21     temp1.extract();
22     temp2.initialize(date.getText(), origin, destination, trafficModel2.getText());
23     temp2.extract();
24     temp3.initialize(date.getText(), origin, destination, trafficModel3.getText());
25     temp3.extract();
26     trafficModels.add(trafficModel1.getText());
27     trafficModels.add(trafficModel2.getText());
28     trafficModels.add(trafficModel3.getText());
29
30     DurationInTrafficExtractor[] tempDuration1;
31     DurationInTrafficExtractor[] tempDuration2;
32     DurationInTrafficExtractor[] tempDuration3;
33     String[] tempHours1;
34     String[] tempHours2;
35     String[] tempHours3;
36     WholeDayExtractor[] tempWeek1 = temp1.getOneWeek();
37     WholeDayExtractor[] tempWeek2 = temp2.getOneWeek();
38     WholeDayExtractor[] tempWeek3 = temp3.getOneWeek();
39     String[] tempDays1 = temp1.getDay();
40
41     for (int i = 0; i < tempWeek1.length && i < tempDays1.length; i++) {
42         tempDuration1 = tempWeek1[i].getWholeDay();
43         tempDuration2 = tempWeek2[i].getWholeDay();
44         tempDuration3 = tempWeek3[i].getWholeDay();
45         tempHours1 = tempWeek1[i].getHours();
46         tempHours2 = tempWeek2[i].getHours();
47         tempHours3 = tempWeek3[i].getHours();
48         for (int j = 0; j < tempDuration1.length && j < tempHours1.length; j++) {
49             data.add(tempDays1[i] + csvSplitBy + tempHours1[j] + csvSplitBy + tempDuration1[j]
50                 .getTime() + csvSplitBy + tempDuration2[j].getTime() + csvSplitBy +
51                 tempDuration3[j].getTime());
52         }
53     }
54 }
55
56 public void saveFile(String directory, String fileName) throws IOException {
57     File file = new File(directory + "\\\" + fileName);
58     try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
59         for (String data1 : data) {
60             writer.write(data1);
61             writer.newLine();
62         }
63         writer.flush();
64     }
65 }
66 }

```

## LAMPIRAN B

### KODE PROGRAM PADA *PACKAGE CONTROLLER*

Listing B.1: DurationTimeController.java

```
3 package Controller;
4
5 import Module.DataProcessor;
6 import java.io.IOException;
7 import java.text.ParseException;
8 import javax.swing.JCheckBox;
9 import javax.swing.JFormattedTextField;
10
11 public class DurationTimeController {
12
13     private final DataProcessor processor;
14
15     public DurationTimeController () {
16         this.processor = new DataProcessor();
17     }
18
19     public void doCalculate(JFormattedTextField date, String origin, String destination, JCheckBox
20         trafficModel1, JCheckBox trafficModel2, JCheckBox trafficModel3) throws ParseException,
21         IOException {
22         if (trafficModel3==null) {
23             if(trafficModel2==null){
24                 processor.initialize(date, origin, destination, trafficModel1);
25             }else{
26                 processor.initialize(date, origin, destination, trafficModel1, trafficModel2);
27             }
28         }else{
29             processor.initialize(date, origin, destination, trafficModel1, trafficModel2,
30                 trafficModel3);
31         }
32     }
33
34     public void saveData(String dir, String filename) throws IOException {
35         processor.saveFile(dir, filename);
36     }
37 }
```



## LAMPIRAN C

### KODE PROGRAM PADA *PACKAGE VIEW*

Listing C.1: main.java

```
3 package View;
4
5 import Controller.DurationTimeController;
6 import java.awt.Desktop;
7 import java.io.File;
8 import java.io.IOException;
9 import java.text.ParseException;
10 import java.util.Calendar;
11 import java.util.Date;
12 import java.util.GregorianCalendar;
13 import java.util.logging.Level;
14 import java.util.logging.Logger;
15 import javax.swing.JFileChooser;
16 import javax.swing.JOptionPane;
17
18 public class main extends javax.swing.JFrame {
19
20     private String nameFile;
21     private String directory;
22     private final DurationTimeController controller;
23     private final FileTypeFilter type = new FileTypeFilter(".csv", "Comma Separated Values File");
24     private final JFileChooser chooser = new JFileChooser();
25     private final String origin = "-6.8746025,107.6024968";
26     private final String sampel1 = "-6.9536001,107.6193958";
27     private final String sampel2 = "-6.937021,107.6643817";
28
29     public main() {
30         this.controller = new DurationTimeController();
31         chooser.setFileFilter(type);
32         initComponents();
33     }
34
35     private void initComponents() {
36
37         buttonGroup1 = new javax.swing.ButtonGroup();
38         trafficModelLabel = new javax.swing.JLabel();
39         dateLabel = new javax.swing.JLabel();
40         originLabel = new javax.swing.JLabel();
41         bestGuessCheckBox = new javax.swing.JCheckBox();
42         optimistCheckBox = new javax.swing.JCheckBox();
43         pessimistCheckBox = new javax.swing.JCheckBox();
44         saveButton = new javax.swing.JButton();
45         sampel1RadioButton = new javax.swing.JRadioButton();
46         sampel2RadioButton = new javax.swing.JRadioButton();
47         datePicker = new org.jdesktop.swingx.JXDatePicker();
48         datePicker.getEditor().setEditable(false);
49
50         setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
51         setTitle("Program Analisis Waktu Bandung");
52         setResizable(false);
53
54         trafficModelLabel.setText("Traffic Model      :");
55
56         dateLabel.setText("Date                      :");
57
58         originLabel.setText("Origin                      : ");
59
60         bestGuessCheckBox.setText("best_guess");
61
62         optimistCheckBox.setText("optimistic");
63
64         pessimistCheckBox.setText("pessimistic");
65
66         saveButton.setText("Save to CSV & Show");
```

```

1      saveButton.addActionListener(new java.awt.event.ActionListener() {
2          public void actionPerformed(java.awt.event.ActionEvent evt) {
3              saveButtonActionPerformed(evt);
4          }
5      });
6
7      buttonGroup1.add(sampel1RadioButton);
8      sampel1RadioButton.setText("Sampel 1 : Amaya Residence");
9
10     buttonGroup1.add(sampel2RadioButton);
11     sampel2RadioButton.setText("Sampel 2 : Jln. Puspa Utara");
12
13     datePicker.setFormats("dd MM yyyy");
14     datePicker.setLinkDay(new Date());
15
16     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
17     getContentPane().setLayout(layout);
18     layout.setHorizontalGroup(
19         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
20             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
21                 .addGap()
22                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
23                     .addGroup(layout.createSequentialGroup()
24                         .createSequentialGroup()
25                         .addComponent(originLabel)
26                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
27                         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
28                             .addComponent(sampel2RadioButton, javax.swing.GroupLayout.Alignment.LEADING)
29                             .addComponent(sampel1RadioButton, javax.swing.GroupLayout.Alignment.LEADING))
30                         .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
31                             .createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
32                             .addComponent(saveButton)
33                             .addGroup(layout.createSequentialGroup()
34                                 .createSequentialGroup()
35                                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
36                                     .addComponent(trafficModelLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
37                                     .addComponent(dateLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
38                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
39                                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
40                                     .addComponent(datePicker, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
41                                     .addGroup(layout.createSequentialGroup()
42                                         .createSequentialGroup()
43                                         .addComponent(bestGuessCheckBox)
44                                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
45                                         .addComponent(optimistCheckBox)
46                                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
47                                         .addComponent(pessimistCheckBox))))))
48                     .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
49             .setVerticalGroup(
50                 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
51                     .addGroup(layout.createSequentialGroup()
52                         .createSequentialGroup()
53                         .addGap()
54                         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
55                             .addComponent(originLabel)
56                             .addComponent(sampel1RadioButton))
57                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
58                         .addComponent(sampel2RadioButton)
59                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
60                         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
61                             .addComponent(dateLabel)
62                             .addComponent(datePicker, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
63                         .addGap(6, 6, 6)
64                         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
65                             .addComponent(trafficModelLabel)
66                             .addComponent(optimistCheckBox)
67                             .addComponent(pessimistCheckBox)
68                             .addComponent(bestGuessCheckBox))
69                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
70                         .addComponent(saveButton)
71                         .addGap(18, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
72                     .setVerticalGroup(
73                         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
74                             .addGroup(layout.createSequentialGroup()
75                                 .createSequentialGroup()
76                                 .addGap()
77                                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
78                                     .addComponent(trafficModelLabel)
79                                     .addComponent(optimistCheckBox)
80                                     .addComponent(pessimistCheckBox)
81                                     .addComponent(bestGuessCheckBox))
82                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
83                                 .addComponent(saveButton)
84                                 .addGap(18, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
85                             .setVerticalGroup(

```



```

1         directory = chooser.getCurrentDirectory().toString();
2         controller.saveData(directory, nameFile);
3         File file = new File(directory + "\\\" + nameFile);
4         dt.open(file);
5     }
6 } else {
7     controller.doCalculate(datePicker.getEditor(), origin, sampel1
8         , optimistCheckBox, null, null);
9     int result = chooser.showSaveDialog(this);
10    if (result == JFileChooser.APPROVE_OPTION) {
11        Desktop dt = Desktop.getDesktop();
12        nameFile = chooser.getSelectedFile().getName() + type.
13            getExtension();
14        directory = chooser.getCurrentDirectory().toString();
15        controller.saveData(directory, nameFile);
16        File file = new File(directory + "\\\" + nameFile);
17        dt.open(file);
18    }
19 }
20 } else if (pessimistCheckBox.isSelected()) {
21     controller.doCalculate(datePicker.getEditor(), origin, sampel1,
22         pessimistCheckBox, null, null);
23     int result = chooser.showSaveDialog(this);
24     if (result == JFileChooser.APPROVE_OPTION) {
25         Desktop dt = Desktop.getDesktop();
26         nameFile = chooser.getSelectedFile().getName() + type.
27             getExtension();
28         directory = chooser.getCurrentDirectory().toString();
29         controller.saveData(directory, nameFile);
30         File file = new File(directory + "\\\" + nameFile);
31         dt.open(file);
32     }
33 } else {
34     JOptionPane.showMessageDialog(this, "Anda harus memilih minimal
35         salah satu dari 3 traffic model yang telah disediakan", "ERROR
36         ", JOptionPane.ERROR_MESSAGE);
37 }
38 } else if (sampel2RadioButton.isSelected()) {
39     if (bestGuessCheckBox.isSelected()) {
40         if (optimistCheckBox.isSelected()) {
41             if (pessimistCheckBox.isSelected()) {
42                 controller.doCalculate(datePicker.getEditor(), origin,
43                     sampel2, bestGuessCheckBox, optimistCheckBox,
44                     pessimistCheckBox);
45                 int result = chooser.showSaveDialog(this);
46                 if (result == JFileChooser.APPROVE_OPTION) {
47                     Desktop dt = Desktop.getDesktop();
48                     nameFile = chooser.getSelectedFile().getName() + type.
49                         getExtension();
50                     directory = chooser.getCurrentDirectory().toString();
51                     controller.saveData(directory, nameFile);
52                     File file = new File(directory + "\\\" + nameFile);
53                     dt.open(file);
54                 }
55             } else {
56                 controller.doCalculate(datePicker.getEditor(), origin,
57                     sampel2, bestGuessCheckBox, optimistCheckBox, null);
58                 int result = chooser.showSaveDialog(this);
59                 if (result == JFileChooser.APPROVE_OPTION) {
60                     Desktop dt = Desktop.getDesktop();
61                     nameFile = chooser.getSelectedFile().getName() + type.
62                         getExtension();
63                     directory = chooser.getCurrentDirectory().toString();
64                     controller.saveData(directory, nameFile);
65                     File file = new File(directory + "\\\" + nameFile);
66                     dt.open(file);
67                 }
68             }
69         } else if (pessimistCheckBox.isSelected()) {
70             controller.doCalculate(datePicker.getEditor(), origin, sampel2
71                 , bestGuessCheckBox, pessimistCheckBox, null);
72             int result = chooser.showSaveDialog(this);
73             if (result == JFileChooser.APPROVE_OPTION) {
74                 Desktop dt = Desktop.getDesktop();
75                 nameFile = chooser.getSelectedFile().getName() + type.
76                     getExtension();
77                 directory = chooser.getCurrentDirectory().toString();
78                 controller.saveData(directory, nameFile);
79                 File file = new File(directory + "\\\" + nameFile);
80                 dt.open(file);
81             }
82         } else {
83             controller.doCalculate(datePicker.getEditor(), origin, sampel2
84                 , bestGuessCheckBox, null, null);
85         }

```



```

1         int result = chooser.showSaveDialog(this);
2         if (result == JFileChooser.APPROVE_OPTION) {
3             Desktop dt = Desktop.getDesktop();
4             nameFile = chooser.getSelectedFile().getName() + type.
5                 getExtension();
6             directory = chooser.getCurrentDirectory().toString();
7             controller.saveData(directory, nameFile);
8             File file = new File(directory + "\\\" + nameFile);
9             dt.open(file);
10        }
11    }
12    } else if (optimistCheckBox.isSelected()) {
13        if (pessimistCheckBox.isSelected()) {
14            controller.doCalculate(datePicker.getEditor(), origin, sampel2
15                , optimistCheckBox, pessimistCheckBox, null);
16            int result = chooser.showSaveDialog(this);
17            if (result == JFileChooser.APPROVE_OPTION) {
18                Desktop dt = Desktop.getDesktop();
19                nameFile = chooser.getSelectedFile().getName() + type.
20                    getExtension();
21                directory = chooser.getCurrentDirectory().toString();
22                controller.saveData(directory, nameFile);
23                File file = new File(directory + "\\\" + nameFile);
24                dt.open(file);
25            }
26        } else {
27            controller.doCalculate(datePicker.getEditor(), origin, sampel2
28                , optimistCheckBox, null, null);
29            int result = chooser.showSaveDialog(this);
30            if (result == JFileChooser.APPROVE_OPTION) {
31                Desktop dt = Desktop.getDesktop();
32                nameFile = chooser.getSelectedFile().getName() + type.
33                    getExtension();
34                directory = chooser.getCurrentDirectory().toString();
35                controller.saveData(directory, nameFile);
36                File file = new File(directory + "\\\" + nameFile);
37                dt.open(file);
38            }
39        }
40    } else if (pessimistCheckBox.isSelected()) {
41        controller.doCalculate(datePicker.getEditor(), origin, sampel2,
42            pessimistCheckBox, null, null);
43        int result = chooser.showSaveDialog(this);
44        if (result == JFileChooser.APPROVE_OPTION) {
45            Desktop dt = Desktop.getDesktop();
46            nameFile = chooser.getSelectedFile().getName() + type.
47                getExtension();
48            directory = chooser.getCurrentDirectory().toString();
49            controller.saveData(directory, nameFile);
50            File file = new File(directory + "\\\" + nameFile);
51            dt.open(file);
52        }
53    } else {
54        JOptionPane.showMessageDialog(this, "Harus pilih minimal salah
55            satu dari 3", "ERROR", JOptionPane.ERROR_MESSAGE);
56    }
57    } else {
58        JOptionPane.showMessageDialog(this, "Harus pilih antara sampel 1 atau
59            sampel 2", "ERROR", JOptionPane.ERROR_MESSAGE);
60    }
61    }
62    }
63    } else {
64        JOptionPane.showMessageDialog(this, "Pilihlah tanggal", "ERROR", JOptionPane.
65            ERROR_MESSAGE);
66    }
67    } catch (ParseException | IOException ex) {
68        Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
69    }
70    }
71
72    public static void main(String args[]) {
73        try {
74            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
75                getInstalledLookAndFeels()) {
76                if ("Nimbus".equals(info.getName())) {
77                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
78                    break;
79                }
80            }
81        } catch (ClassNotFoundException | InstantiationException | IllegalAccessException | javax.
82            swing.UnsupportedLookAndFeelException ex) {
83            java.util.logging.Logger.getLogger(main.class.getName()).log(java.util.logging.Level.
84                SEVERE, null, ex);
85        }

```

```

1  java.awt.EventQueue.invokeLater(new Runnable() {
2      @Override
3      public void run() {
4          new main().setVisible(true);
5      }
6  });
7  }
8
9
10 private javax.swing.JCheckBox bestGuessCheckBox;
11 private javax.swing.ButtonGroup buttonGroup1;
12 private javax.swing.JLabel dateLabel;
13 private org.jdesktop.swingx.JXDatePicker datePicker;
14 private javax.swing.JCheckBox optimistCheckBox;
15 private javax.swing.JLabel originLabel;
16 private javax.swing.JCheckBox pessimistCheckBox;
17 private javax.swing.JRadioButton sampel1RadioButton;
18 private javax.swing.JRadioButton sampel2RadioButton;
19 private javax.swing.JButton saveButton;
20 private javax.swing.JLabel trafficModelLabel;
21 }

```

Listing C.2: FileTypeFilter.java

```

22 package View;
23
24 import java.io.File;
25 import javax.swing.filechooser.*;
26
27 public class FileTypeFilter extends FileFilter{
28
29     private final String extension;
30     private final String description;
31
32     public FileTypeFilter(String ext, String desc){
33         extension = ext;
34         description = desc;
35     }
36
37     @Override
38     public boolean accept(File f) {
39         if (f.isDirectory()) {
40             return true;
41         }else{
42             return f.getName().endsWith(extension);
43         }
44     }
45
46     @Override
47     public String getDescription() {
48         return description + String.format(" (%s)", extension);
49     }
50
51     public String getExtension() {
52         return extension;
53     }
54 }

```

## LAMPIRAN D

### DATA HASIL PENGUJIAN

Pengujian perangkat lunak analisis waktu tempuh kota Bandung menguji ekstraksi data dari tempat asal ke tujuan. Sampel yang digunakan adalah dua sampel yaitu : tujuan asal Amaya Residence ke Universitas Katolik Parahyangan dan tujuan asal Jalan Puspa Utara ke Universitas Katolik Parahyangan. Pengujian ini dilakukan dengan menggunakan input tanggal 8 Mei 2017 dan 15 Mei 2017. Berikut adalah data hasil ekstraksi waktu tempuh dari kedua sampel tersebut :

1	0	33	28	42
1	1	32	29	40
1	2	32	29	37
1	3	31	28	36
1	4	31	27	36
1	5	32	28	38
1	6	36	30	47
1	7	39	31	55
1	8	39	32	54
1	9	40	33	57
1	10	43	35	63
1	11	44	36	66
1	12	44	35	64
1	13	45	36	69
1	14	48	37	72
1	15	49	38	73
1	16	49	39	77
1	17	53	42	78
1	18	46	37	67
1	19	43	35	61
1	20	41	34	57
1	21	40	33	54
1	22	38	31	49
1	23	35	30	45
2	0	34	29	42
2	1	32	28	40
2	2	31	29	37

2	3	31	28	36
2	4	31	28	35
2	5	32	28	37
2	6	36	30	47
2	7	39	32	55
2	8	39	32	55
2	9	40	33	56
2	10	43	34	62
2	11	44	35	63
2	12	45	36	65
2	13	46	37	71
2	14	48	38	71
2	15	48	38	71
2	16	50	39	76
2	17	54	42	79
2	18	46	38	67
2	19	44	36	63
2	20	42	35	59
2	21	41	34	55
2	22	38	32	50
2	23	35	30	45
3	0	34	29	42
3	1	33	29	40
3	2	32	29	37
3	3	31	28	36
3	4	30	28	34
3	5	33	29	38
3	6	36	30	49
3	7	39	32	56
3	8	40	32	56
3	9	41	33	58
3	10	44	35	63
3	11	44	35	65
3	12	44	36	64
3	13	47	37	70
3	14	48	38	73
3	15	50	39	73
3	16	50	40	79
3	17	54	43	82
3	18	48	39	69
3	19	44	36	63
3	20	43	35	59

3	21	41	34	59
3	22	38	32	51
3	23	35	30	46
4	0	34	29	42
4	1	33	29	41
4	2	32	28	37
4	3	32	29	37
4	4	31	28	36
4	5	33	29	38
4	6	37	30	48
4	7	39	32	56
4	8	40	33	55
4	9	41	33	58
4	10	44	35	63
4	11	45	36	66
4	12	45	36	66
4	13	47	37	71
4	14	49	39	74
4	15	49	39	72
4	16	50	40	77
4	17	53	42	78
4	18	46	37	68
4	19	45	37	64
4	20	43	35	61
4	21	42	34	57
4	22	38	32	51
4	23	35	30	46
5	0	33	29	42
5	1	33	28	42
5	2	32	29	39
5	3	32	28	37
5	4	31	28	36
5	5	33	29	39
5	6	36	30	48
5	7	39	32	56
5	8	40	33	57
5	9	41	33	59
5	10	44	35	64
5	11	46	37	68
5	12	41	33	57
5	13	46	36	71
5	14	50	39	75

5	15	49	39	74
5	16	52	40	82
5	17	57	44	85
5	18	51	40	74
5	19	48	38	69
5	20	45	36	66
5	21	44	36	61
5	22	41	34	57
5	23	37	31	49
6	0	34	30	44
6	1	34	29	42
6	2	32	28	40
6	3	32	28	38
6	4	32	28	40
6	5	33	28	41
6	6	34	29	45
6	7	36	30	49
6	8	38	31	53
6	9	40	33	57
6	10	43	35	63
6	11	44	35	65
6	12	46	36	70
6	13	48	38	73
6	14	52	41	79
6	15	50	40	76
6	16	50	40	76
6	17	51	41	76
6	18	51	41	76
6	19	52	42	76
6	20	53	42	77
6	21	51	41	75
6	22	48	39	66
6	23	41	35	57
7	0	36	31	47
7	1	34	29	43
7	2	33	29	40
7	3	32	29	39
7	4	31	28	38
7	5	33	29	39
7	6	34	28	44
7	7	35	29	46
7	8	38	31	52

7	9	40	33	56
7	10	45	36	65
7	11	47	37	70
7	12	49	38	73
7	13	51	40	76
7	14	51	41	78
7	15	51	41	77
7	16	50	40	74
7	17	50	41	73
7	18	49	40	71
7	19	48	39	68
7	20	46	37	66
7	21	43	36	60
7	22	39	32	52
7	23	35	30	46

Tabel D.1: Data sampel 1 pada tanggal 8 Mei 2017 - 14 Mei 2017

1	0	33	28	42
1	1	32	29	40
1	2	32	29	37
1	3	31	28	36
1	4	31	27	36
1	5	32	28	38
1	6	36	30	47
1	7	39	31	55
1	8	39	32	54
1	9	40	33	57
1	10	43	35	63
1	11	44	36	66
1	12	44	35	64
1	13	45	36	69
1	14	48	37	72
1	15	49	38	73
1	16	49	39	77
1	17	53	42	78
1	18	46	37	67
1	19	43	35	61
1	20	41	34	57
1	21	40	33	54
1	22	38	31	49
1	23	35	30	45

2	0	34	29	42
2	1	32	28	40
2	2	31	29	37
2	3	31	28	36
2	4	31	28	35
2	5	32	28	37
2	6	36	30	47
2	7	39	32	55
2	8	39	32	55
2	9	40	33	56
2	10	43	34	62
2	11	44	35	63
2	12	45	36	65
2	13	46	37	71
2	14	48	38	71
2	15	48	38	71
2	16	50	39	76
2	17	54	42	79
2	18	46	38	67
2	19	44	36	63
2	20	42	35	59
2	21	41	34	55
2	22	38	32	50
2	23	35	30	45
3	0	34	29	42
3	1	33	29	40
3	2	32	29	37
3	3	31	28	36
3	4	30	28	34
3	5	33	29	38
3	6	36	30	49
3	7	39	32	56
3	8	40	32	56
3	9	41	33	58
3	10	44	35	63
3	11	44	35	65
3	12	44	36	64
3	13	47	37	70
3	14	48	38	73
3	15	50	39	73
3	16	50	40	79
3	17	54	43	82



3	18	48	39	69
3	19	44	36	63
3	20	43	35	59
3	21	41	34	59
3	22	38	32	51
3	23	35	30	46
4	0	34	29	42
4	1	33	29	41
4	2	32	28	37
4	3	32	29	37
4	4	31	28	36
4	5	33	29	38
4	6	37	30	48
4	7	39	32	56
4	8	40	33	55
4	9	41	33	58
4	10	44	35	63
4	11	45	36	66
4	12	45	36	66
4	13	47	37	71
4	14	49	39	74
4	15	49	39	72
4	16	50	40	77
4	17	53	42	78
4	18	46	37	68
4	19	45	37	64
4	20	43	35	61
4	21	42	34	57
4	22	38	32	51
4	23	35	30	46
5	0	33	29	42
5	1	33	28	42
5	2	32	29	39
5	3	32	28	37
5	4	31	28	36
5	5	33	29	39
5	6	36	30	48
5	7	39	32	56
5	8	40	33	57
5	9	41	33	59
5	10	44	35	64
5	11	46	37	68

5	12	41	33	57
5	13	46	36	71
5	14	50	39	75
5	15	49	39	74
5	16	52	40	82
5	17	57	44	85
5	18	51	40	74
5	19	48	38	69
5	20	45	36	66
5	21	44	36	61
5	22	41	34	57
5	23	37	31	49
6	0	34	30	44
6	1	34	29	42
6	2	32	28	40
6	3	32	28	38
6	4	32	28	40
6	5	33	28	41
6	6	34	29	45
6	7	36	30	49
6	8	38	31	53
6	9	40	33	57
6	10	43	35	63
6	11	44	35	65
6	12	46	36	70
6	13	48	38	73
6	14	52	41	79
6	15	50	40	76
6	16	50	40	76
6	17	51	41	76
6	18	51	41	76
6	19	52	42	76
6	20	53	42	77
6	21	51	41	75
6	22	48	39	66
6	23	41	35	57
7	0	36	31	47
7	1	34	29	43
7	2	33	29	40
7	3	32	29	39
7	4	31	28	38
7	5	33	29	39

7	6	34	28	44
7	7	35	29	46
7	8	38	31	52
7	9	40	33	56
7	10	45	36	65
7	11	47	37	70
7	12	49	38	73
7	13	51	40	76
7	14	51	41	78
7	15	51	41	77
7	16	50	40	74
7	17	50	41	73
7	18	49	40	71
7	19	48	39	68
7	20	46	37	66
7	21	43	36	60
7	22	39	32	52
7	23	35	30	46

Tabel D.2: Data sampel 1 pada tanggal 15 Mei 2017 - 21 Mei 2017

1	0	31	27	40
1	1	31	27	37
1	2	30	28	35
1	3	30	28	34
1	4	29	26	34
1	5	31	27	37
1	6	37	30	50
1	7	38	32	51
1	8	37	31	50
1	9	38	31	52
1	10	41	33	59
1	11	43	35	61
1	12	42	34	60
1	13	42	34	60
1	14	45	36	67
1	15	47	37	68
1	16	50	38	78
1	17	54	42	82
1	18	49	39	72
1	19	42	35	61
1	20	39	32	54

1	21	37	31	49
1	22	35	29	47
1	23	33	28	43
2	0	31	27	39
2	1	30	27	37
2	2	30	27	35
2	3	30	27	34
2	4	30	27	35
2	5	31	27	36
2	6	37	30	50
2	7	37	31	51
2	8	37	31	52
2	9	38	31	52
2	10	40	33	56
2	11	41	34	58
2	12	42	34	58
2	13	43	34	61
2	14	46	37	68
2	15	47	37	69
2	16	50	39	78
2	17	54	42	84
2	18	49	40	72
2	19	45	36	64
2	20	40	33	56
2	21	38	32	51
2	22	35	30	47
2	23	33	28	43
3	0	31	27	39
3	1	31	28	37
3	2	31	28	35
3	3	30	27	34
3	4	30	27	35
3	5	31	27	37
3	6	37	30	51
3	7	38	31	51
3	8	38	32	54
3	9	39	32	54
3	10	41	34	60
3	11	43	34	61
3	12	41	34	60
3	13	44	36	64
3	14	48	39	75

3	15	51	40	76
3	16	53	42	83
3	17	56	44	88
3	18	51	41	75
3	19	45	37	64
3	20	40	34	55
3	21	39	33	54
3	22	36	30	47
3	23	33	28	42
4	0	31	27	40
4	1	31	27	40
4	2	30	27	35
4	3	30	28	35
4	4	30	27	35
4	5	31	27	36
4	6	37	31	50
4	7	38	32	53
4	8	38	32	52
4	9	40	33	55
4	10	42	34	60
4	11	43	35	63
4	12	44	35	64
4	13	45	36	66
4	14	49	38	75
4	15	49	39	76
4	16	51	40	81
4	17	56	43	85
4	18	49	39	72
4	19	45	37	63
4	20	41	34	57
4	21	39	33	53
4	22	36	31	49
4	23	33	28	43
5	0	31	27	39
5	1	31	27	38
5	2	30	28	36
5	3	30	27	35
5	4	30	27	35
5	5	31	27	36
5	6	36	29	50
5	7	37	31	51
5	8	38	32	53

5	9	38	32	54
5	10	42	34	60
5	11	45	35	64
5	12	38	31	53
5	13	44	36	63
5	14	51	40	79
5	15	52	41	82
5	16	54	42	86
5	17	58	45	91
5	18	54	42	80
5	19	50	40	73
5	20	45	37	66
5	21	44	36	63
5	22	41	33	56
5	23	35	29	46
6	0	32	28	41
6	1	32	27	40
6	2	31	27	39
6	3	30	27	37
6	4	31	27	38
6	5	31	27	40
6	6	33	28	42
6	7	34	29	46
6	8	37	31	50
6	9	39	32	53
6	10	41	34	59
6	11	44	37	64
6	12	46	37	71
6	13	51	39	78
6	14	53	41	82
6	15	51	41	76
6	16	49	39	72
6	17	49	39	73
6	18	48	39	70
6	19	50	40	72
6	20	53	43	79
6	21	53	42	77
6	22	48	39	68
6	23	40	33	55
7	0	33	28	44
7	1	32	27	40
7	2	31	27	38

7	3	31	28	38
7	4	30	27	37
7	5	32	28	37
7	6	32	28	42
7	7	36	30	47
7	8	39	32	55
7	9	42	34	59
7	10	46	37	69
7	11	50	39	76
7	12	53	41	82
7	13	56	43	85
7	14	54	42	79
7	15	47	39	68
7	16	46	38	67
7	17	48	39	68
7	18	46	38	67
7	19	46	38	65
7	20	44	37	62
7	21	42	35	58
7	22	37	31	49
7	23	33	28	42

Tabel D.3: Data sampel 2 pada tanggal 15 Mei 2017 - 21 Mei 2017

1	0	31	27	40
1	1	31	27	37
1	2	30	28	35
1	3	30	28	34
1	4	29	26	34
1	5	31	27	37
1	6	37	30	50
1	7	38	32	51
1	8	37	31	50
1	9	38	31	52
1	10	41	33	59
1	11	43	35	61
1	12	42	34	60
1	13	42	34	60
1	14	45	36	67
1	15	47	37	68
1	16	50	38	78
1	17	54	42	82

1	18	49	39	72
1	19	42	35	61
1	20	39	32	54
1	21	37	31	49
1	22	35	29	47
1	23	33	28	43
2	0	31	27	39
2	1	30	27	37
2	2	30	27	35
2	3	30	27	34
2	4	30	27	35
2	5	31	27	36
2	6	37	30	50
2	7	37	31	51
2	8	37	31	52
2	9	38	31	52
2	10	40	33	56
2	11	41	34	58
2	12	42	34	58
2	13	43	34	61
2	14	46	37	68
2	15	47	37	69
2	16	50	39	78
2	17	54	42	84
2	18	49	40	72
2	19	45	36	64
2	20	40	33	56
2	21	38	32	51
2	22	35	30	47
2	23	33	28	43
3	0	31	27	39
3	1	31	28	37
3	2	31	28	35
3	3	30	27	34
3	4	30	27	35
3	5	31	27	37
3	6	37	30	51
3	7	38	31	51
3	8	38	32	54
3	9	39	32	54
3	10	41	34	60
3	11	43	34	61



3	12	41	34	60
3	13	44	36	64
3	14	48	39	75
3	15	51	40	76
3	16	53	42	83
3	17	56	44	88
3	18	51	41	75
3	19	45	37	64
3	20	40	34	55
3	21	39	33	54
3	22	36	30	47
3	23	33	28	42
4	0	31	27	40
4	1	31	27	40
4	2	30	27	35
4	3	30	28	35
4	4	30	27	35
4	5	31	27	36
4	6	37	31	50
4	7	38	32	53
4	8	38	32	52
4	9	40	33	55
4	10	42	34	60
4	11	43	35	63
4	12	44	35	64
4	13	45	36	66
4	14	49	38	75
4	15	49	39	76
4	16	51	40	81
4	17	56	43	85
4	18	49	39	72
4	19	45	37	63
4	20	41	34	57
4	21	39	33	53
4	22	36	31	49
4	23	33	28	43
5	0	31	27	39
5	1	31	27	38
5	2	30	28	36
5	3	30	27	35
5	4	30	27	35
5	5	31	27	36

5	6	36	29	50
5	7	37	31	51
5	8	38	32	53
5	9	38	32	54
5	10	42	34	60
5	11	45	35	64
5	12	38	31	53
5	13	44	36	63
5	14	51	40	79
5	15	52	41	82
5	16	54	42	86
5	17	58	45	91
5	18	54	42	80
5	19	50	40	73
5	20	45	37	66
5	21	44	36	63
5	22	41	33	56
5	23	35	29	46
6	0	32	28	41
6	1	32	27	40
6	2	31	27	39
6	3	30	27	37
6	4	31	27	38
6	5	31	27	40
6	6	33	28	42
6	7	34	29	46
6	8	37	31	50
6	9	39	32	53
6	10	41	34	59
6	11	44	37	64
6	12	46	37	71
6	13	51	39	78
6	14	53	41	82
6	15	51	41	76
6	16	49	39	72
6	17	49	39	73
6	18	48	39	70
6	19	50	40	72
6	20	53	43	79
6	21	53	42	77
6	22	48	39	68
6	23	40	33	55

---

---

7	0	33	28	44
7	1	32	27	40
7	2	31	27	38
7	3	31	28	38
7	4	30	27	37
7	5	32	28	37
7	6	32	28	42
7	7	36	30	47
7	8	39	32	55
7	9	42	34	59
7	10	46	37	69
7	11	50	39	76
7	12	53	41	82
7	13	56	43	85
7	14	54	42	79
7	15	47	39	68
7	16	46	38	67
7	17	48	39	68
7	18	46	38	67
7	19	46	38	65
7	20	44	37	62
7	21	42	35	58
7	22	37	31	49
7	23	33	28	42

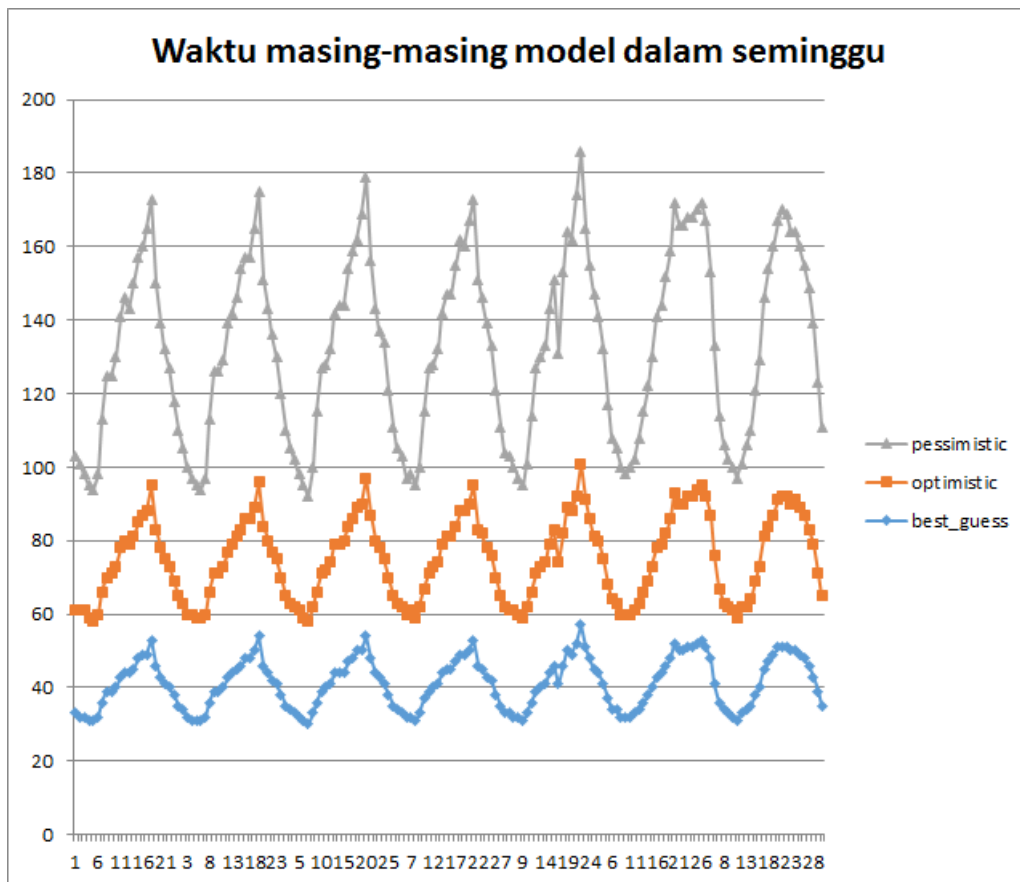
Tabel D.4: Data sampel 2 pada tanggal 8 Mei 2017 - 14 Mei 2017

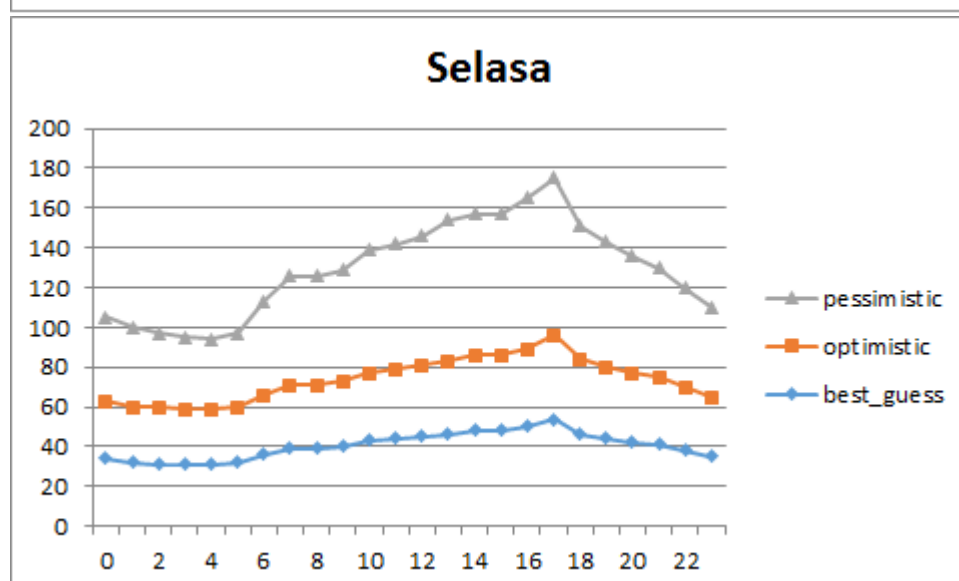
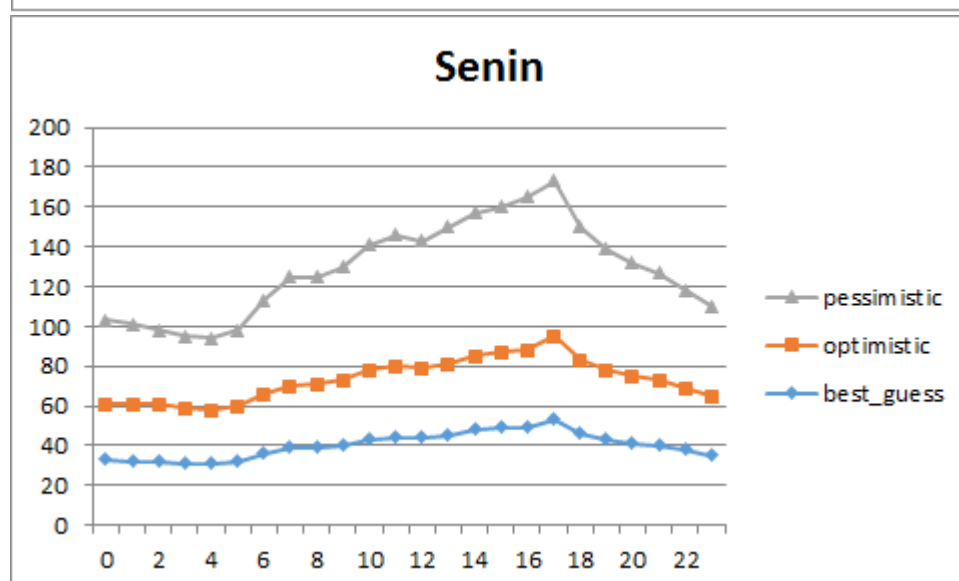
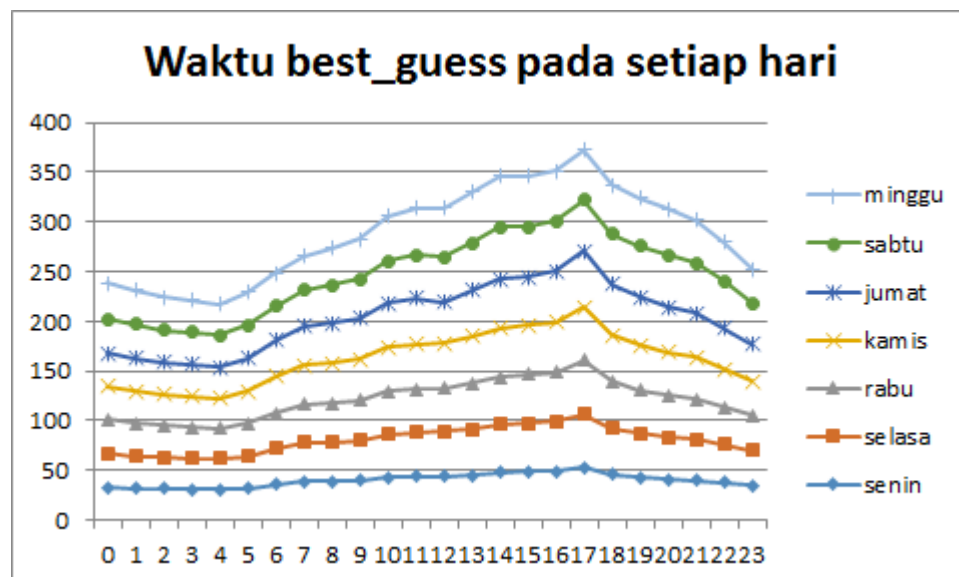


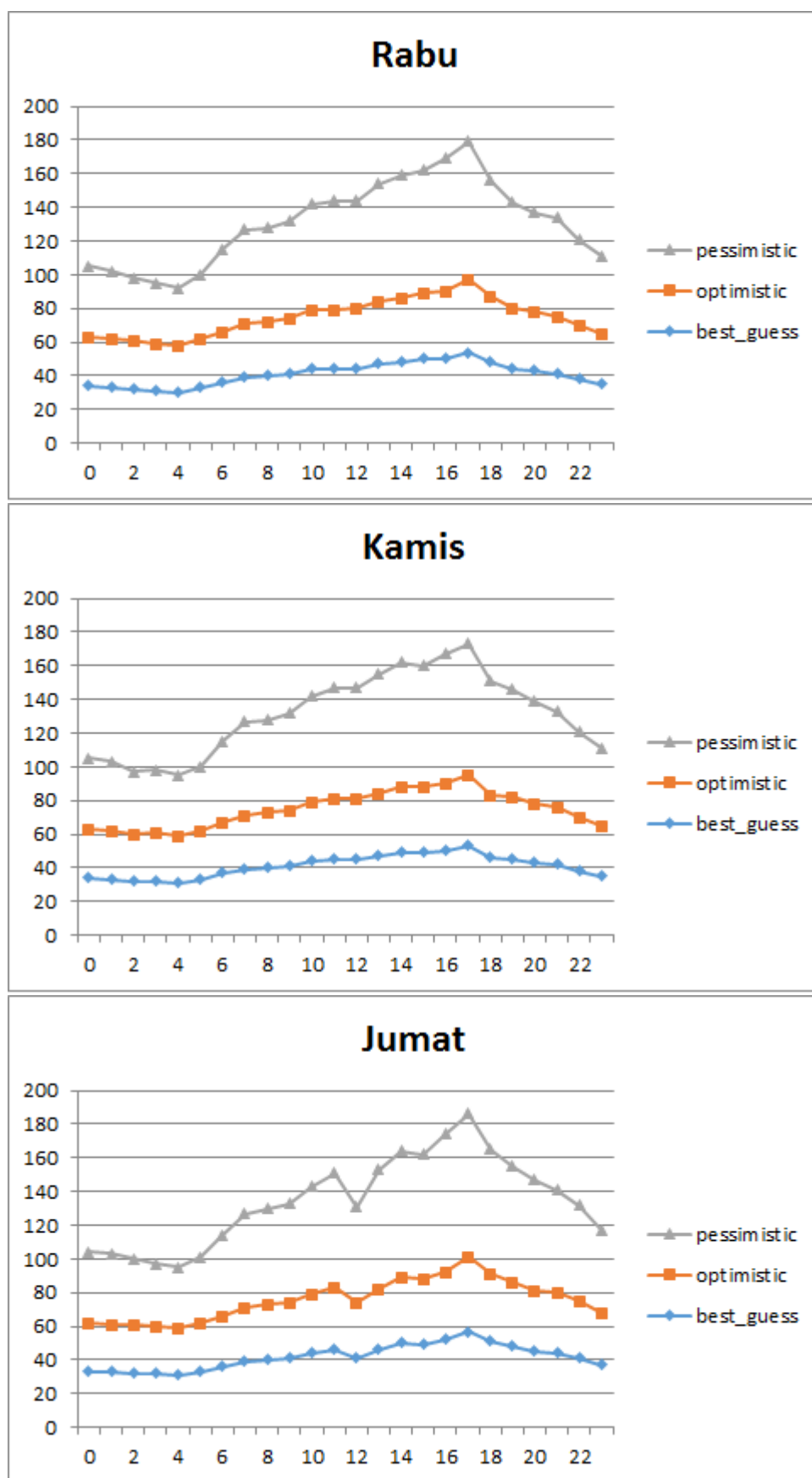
## LAMPIRAN E

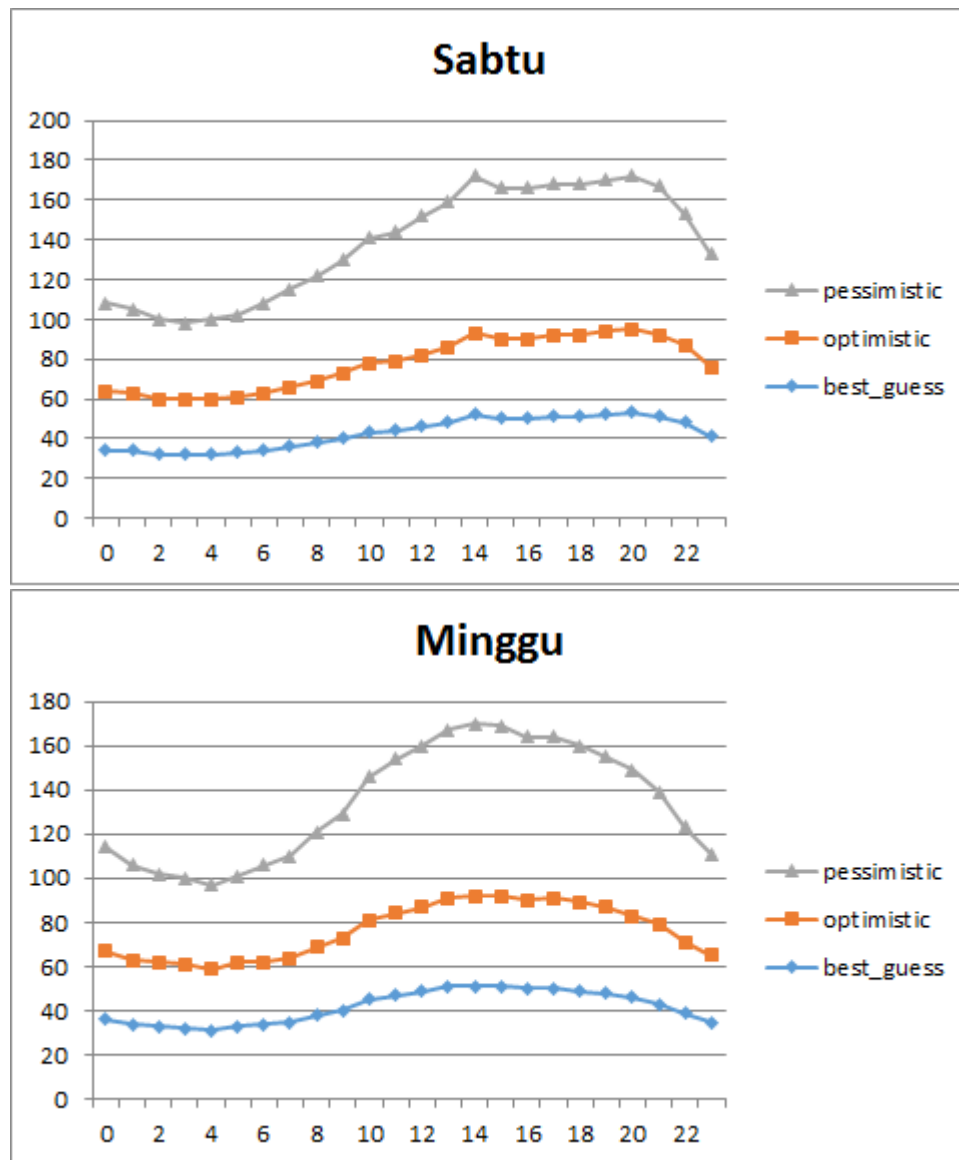
### HASIL PENGUJIAN EKSPERIMENTAL

- Hasil pengujian eksperimental ini berupa bagan yang dibuat dari data pada Lampiran D.  
Berikut adalah bagan hasil dari pengujian eksperimental :



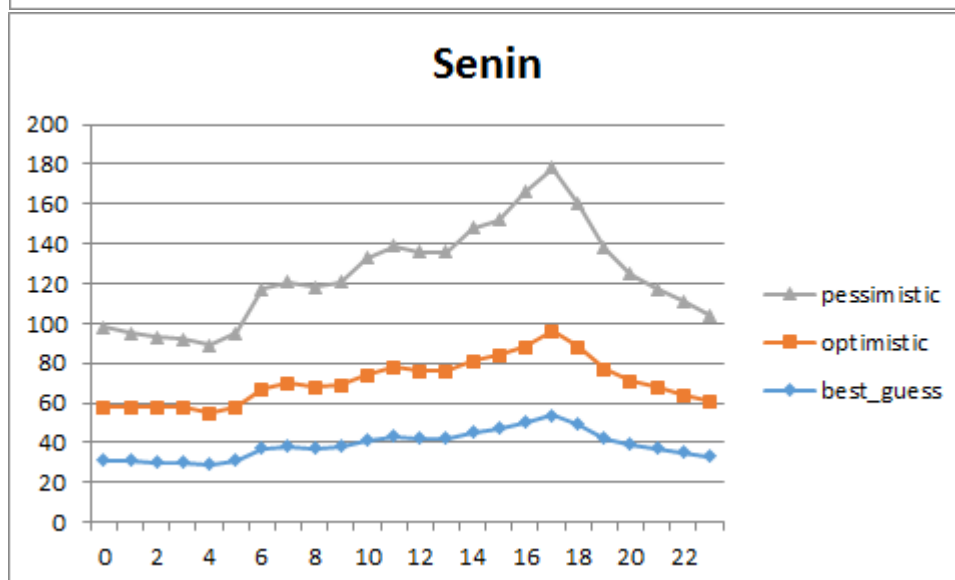
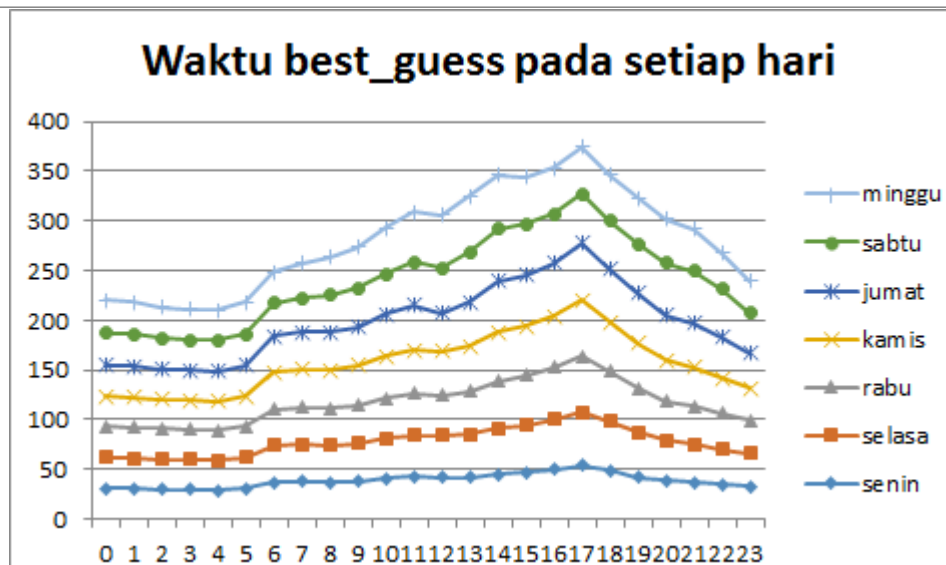
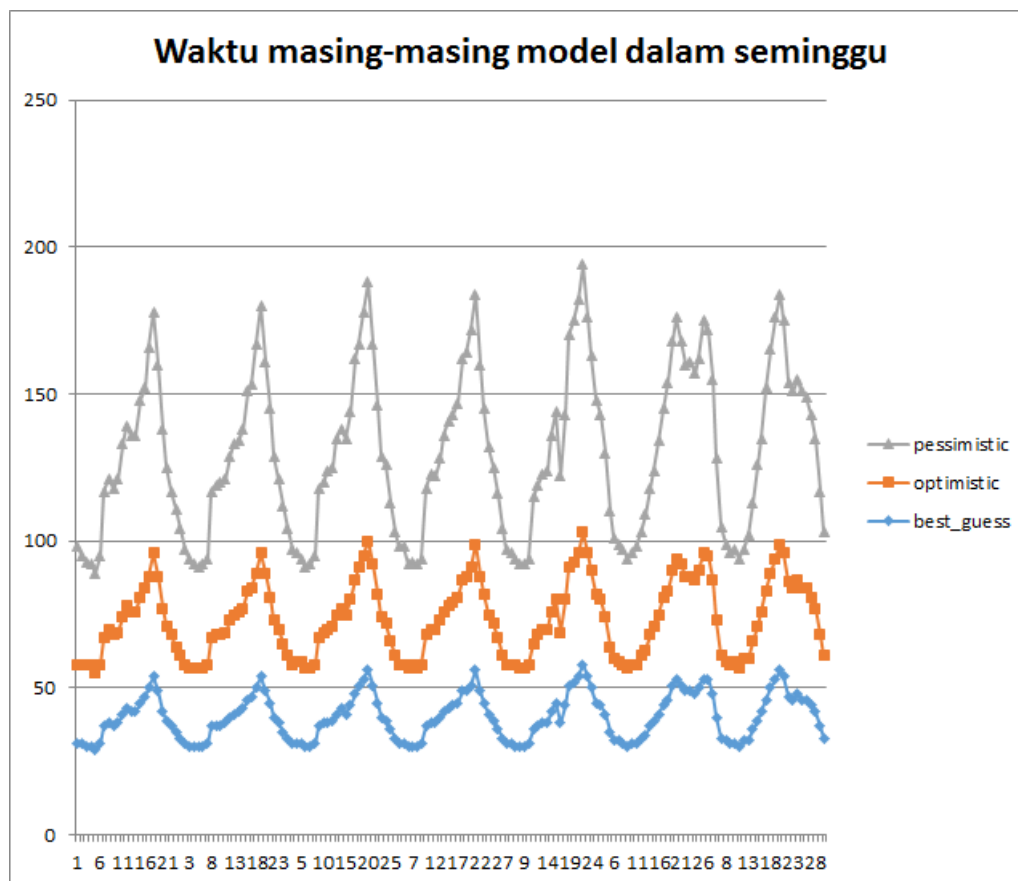


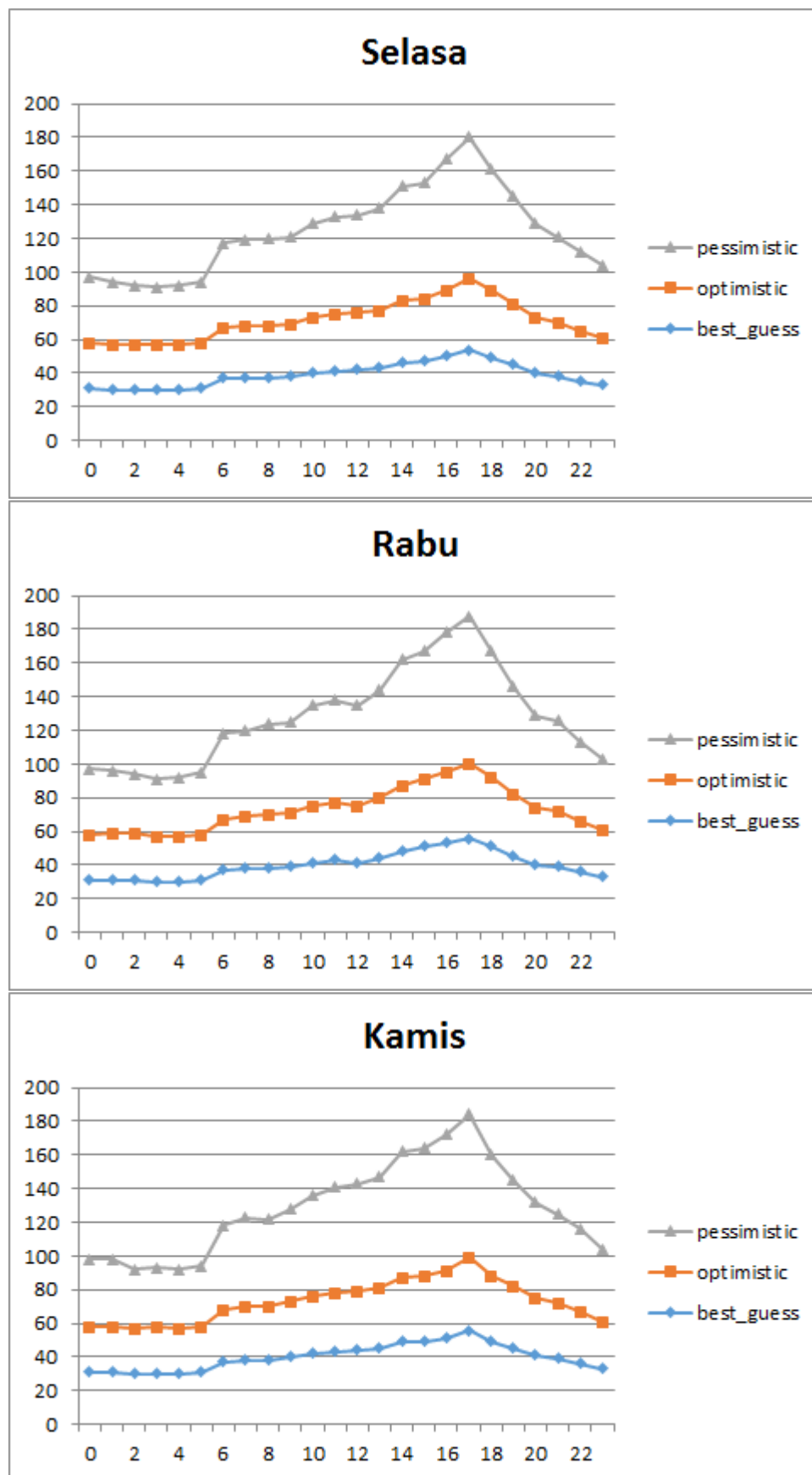


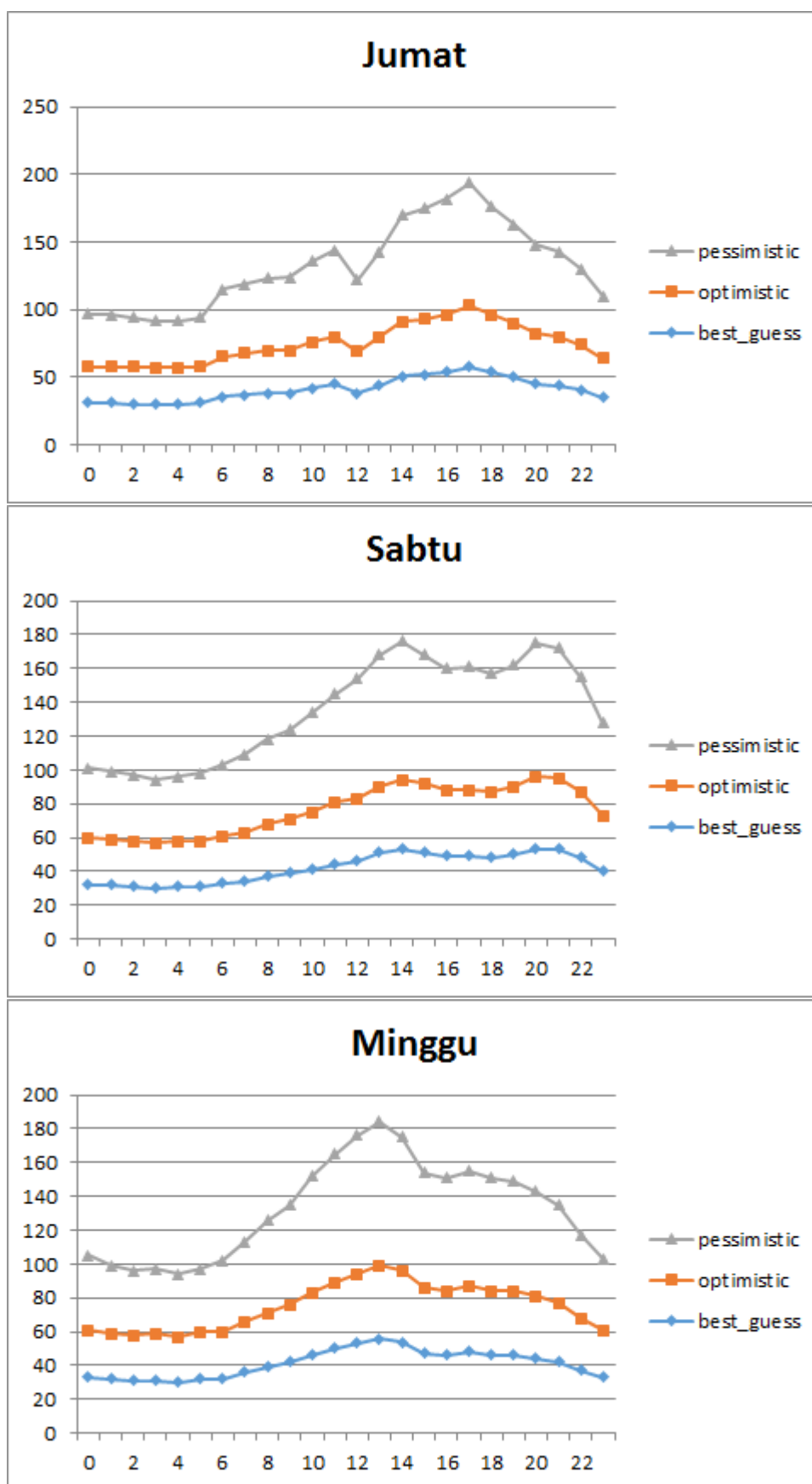


Gambar E.1: Hasil Pengujian Eksperimental sampel 1 08 Mei 2017

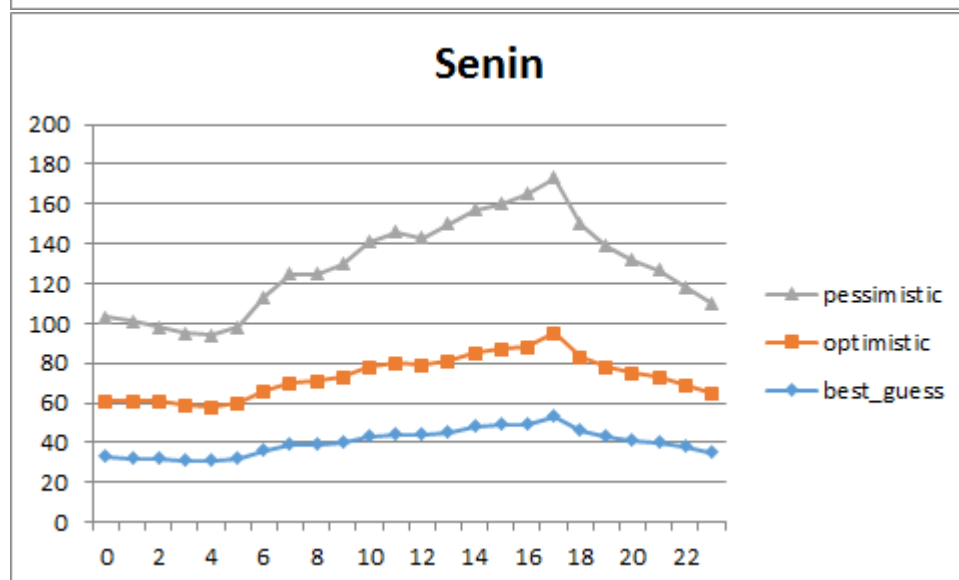
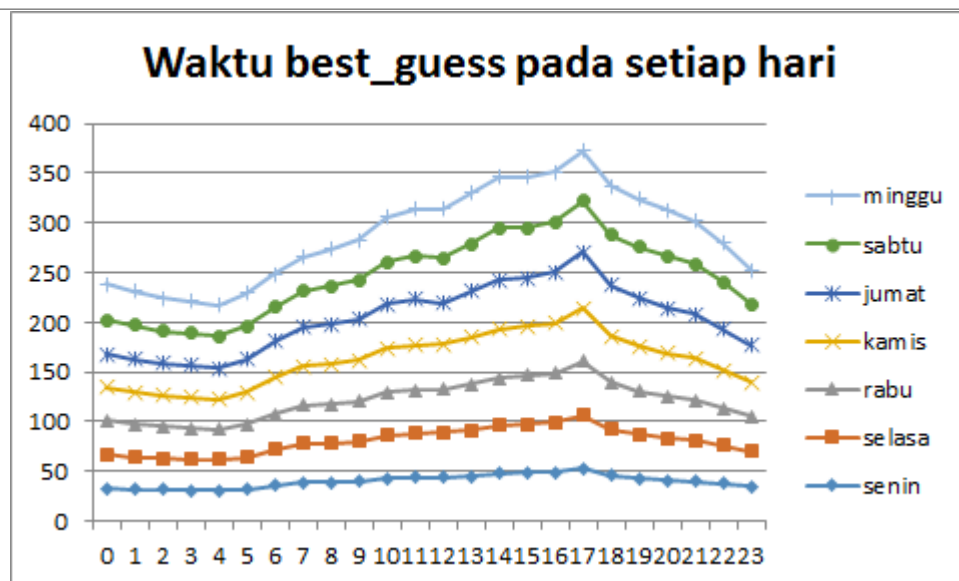
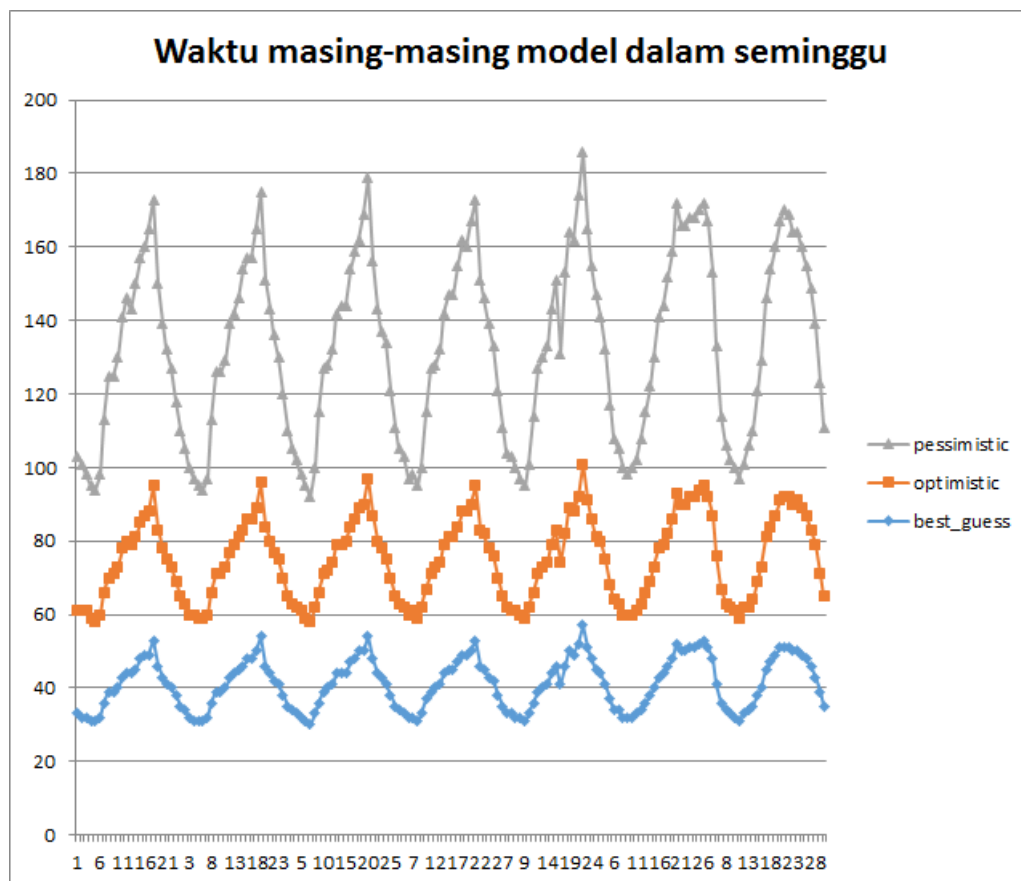




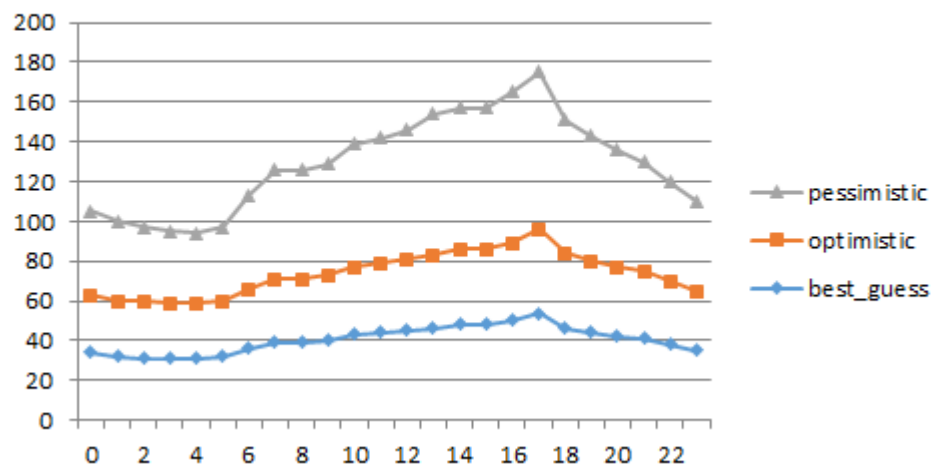




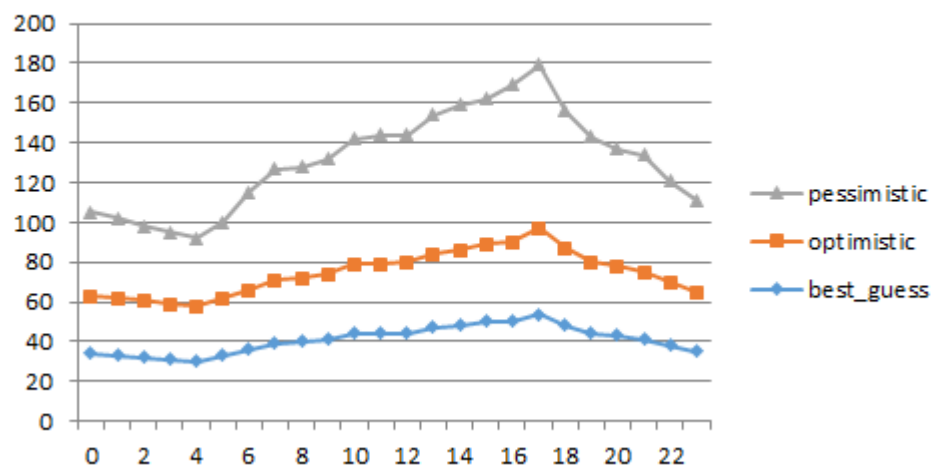
Gambar E.2: Hasil Pengujian Eksperimental sampel 2 15 Mei 2017



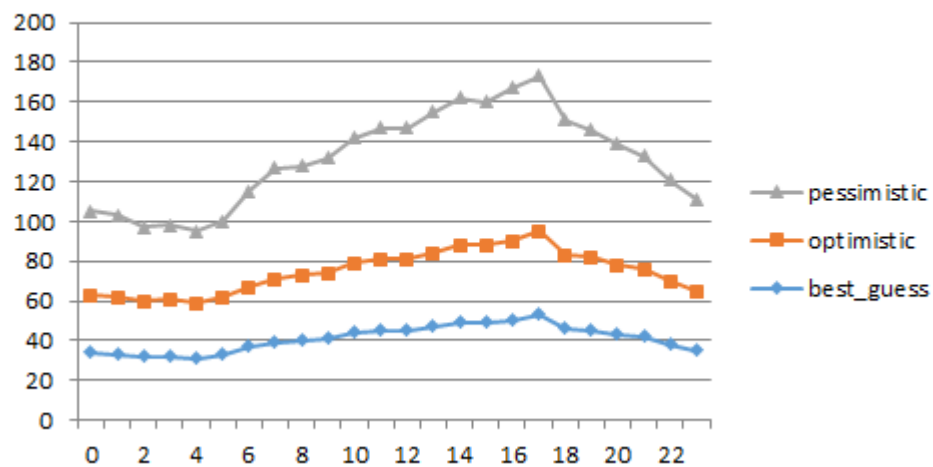
### Selasa

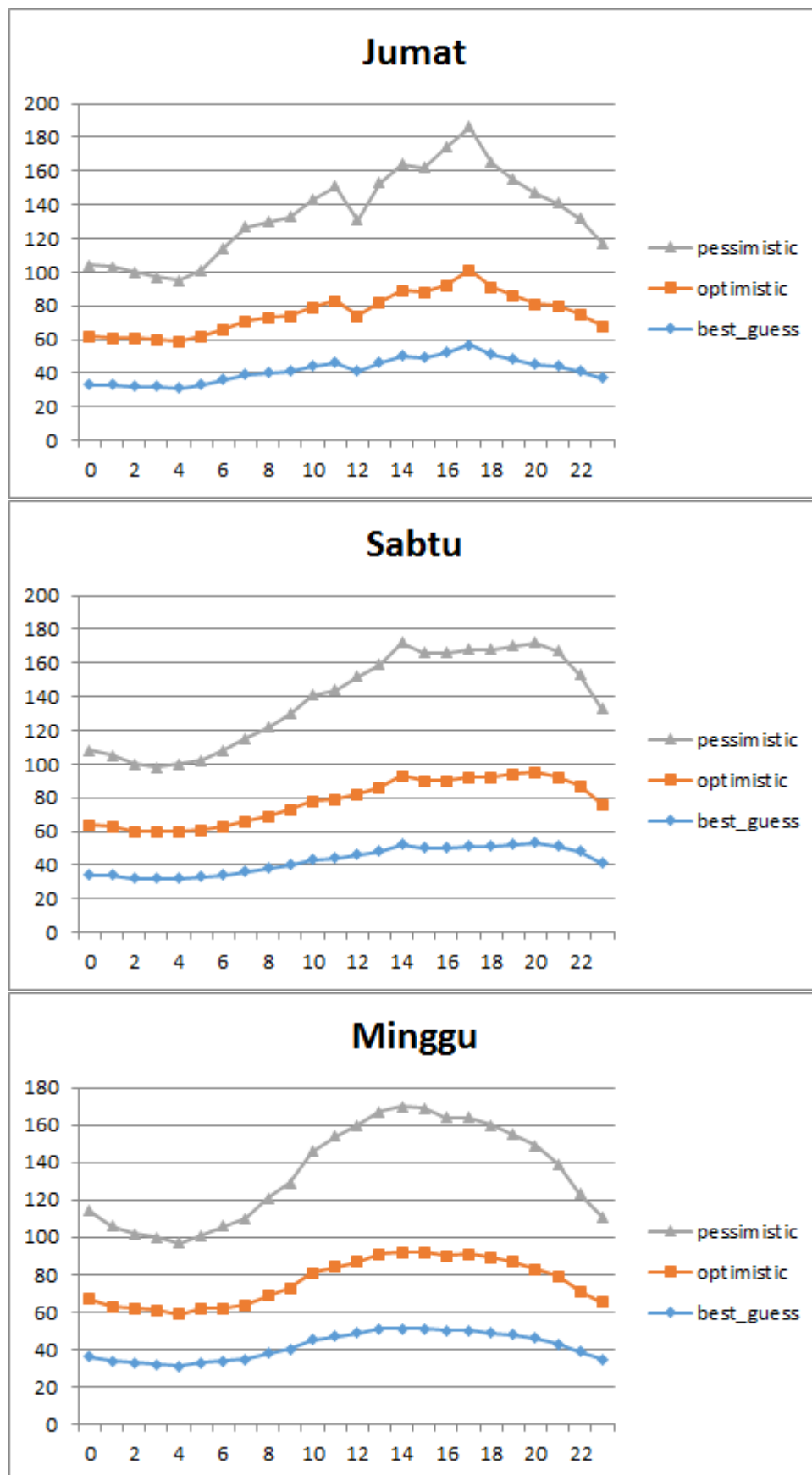


### Rabu



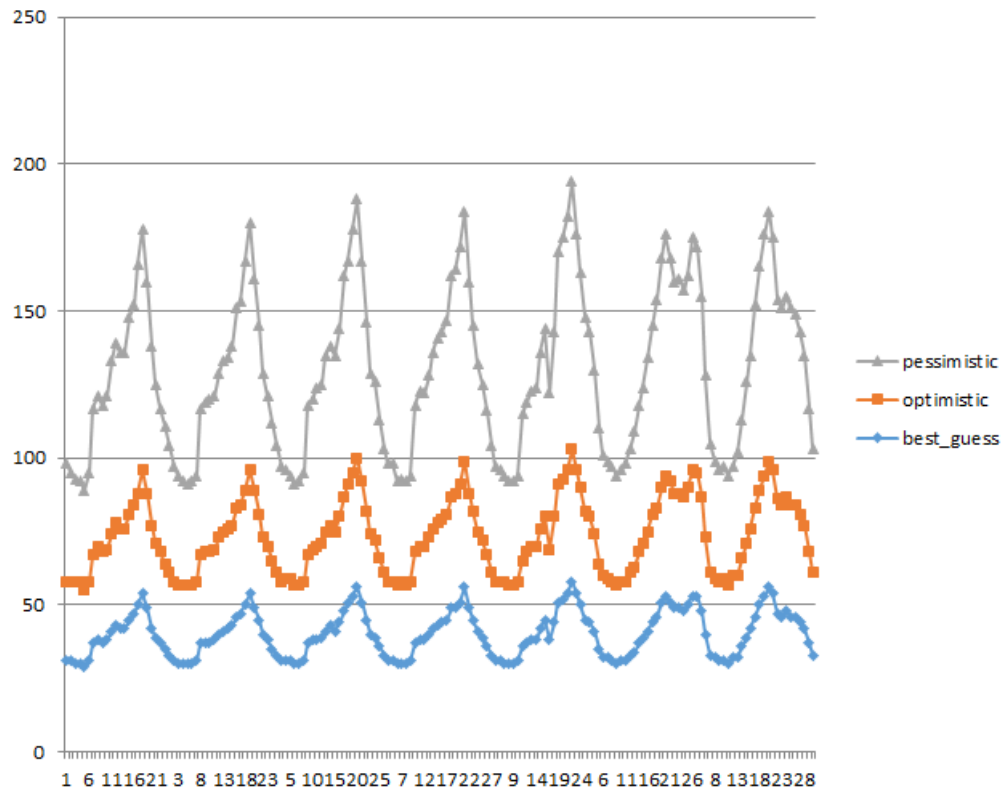
### Kamis



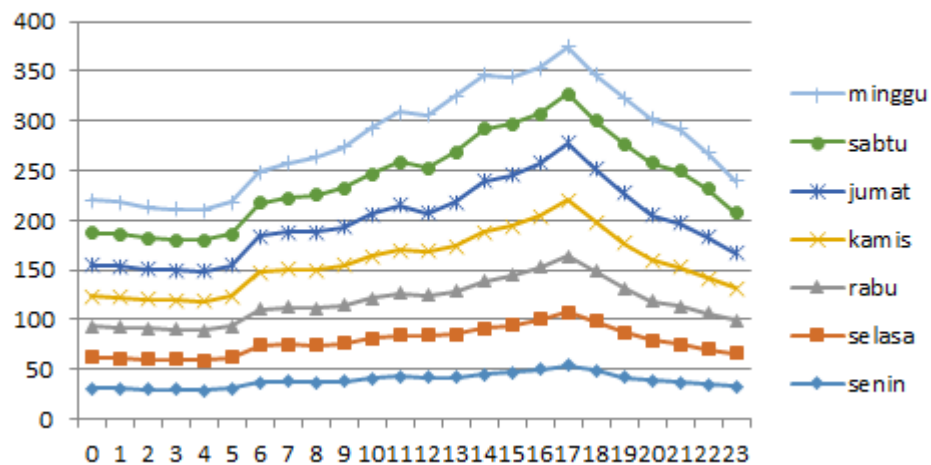


Gambar E.3: Hasil Pengujian Eksperimental sampel 1 08 Mei 2017

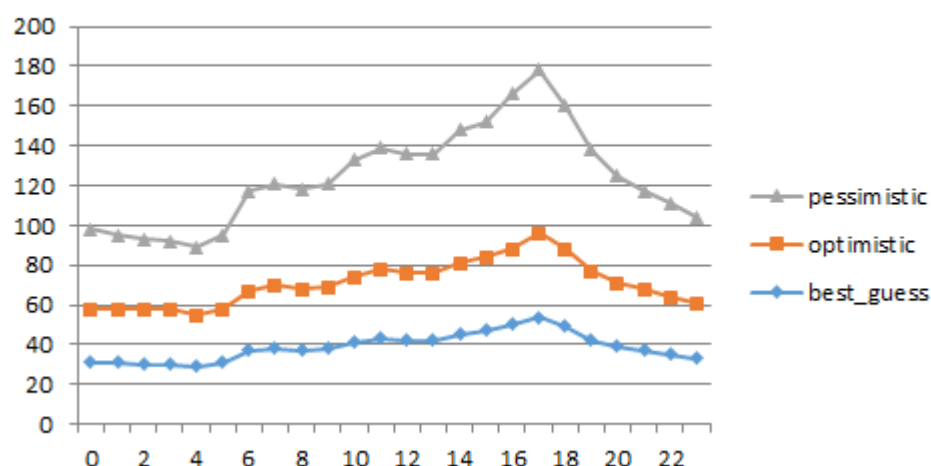
### Waktu masing-masing model dalam seminggu

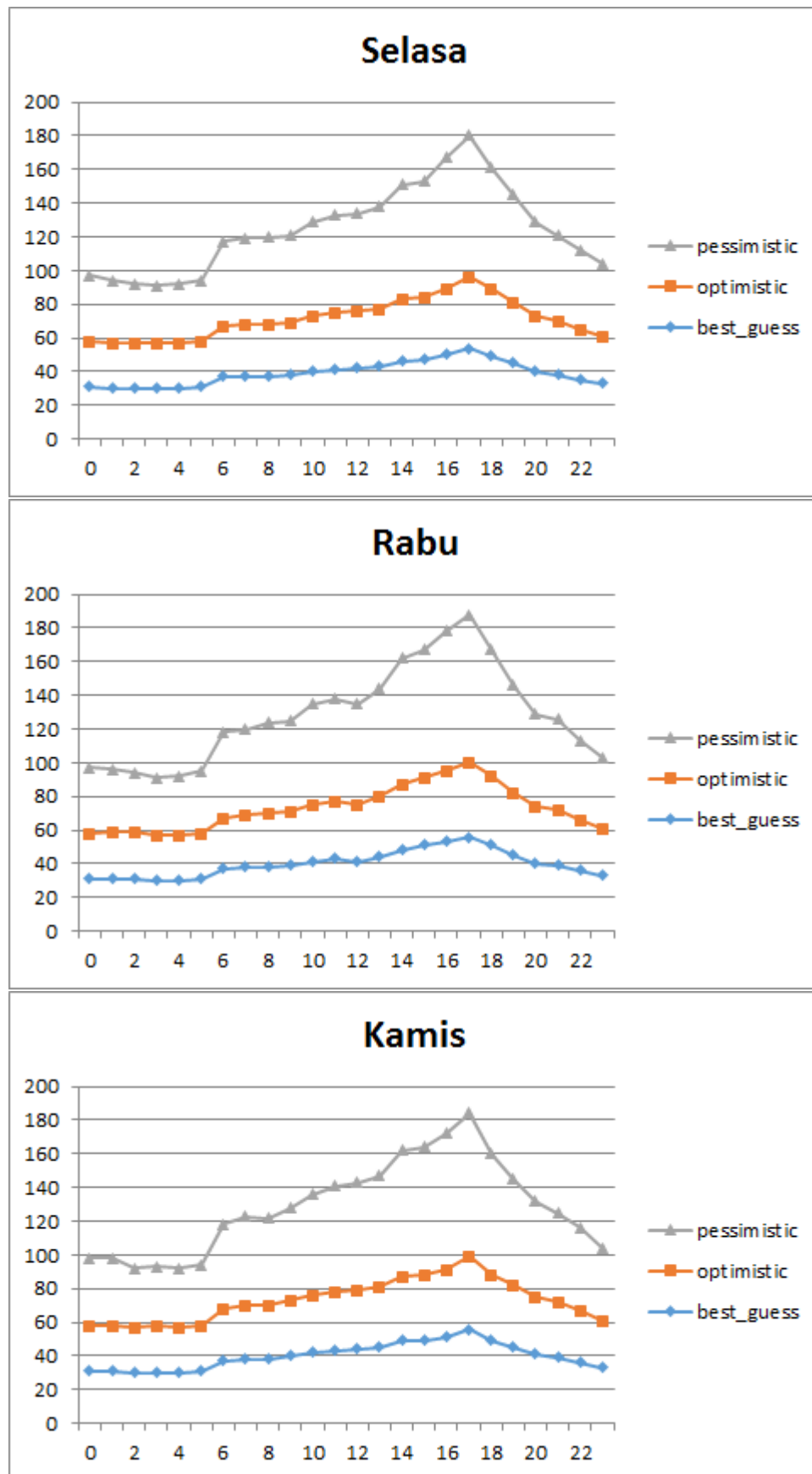


### Waktu best\_guess pada setiap hari

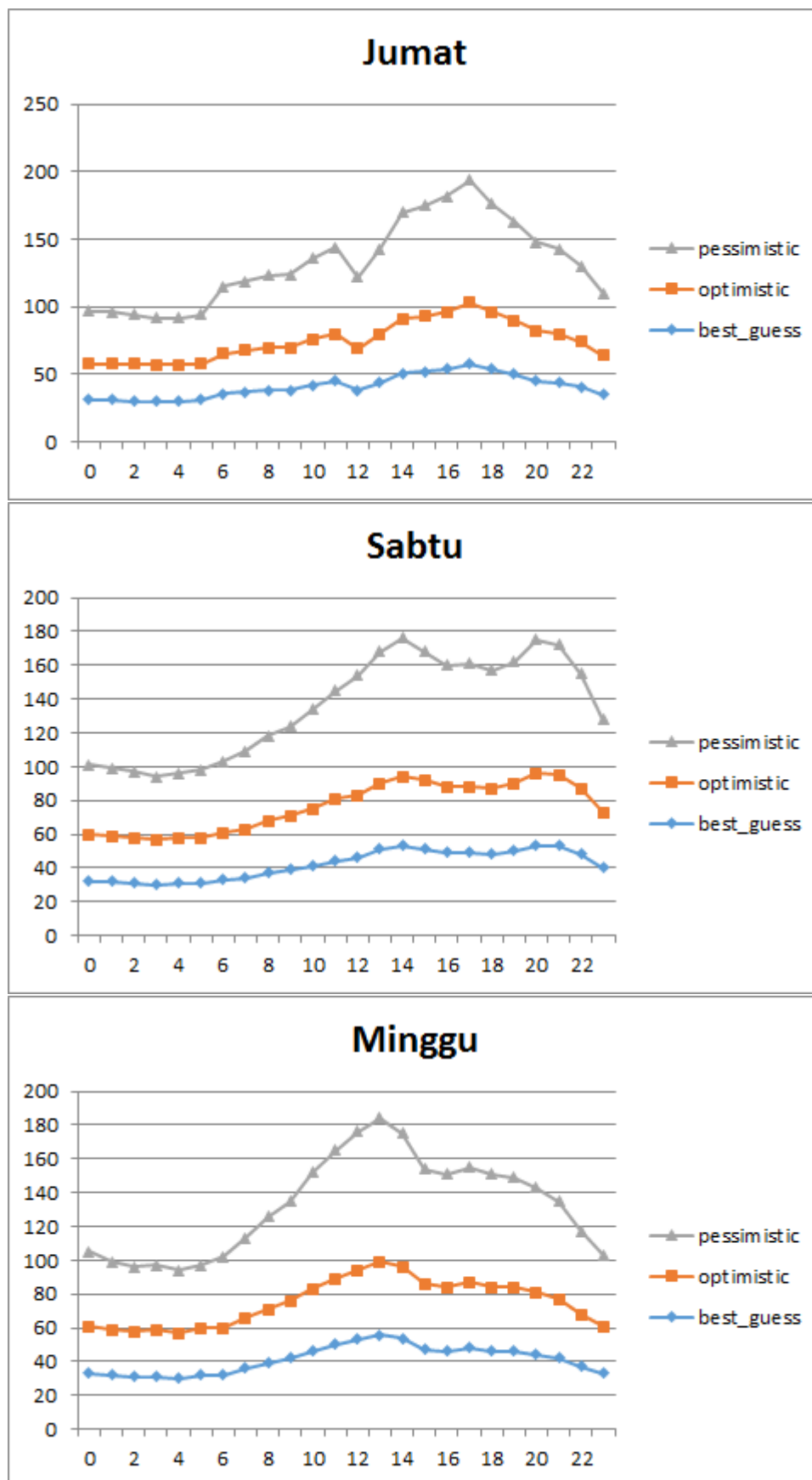


### Senin









Gambar E.4: Hasil Pengujian Eksperimental sampel 2 15 Mei 2017