BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Tahapan implementasi adalah tahapan dimana perancangan yang dibuat pada BAB III diimplementasikan menggunakan bahasa pemrograman tertentu, dalam penelitian tugas akhir ini, digunakan bahasa pemrograman JavaScript. Pada tahap ini, dipaparkan seluruh elemen-elemen yang terlibat pada pengembangan aplikasi *mobile* visualisasi data terbuka Pemprov DKI Jakarta.

4.1.1 Lingkungan Implementasi

Lingkungan implementasi merupakan spesfikasi perangkat keras dan perangkat lunak baik *client* ataupun *server* dimana aplikasi *mobile* visualisasi data akan diimplementasikan. Berikut adalah lingkungan implementasi aplikasi *mobile* visualisasi data terbuka Pemprov DKI Jakarta.

1. Lingkungan Perangkat Keras

Berikut adalah spesifikasi perangkat keras *client* yang diimplementasikan untuk menjalankan aplikasi, yang dapat dilihat pada Tabel IV-1.

Tabel IV-1 Lingkungan Perangkat Keras *Client*

Perangkat Keras	Spesifikasi Android	Spesifikasi iPhone
Prosesor	Quad-core 1.3 GHz	Dual-core 1.4 GHz
Memori	1 GB RAM	1 GB RAM
Resolusi Layar	540 x 960 pixels	750 x 1334 pixels
Disk	8 GB	16 GB
Internet	4G	4G

Berikut adalah spesifikasi perangkat keras *server* (teknologi awan *Infrastructure as a Service* dan *Database as a Service*) yang diimplementasikan untuk mendukung perangkat *client* berjalan, yang dapat dilihat pada Tabel IV-2.

Tabel IV-2 Lingkungan Perangkat Keras IaaS

Perangkat Keras	Spesifikasi
Prosesor	Intel(R) Xeon(R) CPU E5-2630L v2 @ 2.40GHz
Memori	1024 MB
Disk	40 GB SSD

Perangkat Keras	Spesifikasi
Uplink/*Downlink*	1 Gbps
Data Center	DigitalOcean (*Singapore Node*)

Tabel IV-3 Lingkungan Perangkar Keras DaaS

Perangkat Keras	Spesifikasi
Prosesor	*Shared*
Memori	512 MB
Disk	512 MB
Data Center	AWS us-east-1

2. Lingkungan Perangkat Lunak

Berikut adalah spesifikasi perangkat lunak *mobile* yang diimplementasikan untuk menjalankan aplikasi, yang dapat dilihat pada Tabel IV-4.

Tabel IV-4 Lingkungan Perangkat Lunak *Mobile*

Perangkat Lunak	Spesifikasi Android	Spesifikasi iPhone
Sistem Operasi	Android OS v5.1.1	iOS v8

Berikut adalah spesifikasi perangkat lunak *server* yang diimplementasikan untuk mendukung aplikasi *client*, yang dapat dilihat pada Tabel IV-5.

Tabel IV-5 Lingkungan Perangkat Lunak *Server*

Perangkat Lunak	Spesifikasi
Sistem Operasi	Ubuntu 16.04 LTS 64 bit
Task Runner	gulp 3.9.x
Server Proxy	nginx v1.8.x
Software Platform	NodeJS v6.0.0
Database	MongoDB 3.x
Database Driver	Mongoose 4.5.x
Framework	Express 4.14.x
Task Runner	Gulp 3.9.x, gulp-nodemon 2.1.x
Auth Handler	Passport 0.3.x, passport-http-barer 1.0.x

4.1.2 Implementasi Data

Implementasi data yang dilakukan adalah menggunakan MongoDB. Pembuatan *collection* dilakukan melalui *driver* mongoose nodejs.

1. Pembuatan Database

```
1   use dbbatavia
```

2. Implementasi Data 1: Ikhtisar Statistik antar Kerja DKI Jakarta

```
var mongoose = require('mongoose'),
  Schema = mongoose.Schema;

var dataModel = new Schema({
  tahun: Date,
  rincian: String,
  jumlah: Number,
  kategori: String,
  persen: Number
});
var data = mongoose.model('Ikhtisar', dataModel);

module.exports = {
  data: data
}
```

3. Implementasi Data 2: Besar Upan Minimum DKI Jakarta

```
var mongoose = require('mongoose'),
  Schema = mongoose.Schema;

var dataModel = new Schema({
  tahun: Date,
  ump: Number,
  kenaikan_ump: Number,
  inflasi: Number,
  keterangan: String
});
var data = mongoose.model('DataUmpInflasi', dataModel);

var linearModel = new Schema({
  model: {
    coefficient: Number,
    slope: Number,
    correlation_coefficient: Number
  },
  data: [{
    type: Schema.ObjectId,
    ref: 'DataUmpInflasi'
  }]
});
var linear = mongoose.model('LinearUmpInflasi', linearModel);

module.exports = {
  data: data,
  linear: linear
}
```

4. Implementasi Data 3: Laju Pertumbuhan Ekonomi Jakarta dan Nasional

```
var mongoose = require('mongoose'),
  Schema = mongoose.Schema;

```

```
var dataModel = new Schema({
  tahun: Date,
  persen_tumbuh_jakarta: Number,
  persen_tumbuh_nasional: Number,
  keterangan: String

});
var data = mongoose.model('DataPertumbuhanEkonomi', dataModel);

var linearModel = new Schema({
  model: {
    coefficient: Number,
    slope: Number,
    correlation_coefficient: Number
  },
  data: [{
    type: Schema.ObjectId,
    ref: 'DataPertumbuhanEkonomi'
  }]
});
var linear = mongoose.model('LinearPertumbuhanEkonomi',
linearModel);

module.exports = {
  data: data,
  linear: linear
}
```

5. Implementasi Data 4: Tingkat Inflasi Jakarta dan Nasional

```
var mongoose = require('mongoose'),
  Schema = mongoose.Schema;

var dataModel = new Schema({
  tahun: Date,
  inflasi_jakarta: Number,
  inflasi_nasional: Number,
  keterangan: String

});
var data = mongoose.model('DataInflasi', dataModel);

var linearModel = new Schema({
  model: {
    coefficient: Number,
    slope: Number,
    correlation_coefficient: Number
  },
  data: [{
    type: Schema.ObjectId,
    ref: 'DataInflasi'
  }]
});
var linear = mongoose.model('LinearInflasi', linearModel);
```

```
25
26  module.exports = {
27    data: data,
28    linear: linear
29  }
```

6. Implementasi Data 5: Komponen Inflasi Jakarta

```
1   var mongoose = require('mongoose'),
2     Schema = mongoose.Schema;
3
4   var dataModel = new Schema({
5     tahun: Date,
6     komponen: String,
7     persen_inflasi: Number,
8     persen_kontribusi: Number
9   });
10  var data = mongoose.model('KomponenInflasi', dataModel);
11
12  module.exports = {
13    data: data
14  }
```

7. Implementasi Data 6: Ekspor Impor DKI Jakarta

```
1   var mongoose = require('mongoose'),
2     Schema = mongoose.Schema;
3
4   var dataModel = new Schema({
5     tahun: Date,
6     atribut: String,
7     juta_usd: Number
8   });
9   var data = mongoose.model('BesarEkprImpr', dataModel);
10
11  module.exports = {
12    data: data
13  }
```

8. Implementasi Data 7: Volume dan Nilai Ekspor DKI Jakarta

```
1   var mongoose = require('mongoose'),
2     Schema = mongoose.Schema;
3
4   var dataModel = new Schema({
5     tahun: Date,
6     no_hs: String,
7     komoditas: String,
8     volume: Number,
9     nilai: Number,
10    grup: String
11  });
12  var data = mongoose.model('VolumeNilaiEkspor', dataModel);
```

```
13
14  module.exports = {
15    data: data
16  }
```

9. Implementasi Data 8: Nilai Impor Produk Menurut Golongan

```
1   var mongoose = require('mongoose'),
2     Schema = mongoose.Schema;
3
4   var dataModel = new Schema({
5     bulan: Date,
6     golongan: String,
7     nilai_fob: Number
8   });
9   var data = mongoose.model('DataNilaiImprProgukGol', dataModel);
10
11  var histogramModel = new Schema({
12    kelas_interval: String,
13    frekuensi: Number,
14    frekuensi_relatif: Number
15  });
16  var histogram = mongoose.model('HistogramNilaiImprProdukGol',
    histogramModel);
17
18  module.exports = {
19    data: data,
20    histogram: histogram
21  }
```

10. Implementasi Data 9: Struktur Ekonomi Jakarta Berdasarkan Sektor

```
1   var mongoose = require('mongoose'),
2     Schema = mongoose.Schema;
3
4   var dataModel = new Schema({
5     tahun: Date,
6     persen_kontribusi: Number,
7     jenis_sektor: String,
8     keterangan: String
9   });
10  var data = mongoose.model('StrukturEkonomi', dataModel);
11
12  module.exports = {
13    data: data
14  }
```

11. Implementasi Data 10: Harga Pangan Tingkat Konsumen DKI Jakarta

```
1   var mongoose = require('mongoose'),
2     Schema = mongoose.Schema;
3
4   var dataModel = new Schema({
```

```
  tanggal: Date,
  wilayah: String,
  komoditi: String,
  harga: Number,
  satuan: String
});
var data = mongoose.model('DataHargaPanganKonsumen', dataModel);

var histogramModel = new Schema({
  kelas_interval: String,
  frekuensi: Number,
  frekuensi_relatif: Number
});
var histogram = mongoose.model('HistogramHargaPanganKonsumen',
histogramModel);

module.exports = {
  data: data,
  histogram: histogram
}
```

12. Implementasi Data 11: Perkembangan Harga Grosir di Pasar Induk

```
var mongoose = require('mongoose'),
  Schema = mongoose.Schema;

var dataModel = new Schema({
  tanggal: Date,
  komoditas: String,
  harga: Number
});
var data = mongoose.model('DataPerkemHargaGrosir', dataModel);

var histogramModel = new Schema({
  kelas_interval: String,
  frekuensi: Number,
  frekuensi_relatif: Number
});
var histogram = mongoose.model('HistogramPerkemHargaGrosir',
histogramModel);

module.exports = {
  data: data,
  histogram: histogram
}
```

13. Implementasi Data 12: Pendapatan Perkapita Jakarta dan Nasional

```
var mongoose = require('mongoose'),
  Schema = mongoose.Schema;

var dataModel = new Schema({
  tahun: Date,
```

```
6     perkapita_jakarta: Number,
7     perkapita_nasional: Number,
8     keterangan: String
9
10  });
11  var data = mongoose.model('DataPendapatanPerkapita', dataModel);
12
13  var linearModel = new Schema({
14    model: {
15      coefficient: Number,
16      slope: Number,
17      correlation_coefficient: Number
18    },
19    data: [{
20      type: Schema.ObjectId,
21      ref: 'DataPendapatanPerkapita'
22    }]
23  });
24  var linear = mongoose.model('LinearPendapatanPerkapita',
25  linearModel);
26
27  module.exports = {
28    data: data,
29    linear: linear
30  }
```

4.1.3 Implementasi Antarmuka

Bagian ini merupakan implementasi antarmuka dari subsistem aplikasi *mobile* visualisasi data terbuka Pemprov DKI Jakarta. Implementasi ini terdiri dari nama antarmukan dan nama file yang mewakilinya. Adapun rincian implementasi antarmuka dapat dilihat pada lampiran. Implementasi antarmuka ini dapat dilihat pada Tabel IV-6.

Tabel IV-6 Implementasi Antarmuka

No.	Nama Antarmuka	Nama Berkas	No. Lampiran
1	Perkenalan Aplikasi	intro.html	A-1
2	Beranda	main.html	A-2
3	Halaman Kredit	credit.html	A-3
4	Menu Ekonomi Makro	makro-menu.html	A-4
5	Menu Ekonomi Mikro	mikro-menu.html	A-5
6	Halaman Visualisasi Ikhtisar Statistik	ikhtisar-statistik-tab.html	A-6
7	Halaman Visualisasi UMP	upah-minimum-provinsi-tab.html	A-7
8	Halaman Visualisasi Pertumbuhan Ekonomi	pertumbuhan-ekonomi-tab.html	A-8
9	Halaman Visualisasi Tingkat Inflasi	tingkat-inflasi-tab.html	A-9

No.	Nama Antarmuka	Nama Berkas	No. Lampiran
10	Halaman Visualisasi Komponen Inflasi	komponen-inflasi-tab.html	A-10
11	Halaman Visualisasi Struktur Ekonomi	struktur-ekonomi-pie.html	A-11
12	Halaman Visualisasi Pendapatan Perkapita	pendapatan-perkapita-tab.html	A-12
13	Halaman Visualisasi Ekspor dan Impor	ekspor-impor-line.html	A-13
14	Halaman Visualisasi Volume dan Nilai Ekspor	volume-nilai-ekspor-tab.html	A-14
15	Halaman Visualisasi Nilai Impor Produk	nilai-impor-produk-tab.html	A-15
16	Halaman Visualisasi Pangan Konsumen	pangan-konsumen-tab.html	A-16
17	Halaman Visualisasi Harga Grosiran	harga-grosiran-tab.html	A-17

4.2 Pengujian Sistem

Pengujian sistem merupakan tahapan untuk mencari tahu kekurangan-kekurangan pada aplikasi yang dibangun sehingga bisa memenuhi tujuan untuk selanjutnya disempurnakan dan di-*deploy* sebaik mungkin.

4.2.1 Rencana Pengujian

Rencana pengujian merupakan strategi dokumentasi yang digunakan untuk melakukan verifikasi dan memastikan bahwa aplikasi yang dibangun telah memenuhi desain dan tujuan yang dipaparkan pada bagian sebelumnya. Terdapat empat rencana pengujian yang dilakukan, yaitu rencana pengujian unit, pengujian integrasi, pengujian algoritma, dan pengujian *beta*.

1. Rencana Pengujian Unit

Pengujin unit yang akan dilakukan, ditujukan untuk kelas *controller* pada subsistem RESTful API yang terdiri dari tiga kelas. Dari setiap kelas tersebut kemudian akan menguji setiap method yang ada di kelas tersebut. Berikut adalah tabel lengkap rencana pengujian unit, dapat dilihat pada Tabel IV-7.

Tabel IV-7 Rencana Pengujian Unit

Kelas Uji	Method Uji	Jenis Pengujian
Controller	create()	*Unit Testing*
	delete()	*Unit Testing*
	getDateBetween()	*Unit Testing*
	getMaxMin()	*Unit Testing*
	limitSortByKey()	*Unit Testing*
	select()	*Unit Testing*
	selectById()	*Unit Testing*
	sortByKey()	*Unit Testing*
	update()	*Unit Testing*
	updateAll()	*Unit Testing*
LinearCtrl	createModel()	*Unit Testing*
	deleteModel()	*Unit Testing*
	selectModel()	*Unit Testing*
	selectModelById()	*Unit Testing*
	updateModel()	*Unit Testing*
HistogramCtrl	createHist()	*Unit Testing*
	deleteHist()	*Unit Testing*
	getDateBetween()	*Unit Testing*
	selectHist()	*Unit Testing*
	selectHistById()	*Unit Testing*
	sortHist()	*Unit Testing*
	updateHist()	*Unit Testing*
	updateHistAll()	*Unit Testing*

2. Rencana Pengujian Integrasi

Pengujin integrasi ditujukan untuk router dari subsistem RESTful API. Tujuannya untuk memeriksa integrasi dengan database. Berikut adalah tabel lengkap rencana pengujian integrasi, dapat dilihat pada Tabel IV-8 sampai Tabel IV-19.

Tabel IV-8 Route Data 1: Ikhtisar Statistik antar Kerja DKI Jakarta

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/ikhtisar-statistik-antar-kerja	Digunakan untuk membuat data baru Ikhtisar Statistik antar Kerja Jakarta	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja	Digunakan untuk me-*retrive* data Ikhtisar Statistik antar Kerja Jakarta antar kerja	*Integration Testing*
GET	{ Base URL }/ ikhtisar-statistik-antar-kerja /limit={ :n }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
	&sortBy={ :key } &order={ [asc]/[desc] }		
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/id={ :id }	Digunakan untum me-*retrive* satu data Ikhtisar Statistik antar Kerja Jakarta dengan berdasarkan *field* id	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/key={ field } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *field*	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/ikhtisar-statistik-antar-kerja/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/ikhtisar-statistik-antar-kerja/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/ikhtisar-statistik-antar-kerja/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

Tabel IV-9 Route Data 2: Besar Upan Minimum DKI Jakarta

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/ump-inflasi	Digunakan untuk membuat data model linear UMP DKI Jakarta	*Integration Testing*
GET	{ Base URL }/ump-inflasi/id={ :id }	Digunakan untum me-*retrive* data model linear UMP DKI Jakarta berdasarkan *field* _id-nya	*Integration Testing*
GET	{ Base URL }/ump-inflasi	Digunakan untum me-*retrive* data model linear UMP DKI Jakarta dan data-nya	*Integration Testing*
PATCH	{ Base URL }/ump-inflasi/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
DELETE	{ Base URL }/ump-inflasi/id={ :id }	Digunakan untuk menghapus model linear UMP DKI Jakarta berdasarkan id-nya	*Integration Testing*
POST	{ Base URL }/ump-inflasi/data	Digunakan untuk membuat data baru UMP DKI Jakarta	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data	Digunakan untuk me-*retrive* data UMP DKI Jakarta	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data/ id={ :id }	Digunakan untum me-*retrive* satu data UMP DKI Jakarta berdasarkan id-nya	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data/ /key={ :key }&get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/ump-inflasi/data/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/ump-inflasi/data/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/ump-inflasi/data/ id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

Tabel IV-10

Route Data 3: Laju Pertumbuhan Ekonomi Jakarta dan Nasional

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/ pertumbuhan-ekonomi	Digunakan untuk membuat data model linear Laju Pertumbuhan Ekonomi	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Digunakan untum me-*retrive* data model linear Laju Pertumbuhan Ekonomi berdasarkan *field* _id-nya	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
GET	{ Base URL }/ pertumbuhan-ekonomi	Digunakan untum me-*retrive* data model linear Laju Pertumbuhan Ekonomi dan data-nya	*Integration Testing*
PATCH	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Digunakan untuk menghapus model linear Laju Pertumbuhan Ekonomi berdasarkan id-nya	*Integration Testing*
POST	{ Base URL }/ pertumbuhan-ekonomi/data	Digunakan untuk membuat data baru Laju Pertumbuhan Ekonomi	*Integration Testing*
GET	{ Base URL }/pertumbuhan-ekonomi/data	Digunakan untuk me-*retrive* data Laju Pertumbuhan Ekonomi	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/data/id={ :id }	Digunakan untum me-*retrive* satu data Laju Pertumbuhan Ekonomi berdasarkan id-nya	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/data &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL } /pertumbuhan-ekonomi/data/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/data/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/pertumbuhan-ekonomi/data/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/pertumbuhan-ekonomi/data/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
DELETE	{ Base URL }/ pertumbuhan-ekonomi/data/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

Tabel IV-11 Route Data 4: Tingkat Inflasi Jakarta dan Nasional

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/tingkat-inflasi	Digunakan untuk membuat data model linear Inflasi Jakarta dan Nasional	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/id={ :id }	Digunakan untum me-*retrive* data model linear Inflasi Jakarta dan Nasional berdasarkan *field* - id-nya	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi	Digunakan untum me-*retrive* data model linear Inflasi Jakarta dan Nasional dan data-nya	*Integration Testing*
PATCH	{ Base URL }/tingkat-inflasi/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/tingkat-inflasi/id={ :id }	Digunakan untuk menghapus model linear Inflasi Jakarta dan Nasional berdasarkan id-nya	*Integration Testing*
POST	{ Base URL }/tingkat-inflasi/data	Digunakan untuk membuat data baru Inflasi Jakarta dan Nasional	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data	Digunakan untuk me-*retrive* data Inflasi Jakarta dan Nasional	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data/id={ :id }	Digunakan untum me-*retrive* satu data Inflasi Jakarta dan Nasional berdasarkan id-nya	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data&sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
PUT	{ Base URL }/tingkat-inflasi/data/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/tingkat-inflasi/data/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/tingkat-inflasi/data/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

Tabel IV-12 Route Data 5: Komponen Inflasi Jakarta

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/ komponen-inflasi	Digunakan untuk membuat data baru Komponen Inflasi Jakarta	*Integration Testing*
GET	{ Base URL }/ komponen-inflasi	Digunakan untuk me-*retrive* data Komponen Inflasi Jakarta	*Integration Testing*
GET	{ Base URL }/ komponen-inflasi/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/ komponen-inflasi/ id={ :id }	Digunakan untum me-*retrive* satu Komponen Inflasi Jakarta berdasarkan *key* id	*Integration Testing*
GET	{ Base URL }/komponen-inflasi &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/komponen-inflasi/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/komponen-inflasi/key=tahun &gd="{ :tahun }"&ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/ komponen-inflasi/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/ komponen-inflasi/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
DELETE	{ Base URL }/ komponen-inflasi/ id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

Tabel IV-13 Route Data 6: Ekspor Impor DKI Jakarta

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/besar-ekspor-impor	Digunakan untuk membuat data baru Besar Ekspor Impor Jakarta	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor	Digunakan untuk me-*retrive* data Besar Ekspor Impor Jakarta	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor/id={ :id }	Digunakan untum me-*retrive* satu Besar Ekspor Impor Jakarta berdasarkan *field* _id	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/besar-ekspor-impor/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/besar-ekspor-impor/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/besar-ekspor-impor/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

Tabel IV-14 Route Data 7: Volume dan Nilai Ekspor DKI Jakarta

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/volume-dan-nilai-ekspor	Digunakan untuk membuat data baru Volume dan Nilai Ekspor Jakarta	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor	Digunakan untuk me-*retrive* data Volume dan Nilai Ekspor Jakarta	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Digunakan untum me-*retrive* satu Volume dan Nilai Ekspor Jakarta berdasarkan *field* _id	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor&sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

Tabel IV-15 Route Data 8: Nilai Impor Produk Menurut Golongan

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/nilai-impor-produk-menurut-golongan	Digunakan untuk membuat data baru Nilai Impor Produk Menurut Golongan	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
GET	{ Base URL }/nilai-impor-produk-menurut-golongan	Digunakan untuk me-*retrive* data Nilai Impor Produk Menurut Golongan	
GET	{ Base URL }/nilai-impor-produk-menurut-golongan /id={ :id }	Digunakan untum me-*retrive* satu Nilai Impor Produk Menurut Golongan berdasarkan *field* _id	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan /key=tahun &gd=”{ :tahun }” &ld”{ :tahun }”	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/nilai-impor-produk-menurut-golongan/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/nilai-impor-produk-menurut-golongan/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/nilai-impor-produk-menurut-golongan/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*
POST	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram	Digunakan untuk membuat data histogram dari data Nilai Impor Produk Menurut Golongan	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram	Digunakan untuk me-*retrive* data histogram dari data Nilai Impor Produk Menurut Golongan	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ _id }	Digunakan untuk me-*retrive* data histogram dari data Nilai Impor Produk Menurut Golongan	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram &sortBy{ :key} &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data histogram secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data histogram berdasarkan *key*	*Integration Testing*
PUT	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id histogram	*Integration Testing*
PATCH	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id histogram	*Integration Testing*
DELETE	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Digunakan untuk menghapus seluruh data histogram berdasarkan id-nya	*Integration Testing*

Tabel IV-16 Route Data 9: Struktur Ekonomi Jakarta Berdasarkan Sektor

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/struktur-ekonomi	Digunakan untuk membuat data baru Struktur Ekonomi Jakarta	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi	Digunakan untuk me-*retrive* data Struktur Ekonomi Jakarta	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi /id={ :id }	Digunakan untum me-*retrive* satu Struktur Ekonomi Jakarta berdasarkan *field* _id	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
GET	{ Base URL }/struktur-ekonomi /key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/struktur-ekonomi/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/struktur-ekonomi/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/struktur-ekonomi/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

Tabel IV-17 Route Data 10: Harga Pangan Tingkat Konsumen DKI Jakarta

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/harga-pangan-tingkat-konsumen	Digunakan untuk membuat data baru Harga Pangan Tingkat Konsumen Jakarta	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen	Digunakan untuk me-*retrive* data Nilai Harga Pangan Tingkat Konsumen Jakarta	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Digunakan untum me-*retrive* satu Harga Pangan Tingkat Konsumen Jakarta berdasarkan *field* _id	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen &{ :key }={ :value } &sortBy={ key} &order{ [asc]/[desc] }	Digunakan untuk melakukan filtering data dan diurutkan secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
GET	{ Base URL }/harga-pangan-tingkat-konsumen /key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/harga-pangan-tingkat-konsumen/id={ _id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*
POST	{ Base URL }/harga-pangan-tingkat-konsumen/histogram	Digunakan untuk membuat data histogram dari data Harga Pangan Tingkat Konsumen Jakarta	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram	Digunakan untuk me-*retrive* data histogram dari data Harga Pangan Tingkat Konsumen Jakarta	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen /histogram/id={ :id }	Digunakan untuk me-*retrive* data histogram dari data Harga Pangan Tingkat Konsumen Jakarta	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data histogram secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data histogram berdasarkan *key*	*Integration Testing*
PUT	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id histogram	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
PATCH	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id histogram	*Integration Testing*
DELETE	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Digunakan untuk menghapus seluruh data histogram berdasarkan id-nya	*Integration Testing*

Tabel IV-18 Route Data 11: Perkembangan Harga Grosir di Pasar Induk

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/ perkembangan-harga-grosir	Digunakan untuk membuat data baru Perkembangan Harga Grosir di Pasar Induk	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir	Digunakan untuk me-*retrive* data Perkembangan Harga Grosir di Pasar Induk	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Digunakan untum me-*retrive* satu Perkembangan Harga Grosir di Pasar Induk berdasarkan *field* _id	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir&sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/key=tahun &gd=”{ :tahun }” &ld”{ :tahun }”	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/ perkembangan-harga-grosir /id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
DELETE	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*
POST	{ Base URL }/ perkembangan-harga-grosir/histogram	Digunakan untuk membuat data histogram dari data Perkembangan Harga Grosir di Pasar Induk	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/histogram	Digunakan untuk me-*retrive* data histogram dari data Perkembangan Harga Grosir di Pasar Induk	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/histogram/ id={ :id }	Digunakan untuk me-*retrive* data histogram dari data Perkembangan Harga Grosir di Pasar Induk	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/histogram &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data histogram secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/perkembangan-harga-grosir/histogram/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data histogram berdasarkan *key*	*Integration Testing*
PUT	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id histogram	*Integration Testing*
PATCH	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id histogram	*Integration Testing*
DELETE	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Digunakan untuk menghapus seluruh data histogram berdasarkan id-nya	*Integration Testing*

Tabel IV-19 Route Data 12: Pendapatan Perkapita Jakarta dan Nasional

Method	URL	Keterangan	Jenis Pengujian
POST	{ Base URL }/pendapatan-perkapita	Digunakan untuk membuat data model linear Pendapatan Perkapita Jakarta dan Nasional	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Digunakan untum me-*retrive* data model linear Pendapatan	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
		Perkapita Jakarta dan Nasional berdasarkan *field* _id-nya	
GET	{ Base URL }/ pendapatan-perkapita	Digunakan untum me-*retrive* data model linear Pendapatan Perkapita Jakarta dan Nasional dan data-nya	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	*Integration Testing*
PATCH	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	*Integration Testing*
DELETE	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Digunakan untuk menghapus model linear Pendapatan Perkapita Jakarta dan Nasional berdasarkan id-nya	*Integration Testing*
POST	{ Base URL }/ pendapatan-perkapita/ data	Digunakan untuk membuat data baru Pendapatan Perkapita Jakarta dan Nasional	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ data	Digunakan untuk me-*retrive* data Pendapatan Perkapita Jakarta dan Nasional	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Digunakan untum me-*retrive* satu data Pendapatan Perkapita Jakarta dan Nasional berdasarkan id-nya	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/data &sortBy{ :key} &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ data/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	*Integration Testing*
GET	{ Base URL }/pendapatan-perkapita/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	*Integration Testing*
PUT	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	*Integration Testing*
PATCH	{ Base URL }/	Digunakan untuk melakukan *update* terhadap sebagian atau	*Integration Testing*

Method	URL	Keterangan	Jenis Pengujian
	pendapatan-perkapita/ data/id={ :id }	seluruh *fields* dari id yang dipilih	
DELETE	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	*Integration Testing*

3. Rencana Pengujian Keluaran Algoritma

Rencana pengujian keluaran *algoritma* merupakan pengujian dengan cara memban-dingkan hasil keluaran yang dihasilkan dari penerapan algoritma *linear regression* dan *k*-Means dengan algoritma yang sama dari perangkat lunak Weka versi 3.8.x.

Adapun rencana pengujian algoritma adalah sebagai berikut:

a. Pengujian menggunakan data yang sama.
b. Untuk algoritma *k*-Means, banyaknya grup ditentukan sama seperti pada *k*-Means yang dibangun, selanjutnya hasil keluaran setiap *cluster* akan dibandingkan.
c. Untuk algoritma *linear regression,* hasil yang akan dibandingkan adalah kesamaan *correlation coeficient* dan model-nya.

4. Rencana Pengujian Beta

Rencana pengujian *beta* yang akan dilakukan adalah dengan melakukan kuesioner ke beberapa pengguna terpilih untuk segmen pelaku ekonomi makro dan mikro (skala *likert*) dan juga wawancara kepada *early adaptor user* yang sudah ditetapkan pada bab III terhadap subsistem aplikai *mobile* visualisasi data yang dibangun, Ibu Sri Dewi Anggadini, SE., M.Si, dosen tetap Fakultas Ekonomi di Universitas Komputer Indonesia (UNIKOM).

4.2.2 Sekenario Pengujian

Skenario pengujian menjabarkan urutan dalam pengujian yang dilakukan pada perangkat lunak yang dibangun. Skenario pengujian dilakukan berdasarkan pengujian unit, pengujian integrasi, pengujian algoritma, dan pengujian *beta.*

1. Skenario Pengujian Unit

Bagian ini memaparkan bagaimana bentuk skenario pengujian unit pada kelas *controller* subsistem RESTful API. Skenario pengujiannya yaitu, jika pengujiannya sukses maka ia akan menampilkan pesan sesuai tugas fungsi yang di harapkan. Berikut adalah tabel lengkap skenario pengujian unit *controller* yang dapat dilihat pada Tabel IV-20.

Tabel IV-20 Skenario Pengujian Unit

Kelas Uji	Method Uji	Poin Pengujian	Jenis Pengujian
Controller	create()	Jika pengujian sukses, akan me-respon kode status 200 *sucessful*.	*Unit Testing*
	delete()	Jika pengujian sukses, akan me-*respon* kode status 204 *deleted sucessful*.	*Unit Testing*
	getDateBetween()	Jika pengujian sukses, akan me-*return* JSON dengan rentang yang dipilih dan merespon kode status 200 *sucessful*.	*Unit Testing*
	getMaxMin()	Jika pengujian sukses, akan me-*return* JSON *value* maksimum atau minimum dan merespon kode status 200 *sucessful*.	*Unit Testing*
	limitSortByKey()	Jika pengujian sukses, akan me-*return* JSON dengan jumlah tertentu secara terurut dan merespon kode status 200 *sucessful*.	*Unit Testing*
	select()	Jika pengujian sukses, akan me-*return* seluruh JSON data dan merespon kode status 200 *sucessful*.	*Unit Testing*
	selectById()	Jika pengujian sukses, akan me-*return* JSON dengan id tertentu dan merespon kode status 200 *sucessful*.	*Unit Testing*
	sortByKey()	Jika pengujian sukses, akan me-*return* JSON dengan urutan *ascending* atau *descending* sesuai *key* yang dipilih dan merespon kode status 200 *sucessful*.	*Unit Testing*

Kelas Uji	Method Uji	Poin Pengujian	Jenis Pengujian
	update()	Jika pengujian sukses, akan merespon kode status 200 *update sucessful*.	*Unit Testing*
	updateAll()	Jika pengujian sukses, akan merespon kode status 200 *update sucessful*.	*Unit Testing*
LinearCtrl	createModel()	Jika pengujian sukses, akan merespon kode status 200 *sucessful*.	*Unit Testing*
	deleteModel()	Jika pengujian sukses, akan me-*respon* kode status 204 *deleted sucessful*.	*Unit Testing*
	selectModel()	Jika pengujian sukses, akan me-*return* seluruh JSON data dan merespon kode status 200 *sucessful*.	*Unit Testing*
	selectModelById()	Jika pengujian sukses, akan me-*return* JSON dengan id tertentu dan merespon kode status 200 *sucessful*.	*Unit Testing*
	updateModel()	Jika pengujian sukses, akan merespon kode status 200 *update sucessful*.	*Unit Testing*
HistogramCtrl	createHist()	Jika pengujian sukses, akan merespon kode status 200 *sucessful*.	*Unit Testing*
	deleteHist()	Jika pengujian sukses, akan me-*respon* kode status 204 *deleted sucessful*.	*Unit Testing*
	getDateBetween()	Jika pengujian sukses, akan me-*return* JSON dengan rentang yang dipilih dan merespon kode status 200 *sucessful*.	*Unit Testing*
	selectHist()	Jika pengujian sukses, akan me-*return* seluruh JSON data dan merespon kode status 200 *sucessful*.	*Unit Testing*
	selectHistById()	Jika pengujian sukses, akan me-*return* JSON dengan id tertentu dan merespon kode status 200 *sucessful*.	*Unit Testing*
	sortHist()	Jika pengujian sukses, akan me-*return* JSON dengan urutan *ascending* atau *descending* dan merespon kode status 200 *sucessful*.	*Unit Testing*

Kelas Uji	Method Uji	Poin Pengujian	Jenis Pengujian
	updateHist()	Jika pengujian sukses, akan merespon kode status 200 *update sucessful*.	*Unit Testing*
	updateHistAll()	Jika pengujian sukses, akan merespon kode status 200 *update sucessful*.	*Unit Testing*

2. Skenario Pengujian Integrasi

Bagian ini akan memaparkan bagaimana bentuk skenario pengujian integrasi pada *route*. Skenario bertujuan untuk melihat integrasi *route* dengan *database*. Skenarionya yaitu, jika pengujiannya sukses maka ia akan menampilkan, menambahkan, memperbahurui, ataupun menghapus data yang ada di database. Berikut adalah tabel lengkap skenario pengujian integrasi *route* yang dapat dilihat pada Tabel IV-21 sampai Tabel IV-32.

Tabel IV-21 Skenario Pengujian Integrasi *Route* Data 1

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/ikhtisar-statistik-antar-kerja	Digunakan untuk membuat data baru Ikhtisar Statistik antar Kerja Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja	Digunakan untuk me-*retrive* data Ikhtisar Statistik antar Kerja Jakarta antar kerja	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/ ikhtisar-statistik-antar-kerja /limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/id= { :id }	Digunakan untum me-*retrive* satu data Ikhtisar Statistik antar Kerja Jakarta dengan berdasarkan *field* id	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/key={ field } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *field*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
PUT	{ Base URL }/ikhtisar-statistik-antar-kerja/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/ikhtisar-statistik-antar-kerja/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/ikhtisar-statistik-antar-kerja/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-22 Skenario Pengujian Integrasi *Route* Data 2

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/ump-inflasi	Digunakan untuk membuat data model linear UMP DKI Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/ump-inflasi/id={ :id }	Digunakan untum me-*retrive* data model linear UMP DKI Jakarta berdasarkan *field* _id-nya	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/ump-inflasi	Digunakan untum me-*retrive* data model linear UMP DKI Jakarta dan data-nya	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
PATCH	{ Base URL }/ump-inflasi/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
DELETE	{ Base URL }/ump-inflasi/id={ :id }	Digunakan untuk menghapus model linear UMP DKI Jakarta berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*
POST	{ Base URL }/ump-inflasi/data	Digunakan untuk membuat data baru UMP DKI Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data	Digunakan untuk me-*retrive* data UMP DKI Jakarta	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data/ id={ :id }	Digunakan untum me-*retrive* satu data UMP DKI Jakarta berdasarkan id-nya	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data/ /key={ :key }&get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/ump-inflasi/data/key=tahun &gd=”{ :tahun }” &ld”{ :tahun }”	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
PUT	{ Base URL }/ump-inflasi/data/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/ump-inflasi/data/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/ump-inflasi/data/ id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-23 Skenario Pengujian Integrasi *Route* Data 3

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/ pertumbuhan-ekonomi	Digunakan untuk membuat data model linear Laju Pertumbuhan Ekonomi	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Digunakan untum me-*retrive* data model linear Laju Pertumbuhan Ekonomi berdasarkan *field* _id-nya	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi	Digunakan untum me-*retrive* data model linear Laju Pertumbuhan Ekonomi dan data-nya	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
PATCH	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Digunakan untuk menghapus model linear Laju Pertumbuhan Ekonomi	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
		berdasarkan id-nya		
POST	{ Base URL }/ pertumbuhan-ekonomi/data	Digunakan untuk membuat data baru Laju Pertumbuhan Ekonomi	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/pertumbuhan-ekonomi/data	Digunakan untuk me-*retrive* data Laju Pertumbuhan Ekonomi	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/data/id={ :id }	Digunakan untum me-*retrive* satu data Laju Pertumbuhan Ekonomi berdasarkan id-nya	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/data &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL } /pertumbuhan-ekonomi/data/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/ pertumbuhan-ekonomi/data/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
GET	{ Base URL }/ pertumbuhan-ekonomi/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/pertumbuhan-ekonomi/data/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/pertumbuhan-ekonomi/data/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/ pertumbuhan-ekonomi/data/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-24 Skenario Pengujian Integrasi *Route* Data 4

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/tingkat-inflasi	Digunakan untuk membuat data model linear Inflasi Jakarta dan Nasional	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/id={ :id }	Digunakan untum me-*retrive* data model linear Inflasi Jakarta dan Nasional berdasarkan *field* id-nya	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi	Digunakan untum me-*retrive* data model linear Inflasi Jakarta dan Nasional dan data-nya	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
PATCH	{ Base URL }/tingkat-inflasi/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/tingkat-inflasi/id={ :id }	Digunakan untuk menghapus model linear Inflasi Jakarta dan Nasional berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*
POST	{ Base URL }/tingkat-inflasi/data	Digunakan untuk membuat data baru Inflasi Jakarta dan Nasional	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data	Digunakan untuk me-*retrive* data Inflasi Jakarta dan Nasional	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data/id={ :id }	Digunakan untum me-*retrive* satu data Inflasi Jakarta dan Nasional berdasarkan id-nya	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data&sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/tingkat-inflasi/data/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
GET	{ Base URL }/tingkat-inflasi/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/tingkat-inflasi/data/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/tingkat-inflasi/data/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/tingkat-inflasi/data/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-25 Skenario Pengujian Integrasi *Route* Data 5

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/ komponen-inflasi	Digunakan untuk membuat data baru Komponen Inflasi Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/ komponen-inflasi	Digunakan untuk me-*retrive* data Komponen Inflasi Jakarta	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/ komponen-inflasi/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
GET	{ Base URL }/ komponen-inflasi/ id={ :id }	Digunakan untum me-*retrive* satu Komponen Inflasi Jakarta berdasarkan *key* id	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/komponen-inflasi &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/komponen-inflasi/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/komponen-inflasi/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/ komponen-inflasi/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/ komponen-inflasi/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/ komponen-inflasi/ id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-26 Skenario Pengujian Integrasi *Route* Data 6

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/besar-ekspor-impor	Digunakan untuk membuat data baru Besar Ekspor Impor Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor	Digunakan untuk me-*retrive* data Besar Ekspor Impor Jakarta	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor/id={ :id }	Digunakan untum me-*retrive* satu Besar Ekspor Impor Jakarta berdasarkan *field* _id	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/besar-ekspor-impor/key=tahun &gd=”{ :tahun }” &ld”{ :tahun }”	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
PUT	{ Base URL }/besar-ekspor-impor/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/besar-ekspor-impor/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/besar-ekspor-impor/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-27 Skenario Pengujian Integrasi *Route* Data 7

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/volume-dan-nilai-ekspor	Digunakan untuk membuat data baru Volume dan Nilai Ekspor Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor	Digunakan untuk me-*retrive* data Volume dan Nilai Ekspor Jakarta	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Digunakan untum me-*retrive* satu Volume dan Nilai Ekspor Jakarta berdasarkan *field* _id	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor/ limit={ :n }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
	&sortBy={ :key } &order={ [asc]/[desc] }			
GET	{ Base URL }/volume-dan-nilai-ekspor&sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/volume-dan-nilai-ekspor/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-28 Skenario Pengujian Integrasi *Route* Data 8

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/nilai-impor-produk-menurut-golongan	Digunakan untuk membuat data baru Nilai Impor Produk Menurut Golongan	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan	Digunakan untuk me-*retrive* data Nilai Impor Produk Menurut Golongan	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan /id={ :id }	Digunakan untum me-*retrive* satu Nilai Impor Produk Menurut Golongan berdasarkan *field* _id	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
GET	{ Base URL }/nilai-impor-produk-menurut-golongan /key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/nilai-impor-produk-menurut-golongan/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/nilai-impor-produk-menurut-golongan/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/nilai-impor-produk-menurut-golongan/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*
POST	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram	Digunakan untuk membuat data histogram dari data Nilai Impor Produk Menurut Golongan	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram	Digunakan untuk me-*retrive* data histogram dari data Nilai Impor Produk Menurut Golongan	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Digunakan untuk me-*retrive* data histogram dari data Nilai Impor Produk Menurut Golongan	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram &sortBy{ :key} &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data histogram secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data histogram berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id histogram	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id histogram	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Digunakan untuk menghapus seluruh data histogram berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-29 Skenario Pengujian Integrasi *Route* Data 9

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/struktur-ekonomi	Digunakan untuk membuat data baru Struktur Ekonomi Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi	Digunakan untuk me-*retrive* data Struktur Ekonomi Jakarta	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi /id={ :id }	Digunakan untum me-*retrive* satu Struktur Ekonomi Jakarta berdasarkan *field* _id	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi /key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/struktur-ekonomi/key=tahun &gd=”{ :tahun }” &ld”{ :tahun }”	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
PUT	{ Base URL }/struktur-ekonomi/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/struktur-ekonomi/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/struktur-ekonomi/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-30 Skenario Pengujian Integrasi *Route* Data 10

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/harga-pangan-tingkat-konsumen	Digunakan untuk membuat data baru Harga Pangan Tingkat Konsumen Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen	Digunakan untuk me-*retrive* data Nilai Harga Pangan Tingkat Konsumen Jakarta	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Digunakan untum me-*retrive* satu Harga Pangan Tingkat Konsumen Jakarta berdasarkan *field* _id	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
GET	{ Base URL }/harga-pangan-tingkat-konsumen &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen &{ :key }={ :value } &sortBy={ key} &order{ [asc]/[desc] }	Digunakan untuk melakukan filtering data dan diurutkan secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan sesuai key yang dipilih dan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen /key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/harga-pangan-tingkat-konsumen/id={ _id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/harga-pangan-tingkat-konsumen/histogram	Digunakan untuk membuat data histogram dari data Harga Pangan Tingkat Konsumen Jakarta	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram	Digunakan untuk me-*retrive* data histogram dari data Harga Pangan Tingkat Konsumen Jakarta	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen /histogram/id={ :id }	Digunakan untuk me-*retrive* data histogram dari data Harga Pangan Tingkat Konsumen Jakarta	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data histogram secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data histogram berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id histogram	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id histogram	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
DELETE	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Digunakan untuk menghapus seluruh data histogram berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-31 Skenario Pengujian Integrasi *Route* Data 11

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL }/ perkembangan-harga-grosir	Digunakan untuk membuat data baru Perkembangan Harga Grosir di Pasar Induk	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir	Digunakan untuk me-*retrive* data Perkembangan Harga Grosir di Pasar Induk	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Digunakan untum me-*retrive* satu Perkembangan Harga Grosir di Pasar Induk berdasarkan *field* _id	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir&sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/limit={ :n } &sortBy={ :key }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
	&order={ [asc]/[desc] }			
GET	{ Base URL }/ perkembangan-harga-grosir/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/ perkembangan-harga-grosir /id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*
POST	{ Base URL }/ perkembangan-harga-grosir/histogram	Digunakan untuk membuat data histogram dari data Perkembangan Harga Grosir di Pasar Induk	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/histogram	Digunakan untuk me-*retrive* data histogram dari data Perkembangan Harga Grosir di Pasar Induk	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
GET	{ Base URL }/ perkembangan-harga-grosir/histogram/ id={ :id }	Digunakan untuk me-*retrive* data histogram dari data Perkembangan Harga Grosir di Pasar Induk	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/ perkembangan-harga-grosir/histogram &sortBy{ :key } &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data histogram secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/perkembangan-harga-grosir/histogram/ key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data histogram berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id histogram	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id histogram	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Digunakan untuk menghapus seluruh data histogram berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

Tabel IV-32 Skenario Pengujian Integrasi *Route* Data 12

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
POST	{ Base URL}/ pendapatan-perkapita	Digunakan untuk membuat data model linear Pendapatan Perkapita Jakarta dan Nasional	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Digunakan untum me-*retrive* data model linear Pendapatan Perkapita Jakarta dan Nasional berdasarkan *field* _id-nya	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita	Digunakan untum me-*retrive* data model linear Pendapatan Perkapita Jakarta dan Nasional dan data-nya	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Digunakan untuk me-*retrive* data dengan n buah objek	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Digunakan untuk menghapus model linear Pendapatan Perkapita Jakarta dan Nasional berdasarkan id-nya	Jika pengujian sukses maka data baru akan ditambahkan ke *dabatabase*	*Integration Testing*
POST	{ Base URL }/ pendapatan-perkapita/ data	Digunakan untuk membuat data baru Pendapatan Perkapita Jakarta dan Nasional	Jika pengujian sukses maka seluruh data yang terdapat dalam *database* akan ditampilkan	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ data	Digunakan untuk me-*retrive* data Pendapatan Perkapita Jakarta dan Nasional	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan berdasarkan parameter id	*Integration Testing*

Method	URL	Keterangan	Poin Pengujian	Jenis Pengujian
GET	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Digunakan untum me-*retrive* satu data Pendapatan Perkapita Jakarta dan Nasional berdasarkan id-nya	Jika pengujian sukses maka n buah data yang terdapat dalam *database* akan ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/data &sortBy{ :key} &order{ [asc]/[desc] }	Digunakan untuk mengurutkan data secara *ascending* atau *descending* berdasarkan *key*	Jika pengujian sukses maka data yang terdapat dalam *database* akan ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Integration Testing*
GET	{ Base URL }/ pendapatan-perkapita/ data/key={ :key } &get{ [max]/[min] }	Digunakan untuk me-*retrive* maksimum atau minumum data berdasarkan *key*	Jika pengujian sukses maka data maksimum atau minum yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
GET	{ Base URL }/pendapatan-perkapita/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Digunakan untuk me-*retrive* data antara tahun	Jika pengujian sukses maka rentang data yang terdapat dalam *database* akan ditampilkan sesuai parameter	*Integration Testing*
PUT	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Digunakan untuk melakukan *update* terhadap seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Integration Testing*
PATCH	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Digunakan untuk melakukan *update* terhadap sebagian atau seluruh *fields* dari id yang dipilih	Jika pengujian sukses maka akan memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Integration Testing*
DELETE	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Digunakan untuk menghapus data berdasarkan id-nya	Jika pengujian sukses maka akan menghapus data dalam database berdasarkan parameter id	*Integration Testing*

3. Skenario Pengujian Keluaran Algoritma

Skenario pengujian keluaran algoritma merupakan suatu proses pengujian mengenai algoritma *linear regression* dan *k*-Means. Keluaran dari algoritma yang dibangun akan dicocokan dengan keluaran dari algoritma yang sama pada perangkat lunak weka v3.8.x dan perhitungan dari Microsoft excel untuk *k*-Means. Tujuan dari pengujian ini untuk mencari tahu kesalahan dalam implementasi algoritma *linear regression* dan *k*-Means pada sistem yang dibangun.

4. Skenario Pengujian Beta

Skenario pengujian *beta* yang dilakukan adalah melakukan wawancara kepada *early adaptor user* yang sudah ditetapkan pada bab III terhadap subsistem aplikai *mobile* visualisasi data yang dibangun, Ibu Sri Dewi Anggadini, SE., M.Si, dosen tetap Fakultas Ekonomi di Universitas Komputer Indonesia (UNIKOM). Adapun pertanyaan untuk wawancara ditunjukan tabel A dan skenario pengujian beta dengan cara kuesioner ditunjukan tabel B.

Tabel IV-33 Skenarion Pengujian Beta dengan Wawancara

No	Pertanyaan
1	Apakah aplikasi ini dapat menunjukan seberapa besar sumber daya yang sudah digunakan?
2	Apakah aplikasi ini dapat menunjukan pergerakan variabel-variabel ekonomi?
3	Apakah aplikasi ini dapat membantu melihat distribusi harga?
4	Apakah aplikasi visualisasi ini dapat memnunjukan laju pergerakan permintaan dan penawaran?
5	Apakah aplikasi ini mudah digunakan?

Tabel IV-34 Skenario Pengujian Beta dengan Kuesioner

No	Kuesioner	1	2	3	4	5	6	7
1	Apakah Bp/Ibu dapat membandingkan lowongan kerja yang sudah dipenuhi dan lowongan kerja yang belum dipenuhi pada tahun 2011? Dari skala 1 (sangat tidak dapat) sampai skala 7 (sangat dapat).							
2	Apakah Bp/Ibu dapat melihat pergerakan inflasi Jakarta dan Nasional dengan mudah?							

No	Kuesioner	1	2	3	4	5	6	7
	Dari skala 1 (sangat tidak dapat) sampai skala 7 (sangat dapat).							
3	Apakah Bp/Ibu melihat kumpulan harga pangan yang paling banyak muncl dan paling banyak tidak muncul? Dari skala 1 (sangat tidak dapat) sampai skala 7 (sangat dapat).							
4	Apakah Bp/Ibu dapat melihat harga pangan tanggal pada tanggal tertentu dan tanggal lainnya? Dari skala 1 (sangat tidak dapat) sampai skala 7 (sangat dapat).							
5	Apakah aplikasi ini mudah digunakan? Dari skala 1 (sangat mudah) sampai skala 7 (sangat sulitt).							

4.2.3 Hasil Pengujian

Hasil pengujian memaparkan hasil dari skenario pengujian yang telah disusun pada bagian sebelumnya. Hasil pengujian ini terdiri dari hasil pengujian unit, hasil pengujian integritas, hasil pengujian algoritma, dan hasil pengujian beta.

1. Hasil Pengujian Unit

Berikut adalah hasil pengujian unit kelas *controller* pada subsistem RESTful API, dapat dilihat pada Tabel IV-35 sampai Tabel IV-37.

Tabel IV-35 Hasil Pengujian Unit Kelas Controller

Method Uji	Hasil yang Diharapkan	Kesimpulan
create()	Merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
delete()	Merespon kode status 204 *deleted sucessful*	*Method* sudah berkerja dengan benar
getDateBetween()	Me-*return* JSON dengan rentang yang dipilih dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
getMaxMin()	Me-*return* JSON *value* maksimum atau minimum dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
limitSortByKey()	Me-*return* JSON dengan jumlah tertentu secara terurut dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
select()	Me-*return* seluruh JSON data dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
selectById()	Me-*return* JSON dengan id tertentu dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar

Method Uji	Hasil yang Diharapkan	Kesimpulan
sortByKey()	Me-*return* JSON dengan urutan *ascending* atau *descending* sesuai *key* yang dipilih dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
update()	Merespon kode status 200 *update sucessful*	*Method* sudah berkerja dengan benar
updateAll()	Merespon kode status 200 *update sucessful*	*Method* sudah berkerja dengan benar

Tabel IV-36 Hasil Pengujian Unit Kelas ModelCtrl

Method Uji	Hasil yang Diharapkan	Kesimpulan
createModel()	Merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
deleteModel()	Merespon kode status 204 *deleted sucessful*	*Method* sudah berkerja dengan benar
selectModel()	Me-*return* seluruh JSON data dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
selectModelById()	Me-*return* JSON dengan id tertentu dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
updateModel()	Merespon kode status 200 *update sucessful*	*Method* sudah berkerja dengan benar

Tabel IV-37 Hasil Pengujian Unit Kelas HistogramCtrl

Method Uji	Hasil yang Diharapkan	Kesimpulan
createHist()	Merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
deleteHist()	Merespon kode status 204 *deleted sucessful*	*Method* sudah berkerja dengan benar
getDateBetween()	Me-*return* JSON dengan rentang yang dipilih dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
selectHist()	Me-*return* seluruh JSON data dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
selectHistById()	Me-*return* JSON dengan id tertentu dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
sortHist()	Me-*return* JSON dengan urutan *ascending* atau *descending* dan merespon kode status 200 *sucessful*	*Method* sudah berkerja dengan benar
updateHist()	Merespon kode status 200 *update sucessful*	*Method* sudah berkerja dengan benar
updateHistAll()	Merespon kode status 200 *update sucessful*	*Method* sudah berkerja dengan benar

2. Hasil Pengujian Integrasi

Berikut adalah hasil pengujian integrasi router pada subsistem RESTful API, dapat dilihat pada Tabel IV-38 sampai Tabel IV-49.

Tabel IV-38 Hasil Pengujian Integrasi Data 1

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/ikhtisar-statistik-antar-kerja	Pengujian sukses, data baru akan ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ ikhtisar-statistik-antar-kerja /limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/id= { :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/key={ field } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ikhtisar-statistik-antar-kerja/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/ikhtisar-statistik-antar-kerja/id= { :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/ikhtisar-statistik-antar-kerja/id= { :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
DELETE	{ Base URL }/ikhtisar-statistik-antar-kerja/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-39 Hasil Pengujian Integrasi Data 2

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/ump-inflasi	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ump-inflasi/id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ump-inflasi	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/ump-inflasi/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/ump-inflasi/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
POST	{ Base URL }/ump-inflasi/data	Pengujian sukses, maka data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ump-inflasi/data	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ump-inflasi/data/ id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ump-inflasi/data/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ump-inflasi/data	Pengujian sukses, data yang terdapat dalam *database*	*Route* sudah terintegrasi

Method	URL	Hasil yang Diharapkan	Kesimpulan
	&sortBy{ :key } &order{ [asc]/[desc] }	ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	dengan *database*
GET	{ Base URL }/ump-inflasi/data/ /key={ :key }&get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ump-inflasi/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/ump-inflasi/data/ id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/ump-inflasi/data/ id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/ump-inflasi/data/ id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-40 Hasil Pengujian Integrasi Data 3

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/ pertumbuhan-ekonomi	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pertumbuhan-ekonomi	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/ pertumbuhan-ekonomi/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/ pertumbuhan-ekonomi/data	Pengujian sukses, maka data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/pertumbuhan-ekonomi/data	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pertumbuhan-ekonomi/data/id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pertumbuhan-ekonomi/data &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL } /pertumbuhan-ekonomi/data/key={ :key } &get{ [max]/[min] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pertumbuhan-ekonomi/data/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pertumbuhan-ekonomi/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/pertumbuhan-ekonomi/data/id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/pertumbuhan-ekonomi/data/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/ pertumbuhan-ekonomi/data/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-41 Hasil Pengujian Integrasi Data 4

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/tingkat-inflasi	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/tingkat-inflasi/id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/tingkat-inflasi	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/tingkat-inflasi/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/tingkat-inflasi/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
POST	{ Base URL }/tingkat-inflasi/data	Pengujian sukses, maka data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/tingkat-inflasi/data	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/tingkat-inflasi/data/id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/tingkat-inflasi/data/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/tingkat-inflasi/data&sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/tingkat-inflasi/data/key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
GET	{ Base URL }/tingkat-inflasi/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/tingkat-inflasi/data/id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/tingkat-inflasi/data/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/tingkat-inflasi/data/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-42 Hasil Pengujian Integrasi Data 5

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/ komponen-inflasi	Pengujian sukses, data baru akan ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ komponen-inflasi	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ komponen-inflasi/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ komponen-inflasi/ id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/komponen-inflasi &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/komponen-inflasi/key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
GET	{ Base URL }/komponen-inflasi/key=tahun &gd=”{ :tahun }” &ld”{ :tahun }”	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/ komponen-inflasi/ id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/ komponen-inflasi/ id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/ komponen-inflasi/ id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-43 Hasil Pengujian Integrasi Data 6

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/besar-ekspor-impor	Pengujian sukses, data baru akan ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/besar-ekspor-impor	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/besar-ekspor-impor/id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/besar-ekspor-impor/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/besar-ekspor-impor &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/besar-ekspor-impor/ key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
GET	{ Base URL }/besar-ekspor-impor/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/besar-ekspor-impor/id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/besar-ekspor-impor/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/besar-ekspor-impor/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-44 Hasil Pengujian Integrasi Data 7

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/volume-dan-nilai-ekspor	Pengujian sukses, data baru akan ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/volume-dan-nilai-ekspor	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/volume-dan-nilai-ekspor/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/volume-dan-nilai-ekspor&sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/volume-dan-nilai-ekspor/ key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
GET	{ Base URL }/volume-dan-nilai-ekspor/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/volume-dan-nilai-ekspor/ id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-45 Hasil Pengujian Integrasi Data 8

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/nilai-impor-produk-menurut-golongan	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan /id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
GET	{ Base URL }/nilai-impor-produk-menurut-golongan /key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/nilai-impor-produk-menurut-golongan/ id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/nilai-impor-produk-menurut-golongan/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/nilai-impor-produk-menurut-golongan/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
POST	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram &sortBy{ :key} &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
PATCH	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/nilai-impor-produk-menurut-golongan/histogram/ id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-46 Hasil Pengujian Integrasi Data 9

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/struktur-ekonomi	Pengujian sukses, data baru akan ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/struktur-ekonomi	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/struktur-ekonomi /id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/struktur-ekonomi/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/struktur-ekonomi &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/struktur-ekonomi /key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/struktur-ekonomi/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/struktur-ekonomi/id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/struktur-ekonomi/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field*	*Route* sudah terintegrasi

Method	URL	Hasil yang Diharapkan	Kesimpulan
		data dalam database berdasarkan parameter id	dengan *database*
DELETE	{ Base URL }/struktur-ekonomi/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-47 Hasil Pengujian Integrasi Data 10

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/harga-pangan-tingkat-konsumen	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen &{ :key }={ :value } &sortBy={ key} &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan sesuai key yang dipilih dan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen /key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/key=tahun &gd=”{ :tahun }” &ld”{ :tahun }”	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
PUT	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/harga-pangan-tingkat-konsumen/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/harga-pangan-tingkat-konsumen/id={ _id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
POST	{ Base URL }/harga-pangan-tingkat-konsumen/histogram	Pengujian sukses, baru akan ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram	Pengujian sukses, seluruh data yang terdapat dalam *database* akan ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen /histogram/id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/harga-pangan-tingkat-konsumen/histogram/ id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-48 Hasil Pengujian Integrasi Data 11

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL }/ perkembangan-harga-grosir	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ perkembangan-harga-grosir	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ perkembangan-harga-grosir&sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ perkembangan-harga-grosir/limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ perkembangan-harga-grosir/key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ perkembangan-harga-grosir/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/ perkembangan-harga-grosir /id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/ perkembangan-harga-grosir/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
POST	{ Base URL }/ perkembangan-harga-grosir/histogram	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
GET	{ Base URL }/ perkembangan-harga-grosir/histogram	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ perkembangan-harga-grosir/histogram/ id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ perkembangan-harga-grosir/histogram &sortBy{ :key } &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/perkembangan-harga-grosir/histogram/ key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/perkembangan-harga-grosir/histogram/ id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Tabel IV-49 Hasil Pengujian Integrasi Data 12

Method	URL	Hasil yang Diharapkan	Kesimpulan
POST	{ Base URL}/ pendapatan-perkapita	Pengujian sukses, data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pendapatan-perkapita	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
GET	{ Base URL }/ pendapatan-perkapita/ limit={ :n } &sortBy={ :key } &order={ [asc]/[desc] }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
DELETE	{ Base URL }/ pendapatan-perkapita/ id={ :id }	Pengujian sukses, maka data baru ditambahkan ke *dabatabase*	*Route* sudah terintegrasi dengan *database*
POST	{ Base URL }/ pendapatan-perkapita/ data	Pengujian sukses, seluruh data yang terdapat dalam *database* ditampilkan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pendapatan-perkapita/ data	Pengujian sukses, data yang terdapat dalam *database* ditampilkan berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Pengujian sukses, n buah data yang terdapat dalam *database* ditampilkan dan diurutkan sesuai parameter banyaknya data dan pengurutan	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pendapatan-perkapita/data &sortBy{ :key} &order{ [asc]/[desc] }	Pengujian sukses, data yang terdapat dalam *database* ditampilkan secara terurut *ascending* atau *descending* berdasarkan parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/ pendapatan-perkapita/ data/key={ :key } &get{ [max]/[min] }	Pengujian sukses, data maksimum atau minum yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
GET	{ Base URL }/pendapatan-perkapita/data/key=tahun &gd="{ :tahun }" &ld"{ :tahun }"	Pengujian sukses, rentang data yang terdapat dalam *database* ditampilkan sesuai parameter	*Route* sudah terintegrasi dengan *database*
PUT	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Pengujian sukses, memperbaharui seluruh *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*
PATCH	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Pengujian sukses, memperbaharui bebrapa *field* data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

Method	URL	Hasil yang Diharapkan	Kesimpulan
DELETE	{ Base URL }/ pendapatan-perkapita/ data/id={ :id }	Pengujian sukses, menghapus data dalam database berdasarkan parameter id	*Route* sudah terintegrasi dengan *database*

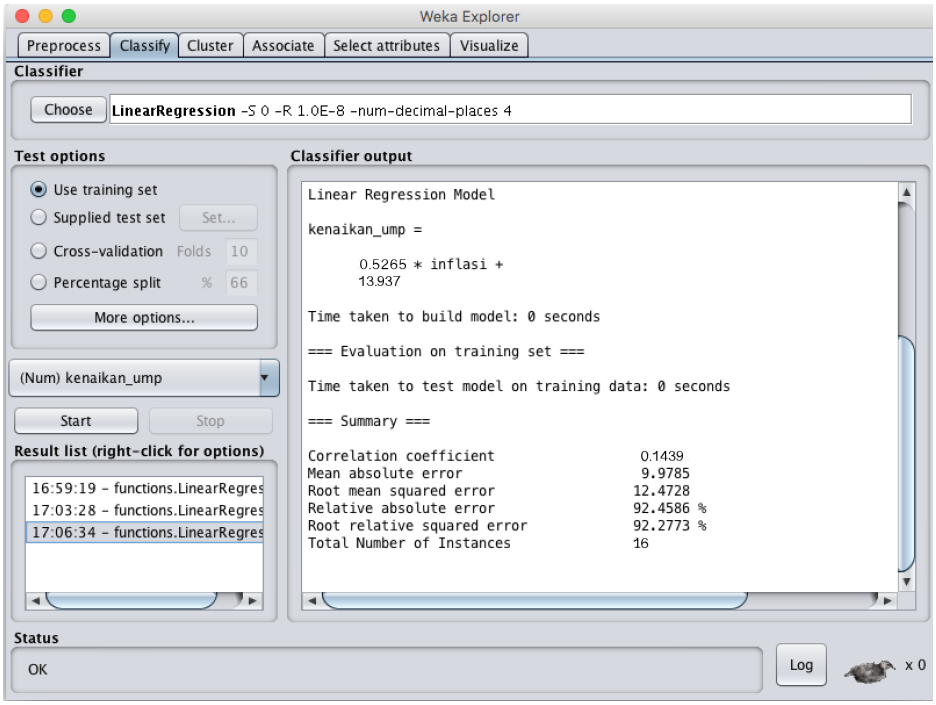
3. Hasil Pengujian Keluaran Algoritma

Berdasarkan rencana pengujian algoritma, keluaran data yang dihasilkan oleh sistem yang dibangun dan dengan yang dihasilkan oleh perangkat lunak Weka 3.8.x tidak jauh berbeda. Berikut adalah hasil dari pengujian algoritma.

a. Hasil Pengujian Algoritma *Linear Regression* untuk Data 2

Tabel IV-50 Hasil Pengujian Algoritma *Linear Regression* untuk Data 2

Variabel Uji	Hasil Implementasi	Hasil Weka v3.8.0	Kesimpulan
Independent Variable	inflasi	inflasi	Pengujian Terpenuhi
Dependent Variable	kenaikan_ump	kenaikan_ump	Pengujian Terpenuhi
Correlation Coefficient	0.1439	0.1439	Pengujian Terpenuhi
Model	$y = 0.5265x + 13.937$	$y = 0.5265x + 13.937$	Pengujian Terpenuhi

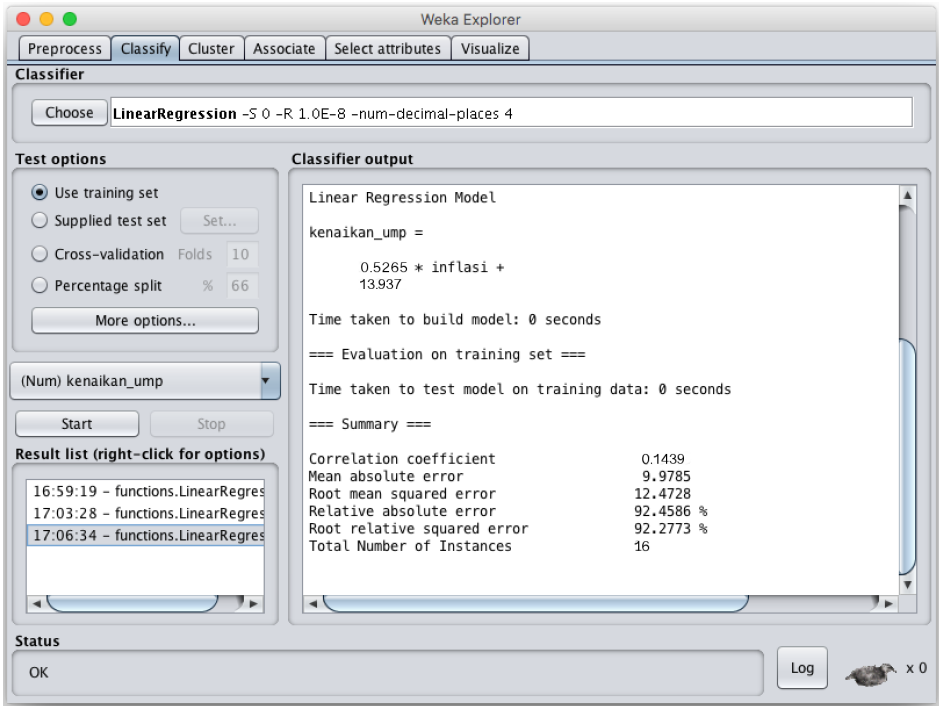


Gambar IV-3 Hasil Pengujian dengan Weka v3.8.x untuk Data 2

b. Hasil Pengujian Algoritma *Linear Regression* untuk Data 3

Tabel IV-51 Hasil Pengujian Algoritma *Linear Regression* untuk Data 3

Variabel Uji	Hasil Implementasi	Hasil Weka v3.8.0	Kesimpulan
Independent Variable	persen_tumbuh_jakarta	persen_tumbuh_jakarta	Pengujian Terpenuhi
Dependent Variable	persen_tumbuh_nasional	persen_tumbuh_nasional	Pengujian Terpenuhi
Correlation Coefficient	0.9078	0.9078	Pengujian Terpenuhi
Model	$y = 1.263x - 2.062$	$y = 1.2633x - 2.062$	Pengujian Terpenuhi



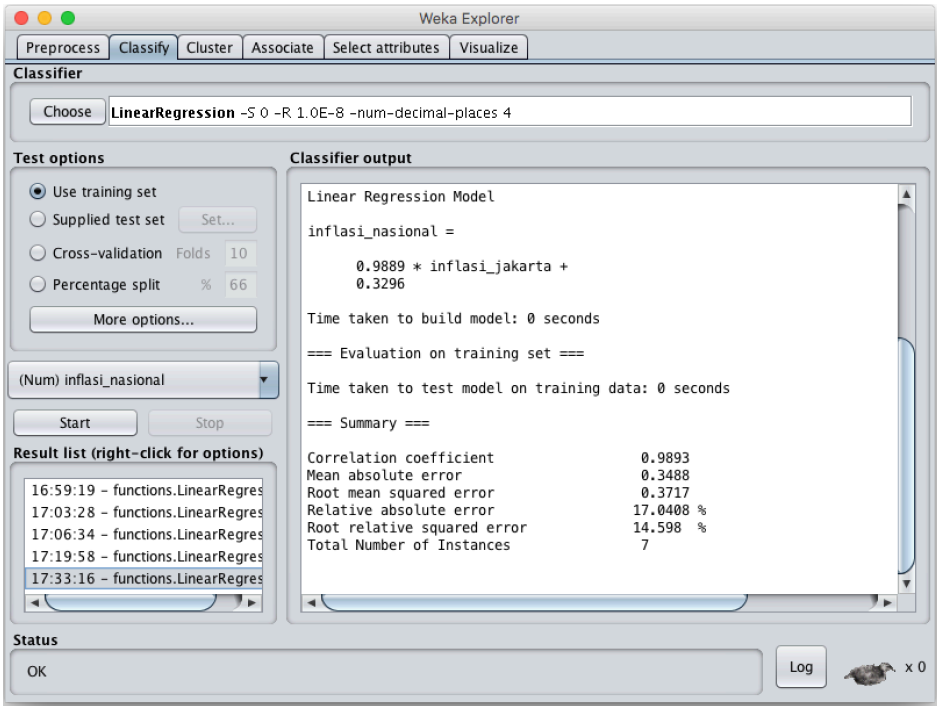
Gambar IV-4 Hasil Pengujian dengan Weka v3.8.x untuk Data 3

c. Hasil Pengujian Algoritma *Linear Regression* utnuk Data 4

Tabel IV-52 Hasil Pengujian Algoritma *Linear Regression* untuk Data 4

Variabel Uji	Hasil Implementasi	Hasil Weka v3.8.0	Kesimpulan
Independent Variable	inflasi_jakarta	inflasi_jakarta	Pengujian Terpenuhi
Dependent Variable	inflasi_nasional	inflasi_nasional	Pengujian Terpenuhi

Variabel Uji	Hasil Implementasi	Hasil Weka v3.8.0	Kesimpulan
Correlation Coefficient	0.9993	0.9893	Pengujian Terpenuhi
Model	$y = 0.9889x + 0.3296$	$y = 0.9889x + 0.3296$	Pengujian Terpenuhi

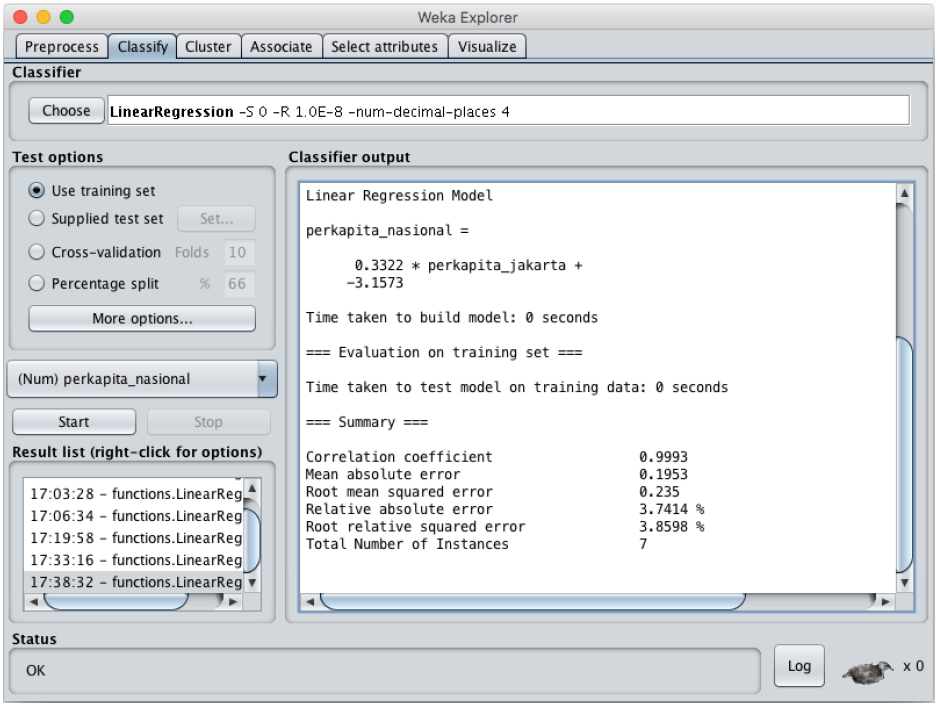


Gambar IV-5 Hasil Pengujian dengan Weka v3.8.x untuk Data 4

d. Hasil Pengujian Algoritma *Linear Regression* untuk Data 12

Tabel IV-53 Hasil Pengujian Algoritma *Linear Regression* untuk Data 12

Variabel Uji	Hasil Implementasi	Hasil Weka v3.8.0	Kesimpulan
Independent Variable	perkapita_jakarta	perkapita_jakarta	Pengujian Terpenuhi
Dependent Variable	perkapita_nasional	perkapita_nasional	Pengujian Terpenuhi
Correlation Coefficient	0.9892	0.9993	Pengujian Terpenuhi
Model	$y = 0.3322x - 3.1573$	$y = 0.3322x - 3.1573$	Pengujian Terpenuhi



Gambar IV-6 Hasil Pengujian dengan Weka v3.8.x untuk Data 12

e. Hasil Pengujian Algoritma *k*-means untuk Data 7

Berikut adalah banyaknya *instance* untuk setiap *cluster* yang dibandingkan antara keluaran *k-means* dari perhitungan Microsoft excel dengan algoritma *k*-means yang diintegrasikan dalam sistem.

Tabel IV-54 Banyaknya *Instance* Algoritma *k*-Means untuk Data 7

Cluster 1		**Cluster 2**		**Cluster 3**	
Excel	Integrasi	Excel	Integrasi	Excel	Integrasi
165 (85.05 %)	165 (85.05 %)	17 (8.76 %)	17 (8.76 %)	12 (6.19 %)	12 (6.19 %)

Tabel IV-55 Centroid Akhir Pengujian *k*-Means untuk Data 7

Cluster	Keluaran	*Feature*$_1$	*Feature*$_2$
Cluster 1	Excel	301900.4692	3933797.801
	Integrasi	301900.47	3933797.8
Cluster 2	Excel	68058.42812	138135.4892
	Integrasi	68058.43	138135.5
Cluster 3	Excel	565182.7665	138135.4892
	Integrasi	565182.77	1454812.0

Berdasarkan Tabel IV-54 dan Tabel IV-55, dapat diambil kesimpulan bahwa algoritma *k*-Means yang diintegrasikan dalam sistem sudah berjalan dengan semestinya.

4. Hasil Pengujian Beta

Berdasarkan rencana pengujian *beta* yang dilakukan, yaitu melakukan wawancara kepada *early adaptor user* yang sudah ditetapkan, Ibu Sri Dewi Anggadini, SE., M.Si, dosen tetap Fakultas Ekonomi di Universitas Komputer Indonesia (UNIKOM), berikut adalah hasil wawancara tersebut.

Tabel IV-56 Hasil Pengujian Beta Wawancara

Pertanyaan	Respon	Kesimpulan
Apakah aplikasi ini dapat menunjukkan seberapa besar sumber daya yang sudah digunakan?	"Aplikasi ini dapat menunjukan banyknya sumber daya yang sudah digunakan, tampilannya sangan sederhana dan mudah digunakan."	Tujuan terpenuhi
Apakah aplikasi ini dapat menunjukkan pergerakan variabel-variabel ekonomi?	"Tentu, variabel-variabel yang memang seharusnya ditunjukan pada data. Linear regression dan distribusi statistik adalah hal yang diperlukan seorang ekonom dalam mengambil keputusan, dan aplikasi ini dapat menunjukkan visualisasi itu."	Tujuan terpenuhi
Apakah aplikasi ini dapat membantu melihat distribusi harga?	"Aplikasi memang dapat menunjukan distribusi harga, hanya saja pemecahan rentang harganya kurang bagus. Hal ini juga diperlukan dalam pengambilan keputusan."	Tujuan terpenuhi
Apakah aplikasi visualisasi ini dapat memnunjukan laju pergerakan permintaan dan penawaran?	"Sayang, datanya tidak *up-to-date,* untuk permasalahan ini seharusnya data yang terbaru yang diperlihatkan. Jika mungkin data hari ini juga, karena sangat penting. Tetapi, data yang disediakan seperti ini juga, aplikasi ini sudah dapat	Tujuan terpenuhi

Pertanyaan	Respon	Kesimpulan
	menunjukkan pergerakan permintaan dan penawaran.	
Apakah aplikasi ini mudah digunakan?	"Ya, sangat mudah digunakan. Siapapun pasti tidak akan sulit menggunakannya. Tidak banyak tombol yang membuat bingung dan harus dipelajari. Tampilannya sangat sederhana dan cepat. *Good Job!*"	Tujuan terpenuhi

Setelah didapatkan hasil kuesioner, maka dilakukan perhitungan kuesioner dengan formula menghitung rata-rata sub skala untuk skala *likert*. Formulanya adalah sebagai berikut:

$$Nila\ Persentase = 100 * \frac{\text{jumlah skor penilaian responden dari setiap sub skala}}{\text{jumlah skor ideal setiap sub skala}}$$

Keterangan skor adalah sebagai berikut:

Tabel IV-57 Keterangan Skor

No.	Rentang Skor	Keterangan
1	80 ≤ Skor ≤ 100	Sangat Baik
2	60 ≤ Skor < 80	Baik
3	40 ≤ Skor < 60	Cukup
4	20 ≤ Skor < 40	Buruk
5	0 ≤ Skor < 20	Sangat Buruk

Tabel IV-58 Hasil Pengujian Beta Kuesioner Pelaku Ekonomi Makro

No. Responden	Pertanyaan				
	1	2	3	4	5
1	7	4	5	6	7
2	7	5	4	5	7
Σ	14	9	9	11	14
Rata-rata	100%	64.29%	64.29%	78.57%	100%
Kesimpulan	Sangat Baik	Baik	Baik	Baik	Sangat Baik

Tabel IV-59 Hasil Pengujian Beta Kuesioner Pelaku Ekonomi Mikro

No. Responden	Pertanyaan				
	1	2	3	4	5
1	6	6	5	6	5
2	5	5	6	6	4
3	3	4	6	3	1
4	6	3	4	4	2

No. Responden	Pertanyaan				
	1	2	3	4	5
5	4	3	4	5	5
6	5	5	5	5	5
7	5	6	6	6	6
8	6	6	6	6	6
9	3	2	6	4	3
10	5	5	5	6	5
Σ	47	48	52	52	54
Rata-rata	68.57%	64.29%	71.43%	77.14%	60%
Kesimpulan	Baik	Baik	Baik	Baik	Baik

Berdasarkan Tabel IV-58 dan Tabel IV-59 dapat disimpulkan bahwa pengujian sudah terpenuhi dengan baik.

4.2.4 Evaluasi Pengujian

Berdasarkan hasil pengujian unit, pengujian integrasi, pengujian algoritma, dan pengujian beta yang dilakukan, maka dapat disimpulkan bahwa:

1. Pada pengujian unit, setiap *method* yang diuji pada setiap *class* sudah berkerja sesuai tugas yang diharapkan.
2. Pada pengujian integrasi, setiap *route* yang diuji telah berintegrasi dengan baik dengan *database*.
3. Pada pengujian algoritma, keluaran algoritma yang dilakukan pada perangkat lunak weka v3.8.x sudah sama dengan keluaran algoritma yang diintegrasikan.
4. Pada pengujian beta, aplikasi dapat mencapai tujuan yang diharapkan berdasarkan pemaparan maksud dan tujuan yang ditulis pada Bab I.