W

;/--n

**TUGAS AKHIR – KI141502**

**RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN**

**HADRIAN BAYANULHAQ SIREGAR**

**NRP 5112100145**

**Dosen Pembimbing I**

**Waskhito Wibisono, S.Kom., M.Eng,. Ph.D**

**Dosen Pembimbing II**

**Royyana Muslim Ijtihadie, S.Kom., M.Kom,. Ph.D**

**JURUSAN TEKNIK INFORMATIKA**

**Fakultas Teknologi Informasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2016**

***[Halaman ini sengaja dikosongkan]***

**TUGAS AKHIR – KI141502**

**RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN**

**HADRIAN BAYANULHAQ SIREGAR**

**NRP 5112100145**

**Dosen Pembimbing I**

**Waskhito Wibisono, S.Kom., M.Eng,. Ph.D**

**Dosen Pembimbing II**

**Royyana Muslim Ijtihadie, S.Kom., M.Kom,. Ph.D**

**JURUSAN TEKNIK INFORMATIKA**

**Fakultas Teknologi Informasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2016**

****

***[Halaman ini sengaja dikosongkan]***

**UNDERGRADUATE THESES – KI1502**

**CLOUD BASED SENSOR VIRTUALIZATION SYSTEM FOR WIRELESS SENSOR MANAGEMENT**

**HADRIAN BAYANULHAQ SIREGAR**

**NRP 5112100145**

**Supervisor I**

**Waskhito Wibisono, S.Kom., M.Eng,. Ph.D**

**Supervisor II**

**Royyana Muslim Ijtihadie, S.Kom., M.Kom,. Ph.D**

**DEPARTMENT OF INFORMATICS**

**FACULTY OF INFORMATION TECHNOLOGY**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA 2015**

# LEMBAR PENGESAHAN

**RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN**

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat

Memperoleh Gelar Sarjana Komputer

pada

Bidang Studi Komputasi Berbasis Jaringan

Program Studi S-1 Jurusan Teknik Informatika

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Oleh

**HADRIAN BAYANULHAQ SIREGAR**

**NRP : 5112 100 145**

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Waskhito Wibisono, S.Kom,. M.Kom., …………………

Ph.D . (Pembimbing 1)

NIP:

1. Royyana Muslim Ijtihadie, S.Kom., ..………………

M.Kom., Ph.D. (Pembimbing 2)

NIP:

**SURABAYA**

**JUNI, 2016**

***[Halaman ini sengaja dikosongkan]***

**RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN**

|  |  |  |
| --- | --- | --- |
| **Nama Mahasiswa** | **:** | **HADRIAN BAYANULHAQ SIREGAR** |
| **NRP** | **:** | **5112100145** |
| **Jurusan** | **:** | **Teknik Informatika FTIF-ITS** |
| **Dosen Pembimbing 1** | **:** | **Waskhito Wibisono, S.Kom., M.Eng.,**  **Ph.D.** |
| **Dosen Pembimbing 2** | **:** | **Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.** |

# *Abstrak*

**SENSOR VIRTUALIZATION FOR CLOUD BASED SENSOR MANAGEMENT SYSTEM**

|  |  |  |
| --- | --- | --- |
| **Student’s Name** | **:** | **HADRIAN BAYANULHAQ SIREGAR** |
| **Student’s ID** | **:** | **5112100145** |
| **Department** | **:** | **Teknik Informatika FTIF-ITS** |
| **First Advisor** | **:** | **Waskhito Wibisono, S.Kom., M.Eng., Ph.D.** |
| **Second Advisor** | **:** | **Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.** |

# *Abstract*

# KATA PENGANTAR



Alhamdulillahirabbil’alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul ***“*RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN*”*.**

Pengerjaan Tugas Akhir ini merupakan salah satu dari sekian banyak kesempatan yang saya dapatkan, untuk mendapatkan ilmu dan pengalaman berharga selama saya berada di kampus Teknik Informatika ITS ini.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Ibu, Ibu, Ibu, Ayah dan Adik yang selalu memberikan do’a, dukungan, serta motivasi, sehingga penulis selalu termotivasi untuk menyelesaikan Tugas Akhir.
3. Bapak Waskhito Wibisono, S.Kom., M.Eng., Ph.D selaku pembimbing I yang selalu menyemangati dan memotivasi dengan ilmu-ilmu yang diluar dugaan saya.
4. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D selaku pembimbing II yang telah mengoreksi buku ini dengan cermat.
5. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
6. Teman – Teman Datangaja.com yang telah mengisi waktu-waktu penulis dengan tantangan, pengalaman, dan kesempatan bergabung dan merasakan atmosfir menantang dalam sebuah *startup*.
7. Teman – teman angkatan 2012, tanpa mereka, saya tidak akan merasakan apa itu yang dinamakan “Angkatan”.
8. Teman – teman Laboratorium Dasar Terapan Komputasi, yang telah mengajari saya banyak sekali hal secara tidak langsung.

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

Surabaya, Mei 2016

# DAFTAR ISI

[Abstrak vii](#_Toc451337864)

[*Abstract* viii](#_Toc451337865)

[KATA PENGANTAR xi](#_Toc451337866)

[DAFTAR ISI xiii](#_Toc451337867)

[DAFTAR GAMBAR xv](#_Toc451337868)

[DAFTAR TABEL xxi](#_Toc451337869)

[BAB I PENDAHULUAN 1](#_Toc451337870)

[1.1 Latar Belakang 1](#_Toc451337871)

[1.2 Rumusan Masalah 3](#_Toc451337872)

[1.3 Batasan Masalah 3](#_Toc451337873)

[1.4 Tujuan 4](#_Toc451337874)

[1.5 Manfaat 4](#_Toc451337875)

[1.6 Metodologi 4](#_Toc451337876)

[1.7 Sistematika Penulisan Laporan Tugas Akhir 6](#_Toc451337877)

[BAB II TINJAUAN PUSTAKA 9](#_Toc451337878)

[2.1 Mikrokontroler Arduino Uno 9](#_Toc451337879)

[2.2 Komputasi Awan 10](#_Toc451337880)

[2.3 Visualisasi Data 11](#_Toc451337881)

[2.4 Multitenancy 11](#_Toc451337882)

[2.5 JSON 12](#_Toc451337883)

[2.6 PHP 14](#_Toc451337884)

[2.7 Codeigniter 14](#_Toc451337885)

[2.8 MySQL 15](#_Toc451337886)

[BAB III PERANCANGAN PERANGKAT LUNAK 15](#_Toc451337888)

[3.1 Deskripsi Umum Sistem 17](#_Toc451337889)

[3.2 Arsitektur Umum Sistem 18](#_Toc451337890)

[3.3 Perancangan Diagram Kasus Penggunaan 20](#_Toc451337891)

[3.4 Perancangan Basis Data 22](#_Toc451337892)

[3.4.1 Tabel Sensor 22](#_Toc451337893)

[3.4.2 Tabel *Sensor Data* 23](#_Toc451337894)

[3.4.3 Tabel *users* 23](#_Toc451337895)

[3.4.4 Tabel *Sensor Collab* 24](#_Toc451337896)

[3.3.5 Tabel *Sensor Collab Data* 25](#_Toc451337897)

[3.3.6 Tabel *Sensor Rules* 25](#_Toc451337898)

[3.5 Prancangan *Web Service* 26](#_Toc451337899)

[3.6 Perancangan Antarmuka Sistem 26](#_Toc451337900)

[BAB IV IMPLEMENTASI 33](#_Toc451337901)

[4.1 Lingkungan Implementasi 33](#_Toc451337902)

[4.1.1 Lingkungan Implementasi Perangkat Keras 33](#_Toc451337903)

[4.1.2 Lingkungan Implementasi Perangkat Lunak 34](#_Toc451337904)

[4.2 Implementasi Perangkat Lunak 34](#_Toc451337905)

[4.2.1 Implementasi *Layer Sensor-Centric* 35](#_Toc451337906)

[4.2.2 Implementasi *Layer Middleware* 38](#_Toc451337907)

[4.2.2 Implementasi *Layer Client-centric* 44](#_Toc451337908)

[4.3 Implementasi Antarmuka Perangkat Lunak 46](#_Toc451337909)

[BAB V UJI COBA DAN EVALUASI 51](#_Toc451337910)

[5.1 Lingkungan Uji Coba 51](#_Toc451337911)

[5.2 Skenario Uji Coba 52](#_Toc451337912)

[5.2.1 Uji Coba Fungsionalitas 52](#_Toc451337913)

[5.2.2 Uji Coba Performa 74](#_Toc451337914)

[5.3 Evaluasi Hasil Uji Coba 85](#_Toc451337915)

[5.3.1 Evaluasi Hasil Uji Coba Performa 85](#_Toc451337916)

[BAB VI KESIMPULAN DAN SARAN 91](#_Toc451337917)

[6.1 Kesimpulan 91](#_Toc451337918)

[6.2 Saran 92](#_Toc451337919)

[DAFTAR PUSTAKA 93](#_Toc451337920)

# DAFTAR GAMBAR

[Gambar 2.1 11](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817143)

[Gambar 2.2 Diagram Arsitektur *Multitenant* 12](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817144)

[Gambar 2.3 Contoh notasi JSON 13](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817145)

[Gambar 3.1 Rancangan antarmuka halaman *login* 27](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817146)

[Gambar 3.2 Rancangan antarmuka halaman *dashboard* 28](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817147)

[Gambar 3.3 Rancangan antarmuka halaman *Create New Sensor* 29](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817148)

[Gambar 3.4 Rancangan antarmuka halaman *New Collaborative Sensor* 30](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817149)

[Gambar 3.5 Rancangan antarmuka halaman *View Sensor* 31](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817150)

[Gambar 3.6 Rancangan antarmuka halaman *Analyze* 32](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817151)

[Gambar 4.1 Implementasi fungsi *RetrieveData* 35](#_Toc451817152)

[Gambar 4.2 Implementasi fungsi *checkSensorKey* 36](#_Toc451817153)

[Gambar 4.3 Implementasi fungsi *checkCollab* ...........................36](#_Toc451817154)

[Gambar 4.4 Implementasi fungsi *insertData* 36](#_Toc451817155)

[Gambar 4.5 Implementasi fungsi *registerSensor* 37](#_Toc451817156)

[Gambar 4.6 Implementasi fungsi *registerSensor* 37](#_Toc451817157)

[Gambar 4.7 Implementasi fungsi *getAllSensorData* 38](#_Toc451817158)

[Gambar 4.8 Implementasi fungsi *getRealTimeSensorData* 39](#_Toc451817159)

[Gambar 4.9 Implementasi fungsi *getSensorReading* 39](#_Toc451817160)

[Gambar 4.10 Implementasi fungsi *getMaxSensorReading* 40](#_Toc451817161)

[Gambar 4.11 Implementasi fungsi *getMinSensorReading* 41](#_Toc451817162)

[Gambar 4.12 Implementasi fungsi *getAverageSensorReading* 41](#_Toc451817163)

[Gambar 4.13 Implementasi model *getMaxSensorReading* 42](#_Toc451817164)

[Gambar 4.14 Implementasi model *getMinSensorReading* 42](#_Toc451817165)

[Gambar 4.15 Implementasi model *getAverageSensorReading* 43](#_Toc451817166)

[Gambar 4.16 Implementasi fungsi *getMaxSensorReading* 44](#_Toc451817167)

[Gambar 4.17 Implementasi fungsi *viewSensor* 44](#_Toc451817168)

[Gambar 4.18 Implementasi fungsi *analyzeSensor* 45](#_Toc451817169)

[Gambar 4.19 Implementasi antarmuka halaman *login* 46](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817170)

Gambar 4.20 Implementasi antarmuka halaman *dashboard........*45

Gambar 4.21 Implementasi antarmuka halaman *View Sensor*.....45

[Gambar 4.22 Implementasi antarmuka halaman *New Collab* 48](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817171)

[Gambar 4.23 Implementasi antarmuka halaman *View Sensor* 49](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817172)

[Gambar 4.24 Implementasi antarmuka halaman *Analyze* 50](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817173)

[Gambar 5.1 Tampilan antarmuka halaman *View Sensor* 53](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817174)

[Gambar 5.2 Tampilan grafik *real time* pada halaman *View Sensor* 54](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817175)

[Gambar 5.3 *Console* yang menampilkan respon sukses dari *server* 55](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817176)

[Gambar 5.4 Tampilan antarmuka *view sensor* yang menampilkan *datatable* 57](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817177)

[Gambar 5.5 Tampilan antarmuka *Postman* yang menampilkan *raw data* hasil *request* dari *server*. 58](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817178)

[Gambar 5.6 Tampilan antarmuka *View sensor* yang sedang menampilkan data *real time* dari sebuah sensor 60](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817179)

[Gambar 5.7 Tampilan antarmuka *Postman* yang menampilkan data pada suatu waktu (*request* data *real time*) 60](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817180)

[Gambar 5.8 Tampilan antarmuka halaman *analyze* yang menampilkan data *Max* dari sebuah sensor 63](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817181)

[Gambar 5.9 Tampilan antarmuka *Postman* yang menampilkan data *Max* 64](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817182)

[Gambar 5.10 Tampilan antarmuka *Analyze* yang menampilkan data *Min* 64](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817183)

[Gambar 5.11 Tampilan antarmuka *Postman* yang menampilkan data *Min* 65](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817184)

[Gambar 5.12 Tampilan antarmuka *Analyze* yang menampilkan data *Average* 65](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817185)

[Gambar 5.13 Tampilan antarmuka *Postman* yang menampilkan data *Average* 66](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817186)

[Gambar 5.14 Tampilan *Console* yang menampilkan respon sukses dari *server* 67](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817187)

[Gambar 5.15 Tampilan antarmuka halaman *view sensor* yang menampilkan pesan alert “High Rule triggered!” 68](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817188)

[Gambar 5.16 Tampilan *console* yang berisi respon sukses dari *server*. 70](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817189)

[Gambar 5.17 Tampilan antarmuka halaman *View Collab* yang menampilkan data hasil kolaborasi. 71](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817190)

[Gambar 5.18 Tampilanhalaman *View Sensor* 72](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817191)

[Gambar 5.19 Tampilanhalaman *Analyze* 73](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817192)

[Gambar 5.20 Hasil pengujian *load test* 78](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817193)

[Gambar 5.21 Hasil pengujian *load test* 79](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817194)

[Gambar 5.22 Hasil pengujian *load test* 80](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817195)

[Gambar 5.23 Hasil pengujian *load test* 81](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817196)

[Gambar 5.24 Hasil pengujian *load test* 82](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817197)

[Gambar 5.25 Hasil pengujian *load test* 83](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817198)

[Gambar 5.26 Hasil pengujian *load test* 84](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817199)

[Gambar 5.27 Grafik kecepatan waktu penerimaan data terhadap jumlah sensor aktif 86](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817200)

[Gambar 5.28 Grafik kecepatan penerimaan dan pengolahan data kolaborasi sensor terhadap jumlah sensor aktif 87](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817201)

[Gambar 5.29 Grafik kecepatan penerimaan dan pengolahan data kolaborasi sensor pada keadaan 10 sensor aktif 88](file:///D:\Work%20and%20College\TA\511200145_Hadrian_Buku_TA_Draft.docx#_Toc451817202)

***[Halaman ini sengaja dikosongkan***

# DAFTAR TABEL

[Tabel 3.1 Perancangan tabel *sensor* 22](#_Toc451342329)

[Tabel 3.2 Perancangan tabel *sensor\_data* 23](#_Toc451342330)

[Tabel 3.3 Perancangan tabel *users* 23](#_Toc451342331)

[Tabel 3.4 Perancangan tabel *sensor\_collab* 24](#_Toc451342332)

[Tabel 3.5 Perancangan tabel *sensor\_collab\_data* 25](#_Toc451342333)

[Tabel 3.6 Perancangan tabel *sensor\_rules* 25](#_Toc451342334)

[Tabel 5.1 Prosedur uji coba pendaftaran sensor baru 52](#_Toc451342335)

[Tabel 5.2 Prosedur uji coba penerimaan data 53](#_Toc451342336)

[Tabel 5.3 Prosedur uji coba *data fetching* dengan fungsi *getAllSensorData* 55](#_Toc451342337)

[Tabel 5.4 Prosedur uji coba *data fetching* dengan fungsi *getRealTimeSensorData* 58](#_Toc451342338)

[Tabel 5.5 Prosedur uji coba pengolahan data. 61](#_Toc451342339)

[Tabel 5.6 Prosedur uji coba *rules* pada sensor 66](#_Toc451342340)

[Tabel 5.7 Prosedur uji coba kolaborasi sensor 69](#_Toc451342341)

[Tabel 5.8 Prosedur uji coba fungsi *viewSensor* 72](#_Toc451342342)

[Tabel 5.9 Prosedur uji coba fungsi *analyzeSensor* 73](#_Toc451342343)

[Tabel 5.10 Data percobaan penerimaan data. 75](#_Toc451342344)

[Tabel 5.11 Data percobaan penerimaan dan pengolahan data 75](#_Toc451342345)

[Tabel 5.12 Data percobaan pengiriman dan pengolahan data dengan 10 sensor aktif 76](#_Toc451342346)

***[Halaman ini sengaja dikosongkan]***

# BAB I PENDAHULUAN

## Latar Belakang

Aplikasi dari sensor tersebar telah digunakan dibanyak bidang penting seperti pengawasan lingkungan, pengawasan infrastruktur penting, pertanian, bahkan di bidang militer. Namun karena keterbatasan *memory,* komunikasi, komputasi, dan skalabilitas pada sensor tersebar*,* dibutuhkan metode yang efisien untuk mengelola data data dari sensor*.* Manajemen sensor ini membutuhkan infrastruktur dengan kemampuan komputasi yang besar dan *scalable* untuk mengelola, menyimpan, dan melakukan analisa pada data data tersebut. Teknologi komputasi awandapat menjadi solusi yang menjanjikan untuk masalah ini. Komputasi awan dapat menyediakan sumber daya komputasi, penyimpanan, dan layanan perangkat lunak secara fleksibel, dan dengan biaya yang relatif murah.

Komputasi awan (*cloud computing*) adalah sebuah sistem yang memungkinkan akses kepada sekumpulan sumber daya komputasi yang dapat diatur sesuai dengan kebutuhan pengguna dan bisa dengan cepat disediakan tanpa banyak pengaturan dan campur tangan penyedia. Sumber daya tersebut bisa berupa jaringan, *server*, penyimpanan (*storage*), aplikasi, dan layanan (*services*). [1]

Dengan teknologi komputasi awan ini, dapat di bangun sebuah sistem manajemen sensor dengan metode virtualisasi sensor. Virtualisasi sensor yang dimaksud adalah representasi data data hasil bacaan dari sensor fisik yang akan di tampilkan ke pengguna lewat HTTP . Representasi data tersebut berupa sebuah sensor virtual, dengan data data yang di dapatkan dari sensor sensor fisik yang tergabung di dalam suatu kumpulan sensoryang sebelumnya sudah di daftarkan oleh pemiliknya. Hasil akhir dari aplikasi ini berupa sebuah sistem yang dapat melakukan virtualisasi sensor, dan menyediakan beberapa layanan seperti *monitoring* dan visualisasi data hasil sensor. Infrastruktur atau sistem ini disebut sebagai *sensor cloud*.

Istilah Sensor-Cloud sendiri sebenarnya masih baru. Menurut IntelliSys, Sensor-Cloud sendiri dapat didefinisikan sebagai sebuah infrastruktur yang memungkinkan komputasi yang *pervasive* dengan menggunakan sensor sensor sebagai *interface* antara dunia nyata dan dunia may, dengan menggunakan data data sebagai *backbone* dan internet sebagai media komunikasi nya. [2, 3]. MicroStrain juga mendefenisikan Sensor-Cloud sebagai sebuah tempat penyimpanan data sensor yang unik, visualisasi, dan pengelolaan secara *remote* yang memanfaatkan kekuatan dari teknologi komputasi awan untuk menyediakan skalabilitas, visualisasi, dan analisis yang bisa di atur oleh pengguna nya. [1]

Sebuah infrastruktur Sensor-Cloud mengumpulkan dan memproses informasi dari beberapa sensor tersebar. Ini memungkinkan pengguna untuk berbagi informasi, dan berkolaborasi menggunakan aplikasi dengan basis komputasi awan. Dengan kata lain, Sensor-Cloud memungkinkan pengguna untuk mengumpulkan, mengakses, memproses, memvisualisasi, menganalisa, menyimpan, membagi, dan mencari data dalam jumlah yang besar dengan memanfaatkan teknologi komputasi awan.

## Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut :

1. Bagaimana cara mengumpulkan data dari sensor fisik untuk ditampilkan pada sensor virtual

2. Bagimana cara pemilihan data yang akan ditampilkan pada sensor virtual

3. Bagaimana cara mendesain arsitektur sistem yang baik

4. Bagaimana cara membuat abstraksi data dari kolaborasi antar sensor

## Batasan Masalah

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

1. Interaksi dengan pengguna berbasis *web*.

2. Pemilik sensor harus mengikuti protocol yang disediakan oleh sistem ini agar data dari sensor nya bisa di simpan dan di tampilkan pada aplikasi ini

3. Sistem dibangun dengan Bahasa PHP, dengan *web framework* Codeigniter dan menggunakan MySQL sebagai basis data

4. Uji coba akan dilakukan dengan mikrokontroler Arduino yang akan di *install* beberapa sensor.

5. Uji coba performa aplikasi akan dilakukan dengan mikrokontroler Arduino Uno, dan program Python yang akan mensimulasikan beberapa sensor.

## Tujuan

Tujuan dari pembuatan tugas akhir ini antara lain :

1. Membuat sistem yang dapat melakukan manajemen sensor berbasis komputasi awan.
2. Melakukan virtualisasi sensor.
3. Menyediakan layanan berupa visualisasi data, kolaborasi, dan data feed kepada pengguna yang dapat diaplikasikan sebagai pengawasan / *monitoring.*
4. Menyediakan layanan yang bersifat *multitenant*.

## Manfaat

Manfaat dari pembuatan tugas akhir ini, antara lain:

1. Mempermudah dalam manajemen sensor tersebar.
2. Memanfaatkan teknologi komputasi awan untuk membangun sistem yang fleksibel dan *scalable*.

## Metodologi

1. Penyusunan proposal tugas akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan pada proposal tugas akhir ini terdiri dari latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat hasil dari pembuatan tugas akhir. Selain itu dijelaskan pula tinjauan pustaka yang digunakan sebagai referensi pendukung implementasi tugas akhir. Pada proposal ini juga terdapat perencanaan jadwal pengerjaan tugas akhir.

1. Studi literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam implementasi sistem, yaitu mengenai mikrokontroler arduino, bahasa pemrograman PHP, kerangka kerja *web* Codeigniter, JSON, dan MySQL

1. Analisis dan desain perangkat lunak

Tahap ini meliputi perancangan sitem berdasarkan studi literature dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat prototype sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

1. Implementasi perangkat lunak

Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

1. Pengujian dan evaluasi

Pada tahapan ini dilakukan uji coba pada alat yang telah dirancang. Tahapan ini dimaksudkan untuk mengevaluasi tingkat akurasi dari alat tersebut serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

1. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat

## Sistematika Penulisan Laporan Tugas Akhir

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

**Bab I Pendahuluan**

Bab yang berisi latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

**Bab II Dasar Teori**

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

**Bab III Perancangan Perangkat Lunak**

Bab ini berisi implementasi dari perancangan perangkat lunak yang telah dibuat pada bab sebelumnya. Implementasi berupa *pseudocode* dari fungsi utama dan *screenshot* perangkat lunak.

**Bab IV Implementasi**

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa code yang digunakan untuk proses implementasi.

**Bab V Uji Coba Dan Evaluasi**

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

**Bab VI Kesimpulan Dan Saran**

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

***[Halaman ini sengaja dikosongkan]***

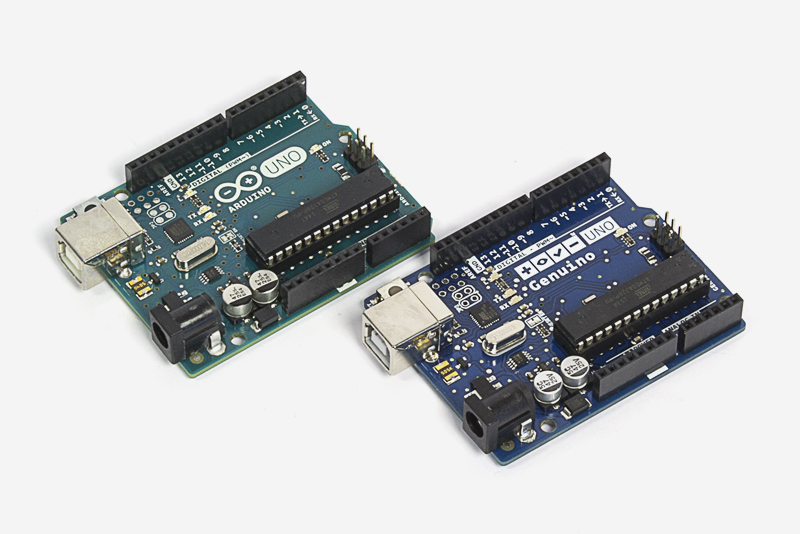
# BAB II TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan rancangan alat yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap alat yang dirancang dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

## Mikrokontroler Arduino Uno

Mikrokontroler Arduino Uno [4] merupaka sebuah kit elektronik yang bersifat *open source*. Mikrokontroller Arduino ini menggunakan chip mikrokontorler AVR dari perusahaan Atmel. Mikrokontroler itu sendiri adalah sebuah *chip* yang bisa di program menggunakan perangkat komputer. Mikrokontroler ada pada perangkat elektronik yang kita pakai sehari-hari, misalnya pada AC, TV, pemutar DVD, dll. Mikrokontroler juga dipakai untuk mengendalikan robot.

Arduino Uno merupakan jenis dari mikrokontroler Arduino yang paling banyak digunakan, terutama untuk pemula yang masih ingin melakukan eksplorasi pada mirkokontroler. Versi terakhir Arduino Uno adalah R3, dengan ATMEGA328 sebagai mikrokontroler nya. Arduino Uno memiliki 14 pin I/O digital dan 6 pin I/O analog .

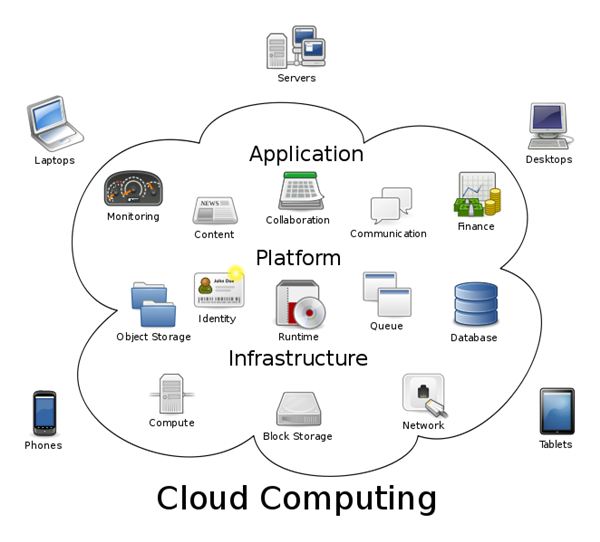


Gambar 2.1 Arduino Uno[4]

## Komputasi Awan

Komputasi awan [5] adalah sebuah gabungan dari dua teknologi, yaitu teknologi komputasi, dan pengembangan berbasis internet (awan). “Awan” merupakan metafora dari internet, seperti yang sering digambarkan pada diagram jaringan komputer. Awan dalam diagram jaringan komputer merepresentasikan infrastruktur kompleks yang disembunyikannya.

Teknologi komputasi awan merupakan sebuah teknologi yang menggunakan internet sebagai pusat *server* untuk mengelola data dan juga aplikasi pengguna. Teknologi ini mengizinkan pada pengguna untuk menjalankan program tnapa instalasi dan mengizinkan pengguna untuk mengakses data data mereka melalui komputer dimana saja, selama mempunyai akses internet.



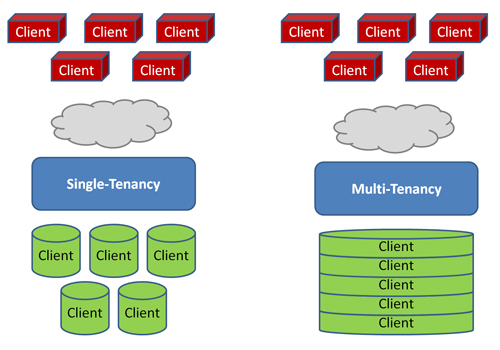
Gambar 2.2 Diagram ilustrasi Komputasi Awan [5]

## Visualisasi Data

Visualisasi data [5] merupakan sebuah komunikasi visual modern yang digunakan oleh banyak bidang ilmu. Visualisasi data tidak berada di bawah bidang manapun, melainkan interpretasi di antara banyak bidang. Visualisasi data mengikutkan pembuatan dan kajian dari representasi visual data, yang artinya meruakan suatu “informasi yang telah di abstraksikan dalam bentuk skematis, termasuk atribut atau variabel dari unit informasi”.

## Multitenancy

Pada bidang rekayasa perangkat lunak (*software engineering*), *multitenancy* mmerujuk pada sebuah arsitektur perangkat lunak dimana sebuah *instance* dari sebuah perangkat lunak yang berjalan pada suatu *server* bisa melayani lebih dari satu *tenant* dalam suatu waktu. *Tenant* itu sendiri merupakan sekumpulan atau suatu pengguna yang mempunyai akses yang sama pada sebuah perangkat lunak atau layanan, dengan hak akses yang spesifik pada masing masing pengguna dalam *group* tersebut. Dengan arsitektur *multitenant* ini, sebuah layanan di desain agar bisa menyediakan semua fungsionalitas dari sebuah layanan kepada semua *tenant* secara penuh.



Gambar 2.3 Diagram Arsitektur *Multitenant [6]*

## JSON

JSON [7] merupakan singkatan dari *Javascript Object Notation*. JSON merupakan sebuah format data yang mudah diolah oleh berbagai bahasa pemrograman berbeda. JSON memudahkan manusia untuk membaca dan menulis dan memudahkan mesin untuk mengurai dan menghasilkan data. JSON merupakan sebuah format teks yang independen, namun memiliki pustaka yang dapat dikenali oleh berbagai macam bahasa pemrograman, seperti C++, PHP, Python, Ruby, Perl, dan lain-lain. JSON dibangun dalam dua bentuk, antara lain :

1. Sebuah objek, merupakan sekumpulan pasangan nama dan nilai. Sebuah objek dimulai dengan karakter “{“ dan diakhiri dengan karakter “}”. Setiap nama diikuti oleh karakter “:” dan setiap elemen JSON yang berisi nama dan nilai dipisahkan dengan karakter “,”.
2. Sebuah nilai yang berurutan, dalam bahasa pemrograman dikenal dengan istilah *array vector,* atau *list*. Sebuah *array* dimulai dengan karakter “[“ dan diakhiri dengan karakter “]”. Setiap nilai yang berbeda dipisahkan dengan tanda koma.



Gambar 2.4 Contoh notasi JSON

## PHP

PHP [8] merupakan sebuah bahasa pemrograman yang berjalan pada HTTP *server* dan digunakan pada pengembangan *web*. Selain itu, PHP juga dapat digunakan untuk membuat berkas XML dan melakukan pengolahan basis data seperti menyeleksi, menambah, membaca, menghapus, maupun memperbaharui data yang ada pada basis data tersebut. PHP biasanya dikombinasikan dengan HTML (*Hyptertext Markup Language)* untuk mengasilkan interaksi pengguna dari sebuah aplikasi web.

Pada tugas akhir ini, PHP digunakan untuk membangun sistem virtualisasi sensor untuk manajemen sensor tersebar. Sistem dibuat seutuhnya dengan PHP.

## Codeigniter

Codeigniter (CI) [9] adalah *framework* pengembangan aplikasi (*Application Development Framework*) berbasis bahasa pemrograman PHP. CI merupakan suatu kerangka kerja *web* untuk membuat aplikasi *web* dengan bahasa pemrograman PHP yang lebih sistematis dan cepat dalam hal pengembangannya. CI menggunakan pola pengembangan MVC (*Model – View – Controller*)[4-nisaa].

* *View,* merupakan bagian yang menangani *presentation* *logic*. Pada suatu aplikasi *web*, bagian ini biasanya berupa file *template* HTML, yang diatur oleh *Controller*. *View* berfungsi untuk menerima dan merepresentasikan data yang di susun oleh *Controller* kepada pengguna. Bagian ini tidak mempunyai akses langsung pada bagian *Model*.
* *Model*, merupakan bagian yang berhubungan langsung dengan basis data dari suatu aplikasi. Karena bagian ini berhubungan langsung dengan basis data, *Model* biasanya digunakan untuk melakukan manipulasi data seperti CRUD (*Create, Read, Update, Delete*)
* *Controller*, merupakan bagian yang mengatur hubungan antar *model* dan *view*. *Controller* berfungsi menerima dan memproses *request* yang di terima dari *web browser* pengguna. [9]

## MySQL

MySQL [10] merupakan sebuah RDBMS (*Relational Database Management System*) yang bersifat *open source*. MySQL merupakan pilihan yang cukup populer untuk sistem basis data, dan merupakan sebuah komponen utama pada LAMP (Linux, Apache, MySQL, PHP/Perl/Python) *stack*. Beberapa aplikasi web yang menggunakan MySQL adalah Joomla, Wordpress, phpBB, Drupal, Facebook, Twitter, Flickr, dan Youtube. [11][12]



## Web Service

*Web Service* [13] merupakan antarmuka yang menggambarkan operasi-operasi yang bisa di akses dalam jaringan, dan mengembalikan respon dalam bentuk XML atau JSON. Antarmuka *web service* menyembunyikan detail implementasi dari layanan, yang memungkinkan pengunaannya secara independen dari *platform hardware* atau *software* dimana *web service* itu di implementasikan.

***[Halaman ini sengaja dikosongkan]***

# BAB III PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dijelaskan mengenai analisis dan perancangan perangkat lunak yang akan dikembangkan. Perancangan merupakan bagian penting dari pengembangan perangkat lunak karena merupakan perencanaan perangkat lunak secara teknis. Adapun hal-hal yang dibahas dalam bab ini adalah deskripsi umum perangkat lunak, arsitektur perangkat lunak, diagram kasus penggunaan, perancangan basis data, diagram alur, dan desain antar muka perangkat lunak.

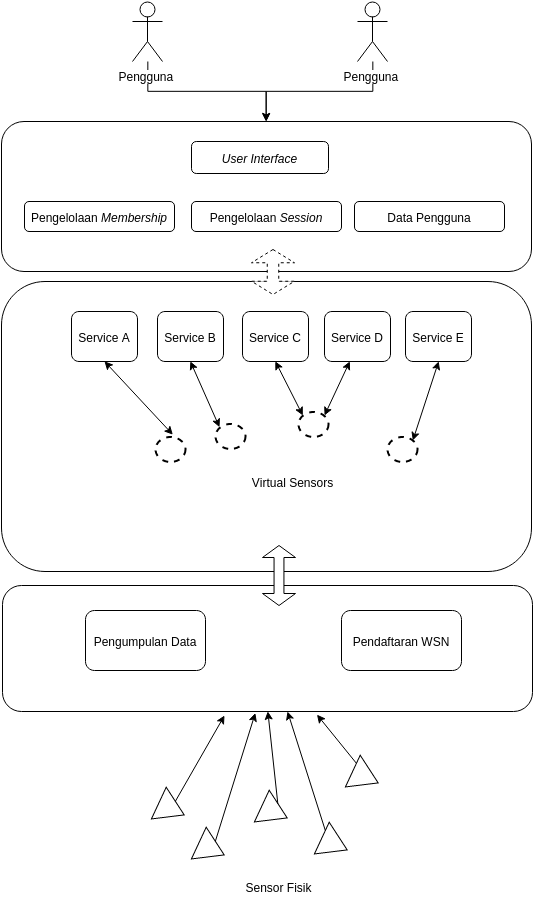
3.1 Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun sebuah sistem yang mampu melakukan virtualisasi sensor fisik untuk keperluan manajemen sensor tersebar. Pengguna bisa berupa aplikasi yang memerlukan akses data ke sebuah sensor tanpa harus membuat *server* data nya sendiri, atau orang yang ingin mengawasi data sensor yang dimiliknya (misalnya sensor di rumah, kebun, dll).

Pada aplikasi ini, pengguna dapat menggunakan sensor yang terpasang pada mikrokontroler Arduino, maupun Raspberry Pi selama mengikuti protokol yang disediakan oleh sistem. Data dari snesor yang tergabung dalam sistem akan di *record* ke dalam basis data, dan pengguna bisa melakukan pengawasan dan analisa terhadap data data tersebut.

Terdapat fitur kolaborasi sensor, yang berarti data dari dua atau tiga sensor bisa di kolaborasikan (bisa melakukan operasi matematika terhadap data data antar sensor tersebut).

3.2 Arsitektur Umum Sistem

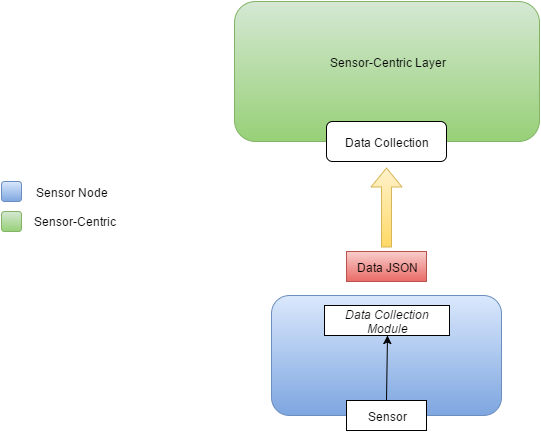
Perangkat lunak yang akan di bangun terdiri dari tiga *layer* seperti pada diagram dibawah

Gambar 3.1 Diagram Arsitektur Umum Sistem

*Layer* pertama bersifat *client-centric*. *Layer* ini meruipakan “jembatan” antara sistem dan pengguna (antarmuka pengguna). Komponen komponen pada *layer* ini memfasilitasi dan mengelola interaksi pengguna dengan sistem, mengelola *membership* dan *session* dari tiap pengguna yang terdaftar pada sistem. Hak akses pada layer ini juga di atur pada *layer* ini.

*Layer* kedua adalah *middleware layer*. *Layer* ini berfungsi sebagai penghubung antara *client-centric layer* dan *sensor-centric layer*. Proses virtualisasi sensor fisik yang tergabung pada sistem akan dilakukan pada *layer* ini.

*Layer* ketiga adalah layer yang bersifat *sensor-centric,* artinya *layer* ini berhubungan langsung dengan sensor sensor fisik yang terdaftar pada sistem. Beberapa fungsi yang dijalankan oleh *layer* ini salah satunya adalah pengumpulan data dari sensor fisik. Agar bisa melakukan pengumpulan data, *node* sensor fisik harus mengikuti protokol yang ditentukan oleh sistem. Pengiriman data dari sensor fisik ke sistem dilakukan dalam format JSON



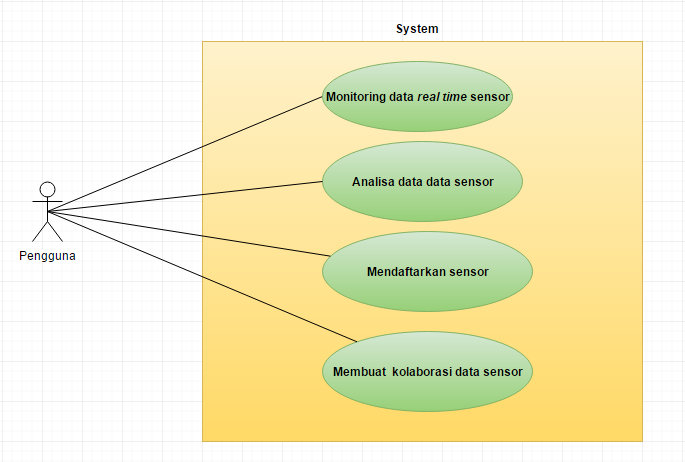
Gambar 3.2 Diagram *Data Collection Module*

Modul pengumpulan data (*Data Collection Module*) merupakan modul yang harus di *install* pada *node* sensor yang akan mengirimkan data ke sistem. *Module* ini berfungsi untuk melakukan enkapsulasi data yang telah di baca oleh sensor ke format JSON, dan mengirimkan data tersebut ke modul pengumpulan data tyang ada pada sistem.

1. Perancangan mikrokontroler arduino dan integrasi sensir

* 1. Perancangan Diagram Kasus Penggunaan

Diagram kasus penggunaan menggambarkan peran aktor yang terlibat dalam fungsionalitas perangkat lunak yang dibangun. Adapun perancangan diagram kasus perangkat lunak dalam Tugas Akhir ini dapat dilihat pada gambar dibawah.



Gambar 3.3 Diagram kasus penggunaan perangkat lunak

Pada Tugas Akhir ini, aktor yang merupakan pengguna sistem manajemen sensor ini berperan sebagai pengguna, dan pemilik sensor tersebar yang ada pada sistem. Pengguna dapat menjalankan fitur yang ada pada perangkat lunak, diantaranya :

* *Monitoring* data *real time* sensor

Pengguna dapat menampilkan dan mengawasi data sensor yang dimiliknya secara *real time* melalui antar muka sistem yang berbasis *web.* Data yang ditampilkan berupa grafik / *chart* data sensor dengan *series* berupa waktu, dan *value* dari hasil bacaan sensor.

* Analisa data data sensor

Pengguna dapat melakukan analisa data-data sensor yang telah dikumpulkan sebelumnya. Analisa berupa rerata, maksimum, dan minimum dari data-data sebuah sensor per hari nya.

* Mendaftarkan sensor

Pengguna juga bertindak sebagai pemilik sensor, artinya pengguna dapat mendaftarkan sensor fisik yang dimilikinya agar bisa memanajemen sensor fisiknya dari internet.

* Membuat kolaborasi data sensor

Pengguna bisa membuat kolaborasi antar dua atau tiga sensor. Data data sensor yang dikolaborasikan bisa di jumlah, kurang, bagi, dan kali kan. Fitur analisa dan *monitoring* *real time* data juga berlaku pada sensor yang dikolaborasi kan, karena sensor yang dikolaborasikan di anggap sebagai satu sensor.

* 1. Perancangan Basis Data

Pada perancangan basis data merupakan tahapan untuk merancang basis data yang akan digunakan pada Tugas Akhir ini. Basis data ini berfungsi untuk menyimpan data-data sensor, data pengguna, dan data-data *rules* sensor.

3.4.1 Tabel Sensor

Tabel 3.1 Perancangan tabel *sensor*

|  |  |  |  |
| --- | --- | --- | --- |
| No | Nama Atribut | Tipe Data | Keterangan |
| 1 | *id* | *integer* | Sebagai *primary key* untuk sensor |
| 2 | *user\_id* | *integer* | Sebagai *id* dari akun pemilik sensor |
| 3 | *sensor\_key* | *varchar* | Sebagai *token* dari sebuah sensor agar bisa mengirim data ke sistem |
| 4 | *sensor\_type* | *varchar* | Tipe sensor, bisa berupa sensor biasa atau GPS |
| 5 | *sensor\_name* | *varchar* | Nama sensor yang di daftarkan |
| 6 | *sensor\_description* | *varchar* | Deskripsi dari sensor |
| 7 | *created\_on* | *timestamp* | Tanggal dan waktu di daftarkan nya sensor |
| 8 | *last\_updated* | *timestamp* | Tanggal dan waktu sensor terakhir kali di update |
| 9 | *status* | *varchar* | Status dari sensor. |

* + 1. Tabel *Sensor Data*

Tabel 3.2 Perancangan tabel *sensor\_data*

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Nama Atribut | Tipe Data | Keterangan |
| 1 | *id* | *integer* | *Primary key* dari sebuah data bacaan sensor |
| 2 | *sensor\_id* | *integer* | *id* dari sensor yang mengirim data |
| 3 | *sensor\_reading* | *float* | data hasil bacaan sensor yang dikirim |
| 4 | *timestamp* | *timestamp* | Waktu data masuk ke sistem |

* + 1. Tabel *users*

Tabel 3.3 Perancangan tabel *users*

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Nama Atribut | Tipe Data | Keterangan |
| 1 | *id* | *int* | *Primary key* dari tabel *users*. |
| 2 | *ip\_address* | *varchar* | Alamat *IP* dari pengguna ketika login terakhir |
| 3 | *username* | *varchar* | *username* dari pengguna yang digunakan untuk login ke sistem |
| 4 | *password* | *varchar* | *password* pengguna yang digunakan utnuk login ke sistem |
| 5 | *email* | *varchar* | *email* yang digunakan saat pendaftaran |
| 6 | *created\_on* | *integer* | Waktu pembuatan / pendaftaran user dalam *unix time* |
| 7 | *last\_login* | *integer* | Waktu login terakhir pengguna dalam *unix time*. |

* + 1. Tabel *Sensor Collab*

Tabel 3.4 Perancangan tabel *sensor\_collab*

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Nama Atribut | Tipe Data | Keterangan |
| 1 | *sensor\_collab\_id* | *integer* | *Primary key* dari tabel *sensor\_collab\_id* |
| 2 | *user\_id* | *integer* | Pemilik dari sensor kolaborasi |
| 3 | *sensor\_collab\_name* | *varchar* | Nama dari sensor kolaborasi |
| 4 | *sensor\_collab\_desc* | *text* | Deskripsi dari sensor kolaborasi |
| 5 | *sensor\_x\_id* | *integer* | *id* dari sensor yang akan di kolaborasikan |
| 6 | *sensor\_y\_id* | *integer* | *id* dari sensor yang akan di kolaborasikan |
| 7 | *sensor\_z\_id* | *integer* | *id* dari sensor yang akan di kolaborasikan |
| 8 | *sensor\_x\_rule\_id* | *integer* | *id* dari *rule* sensor yang akan di kolaborasikan |
| 9 | *sensor\_y\_rule\_id* | *integer* | *id* dari *rule* sensor yang akan di kolaborasikan |
| 10 | *sensor\_z\_rule\_id* | *integer* | *id* dari *rule* sensor yang akan di kolaborasikan |
| 11 | *comp\_operator* | *varchar* | *operator* komparasi antar sensor (AND , OR) |
| 12 | *operator* | *varchar* | *operator* matematika untuk operasi data antar sensor (penjumlahan, pengurangan, pembagian, perkalian) |

3.3.5 Tabel *Sensor Collab Data*

Tabel 3.5 Perancangan tabel *sensor\_collab\_data*

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Nama Atribut | Tipe Data | Keterangan |
| 1 | *sensor\_collab\_data\_id* | *integer* | *Primary key* dari dari data hasil kolaborasi |
| 2 | *sensor\_x\_value* | *float* | Data bacaan salah satu sensor kolaborasi |
| 3 | *sensor\_y\_value* | *float* | Data bacaan salah satu sensor kolaborasi |
| 4 | *sensor\_z\_value* | *float* | Data bacaan salah satu sensor kolaborasi |
| 5 | *sensor\_collab\_id* | *integer* | *id* dari sensor kolaborasi |
| 6 | *sensor\_reading* | *float* | Data hasil operasi kolaborasi dari dua atau tiga sensor yang dikolaborasikan |
| 7 | *timestamp* | *timestamp* | Waktu data masuk. |

3.3.6 Tabel *Sensor Rules*

Tabel 3.6 Perancangan tabel *sensor\_rules*

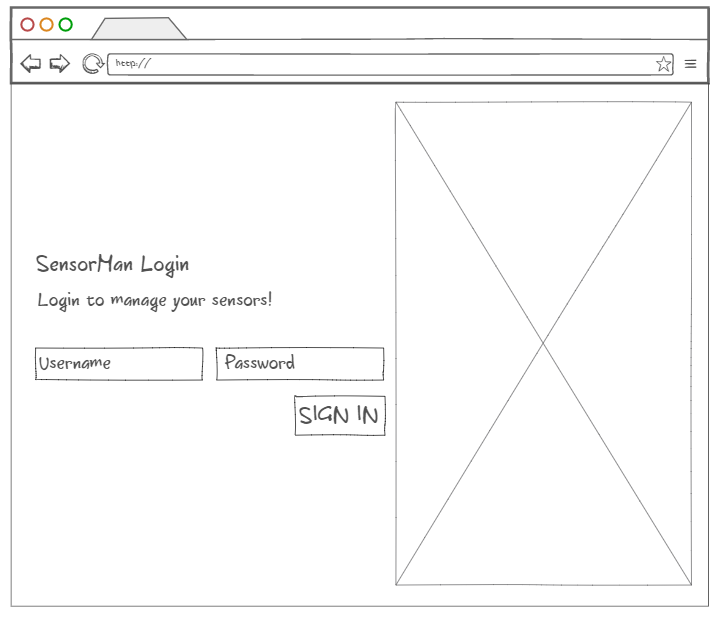
|  |  |  |  |
| --- | --- | --- | --- |
| No. | Nama Atribut | Tipe Data | Keterangan |
| 1 | *rule\_id* | *integer* | *Primary key* dari *rules* sensor |
| 2 | *sensor\_id* | *integer* | *id* dari sensor yang memiliki rule |
| 3 | *rule\_type* | *varchar* | Tipe dari *rule* sensor (*high, low, equal*) |
| 4 | *rule\_value* | *float* | Batas nilai / *threshold* dari *rule* sensor. |

3.5 Prancangan *Web Service*

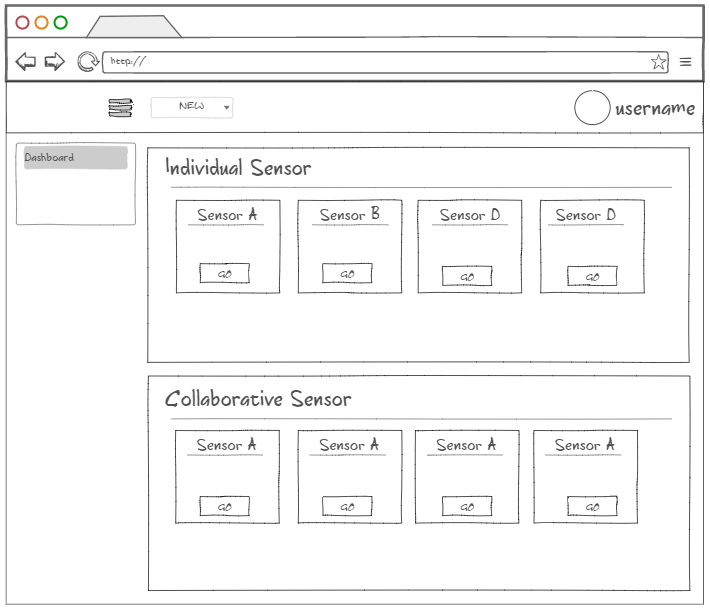
Web service pada Tugas Akhir ini digunakan untuk menerima data dari *node* sensor fisik yang terdaftar. *Web service* yang dirancang merupakan *web service* sederhanda dengan memanfaatkan metode HTTP POST untuk mengirimkan data ke sistem dalam format JSON. Parameter yang ditangkap oleh *web service* ini adalah *sensor\_key* dan *sensor\_reading*. Kemudian data *sensor\_reading* akan di simpan pada tabel *sensor\_data* di basis data.

3.6 Perancangan Antarmuka Sistem

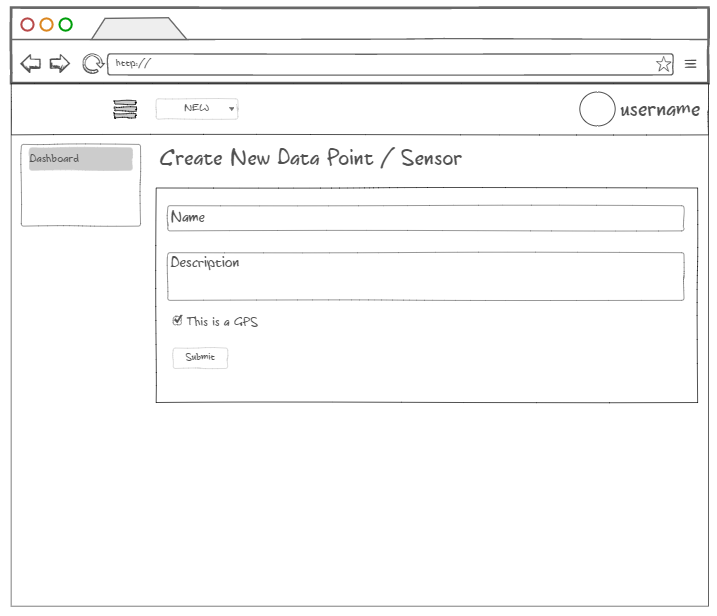
Pada Tugas Akhir ini, antar muka perangkat lunak berupa halaman *web* yang bertujuan untuk segala aktivitas pengguna, baik *monitoring*, analisa, dan pendaftaran. Berikut adalah gambar rancangan antar muka sistem berupa *wireframe*

**

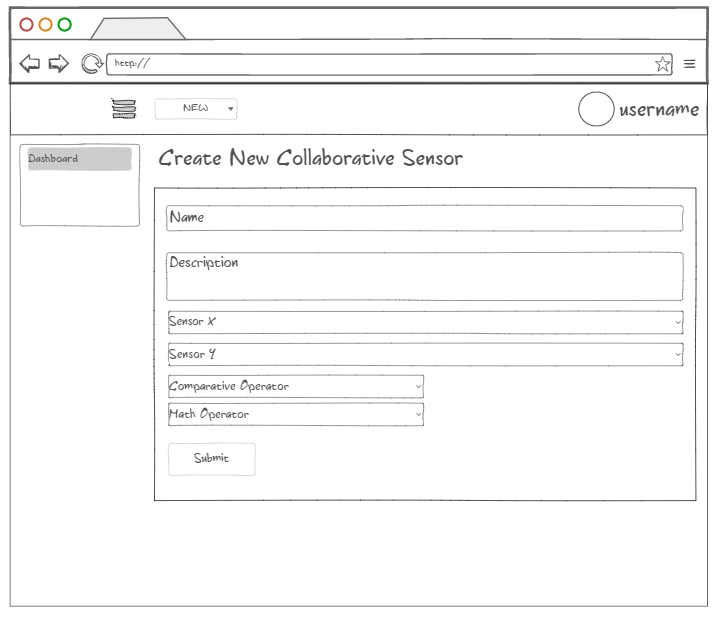
Gambar 3.1 Rancangan antarmuka halaman *login*

**

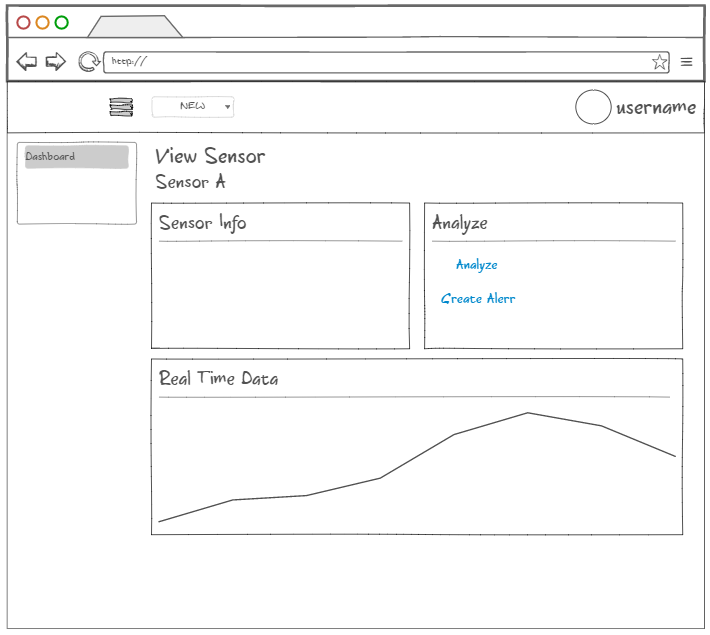
Gambar 3.2 Rancangan antarmuka halaman *dashboard*



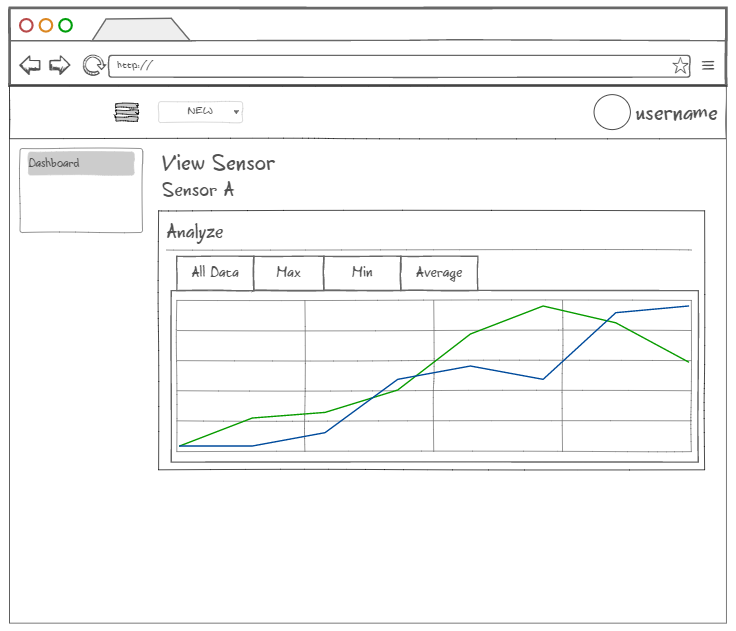
Gambar 3.3 Rancangan antarmuka halaman *Create New Sensor*



Gambar 3.4 Rancangan antarmuka halaman *New Collaborative Sensor*



Gambar 3.5 Rancangan antarmuka halaman *View Sensor*



Gambar 3.6 Rancangan antarmuka halaman *Analyze*

# BAB IV IMPLEMENTASI

Pada bab ini akan dijelaskan tentang implementasi perangkat lunak yang dikembangkan untuk Tugas Akhir ini. Implementasi perangkat lunak merupakan bentuk realisasi dari perancangan perangkat lunak yang telah di jelaskan pada bab sebelumnya. Adapun hal-hal yang akan dibahas pada bab ini adalah lingkungan implementasi perangkat lunak, *pseudocode* dari perangkat lunak, dan *screenshot* hasil implementasi perangkat lunak.

4.1 Lingkungan Implementasi

Pada Tugas Akhir ini, lingkungan untuk mengembangkan perangkat lunak ini terdiri atas perangkat keras dan perangkat lunak. Adapun lingkungan implementasi adalah sebagai berikut :

* + 1. Lingkungan Implementasi Perangkat Keras

Lingkungan implementasi perangkat keras yang digunakan pada Tugas Akhir ini adalah sebuah *laptop* dan mikrokontroler Arduino Uno R3 untuk keperluan pengujian. Adapun spesifikasi dari perangkat keras tersebut adalah sebagai berikut :

* Laptop Apple MacBook Pro
  + OSX 10 El Capitan 10.11.12
  + Processor 2.6 GHz Intel Core i5
  + Memory 8 GB 1600 MHz DDR3
* Arduino Uno R3
  + HanRun HR911105A 15/10 *Ethernet Module*

* + 1. Lingkungan Implementasi Perangkat Lunak

Lingkungan implementasi perangkat lunak yang digunakan untuk menunjang pengembangan perangkat lunak ini terdiri dari sebuah sistem operasi, IDE, dan beberapa pustaka. Adapun lingkungan implementasi perangkat lunak tersebut adalah sebagai berikut :

* OSX 10 El Capitan 10.11.12 sebagai sistem operasi
* Ninjamock[web] sebagai alat bantu perancangan antarmuka perangkat lunak yang berupa *web*.
* Draw.io[web] sebagai alat bantu perancangan diagram kasus penggunaan perangkat lunak.
* SublimeText sebagai IDE utama dalam pengembangan perangkat lunak.
* Arduino IDE Versi 1.6.8 sebagai IDE dalam pengembangan modul koleksi data pada mikrokontroler Arduino Uno R3 untuk keperluan pengujian.
* MySQL versi 5.6 sebagai RDBMS (*Relational Database Management System*) untuk menyimpan data dan informasi sensor.
* Apache 2.4.17 sebagai *platform webserver* untuk *deployment* situs *web* secara lokal.
* HighCharts sebagai pustaka Javascript utama untuk menampilkan hasil olahan data sebagai grafik.
* Ion\_auth sebagai pustaka untuk *authentication* dan *authorization* pada perangkat lunak.

4.2 Implementasi Perangkat Lunak

Pada subbab ini akan dijelaskan implementasi yang telah dilakukan pada sistem. Berikut akan dijelaskan mengenai implementasi perangkat lunak berupa *web*.

4.2.1 Implementasi *Layer Sensor-Centric*

Bagian ini menjelaskan bagaimana *server* menerima parameter data yang dikirim oleh sensor fisik. *Web Service* yang di implementasikan pada *layer ini* berupa sebuah *method* yang meggunakan HTTP POST untuk menerima data dalam format JSON, yang kemudian akan di *parse* dan di masukkan kedalam basis data. Prosedurnya adalah sebagai berikut :

1. Membuat *method* yang menunggu data POST masuk.
2. Jika ada data masuk, data JSON di *parse* kemudian dilakukan *checking* terhadap parameter *sensor\_key* untuk mengetahui apakah ada sensor yang memiliki *sensor\_key* yang sama di basis data. Jika tidak ditemukan, *server* akan mengembalikan pesan *error* ke *client*, dalam hal ini sensor fisik.
3. Jika terdapat *sensor\_key* yang cocok pada *database,* maka *server* akan menerima parameter data yang dikirim, dan melakukan *checking* untuk *entry* kolaborasi pada database.
4. Jika terdapat data kolaborasi pada sensor tersebut, maka data kolaborasi akan dimasukkan juga kedalam *database.*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | void function RetrieveData  load **model**  raw\_data ← raw\_input\_stream  data ← json\_decode(raw\_data)  **if** checkSensorKey(data[‘sensor\_key’) == FALSE  **then**  **return** error response  **else**  **then**  **if** checkCollab(data[‘sensor\_key’) == TRUE  **then**  model->insert\_collab\_data()  model->insertData(data)  **return** success response |

Gambar 4.1 Implementasi fungsi *RetrieveData*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | function checkSensorKey(sensor\_key)  **load model**  **set** data **to** sensor\_key  **set** query to db->query(“SELECT sensor.id as SENSORID FROM ‘sensor’ WHERE sensor.sensor\_key=’sensor\_key’)  **return** query->result() |

Gambar 4.2 Implementasi fungsi *checkSensorKey*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | function checkCollab(sensor\_id)  **load model**  **set** data **to** sensor\_id  **set** db->from **to** ‘sensor\_collab’  **set** db->where to “’sensor\_x\_id’, data”  **set** db->or\_where to “’sensor\_y\_id’, data”  **set** query **to** db->get()  **return** query->result() |

Gambar 4.3 Implementasi fungsi *checkCollab*

|  |  |
| --- | --- |
| 1  2  3  4 | function insertData(data)  **load model**  **select** database  **insert** data **to** database |

Gambar 4.4 Implementasi fungsi *insertData*

Fungsi *layer sensor-centric* lainnya adalah pendaftaran sensor. Prosedur pendaftaran sensor adalah sebagai berikut :

1. Sebelumnya, pengguna harus sudah mempunyai akun pada aplikasi web ini.
2. Jika telah mempunyai akun, pengguna dapat mendaftarkan sensor pada menu “New” dan memilih “Data Point / Sensor” dan memasukkan data sensor yang di minta.
3. Kemudian aplikasi *web* akan menampilkan *sensor\_key* yang telah digenerate untuk sensor anda.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | function registerSensor(data)  **load model**  **initialize** newSensorData  **set** user\_id **to** getUserData()->id;  **set** sensor\_name **to** input->post(‘name’)  **set** sensor\_description **to** input->post(‘description’)  **set** sensor\_key **to** sha1(sensor\_name + time() + user\_id)  **set** newSensorData **to** array\_push(user\_id, sensor\_name, sensor\_description, sensor\_key)  **model**->createNewSensor(newSensorData)  **return** newSensorData; |

Gambar 4.5 Implementasi fungsi *registerSensor*

|  |  |
| --- | --- |
| 1  2  3  4 | function createNewSensor(newSensorData)  **load model**  **select** database  **insert** newSensorData **to** database |

Gambar 4.6 Implementasi fungsi *registerSensor*

4.2.2 Implementasi *Layer Middleware*

*Layer middleware* merupakan bagian yang berfungsi melakukan virtualisasi sensor fisik. Fungsi yang dijalankan oleh *layer* ini adalah melakukan proses *fetching* dan kalkulasi data dari *database* untuk kemudian ditampilkan pada *client-centric layer*.

Untuk *fetching* data dari *database*, di implementasikan *controller* *getAllSensorData, getRealTimeSensorData,* dan beberapa *model* pendukungnya.

Fungsi *getAllSensorData* berfungsi untuk mengambil semua data sensor yang tersimpan dengan *model getSensorReading* berdasarkan *sensor\_id*, dan menyajikannya dalam format JSON agar bisa di proses *HighCharts* pada *client-centric layer*.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | function getAllSensorData(sensor\_id)  **load model**  **initialize** data  **initialize** querydata  **initialize** datareading  **set** querydata **to** model->getSensorReading(sensor\_id)  **foreach** querydata **as** row  **do**  **set** datetime **to** strtotime(row->timestamp) \* 1000 //konversi ke *javascript time*  **set** datareading **to** row->datareading  **set** data **to** array\_push(datetime, datareading)  **return** json\_encode(data) |

Gambar 4.7 Implementasi fungsi *getAllSensorData*

Fungsi *getRealTimeSensorData* berfungsi untuk mengambil data sensor secara *real time*. Hal ini dilakukan dengan memanfaatkan pustaka *HighCharts* yang memanggil fungsi *getRealTimeSensorData* secara berkala dan menggunakan data yang dikembalikan oleh fungsi ini untuk me *render* grafik secara berkala.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | function getRealTimeSensorData(sensor\_id)  **load model**  **initialize** data  **initialize** result  **set** data **to** model->getSensorReading(sensor\_id)  **foreach** data **as** row  **do**  **set** result **to** array(strtotime(row->timestamp) \* 1000 , row->datareading)  **return** json\_encode(result) |

Gambar 4.8 Implementasi fungsi *getRealTimeSensorData*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | function getSensorReading(sensor\_id)  **set** db->select **to** ‘sensor\_data.sensor\_reading as datareading, sensor\_data.timestamp as timestamp’  **set** db->from **to** ‘sensor\_data’  **set** db->where **to** “ ‘sensor\_data.sensor\_id’, sensor\_id “  **set** db->order\_by **to “** ‘sensor\_data.timestamp’, ‘ASC’ “  **set** query **to** db->get()  **return** query->result() |

Gambar 4.9 Implementasi fungsi *getSensorReading*

Fungsi kalkulasi data juga dilakukan pada *layer* ini. Kalkulasi yang dilakukan berupa nilai rerata, maksimum, dan minimum dari data sensor yang tersimpan di *database*.

Untuk melakukan kalkulasi tersebut, di implementasikan fungsi *getMaxSensorReading*, *getMinSensorReading*, dan *getAverageSensorReading*, dan *model* untuk transaksi *database* dengan nama fungsi yang sama.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | function getMaxSensorReading(sensor\_id)  **load model**  **initialize** data  **initialize** datetime  **initialize** querydata  **initialize** datareading  **set** querydata **to** model->getMaxSensorReading(sensor\_id)  **foreach** querydata **as** row  **do**  **set** datetime **to** strtotime(row->sensordate) \* 1000 //konversi ke *javascript time*  **set** datareading **to** row->sensordata  **set** data **to** array\_push(datetime, datareading)  **return** json\_encode(data) |

Gambar 4.10 Implementasi fungsi *getMaxSensorReading*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | function getMinSensorReading(sensor\_id)  **load model**  **initialize** data  **initialize** datetime  **initialize** querydata  **initialize** datareading  **set** querydata **to** model->getMinSensorReading(sensor\_id)  **foreach** querydata **as** row  **do**  **set** datetime **to** strtotime(row->sensordate) \* 1000 //konversi ke *javascript time*  **set** datareading **to** row->sensordata  **set** data **to** array\_push(datetime, datareading)  **return** json\_encode(data) |

Gambar 4.11 Implementasi fungsi *getMinSensorReading*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | function getAverageSensorReading(sensor\_id)  **load model**  **initialize** data  **initialize** datetime  **initialize** querydata  **initialize** datareading  **set** querydata **to** model->getAverageSensorReading(sensor\_id)  **foreach** querydata **as** row  **do**  **set** datetime **to** strtotime(row->sensordate) \* 1000 //konversi ke *javascript time*  **set** datareading **to** row->sensordata  **set** data **to** array\_push(datetime, datareading)  **return** json\_encode(data) |

Gambar 4.12 Implementasi fungsi *getAverageSensorReading*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | function getMaxSensorReading(sensor\_id)  **set** db->select **to** ‘MAX(sensor\_data.datareading) as sensordata, DATE(sensor\_data.timestamp) as sensordate’  **set** db->from **to** ‘sensor\_data’  **set** db->where **to** “ ‘sensor\_data.sensor\_id’, sensor\_id “  **set** db->group\_by **to** ‘DATE(sensor\_data.timestamp)’  **set** db->order\_by **to “** ‘DATE(sensor\_data.timestamp)’, ‘DESC’ “  **set** query **to** db->get()  **return** query->result() |

Gambar 4.13 Implementasi model *getMaxSensorReading*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | function getMinSensorReading(sensor\_id)  **set** db->select **to** ‘MIN(sensor\_data.datareading) as sensordata, DATE(sensor\_data.timestamp) as sensordate’  **set** db->from **to** ‘sensor\_data’  **set** db->where **to** “ ‘sensor\_data.sensor\_id’, sensor\_id “  **set** db->group\_by **to** ‘DATE(sensor\_data.timestamp)’  **set** db->order\_by **to “** ‘DATE(sensor\_data.timestamp)’, ‘DESC’ “  **set** query **to** db->get()  **return** query->result() |

Gambar 4.14 Implementasi model *getMinSensorReading*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | function getAverageSensorReading(sensor\_id)  **set** db->select **to** ‘AVG(sensor\_data.datareading) as sensordata, DATE(sensor\_data.timestamp) as sensordate’  **set** db->from **to** ‘sensor\_data’  **set** db->where **to** “ ‘sensor\_data.sensor\_id’, sensor\_id “  **set** db->group\_by **to** ‘DATE(sensor\_data.timestamp)’  **set** db->order\_by **to “** ‘DATE(sensor\_data.timestamp)’, ‘DESC’ “  **set** query **to** db->get()  **return** query->result() |

Gambar 4.15 Implementasi model *getAverageSensorReading*

Salah satu fitur pada perangkat lunak ini adalah kolaborasi data antar sensor yang artinya data dari beberapa sensor (dalam hal ini dibatasi dua atau tiga sensor) dapat dikolaborasikan dalam bentuk operasi matematik (penjumlahan, pengurangan, perkalian, dan pembagian). Operasi ini dilakukan pada *sensor-centric* *layer* ketika data masuk, kemudian dikalkulasikan dan di *record* ke dalam *database*. Pada *layer* ini, fungsi yang terkait fitur kolaborasi adalah *fetching* data hasil kolaborasi. Fungsi yang di implementasikan untuk *fetching* data kolaborasi adalah *model* *getCollabData*. *Model getCollabData* mengambil data kolaborasi dari *database* berdasarkan parameter *collab\_id* dan mengembalikan isi tabel *sensor\_collab* untuk disajikan ke pengguna pada *client-centric* baik dalam bentuk tabel maupun grafik.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 20 | function getCollabData(collab\_id)  **set** db->select **to** ‘sensor\_collab.sensor\_collab\_id as COLLABID, sensor\_collab.sensor\_collab\_name as COLLABNAME, sensor\_collab.sensor\_collab\_desc as COLLABDESC, sensor\_collab\_data.timestamp as TIMESTAMP, sensor\_collab\_data.sensor\_reading as DATAREADING’  **set** db->from **to** ‘sensor\_collab  **set** db->join **to** “ ‘sensor\_collab\_data’, ‘sensor\_collab\_data.sensor\_collab\_id = sensor\_collab.sensor\_collab\_id’, ‘left’ “  **set** db->where **to** “ ‘sensor\_collab.sensor\_collab\_id’, collab\_id “  **set** query **to** db->get()  **return** query->result() |

Gambar 4.16 Implementasi fungsi *getMaxSensorReading*

### 4.2.2 Implementasi *Layer Client-centric*

*Layer client-centric* merupakan bagian yang berhubungan langsung dengan pengguna. *Layer* ini memfasilitasi dan mengelola interaksi pengguna dengan sistem, mengelola *membership* dan *session*, dan menyajikan data yang telah di *fetch* dan di olah pada *middleware layer*. Layanan yang dijalankan oleh *layer* ini adalah pendaftaran, *authentication*, dan *authorization* akun pengguna, serta penyajian data yang telah diolah dalam bentuk tabel dan grafik. Layanan tersebut direalisasikan dengan implementasi fungsi *viewSensor*, dan *analyzeSensor*. Kedua fungsi tersebut mengambil data hasil olahan berdasarkan parameter *sensor\_id*.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | function viewSensor(sensor\_id)  **load** model  **initialize** data  **set** data->userdata **to** getUserData();  **set** data->sensordata **to** model->getSensorData(sensor\_id)  **set** load**->**view(data) |

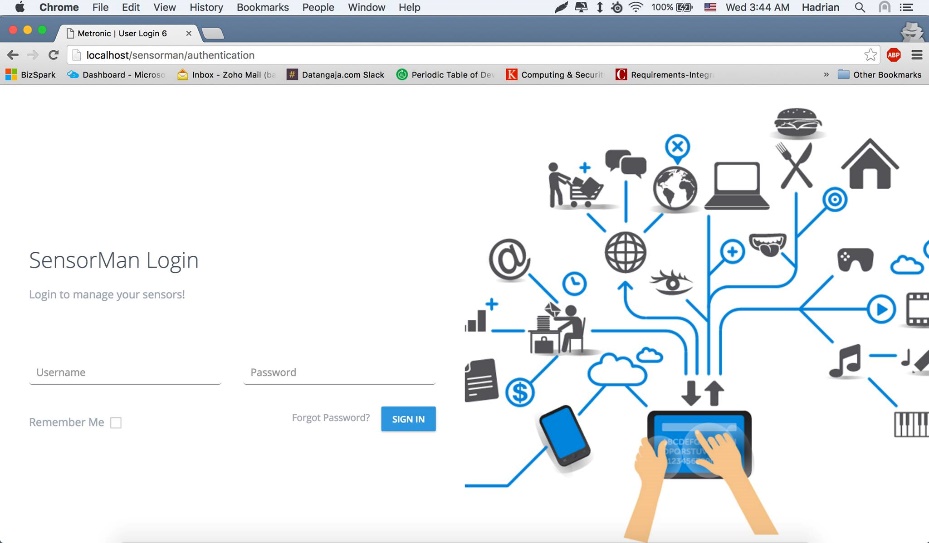
Gambar 4.17 Implementasi fungsi *viewSensor*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | function analyzeSensor(sensor\_id)  **load** model  **initialize** data  **set** data->userdata **to** getUserData();  **set** data->sensordata **to** model->getSensorData(sensor\_id)  **set** data->maxsensordata **to** model->getMaxSensorReading(sensor\_id)  **set** data->minsensordata **to** model->getMinSensorReading(sensor\_id)  **set** data->averagesensordata **to** model->getAverageSensorReading(sensor\_id)  **set** load->view(data) |

Gambar 4.18 Implementasi fungsi *analyzeSensor*

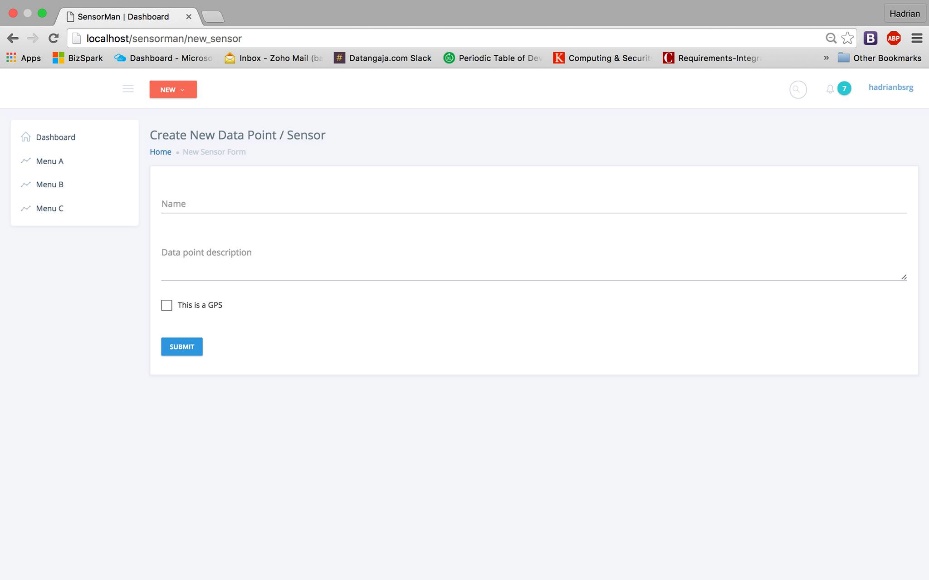
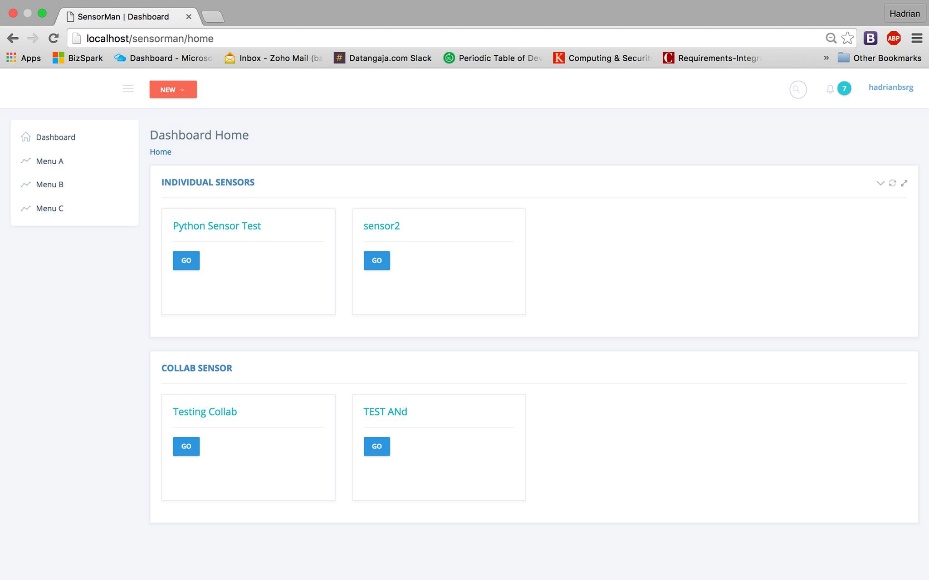
4.3 Implementasi Antarmuka Perangkat Lunak

Implementasi antarmuka perangkat lunak yang dilakukan hanya pada aplikasi *web* karena perangkat lunak memang berbasis *web*. *Web* tersebut memiliki satu *sidebar menu* yaitu **Dashboard**, dan sebuah menu *dropdown* **New** yang berisi menu **New Data Point / Sensor**, **2 Sensor Collaboration**, dan **3 Sensor Collaboration**.



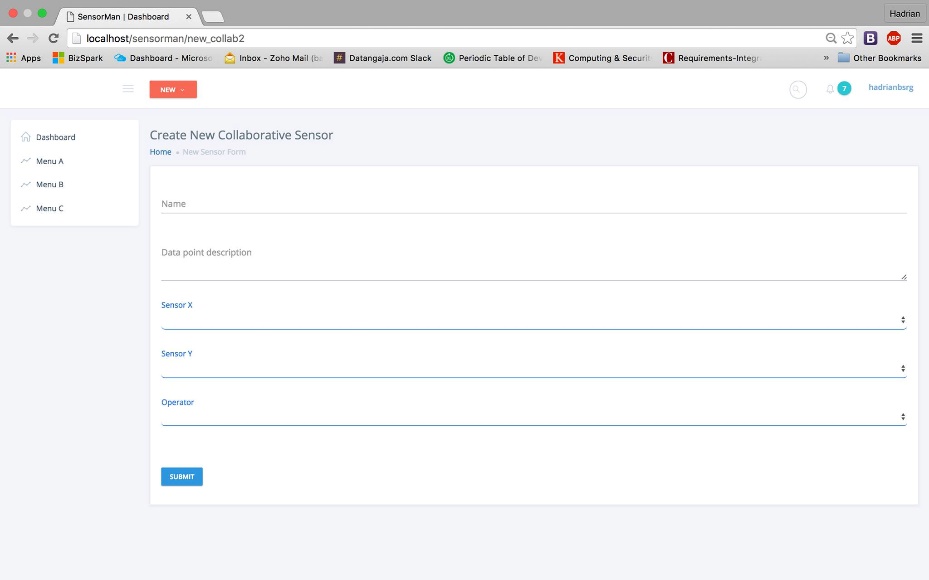
Gambar 4.19 Implementasi antarmuka halaman *login*

Pada halaman utama *dashboard***,** ditampilkan semua sensor dan sensor kolaborasi yang dimiliki oleh pengguna. Pengguna dapat mendaftarkan sensor baru atau membuat sebuah sensor kolaborasi dengan melakukan *hover* pada *dropdown menu* **New** dan memilih salah satu menu nya.



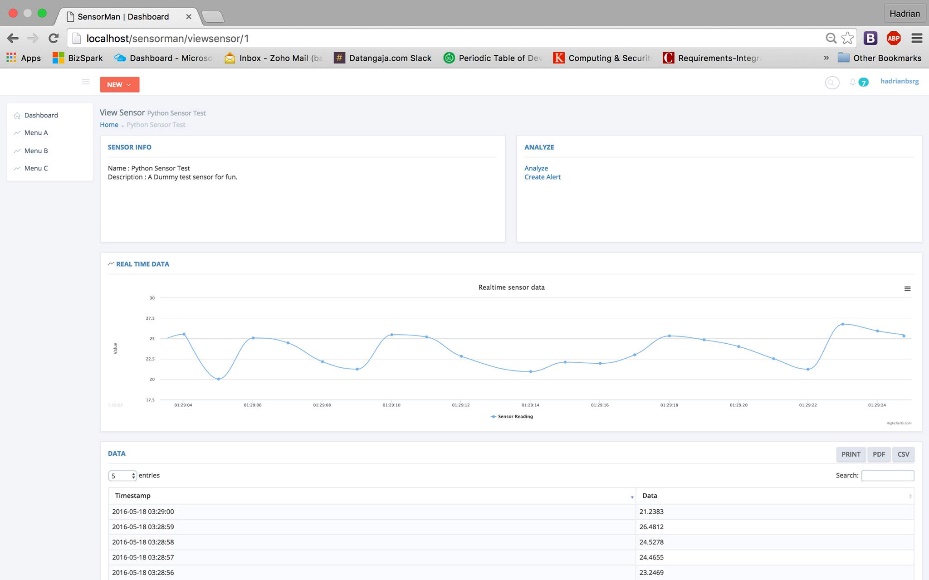
Gambar 4.20 Implementasi antarmuka halaman *dashboard*

Gambar 4.21 Implementasi antarmuka halaman *New Sensor*



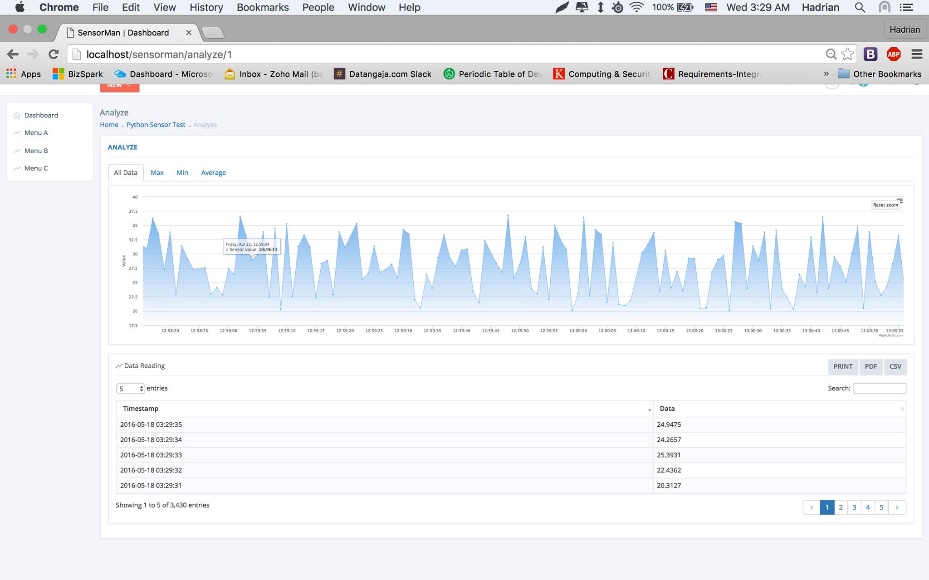
Gambar 4.20 Implementasi antarmuka halaman *New Collab*

Jika pengguna melakukan klik pada salah satu tombol “Go” pada kotak daftar sensor, pengguna akan dibawa ke halaman “*View Sensor*”. Halaman ini berisi informasi detil sebuah sensor yang terdiri dari grafik *real time* data sensor dan *datatable* yang berisi data sensor yang tersimpan di *database*.



Gambar 4.21 Implementasi antarmuka halaman *View Sensor*

Halaman “*Analyze Sensor*” dapat di akses dengan melakukan klik pada menu “*Analyze*” yang terdapat pada halaman “*View Sensor*”. Pada halaman ini disajikan data data hasil olahan dalam bentuk grafik yang terdapat pada masing masing *tab* (*All Data, Max, Min,* dan *Average*).



Gambar 4.22 Implementasi antarmuka halaman *Analyze*

# BAB V UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai hasil uji coba dan evaluasi dari implementasi perangkat lunak yang telah dilakukan. Adapun hal-hal yang akan dibahas pada bab ini adalah lingkungan uji coba perangkat lunak, uji coba fungsionalitas perangkat lunak, dan uji coba performa perangkat lunak. Uji coba akan dilakukan dengan beberapa skenario.

5.1 Lingkungan Uji Coba

Subbab Lingkungan Uji Coba menjelaskan mengenai lingkungan tempat pengujian perangkat lunak. Lingkungan uji coba pada kasus ini terdiri dari perangkat keras dan perangkat keras dengan spesifikasi sebagai berikut :

* Perangkat Keras
  + Laptop MacBook Pro, dengan spesifikasi :
    - Prosesor 2.6 GHz Intel Core i5
    - Memori 8 GB 1600 MHz DDR3
  + Komputer *Desktop*, dengan spesifikasi :
    - Prosesor 3.80 GHz AMD A10-5800K
    - Memori 8 GB DDR3
  + Arduino Uno R3
* Perangkat Lunak
  + Sistem Operasi Mac OSX El Capitan 10.11.12
  + Sistem Operasi Windows 10 Enterprise
  + *Web Broswer* Google Chrome
  + SublimeText *text editor*
  + Python untuk keperluan simulasi dan pengujian
  + Arduino IDE 1.6.8
  + *Webserver* Apache, dan *database* MySQL
  + Postman yang berfungsi untuk menguji *web service* dan fungsi yang menggunakan HTTP POST.

5.2 Skenario Uji Coba

Pada subbab ini akan dijelaskan tentang skenario uji coba prangkat lunak yang telah dibangun. Skenario uji coba dibagi menjadi dua bagian, yaitu uji coba fungsionalitas dan uji coba performa perangkat lunak.

5.2.1 Uji Coba Fungsionalitas

Uji coba fungsionalitas bertujuan untuk menguji apakah fungsi-fungsi utama pada perangkat lunak berhasil di implementasikan dan berjalan sesuai dengan yang diharapkan.

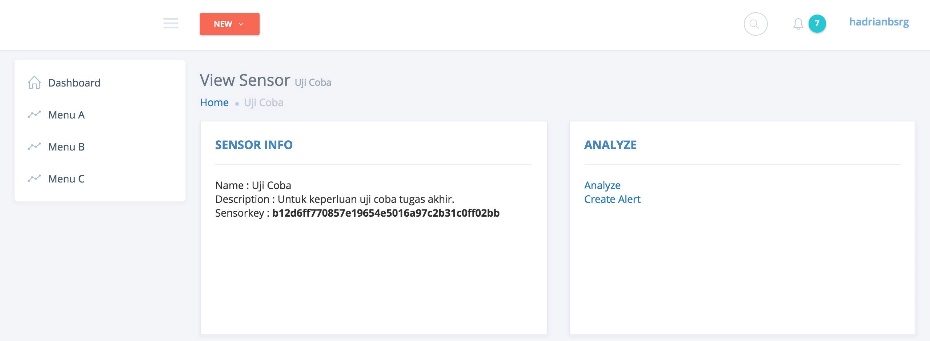
#### 5.2.1.1 Pengujian *Sensor-centric Layer*

Pengujian fungsi fungsi pada *sensor-centric layer* dilakukan dengan mecoba mendaftarkan sensor, dan melakukan pengiriman data dari sensor yang telah di daftarkan. Pendaftaran sensor dilakukan melalui antarmuka *web*, dan pengiriman data sensor akan dilakukan dengan program Python untuk simulasi dan mikrokontroler Arduino untuk pengujian dengan data yang *real*. Berikut adalah rincian prosedur uji coba *sensor-centric layer*.

Tabel 5.1 Prosedur uji coba pendaftaran sensor baru

|  |  |
| --- | --- |
| **ID** | **UC-01** |
| Nama | Uji Coba Pendaftaran Sensor Baru |
| Tujuan Uji Coba | Menguji fungsionalitas perangkat lunak untuk mendaftarkan dan menggenrasi *sensor\_key* untuk sensor baru. |
| Kondisi Awal | Perangkat lunak dijalankan |
| Skenario | Perangkat lunak dijalankan, kemudian di akses melalui *web browser* Google Chrome, dan melakukan pendaftaran sensor. |
| Masukan | Data sensor baru |
| Keluaran | *sensor\_key* yang telah di *generate*. |
| Hasil Uji Coba | Sensor berhasil di daftarkan dan perangkat lunak berhasil melakukan *generate sensor\_key* baru. |

Tabel diatas merupakan rincian prosedur pengujian Pendaftaran Sensor Baru pada perangkat lunak. Pada gambar dibawah, ditampilkan keluaran dari perangkat lunak, yaitu *sensor\_key* dan data sensor baru yang telah ter-*record* pada *database*.



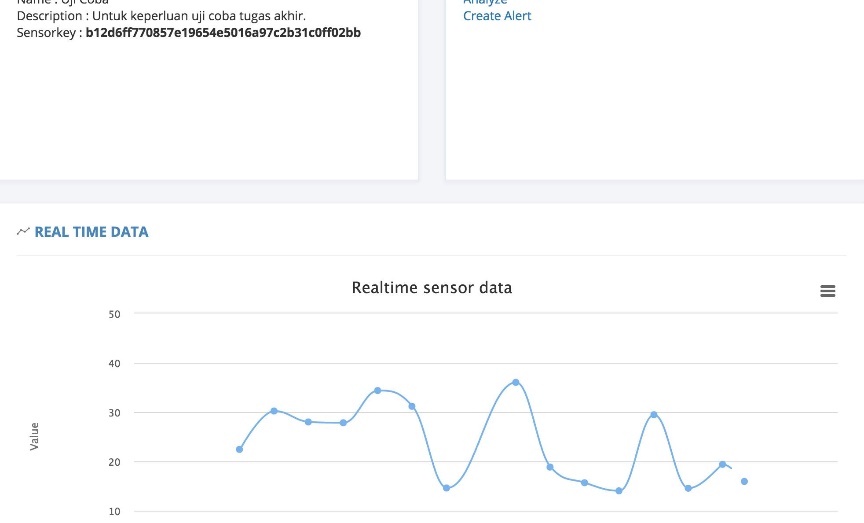
Gambar 5.1 Tampilan antarmuka halaman *View Sensor*

Fungsi berikutnya pada *sensor-centric layer* adalah fungsi penerimaan data. Fungsi ini dijalankan oleh sebuah *web service* yang memanfaatkan metode HTTP POST untuk menerima, *parsing*, mengecek *sensor\_key* dan memasukkan data kedalam *database*.

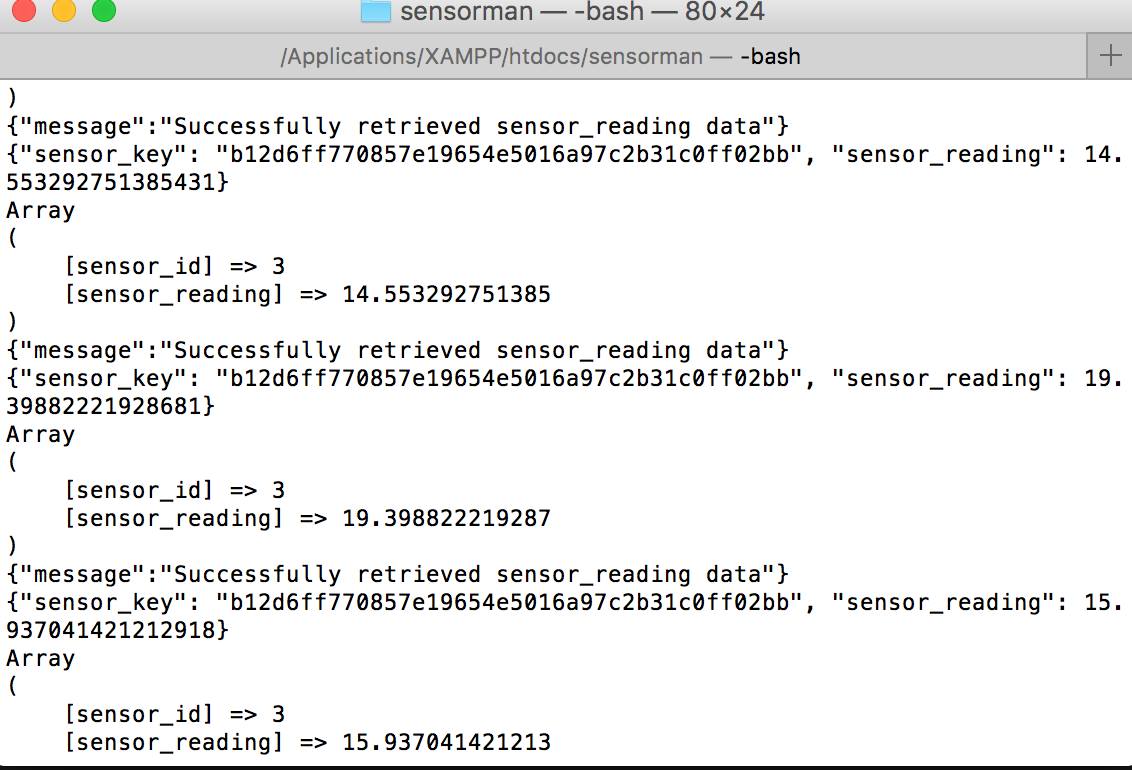
Tabel 5.2 Prosedur uji coba penerimaan data

|  |  |
| --- | --- |
| **ID** | **UC-02** |
| Nama | Uji Coba Penerimaan Data pada *Server*. |
| Tujuan Uji Coba | Menguji fungsionalitas perangkat lunak untuk menerima data dari sensor yang telah di daftarkan. |
| Kondisi Awal | Perangkat lunak dijalankan |
| Skenario | Perangkat lunak dijalankan, kemudian kemudian sebuah program Python akan mengakses *web service* perangkat lunak dan mengirim data dalam format JSON yang berisi *sensor\_key* dan *sensor\_reading*. |
| Masukan | Data dalam format JSON dari program Python untuk simulasi. |
| Keluaran | Respon berhasil dari server, dan data yang berhasil di terima oleh server dalam format JSON. |
| Hasil Uji Coba | *Web service* berhasil menerima, *parsing*, mengecek, dan memasukkan data kedalam *database*. |

Berdasarkan skenario pada tabel diatas, perangkat lunak melakukan penerimaan data dari sensor Python. Hasil keluaran pada antarmuka *web* berupa grafik *real time data* di­ *render* karena terdapat perubahan pada *database*. Keluaran pada *client* Python sensor berupa pesan dalam format JSON



Gambar 5.2 Tampilan grafik *real time* pada halaman *View Sensor*



Gambar 5.3 *Console* yang menampilkan respon sukses dari *server*

#### Pengujian *Middleware Layer*

Pada *middleware layer*, layanan yang dijalankan berupa fungsi *data fetching* dan kalkulasi data. *Core* dari layanan perangkat lunak ini terdapat pada *layer* ini.

Fungsionalitas untuk *data fetching* yang akan di uji coba antara lain adalah fungsi *getAllSensorData* yang berfungsi untuk mengambil seluruh data sensor berdasarkan *sensor\_id*, dan *getRealTimeSensorData* yang berfungsi untuk mengambil data sensor secara *real time* berdasarkan *sensor\_id*.

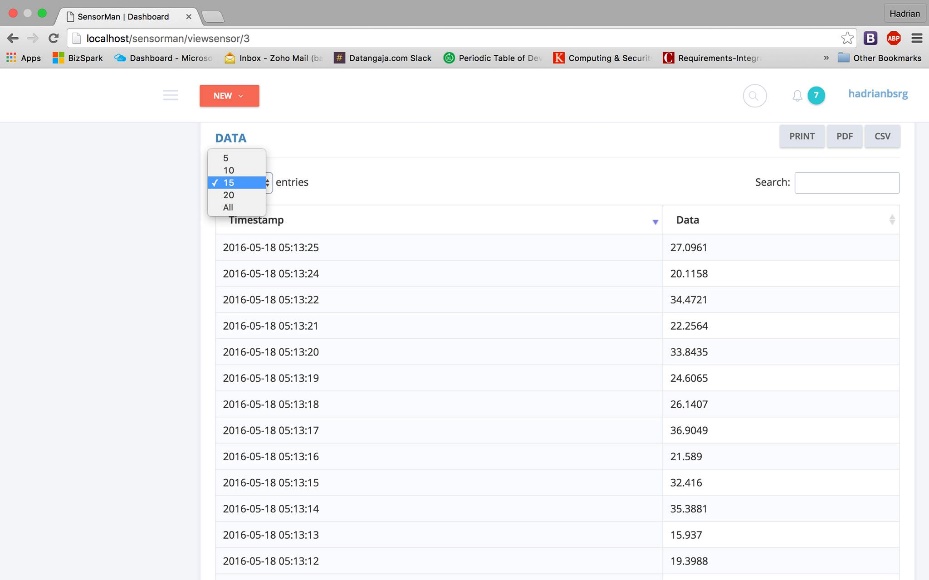
Tabel 5.3 Prosedur uji coba *data fetching* dengan fungsi *getAllSensorData*

|  |  |
| --- | --- |
| **ID** | **UC-03** |
| Nama | Uji Coba *data fetching* dengan fungsi *getAllSensorData*. |
| Tujuan Uji Coba | Menguji fungsionalitas perangkat lunak untuk mengambil data dari *database* berdasarkan *sensor\_id* |
| Kondisi Awal | Perangkat lunak dijalankan |
| Skenario | Perangkat lunak dijalankan, kemudian :   1. Uji coba menggunakan *web browser* Google Chrome dengan cara mengakses halaman “View Sensor”. 2. Uji coba dengan *Postman* dengan mengakses secara langsung fungsi *getAllSensorData*. |
| Masukan | Parameter *sensor\_id* |
| Keluaran | 1. Keluaran pada uji coba dengan *web browser* berupa *datatable* yang berisi data dari sensor terkait. 2. Keluaran pada uji coba dengan Postman berupa data sensor terkait dalam format JSON |
| Hasil Uji Coba | Fungsi *getAllSensorData* berhasil mengambil dan mengembalikan data sensor terkait. |

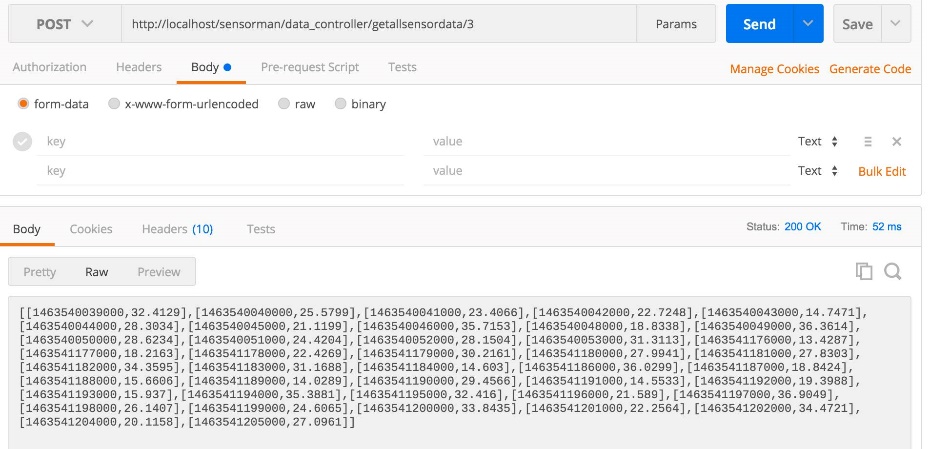
Seperti yang terdapat pada tabel diatas, skenario uji coba dilakukan dengan mengunakan dua metode.

Metode pertama berupa pengujian dengan *web browser* Google Chrome yang bertujuan untuk mengetahui apakah fungsi *getAllSensorData* dapat menjalankan fungsinya untuk mengambil, menyajikan, dan mem­-*passing* data tersebut ke *View* untuk ditampilkan pada antarmuka *web*.

Metode kedua berupa pengujian dengan kakas bantu Postman untuk melihat keluaran data mentah dalam format JSON. Hasil keluaran dari kedua uji coba tersebut bisa dilihat pada dua gambar dibawah.



Gambar 5.4 Tampilan antarmuka *view sensor* yang menampilkan *datatable*



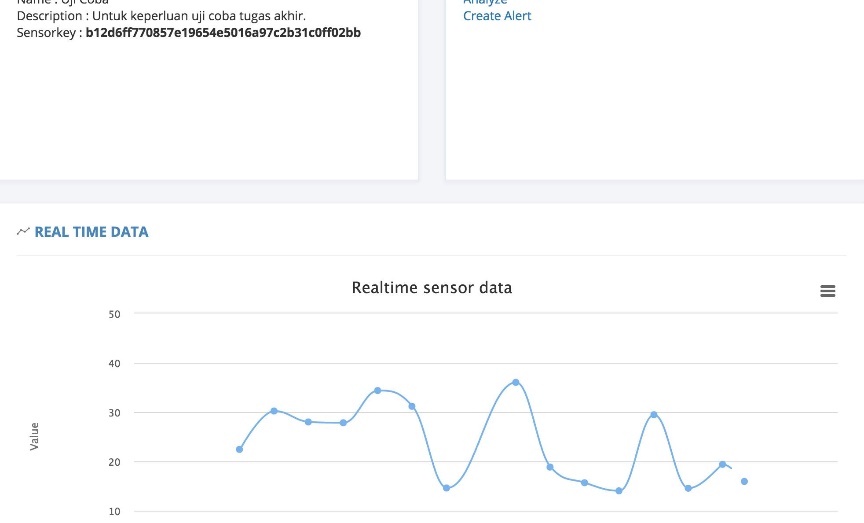
Gambar 5.5 Tampilan antarmuka *Postman* yang menampilkan *raw data* hasil *request* dari *server*.

Fungsionalitas *middleware layer* berikutnya adalah *getRealTimeSensorData* yang berfungsi untuk mengambil data sensor dari *database* secara *real time*, artinya keluaran data mentah akan berupa satu data pada setiap *request*.

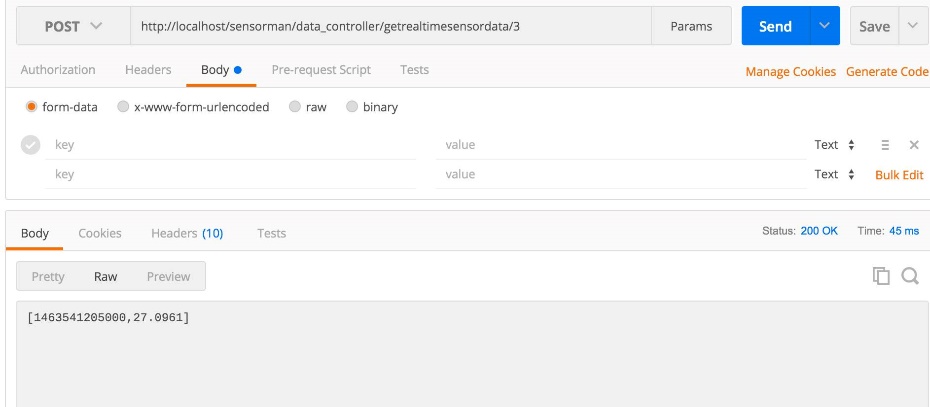
Tabel 5.4 Prosedur uji coba *data fetching* dengan fungsi *getRealTimeSensorData*

|  |  |
| --- | --- |
| **ID** | **UC-04** |
| Nama | Uji Coba *data fetching* dengan fungsi *getRealTimeSensorData* |
| Tujuan Uji Coba | Menguji fungsionalitas perangkat lunak untuk mengambil data dari *database* berdasarkan *sensor\_id* scara *real time*. |
| Kondisi Awal | Perangkat lunak dijalankan |
| Skenario | Perangkat lunak dijalankan, kemudian :   1. Uji coba menggunakan *web browser* Google Chrome dengan cara mengakses halaman “View Sensor”. 2. Uji coba dengan *Postman* dengan mengakses secara langsung fungsi *getRealTimeSensorData*. |
| Masukan | Parameter *sensor\_id* |
| Keluaran | 1. Keluaran pada uji coba dengan *web browser* berupa grafik *real time* yang merepresentasikan tiap data baru yang masuk ke *database*. 2. Keluaran pada uji coba dengan Postman berupa data sensor terkait dalam format JSON |
| Hasil Uji Coba | Fungsi *getRealTimeSensorData()* berhasil mengambil dan mengembalikan data sensor terkait. |

Sama seperti pengujian sebelumnya, ujicoba UC-04 dilakukan dengan dua metode yaitu dengan *web browser* dan Postman. Hasil uji coba dapat dilihat pada gambar dibawah.



Gambar 5.6 Tampilan antarmuka *View sensor* yang sedang menampilkan data *real time* dari sebuah sensor



Gambar 5.7 Tampilan antarmuka *Postman* yang menampilkan data pada suatu waktu (*request* data *real time*)

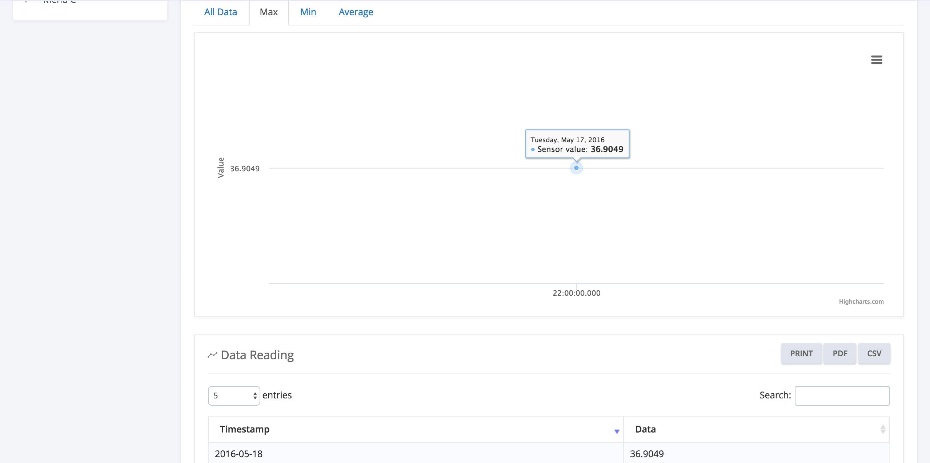
Fungsionalitas *middleware layer* berikutnya adalah fungsi kalkulasi. Fungsi kalkulasi merupakan sebuah fitur pengolahan data sensor yang tersimpan di dalam *database*. Fitur kalkulasi ini juga berupa fungsi *rules* sensor, yaitu nilai *threshold* dari sebuah sensor. Jika nilai tersebut melewati batas *threshold* , maka akan di tampilkan warning pada halaman “*View Sensor”*. Kolaborasi termasuk pada fungsi kalkulasi pada perangkat lunak ini. Kolaborasi memungkinkan pengguna untuk membuat abstraksi data dari beberapa sensor (dalam kasus ini dibatasi dua hingga tiga sensor), dengan operator pembanding (AND atau OR) dan operator matematika (penjumlahan, pengurangan, perkalian, dan pembagian)

Pengolahan data pada perangkat lunak ini berupa fungsi maksimum, minimum, dan rerata dari data data sensor yang telah dikumpulkan di *database*. Data hasil olahan tersebut kemudian disajikan dalam bentuk grafik statis pada antarmuka *web*, dan data dalam bentuk JSON. Skenario uji coba UC-05 dibawah akan membahas pengujian ketiga fungsi *getMaxSensorReading, getMinSensorReading,* dan *getAverageSensorReading*.

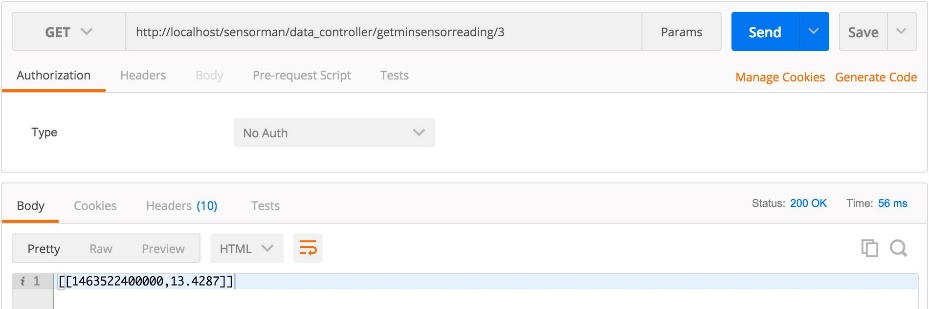
Tabel 5.5 Prosedur uji coba pengolahan data.

|  |  |
| --- | --- |
| **ID** | **UC-05** |
| Nama | Uji Coba Pengolahan Data dengan Fungsi *getMaxSensorReading, getMinSensorReading,* dan *getAverageSensorReading*. |
| Tujuan Uji Coba | Menguji fungsionalitas pengolahan data sensor yang tersimpan di *database*. |
| Kondisi Awal | Perangkat lunak dijalankan |
| Skenario | Perangkat lunak dijalankan, kemudian masing masing dari tiga fungsi diatas akan diuji dengan :   1. Uji coba menggunakan *web browser* Google Chrome dengan cara mengakses halaman “View Sensor”. 2. Uji coba dengan *Postman* dengan mengakses secara langsung fungsi *getRealTimeSensorData*. |
| Masukan | Parameter *sensor\_id* |
| Keluaran | 1. Keluaran pada uji coba dengan *web browser* berupa grafik *real time* yang merepresentasikan hasil pengolahan data dari masing masing fungsi. 2. Keluaran pada uji coba dengan Postman berupa data sensor terkait dalam format JSON |
| Hasil Uji Coba | Ke tiga fungsi berhasil mengolah data sesuai dengan fungsinya. |

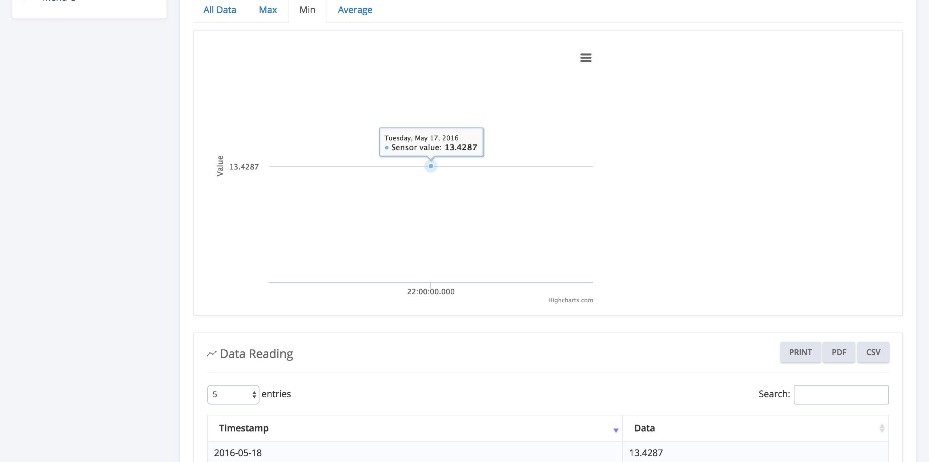
Dari tabel skenario uji coba UC-05 , masing masing dari tiga fungsi diatas diuji coba dengan dua metode, yaitu metode *web browser* dan metode Postman. Hasil uji coba ketiga fungsi diatas dapat dilihat pada gambar dibawah.



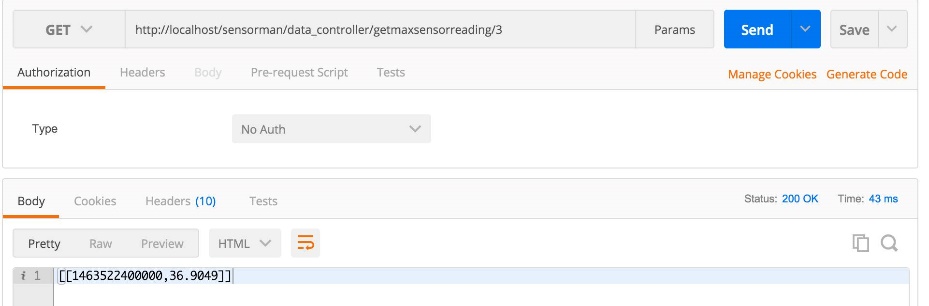
Gambar 5.8 Tampilan antarmuka halaman *analyze* yang menampilkan data *Max* dari sebuah sensor



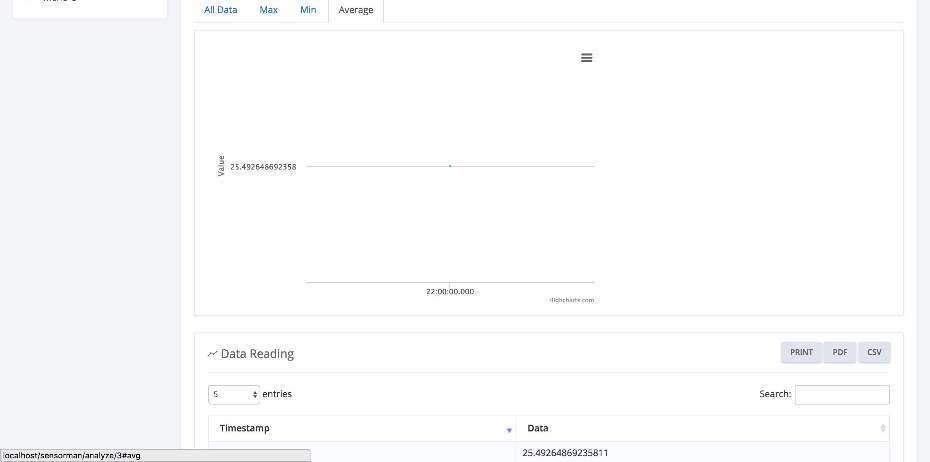
Gambar 5.9 Tampilan antarmuka *Postman* yang menampilkan data *Max*



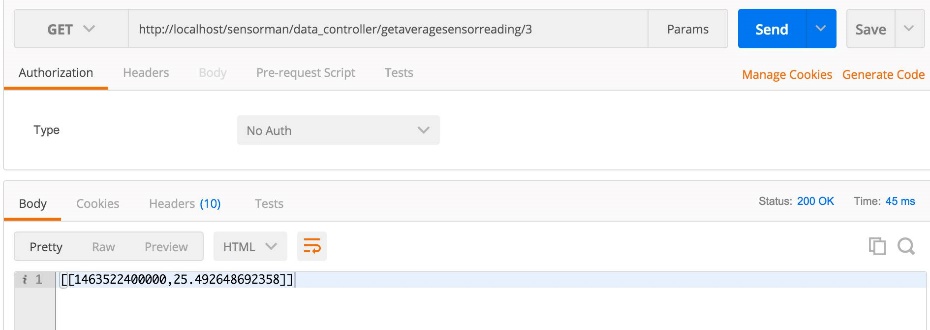
Gambar 5.10 Tampilan antarmuka *Analyze* yang menampilkan data *Min*



Gambar 5.11 Tampilan antarmuka *Postman* yang menampilkan data *Min*



Gambar 5.12 Tampilan antarmuka *Analyze* yang menampilkan data *Average*



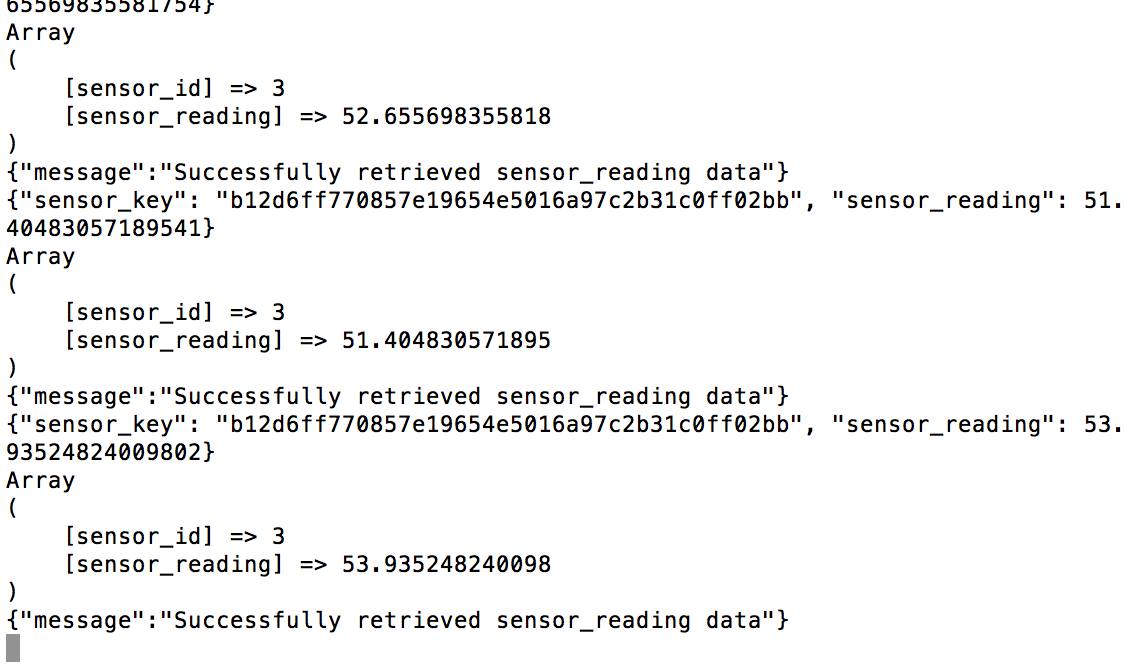
Gambar 5.13 Tampilan antarmuka *Postman* yang menampilkan data *Average*

Fungsi yang dijalankan *middleware layer* selanjutnya adalah fungsi *rules* pada sensor. Fungsi ini berupa *threshold* nilai dari sensor tersebut, baik nilai batas atas (*high*), batas bawah (*low*), maupun nilai spesifik yang ditentukan oleh pengguna (*equal*). Implementasi dari *rules* ini adalah fungsi *checkSensorRules*. Fungsi ini dijalankan secara terus menerus dengan *looping* karena berfungsi melakukan pengecekan secara *real time* pada data sensor yang ada. Tabel uji coba UC-06 akan membahas secara rinci pengujian fungsi ini.

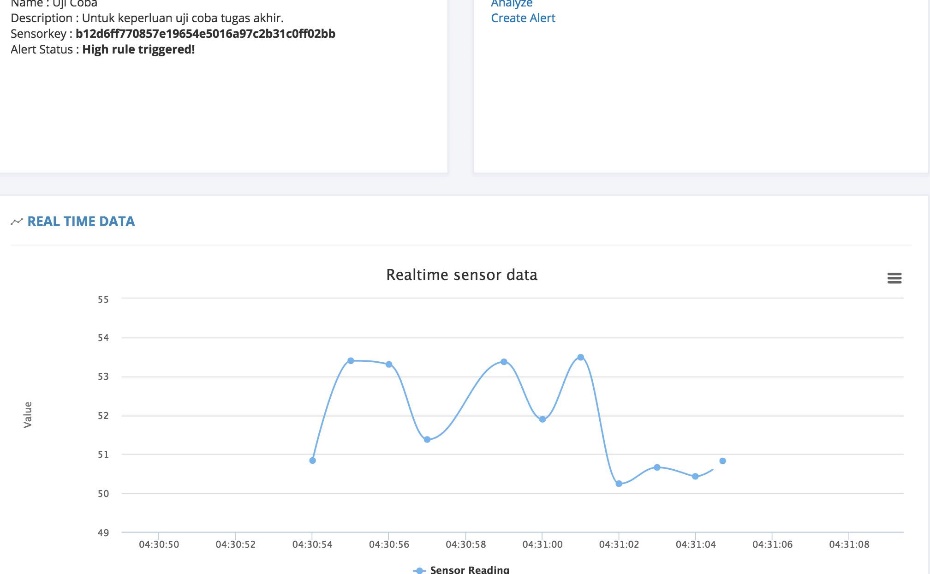
Tabel 5.6 Prosedur uji coba *rules* pada sensor

|  |  |
| --- | --- |
| **ID** | **UC-06** |
| Nama | Uji Coba *Rules* Sensor. |
| Tujuan Uji Coba | Menguji fungsionalitas *rules* dan *alert* pada perangkat lunak.. |
| Kondisi Awal | Perangkat lunak dijalankan, *rule* high dengan *value* 50 telah ditambahkan untuk sensor terkait. |
| Skenario | Perangkat lunak dijalankan, kemudian sensor akan mengirimkan nilai yang berada di luar batas threshold dari *rule* sensor tersebut. |
| Masukan | Parameter *sensor\_key* dan *sensor\_reading*. |
| Keluaran | Berupa peringatan yang ditampilkan pada halaman *web* “*View Sensor*”. |
| Hasil Uji Coba | Berhasil mendeteksi nilai yang berada diluar batas *rules*. |

Berdasarkan tabel diatas, pengujian dilakukan dengan mengirim nilai bacaan sensor diatas 50, karena *rule* sensor yang berupa *high threshold value* di set pada nilai 50. Ketika sensor Python mengirim data dengan *value* diatas 50, maka *rule high* tadi akan ter-*trigger*, dan ditampilkan pada halaman “*View Sensor*”.



Gambar 5.14 Tampilan *Console* yang menampilkan respon sukses dari *server*



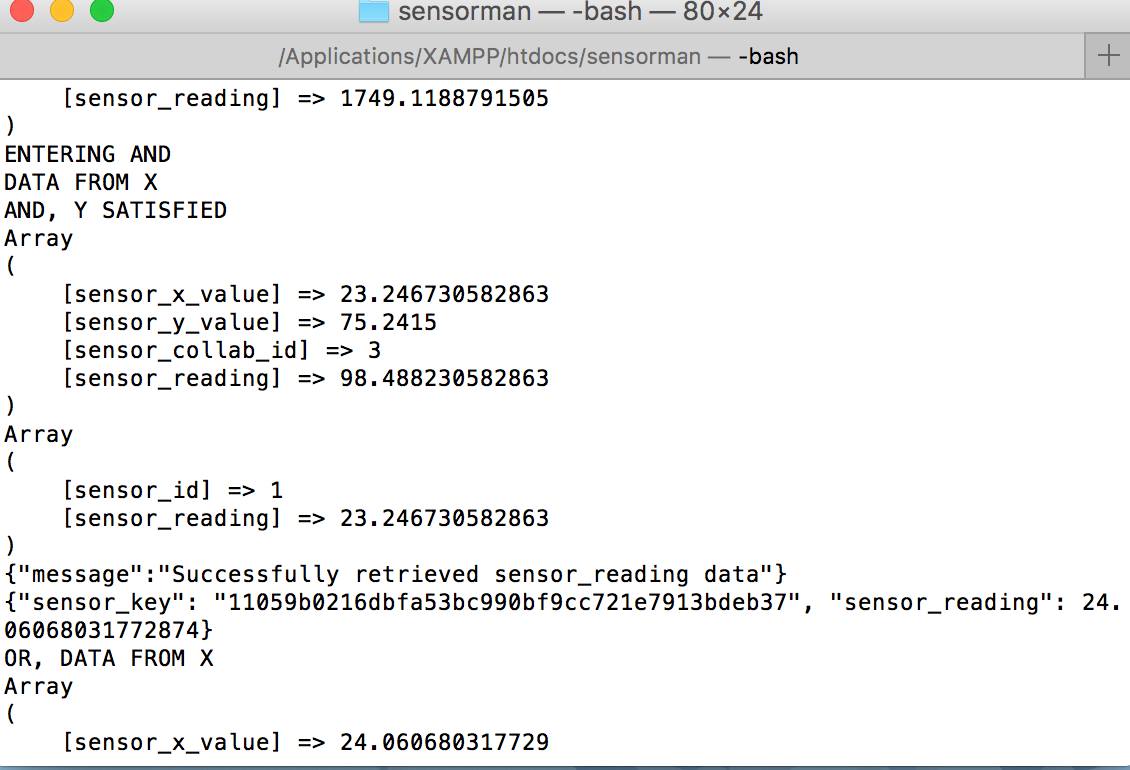
Gambar 5.15 Tampilan antarmuka halaman *view sensor* yang menampilkan pesan alert “High Rule triggered!”

Fungsi kolaborasi juga dijalankan pada *middleware layer*. Fungsi ini memungkinkan pengguna untuk melakukan abstraksi data dari dua hingga tiga sensor. Abstraksi data berupa operasi matematika dasar antar nilai sensor tersebut. Tabel UC-07 akan membahas rincian prosedur uji coba fitur kolaborasi ini.

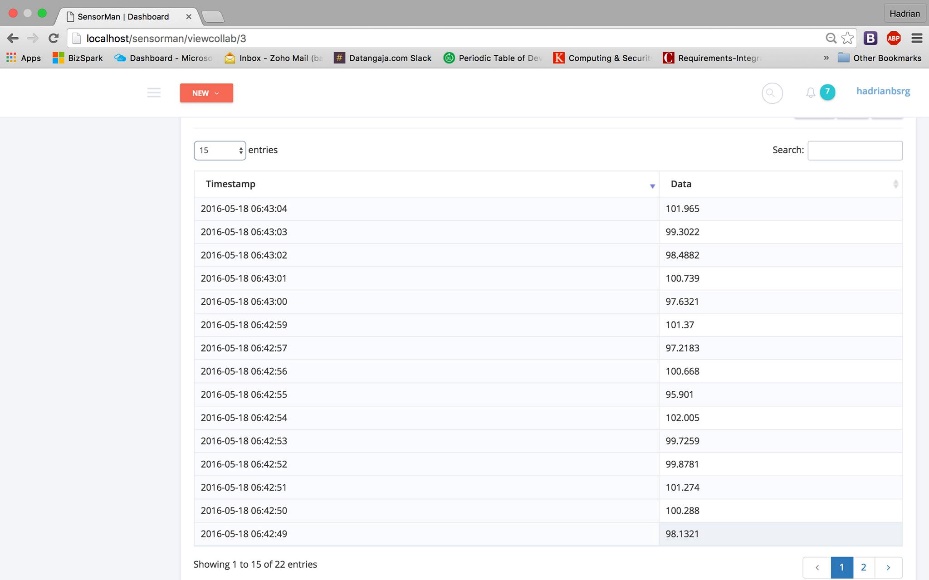
Tabel 5.7 Prosedur uji coba kolaborasi sensor

|  |  |
| --- | --- |
| **ID** | **UC-07** |
| Nama | Uji Coba Kolaborasi Sensor |
| Tujuan Uji Coba | Menguji fungsionalitas kolaborasi sensor pada pernagkat lunak. |
| Kondisi Awal | Perangkat lunak dijalankan, sudah terdapat sensor kolaborasi pada *database*. |
| Skenario | Perangkat lunak dijalankan, kemudian sensor akan mengirimkan data ke *server*. |
| Masukan | Parameter *sensor\_key* dan *sensor\_reading*. |
| Keluaran | Data hasil kalkulasi antar sensor dan respon JSON dari *server*. |
| Hasil Uji Coba | Berhasil melakukan abstraksi data berupa operasi matematika dasar pada data sensor. |

Berdasarkan skenario diatas, pengujian dilakukan dengan mengirimkan data dari sensor yang telah terdaftar pada sebuah *instance* kolaborasi. Ketika data sensor tersebut masuk, *server* melakukan pengecekan terhadap *sensor\_key* yang masuk, apakah terdaftar pada *server* dan terdaftar pada tabel *sensor\_collab*. Jika ya, maka data akan di olah sesuai dengan aturan pada tabel *sensor\_collab* terkait. Berikut gambar hasil pengujian UC-07



Gambar 5.16 Tampilan *console* yang berisi respon sukses dari *server*.



Gambar 5.17 Tampilan antarmuka halaman *View Collab* yang menampilkan data hasil kolaborasi.

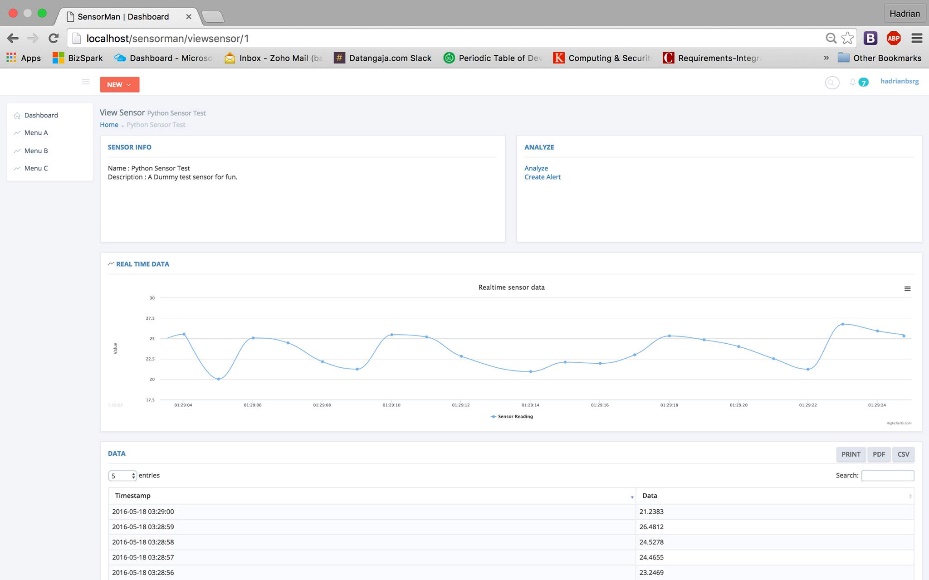
#### Pengujian *Client-centric Layer*

*Client-centric layer*, seperti namanya, merupakan *layer* yang berhubungan langsung dengan pengguna. *Layer* ini menjalankan fungsi manajemen *user*, pendaftaran *user*, dan menampilkan data-data yang telah di kumpulkan *Sensor-centric layer*, di olah oleh *Middleware layer*.

Implementasi utama dari fungsi *client-centric layer* ini adalah fungsi *viewSensor*, dan *analyzeSensor* yang keduanya berfungsi menampilkan data kepada pengguna melalui antarmuka *web*. UC-08 dan UC-09 akan melakukan uji coba pada masing masing dari kedua fungsi tersebut.

Tabel 5.0.8 Prosedur uji coba fungsi *viewSensor*

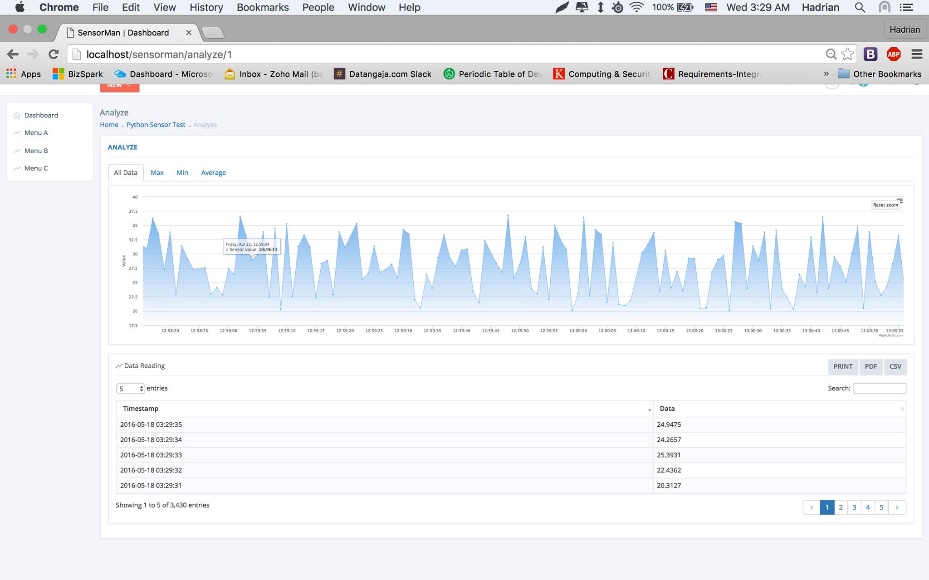
|  |  |
| --- | --- |
| **ID** | **UC-08** |
| Nama | Uji Coba fungsi *viewSensor* untuk menampilkan data sensor. |
| Tujuan Uji Coba | Menguji fungsionalitas *client-centric* layer dalam mempresentasikan data kepada pengguna. |
| Kondisi Awal | Perangkat lunak dijalankan |
| Skenario | Perangkat lunak dijalankan, kemudian di akses menggunakan *web browser* Google Chrome. |
| Masukan | Alamat *web* perangkat lunak. |
| Keluaran | Halaman “*View Sensor*” yang menampilkan data-data sensor. |
| Hasil Uji Coba | Berhasil menampilkan data sensor. |



Gambar 5.18 Tampilanhalaman *View Sensor*

Tabel 5.9 Prosedur uji coba fungsi *analyzeSensor*

|  |  |
| --- | --- |
| **ID** | **UC-09** |
| Nama | Uji Coba fungsi *analyzeSensor* untuk menampilkan hasil olahan dari data sensor. |
| Tujuan Uji Coba | Menguji fungsionalitas *client-centric* layer dalam mempresentasikan data kepada pengguna. |
| Kondisi Awal | Perangkat lunak dijalankan |
| Skenario | Perangkat lunak dijalankan, kemudian di akses menggunakan *web browser* Google Chrome. |
| Masukan | Alamat *web* perangkat lunak. |
| Keluaran | Halaman “*Analyze Sensor*” yang menampilkan data-data sensor. |
| Hasil Uji Coba | Berhasil menampilkan hasil olahan dari data sensor. |



Gambar 5.19 Tampilanhalaman *Analyze*

* + 1. Uji Coba Performa

Uji coba performa merupakan pengujian yang bertujuan untuk menguji tingkat kehandalan suatu perangkat lunak, baik dalam segi kecepatan (*speed*) maupun ketahanan (*robustness*). Uji coba ini juga mengukur penggunaan *bandwidth* saat sensor melakukan pengiriman data ke *server*. Adapun penjelasn setiap bagian dari uji coba performa ini akan dijelaskan pada subbab berikut ini.

#### Uji Coba Performa Kecepatan Pengolahan Data pada *Server*

Uji coba waktu pengolahan data pada *server* bertujuan untuk mengukur seberapa cepat performa *server* dalam mengolah data. Operasi pengolahan data yang akan dilakukan menerima data dari sensor, mengolah dan me *record* data kolaborasi, dan uji coba kecepatan *load* halaman *web*. Masing masing dari operasi yang akan dilakukan diatas akan dicoba berdasarkan jumlah sensor yang aktif mengirimkan data ke *server*.

##### 5.2.2.1.1 Uji Coba Performa Kecepatan Penerimaan Data

Uji coba performa kecepatan penerimaan data dilakukan dengan mencatat waktu mulai saat mengirim data dan waktu berakhir, yaitu waktu saat server telah mengirimkan respon “sukses” dari sebuah transmisi data. Pengujian dilakukan pada *host* terpisah dari tempat *deployment* server perangkat lunak. Berikut adalah data hasil percobaan.

Tabel 5.10 Data percobaan penerimaan data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Jumlah Sensor Aktif** | **Waktu Mulai Pengiriman Data** | **Waktu Diterimanya respon *server*.** | **∆ Waktu (Detik)** |
| 1 | 1 | 07:32:26.301000 | 07:32:26.475000 | 00.174000 |
| 2 | 2 | 08:03:46.394000 | 08:03:46.666000 | 00.272000 |
| 3 | 3 | 08:09:52.233000 | 08:09:52.285000 | 00.052000 |
| 4 | 4 | 08:20:09.860000 | 08:20:10.126000 | 00.266000 |
| 5 | 5 | 08:18:22.517000 | 08:18:22.797000 | 00.280000 |
| **Rata-rata waktu penerimaan dan *insert* data.** | | | | 0.2088 |

##### Uji Coba Performa Kecepatan Penerimaan dan Pengolahan Data Kolaborasi Sensor

Uji coba ini bertujuan untuk mengukur kecepatan peneriman sekaligus pengolahan data kolaborasi sensor. Pada suatu kolaborasi sensor, ketika data masuk, terjadi banyak pengecekan yang dilakukan oleh perangkat lunak, beberapa diantaranya adalah pengecekan *sensor\_key*, pengecekan *sensor\_id* terhadap tabel *sensor\_collab*, pengecekan nilai pada *sensor\_reading* terhadap tabel *sensor\_rule* dan *sensor\_rule\_data*, dan yang pasti adalah operasi matematika jika data yang masuk terdaftar di satu atau lebih kolaborasi sensor.

Pengujian dilakukan dengan menjalankan beberapa sensor kolaborasi sekaligus untuk mengukur kecepatan perangkat lunak.

Tabel 5.11 Data percobaan penerimaan dan pengolahan data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Jumlah Sensor Aktif** | **Waktu Mulai Pengiriman Data** | **Waktu Diterimanya respon *server*.** | **∆ Waktu (Detik)** |
| 1 | 1 | 08:33:32.065000 | 08:33:32.250000 | 00.185000 |
| 2 | 2 | 08:35:23.466000 | 08:35:23.667000 | 00.201000 |
| 3 | 3 | 08:37:09.497000 | 08:37:09.571000 | 00.074000 |
| 4 | 4 | 08:39:32.951000 | 08:39:33.104000 | 00.153000 |
| 5 | 5 | 08:41:24.497000 | 08:41:24.611000 | 00.114000 |
| **Rata-rata waktu penerimaan dan pengolahan data kolaborasi sensor** | | | | 0.1454 |

Dari beberapa hasil uji coba diatas, tidak terlihat adanya kecenderungan kecepatan penerimaan dan pengolahan data yang meningkat seiring bertambahnya jumlah sensor yang aktif. Hal ini terlihat saat uji coba sensor kolaborasi dengan jumlah sensor aktif sebanyak tiga sensor lebih cepat dari uji coba yang menggunakan dua sensor aktif. Berikut merupakan tabel percobaan sensor kolaborasi dengan 5 sensor aktif.

Tabel 5.12 Data percobaan pengiriman dan pengolahan data dengan 10 sensor aktif

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Pengiriman data Ke-** | **Waktu Mulai Pengiriman Data** | **Waktu Diterimanya respon *server*.** | **∆ Waktu (Detik)** |
| 1 | 1 | 08:40:59.174000 | 08:40:59.350000 | 00.176000 |
| 2 | 2 | 08:41:00.402000 | 08:41:00.549000 | 00.147000 |
| 3 | 3 | 08:41:01.594000 | 08:41:01.943000 | 00.349000 |
| 4 | 4 | 08:41:02.993000 | 08:41:03.257000 | 00.264000 |
| 5 | 5 | 08:41:05.537000 | 08:41:05.649000 | 00.112000 |
| 6 | 6 | 08:41:06.704000 | 08:41:06.786000 | 00.082000 |
| 7 | 7 | 08:41:07.840000 | 08:41:07.907000 | 00.067000 |
| 8 | 8 | 08:41:08.956000 | 08:41:09.029000 | 00.073000 |
| 9 | 9 | 08:41:10.084000 | 08:41:10.242000 | 00.158000 |
| 10 | 10 | 08:41:11.292000 | 08:41:11.367000 | 00.075000 |
| **Rata-rata kecepatan pengiriman dan pengolahan data kolaborasi sensor (10 sensor aktif)** | | | | 0,158667 |

.

#### Uji Coba Performa Ketahanan Perangkat Lunak

Pada uji coba ini, parameter yang diuji adalah ketahanan (*robustness*) dari perangkat lunak berbasis *web* ini. Ketahanan perangkat lunak merupakan hal yang harus diperhatikan ketika menyediakan layanan berbasis komputasi awan. Perangkat lunak harus kuat menangani *traffic* data masuk dari banyak sensor, maupun *traffic* *request* untuk menampilkan data sensor dari pengguna.

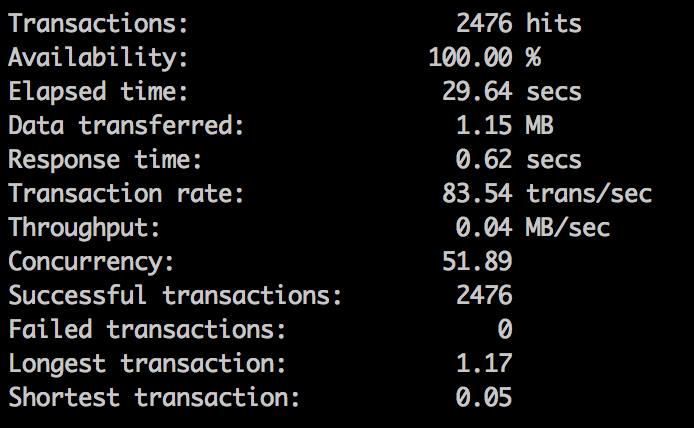
Uji coba performa ketahanan perangkat lunak ini akan diuji menggunakan kakas bantu *siege*. *Siege* merupakan sebuah kakas bantu untuk HTTP *load testing*. *Siege* dapat digunakan untuk melakukan pengukuran dan *benchmark* performa sebuah aplikasi *web* saat dalam *traffic* yang tinggi. Fungsi yang akan diuji adalah pengiriman data sensor ke *server*, dan *request* data dari *database* oleh pengguna. Keduanya akan diuji dengan empat tingkatan, yaitu :

* 100 *concurrent* *user* selama 30 detik
* 100 *concurrent user* selama 60 detik
* 200 *concurrent user* selama 30 detik
* 200 *concurrent user* selama 60 detik

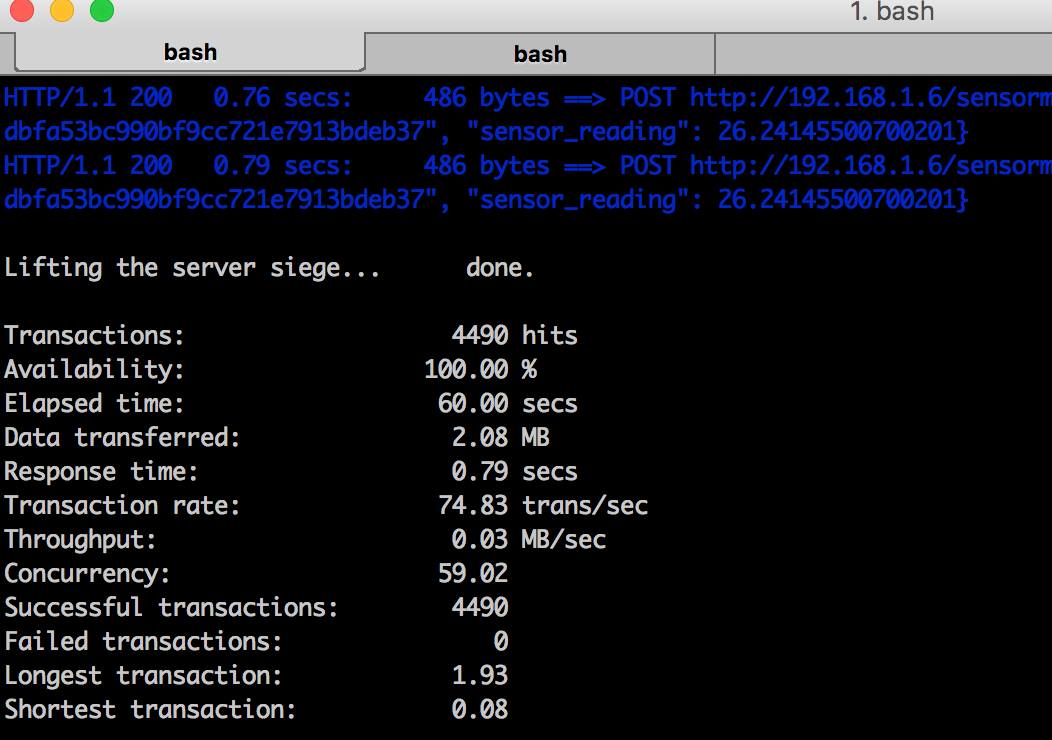
##### 5.2.2.2.1 Uji Coba Performa Ketahanan Perangkat Lunak terhadap *Traffic* Pengiriman Data dari Sensor

Uji coba ini dilakukan dengan mengakses *web service* yang menangani penerimaan data, yaitu fungsi “*api/retrieve\_data*”. Fungsi ini akan di akses oleh *siege* sesuai dengan empat tingkatan percobaan diatas.

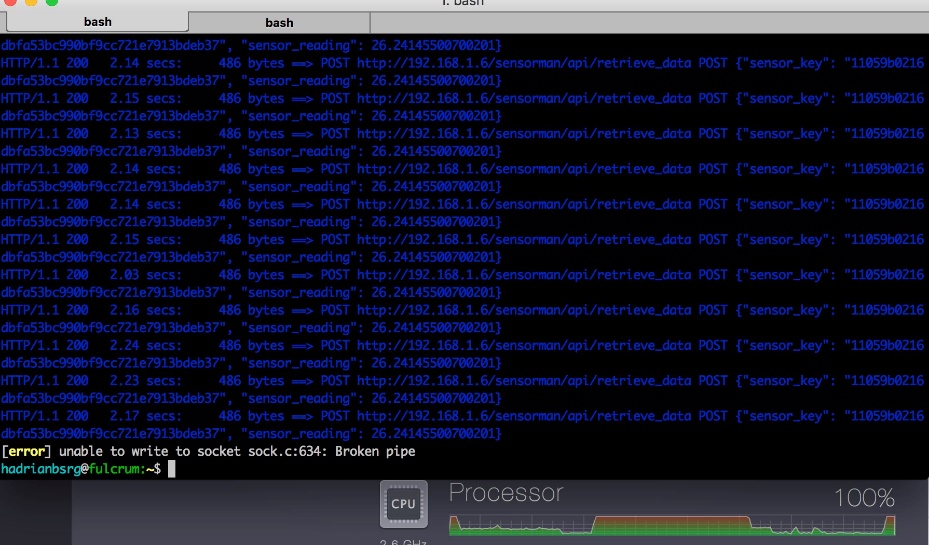
* 100 *concurrent user* selama 30 detik



Gambar 5.20 Hasil pengujian *load test*

* 100 *concurrent user* selama 60 detik

Gambar 5.21 Hasil pengujian *load test*

* 200 *concurrent user* selama 30 detik

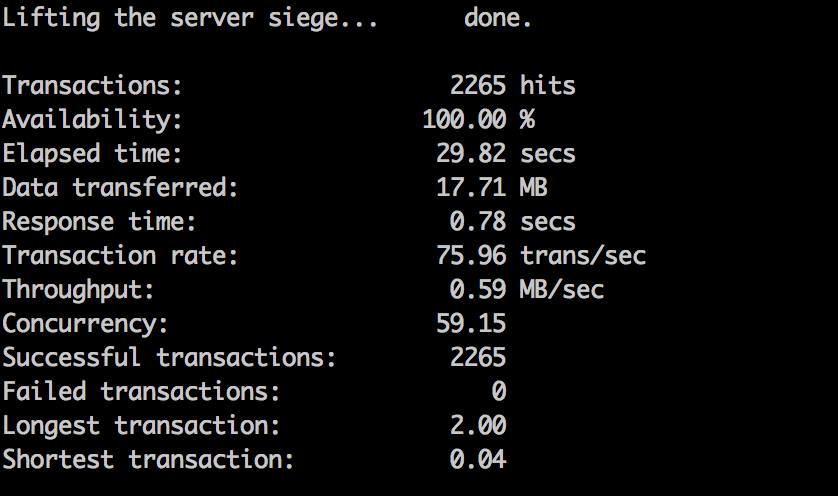
Gambar 5.22 Hasil pengujian *load test*

Dapat dilihat dari hasil uji coba dengan kakas bantu *siege*, *server* perangkat lunak kuat menangani hingga 100 *concurrent user* selama 60 detik. Artinya *server* perangkat lunak kuat jika melayani 100 sensor yang mengirimkan data secara bersamaan selama 60 detik.

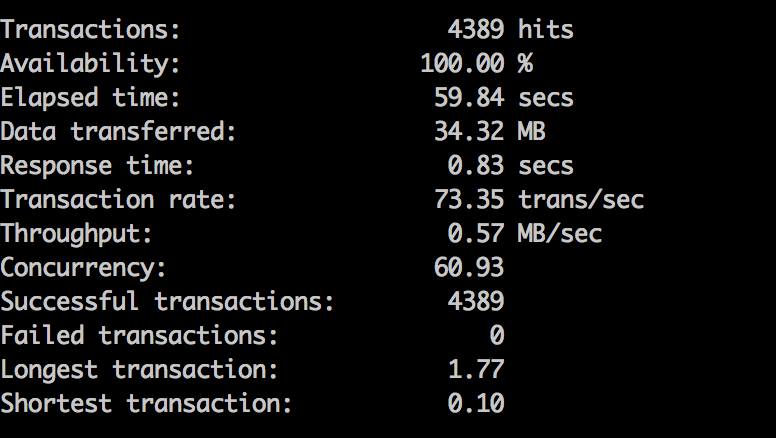
Namun ketika di uji dengan *traffic* sebesar 200 *concurrent user*, *server* perangkat lunak mengalami *down*.

##### 5.2.2.2.2 Uji Coba Performa Ketahanan Perangkat Lunak terhadap *Traffic Request* Data Sensor

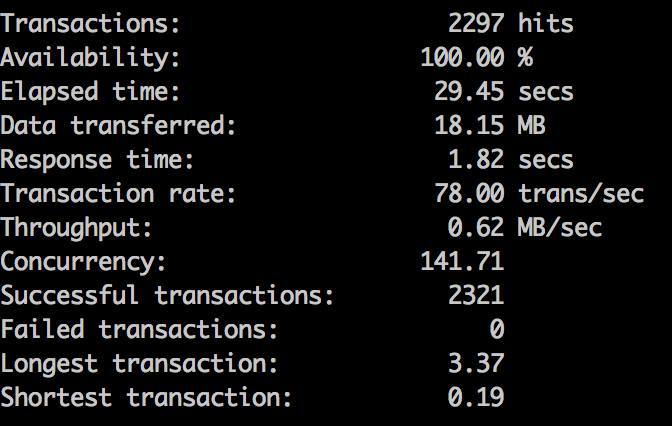
Uji coba ini dilakukan dengan mengakses fungsiyang menangani *data fetching*, yaitu fungsi “*data\_controller/getAllSensorData*”. Fungsi ini akan di akses oleh *siege* sesuai dengan empat tingkatan percobaan diatas.

* 100 *concurrent user* selama 30 detik

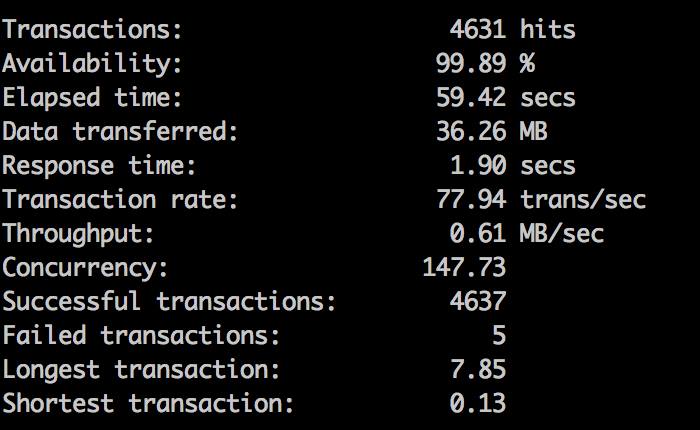
Gambar 5.23 Hasil pengujian *load test.*

* 100 *concurrent user* selama 60 detik

Gambar 5.24 Hasil pengujian *load test*.

* 200 *concurrent user* selama 30 detik

Gambar 5.25 Hasil pengujian *load test*.

* 200 *concurrent user* selama 60 detik

Gambar 5.26 Hasil pengujian *load test*

Dari hasil uji coba diatas terlihat bahwa *server* perangkat lunak masih kuat menangani *traffic* berupa *request* data sensor dari *database* hingga 200 *concurrent user*, walaupun *avaliability nya* tidak 100%. Artinya terdapat transaksi yang gagal (dalam uji coba ini 5 transaksi gagal).

5.3 Evaluasi Hasil Uji Coba

Evaluasi hasil uji coba merupakan pemaparan mengenai hasil uji coba yang telah dilakukan pada subbab sebelumnya.

### 5.3.1 Evaluasi Hasil Uji Coba Performa

Evaluasi hasil uji coba performa dibagi menjadi dua bagian, yaitu evaluasi uji coba performa kecepatan, dan evaluasi uji coba performa ketahanan. Kedua evaluasi tersebut akan dibahas pada subbab berikut.

#### Evaluasi Hasil Uji Coba Performa Kecepatan Pengolahan Data pada *Server*

Evaluasi hasil uji coba ini terdiri atas dua bagian, yaitu evaluasi hasil uji coba performa kecepatan penerimaan data, dan evaluasi hasil uji coba performa kecepatan penerimaan, dan pengolahan data kolaborasi sensor. Masin masing bagian tersebut akan dijelaskan pada subbab berikut.

##### 5.3.1.1.1 Evaluasi Hasil Uji Coba Performa Kecepatan Penerimaan Data

Sesuai dengan hasil uji coba pada subbab sebelumnya, maka di dapat waktu penerimaan data oleh perangkat lunak yang merupakan selisih antara waktu pengiriman data oleh *node* sensor, dan waktu penerimaan respon dari server ke *node* sensor. Grafik dibawah merepresentasikan kecepatan penerimaan data terhadap banyaknya sensor yang aktif.

Gambar 5.27 Grafik kecepatan waktu penerimaan data terhadap jumlah sensor aktif

Berdasarkan gambar diatas, tidak terlihat konsistensi penurunan kecepatan penerimaan data oleh *server* terhadap banyaknya sensor yang aktif pada saat itu. Adapun rata-rata kecepatan penerimaan data oleh *server* adalah 0,2088 detik.

##### 5.3.1.1.2 Evaluasi Hasil Uji Coba Performa Kecepatan Penerimaan dan Pengolahan data Kolaboarsi Sensor

Sesuai dengan hasil uji coba pada subbab sebelumnya, maka didapatkan waktu penerimaan dan pengolahan data kolaborasi sensor. Waktu berupa selisih antara waktu saat data dikirim oleh sensor, dan saat *node* sensor menerima respon dari *server*.

Berikut adalah grafik yang menampilkan waktu kecepatan penerimaan dan pengolahan data kolaborasi sensor terhadap banyaknya sensor yang aktif pada saat itu.

Gambar 5.28 Grafik kecepatan penerimaan dan pengolahan data kolaborasi sensor terhadap jumlah sensor aktif.

Dan berikut adalah grafik yang merepresentasikan kecepatan pengiriman dan pengolahan data sensor kolaborasi dengan sepuluh sensor yang aktif pada saat itu.

Gambar 5.29 Grafik kecepatan penerimaan dan pengolahan data kolaborasi sensor pada keadaan 10 sensor aktif

#### Evaluasi Hasil Uji Coba Performa Ketahanan Perangkat Lunak

Berdasarkan hasil uji coba *roubustness* dari perangkat lunak ini pada subbab sebelumnya, perangkat lunak dapat menangani *traffic* berupa *data request* hingga 200 *concurrent user* untuk selama 60 detik berturut-turut, dengan *availability* 99.89% .

Namum untuk uji coba dengan *traffic* berupa HTTP POST data untuk mengirim data sensor ke *server*, perangkat tidak dapat menangani 200 *concurrent user* karena ketika tiga kali pengujian terjadi *down* pada layanan.

***[Halaman ini sengaja dikosongkan]***

# BAB VI KESIMPULAN DAN SARAN

Bab ini berisi penjelasan mengenai kesimpulan yang dapat di ambil pada pengerjaan Tugas Akhir ini, serta saran-saran tentang pengembangan yang dapat dilakuikan pada Tugas Akhir ini di masa yang akan datang.

## Kesimpulan

Dari hasil pengamatan dan percobaan selama perancangan, implementasi, dan uji coba aplikasi, maka dapat diambil kesimpulan sebagai berikut:

1. Perangkat lunak dapat menjalankan fungsi fungsi utama pada masing masing *layer*.
2. Perangkat lunak dapat melakukan penerimaan dan pengolahan data dari sensor fisik. Pengolahan tersebut berupa fungsi fungsi statistik dasar seperti *Max, Min,* dan *Average*.
3. Perangkat lunak dapat membuat sebuah abstraksi data dari dua atau tiga sensor dengan operasi matematik dasar (penjumlahan, pengurangan, perkalian, dan pembagian)
4. Perangkat lunak dapat menyajikan data data hasil olahan kedalam bentuk visual yang berupa grafik, dengan kata lain perangkat lunak bisa melakukan visualisasi data.
5. Perangkat lunak dapat menampilkan data sensor secara *real time* dalam bentuk grafik *real time*.
6. Kecepatan penerimaan dan pengolahan data sensor maupun data kolaborasi sensor dipengaruhi, walaupun tidak signifikan, oleh banyak nya *traffic* sensor yang aktif.
7. Perangkat lunak dapat menangani *traffic* berupa *data request* hingga 200 *concurrent user* selama 60 detik, dengan *availability* 99.89% . Namun untuk *traffic* yang berupa HTTP POST *request* dari sensor untuk mengirim data, *server* perangkat lunak tidak dapat menangani *traffic load* nya.

## Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah :

1. Untuk melakukan optimasi pada fungsi penerimaan data agar bisa lebih optimal dalam rangkaian proses penerimaan data nya.
2. Untuk menambahkan fitur *alert* secara *real time*.

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | Sensor-Cloud, [Online]. Avaliable: http://sensorcloud.com/system-overview [Diakses 2015] |
| [2] | http://www.ntu.edu.sg/intellisys |
| [3] | K. T. Lan, “What’s Next? Sensor+Cloud?” in *Proceeding of the 7th International Workshop on Data Management for Sensor Networks*, pp. 978-971, ACM Digital Library, 2010. |
| [4] | Arduino Uno, [Online]. Avaliable: https://www.arduino.cc/en/main/arduinoBoardUno [Diakses 2015] |
| [5] | F. Michael, "Milestones in the history of thematic cartography, statistical graphics, and data visualization", 2008 |
| [6] | “IT Asssets Management”, [Online]. Avaliable: http://www.itassetmanagement.net/ [Diakses 2015] |
| [7] | "JSON," JSON. [Online]. Avaliable: http://www.json.org/ [Diakses 2016] |
| [8] | Tatroe, K., MacIntyre, P. dan Lerdorf, R., 2013. Programming PHP, Third Edition. USA: O’Reilly Media, Inc. |
| [9] | Vaswani, Vikram., 2007. Programming PHP Solutions, First Edition. USA: McGraw Hill companies, Inc. |
| [10] | "What is MySQL?". *MySQL 5.1 Reference Manual*. Oracle. |
| [11] | Urlocker, M. Zack (13 December 2005). "Google Runs MySQL". The Open Force. M. Zack Urlocker |
| [12] | Sobel, Jason (21 December 2007). "Keeping Up". *The Facebook Blog*. Facebook |
| [13] | H. Kreger, *Web Services* Conceptual Architecture, New York: International Business Machines Corporation, 2001 |