

---

---

**JURAGAN (JUALAN RAKYAT GABUNGAN  
ONLINE) – MEMBANGUN E-COMMERCE  
DENGAN FRAMEWORK CODEIGNITER**

---

---

**YUSNIAR NUR SYARIF SIDIQ**

**1.16.4.089**



**PROGRAM DIPLOMA IV TEKNIK INFORMATIKA  
POLITEKNIK POS INDONESIA  
BANDUNG  
2020**

## **KATA PENGANTAR**

## DAFTAR ISI

KATA PENGANTAR .....	2
DAFTAR ISI .....	3
DAFTAR GAMBAR .....	5
DAFTAR TABEL .....	10
DAFTAR SOURCE CODE .....	11
BAB I .....	14
BAB II .....	17
2.1 Usaha Kecil Menengah (UKM).....	17
2.1.1 Usaha Mikro.....	17
2.1.2 Usaha Kecil.....	17
2.1.3 Usaha Menengah .....	18
2.2 Pedagang Kaki Lima (PKL).....	18
2.3 Promosi.....	18
2.4 Internet .....	19
2.5 E-Commerce.....	19
2.6 Komunitas .....	19
2.7 Aplikasi.....	20
2.8 Framework .....	20
2.9 CodeIgniter .....	20
2.10 Hypertext Preprocessor (PHP) .....	21
2.11 <i>Database</i> (Basis Data) .....	21
2.12 MariaDB .....	21
BAB III .....	23
3.1 <i>Use Case</i> JURAGAN .....	24
3.1.1 Definisi Aktor .....	28
3.1.2 Definisi <i>Use Case</i> .....	29
3.3.3 <i>Scenario Use Case</i> JURAGAN .....	30
3.2 Perancangan Basis Data ( <i>Database</i> ).....	38

3.2.1 Tabel <i>User</i> .....	39
3.2.2 Tabel <i>User_token</i> .....	41
3.2.3 Tabel tb_barang .....	42
3.3 <i>User Interface</i> Sistem JURAGAN.....	44
BAB IV .....	61
3.1 Xampp.....	61
4.1.1 Perbedaan PHP 7 Dan PHP 5 .....	68
4.2 Visual Studio Code .....	69
4.3 <i>Codeigniter</i> .....	78
4.3.1 <i>Installasi Framework Codeigniter</i> .....	80
BAB V .....	86
5.1 Sejarah Codeigniter.....	86
5.2 Kelebihan Codeigniter.....	86
5.3 Struktur Direktori Codeigniter.....	87
5.4 Konfigurasi Framework Codeigniter .....	93
5.5 Membuat CRUD Dengan Codeigniter.....	101
BAB VI .....	146
6.1 Mengubungkan Boostrap Dengan CI.....	146
5.2 Fungsi Form Validation .....	160
5.3 Membuat Fitur Keranjang Belanja .....	167
6.4 Membuat Fitur Pencarian .....	181
BAB VII .....	190
7.1 Halaman Awal Sistem JURAGAN .....	190
7.2 Halaman Registrasi Dan Login Sistem JURAGAN .....	195
7.3 Mengelola Data Akun Pada Sistem JURAGAN .....	204
7.4 Proses Lupa Password Pada Sistem JURAGAN .....	209
7.5 Mengelola Keranjang Belanja Pada Sistem JURAGAN .....	214
7.6 Menerapkan Fitur Komunitas .....	219
DAFTAR PUSTAKA .....	221

## **DAFTAR GAMBAR**

Gambar 3.1 Komponen Aktor Use Case.....	24
Gambar 3.2 Komponen UseCase.....	25
Gambar 3.3 Komponen Use Case Subject.....	25
Gambar 3.4 Relasi Association.....	26
Gambar 3.5 Relasi Generalization .....	26
Gambar 3.6 Relasi Despendency .....	27
Gambar 3.7 Use Case Sistem JURAGAN .....	27
Gambar 3.8 Perancangan UI Halaman Utama.....	45
Gambar 3.9 Perancangan UI Halaman Login .....	47
Gambar 3.10 Perancangan UI Halaman Daftar Akun .....	48
Gambar 3.11 Perancangan UI Halaman Lupa Password .....	49
Gambar 3.12 Perancangan UI Halaman Reset Password .....	50
Gambar 3.13 Perancangan UI Halaman Utama Kondisi Login.....	51
Gambar 3.14 Perancangan UI Halaman Data Profil.....	53
Gambar 3.15 Perancangan UI Halaman Edit Profil.....	54
Gambar 3.16 Perancangan UI Halaman Ganti Password .....	55
Gambar 3.17 Perancangan UI Halaman Profil Toko .....	56
Gambar 3.18 Perancangan UI Halaman Produk Toko.....	57
Gambar 3.19 Perancangan UI Halaman Input Produk.....	58
Gambar 3.20 Perancangan UI Halaman Keranjang Belanja.....	59
Gambar 4.1 Website Resmi Xampp.....	62
Gambar 4.2 Tahap Dua Dalam Mendownload Xampp .....	62
Gambar 4.3 Halaman Versi Xampp.....	63
Gambar 4.4 Proses Awal Installasi Xampp .....	63
Gambar 4.5 Pemilihan Komponen - Komponen Pada Software Xampp.	64
Gambar 4.6 Pemilihan Tempat Penyimpanan Software Xampp .....	65

Gambar 4.7 Tempat Penyimpanan Xampp .....	65
Gambar 4.8 Proses Installasi Xampp Sedang Berjalan.....	66
Gambar 4.9 Tampilan Sofware Xampp .....	67
Gambar 4.10 Menjalankan Module Apache Dan MySQL .....	67
Gambar 4.11 Halaman Localhost .....	68
Gambar 5.1 Dokumentasi Codeigniter .....	87
Gambar 5.2 Struktur Direktori CI.....	88
Gambar 5.3 Isi Dari Direktori Application .....	89
Gambar 5.4 File Dalam Folder Config .....	90
Gambar 5.5 Contoh Script Code Controller .....	90
Gambar 5.6 Pemanggilan Fungsi Helper .....	91
Gambar 5.7 Contoh Script Code Model .....	92
Gambar 5.8 Pemanggilang Fungsi Model Pada Autoload.....	92
Gambar 5.9 Pemanggilan Fungsi Model Dari Controller .....	92
Gambar 5.10 Folder CI JURAGAN.....	93
Gambar 5.11 Konfigurasi Libraries Sistem JURAGAN.....	94
Gambar 5.12 Konfigurasi Fungsi Helpers Sistem JURAGAN.....	95
Gambar 5.13 Konfigurasi Models Pada Sistem JURAGAN .....	96
Gambar 5.14 Konfigurasi File Config Sistem JURAGAN .....	96
Gambar 5.15 Konfigurasi index_page Sistem JURAGAN.....	97
Gambar 5.16 Membuat File Baru .htaccess .....	97
Gambar 5.17 Membuka Dokumentasi CI .....	98
Gambar 5.18 Mencari Dokumentasi htaccess.....	98
Gambar 5.19 Script Code htaccess .....	99
Gambar 5.20 Mengisikan File .htaccess .....	99
Gambar 5.21 Konfigurasi Default Controller Sistem JURAGAN .....	100
Gambar 5.22 Konfigurasi Database Sistem JURAGAN .....	101

Gambar 5.23 Mempersiapkan Software VSCode .....	102
Gambar 5.24 Open Folder CI Pada VSCode .....	102
Gambar 5.25 Halaman PHP MyAdmin .....	103
Gambar 5.26 Membuat Nama Database .....	103
Gambar 5.27 Membuat Tabel tb_barang .....	104
Gambar 5.28 Membuat Struktur Tabel tb_barang .....	104
Gambar 5.29 Struktur Tabel tb_barang .....	105
Gambar 5.30 Membuat Controllers Barang.php .....	106
Gambar 5.31 Membuat File m_barang Pada Models .....	114
Gambar 5.32 Membuat Folder Barang Pada Views .....	123
Gambar 5.33 Membuat File PHP Pada Folder Barang .....	124
Gambar 6.1 Halaman Start Bootstrap SB ADMIN 2 .....	147
Gambar 6.2 Menu Download Template Bootstrap SB ADMIN 2 .....	148
Gambar 6.3 Meng Extract File Zip Bootstrap .....	148
Gambar 6.4 File Codeigniter Pada Direktori Xampp/htdocs .....	149
Gambar 6.5 Membuat Folder Vendor .....	149
Gambar 6.6 Memindahkan File SB ADMIN 2 Ke Dalam Vendor .....	149
Gambar 6.7 Tampilan View Welcome_message.php .....	150
Gambar 6.8 Membuat Folder Assets .....	150
Gambar 6.9 Mengisi Folder Assets .....	151
Gambar 6.10 Membuat Folder templates Pada Views .....	151
Gambar 6.11 Membuat 4 File PHP Pada Folder Templates .....	152
Gambar 6.12 Isi File Header.php .....	153
Gambar 6.13 Isi File Sidebar.php .....	154
Gambar 6.14 Isi File Topbar.php .....	155
Gambar 6.15 Isi File Footer.php .....	156
Gambar 6.16 Perubahan Sidebar Dan Topbar .....	158

Gambar 6.17 Menerapkan Templates Boostrap Terhadap CI Berhasil .....	160
Gambar 6.18 Contoh Form Validarion Pada Sistem JURAGAN .....	161
Gambar 6.19 Praktek Form Validation Pada Field Name .....	163
Gambar 6.20 Praktek Form Validation Terhadap Field Email .....	164
Gambar 6.21 Praktek Form Validation Pada Field Password.....	165
Gambar 6.22 Mengakses Dokumentasi Shopping Cart .....	168
Gambar 6.23 View Halaman Keranjang Belanja.....	172
Gambar 6.24 Menjalankan Function Add Pada Controllers Cart .....	173
Gambar 6.25 Rowid Delete.....	175
Gambar 6.26 Function Delete Berhasil Diterapkan.....	175
Gambar 6.27 Banyaknya Items Pada Halaman Keranjang Belanja.....	176
Gambar 6.28 Fitur Search Pada Topbar Dalam Sistem JURAGAN .....	183
Gambar 7.1 Halaman Awal Sistem JURAGAN .....	190
Gambar 7.2 Mengaplikasikan Button Detail Toko Pada Halaman Awal .....	191
Gambar 7.3 Halaman Detail Toko Pada Sistem JURAGAN .....	192
Gambar 7.4 Halaman Produk Toko .....	193
Gambar 7.5 Halaman Detail Produk.....	194
Gambar 7.6 Peringatan Wajib Login Terlebih Dahulu .....	195
Gambar 7.7 Halaman Daftar Akun Pelanggan Pada Sistem JURAGAN .....	196
Gambar 7.8 Peringatan Kolom Tidak Boleh Kosong .....	197
Gambar 7.9 Notifikasi Format Email Tidak Sesuai.....	198
Gambar 7.10 Notifikasi Password Terlalu Pendek .....	199
Gambar 7.11 Notifikasi Password Tidak Cocok.....	200
Gambar 7.12 Akun User Berhasil Terdaftar .....	200
Gambar 7.13 Notifikasi Akun Belum Aktif.....	201

Gambar 7.14 Aktivasi Akun Melalui Email .....	202
Gambar 7.15 Akun User Telah Aktif.....	202
Gambar 7.16 Halaman Daftar Akun Toko.....	203
Gambar 7.17 Peringatan Pada Pendaftaran Akun Toko .....	204
Gambar 7.18 Halaman Data Profil.....	205
Gambar 7.19 Halaman Edit Profil .....	205
Gambar 7.20 Melakukan Pengeditan Data Profil .....	206
Gambar 7.21 Kelengkapan Data Profil User .....	207
Gambar 7.22 Halaman Ganti Password Pada Sistem JURAGAN.....	207
Gambar 7.23 Peringatan Password Lama Salah .....	208
Gambar 7.24 Proses Ganti Password Sukses .....	209
Gambar 7.25 Cara Mengakses Halaman Lupa Password .....	210
Gambar 7.26 Halaman Lupa Password Pada Sistem JURAGAN .....	211
Gambar 7.27 Memberikan Email Yang Akan Di Reset Passwordnya ..	211
Gambar 7.28 Notifikasi Cek Email Untuk Melakukan Reset Password	212
Gambar 7.29 Email Untuk Melakukan Reset Password .....	212
Gambar 7.30 Halaman Reset Password Sistem JURAGAN .....	213
Gambar 7.31 Membuat Password Baru .....	213
Gambar 7.32 Fitur Lupa Password Berhasil Dilakukan .....	214
Gambar 7.33 Keranjang Belanja Belum Terisi.....	215
Gambar 7.34 Halaman Produk Sistem JURAGAN .....	216
Gambar 7.35 Keranjang Belanja Terisi .....	216
Gambar 7.36 Halaman Keranjang Belanja Sistem JURAGAN.....	217
Gambar 7.37 Menambah Dan Mengurangkan QTY.....	218
Gambar 7.38 Menghapus Salah Satu Items Pada Halaman Keranjang Belanja .....	218
Gambar 7.39 Menampilkan Produk Berdasarkan Komunitas .....	220

## **DAFTAR TABEL**

Tabel 3. 1 Definisi Aktor Pada Use Case JURAGAN.....	28
Tabel 3. 2 Definisi Komponen Use Case Pada Use Case JURAGAN ....	29
Tabel 3.3 Scenario Use Case Daftar Akun .....	30
Tabel 3.4 Scenario Use Case Login.....	31
Tabel 3.5 Scenario Use Case Kelola Profil.....	32
Tabel 3.6 Scenario Use Case Transaksi .....	33
Tabel 3.7 Scenario Use Case Verifikasi Akun.....	34
Tabel 3.8 Scenario Use Case Kelola Produk (Penjual).....	35
<i>Tabel 3.9 Scenario Use Case Kelola Produk (Admin) .</i>	37
Tabel 3.10 Scenario Use Case Kelola User .....	37
Tabel 3.11 Struktur Tabel User.....	39
Tabel 3.12 Struktur Tabel User_token.....	42
Tabel 3.13 Struktur Tabel tb_barang .....	42
Tabel 4. 1 Spesifikasi Software VS Code .....	70

## **DAFTAR SOURCE CODE**

Source Code 5.1 SQL Create Table .....	106
Source Code 5.2 Code Pertama Controllers .....	107
Source Code 5.3 Pemanggilan Fungsi Models Dan Form_validation ...	108
Source Code 5.4 Membuat Function Index .....	108
Source Code 5.5 Membuat Function Tambah Pada Controllers.....	109
Source Code 5.6 Membuat Function Edit Pada Controllers .....	110
Source Code 5.7 Membuat Function Delete Pada Controllers .....	111
Source Code 5.8 Code Full Controllers Barang.php.....	113
Source Code 5.9 Code Awal Pada m_barang.php .....	114
Source Code 5.10 Mendeskripsikan Field Pada Models .....	115
Source Code 5.11 Function Fungsi Read.....	116
Source Code 5.12 Fungsi Read Berdasarkan Id.....	116
Source Code 5.13 Membuat Function Simpan Pada Models .....	117
Source Code 5.14 Membuat Fungsi Update Pada Models .....	118
Source Code 5.15 Membuat Fungsi Delete Pada Models.....	119
Source Code 5.16 Code Full Models File m_barang.php.....	122
Source Code 5.17 Memanggil Header Pada View v_barang.....	125
Source Code 5.18 Memanggil Template Topbar Dan Sidebar Pada View v_barang.....	125
Source Code 5.19 Membuat Tabel Pada View v_barang .....	127
Source Code 5.20 Membuat Table Body Pada View v_barang.....	128
Source Code 5.21 Membuat Footer Pada View v_barang .....	129
Source Code 5.22 Code Full File v_barang .....	132
Source Code 5.23 Memanggil Flashdata Pada View inputbarang.....	133
Source Code 5.24 Membuat Button Kembali Pada Form inputbarang .	134
Source Code 5.25 Membuat Textbox Pada View inputbarang.....	136

Source Code 5.26 Code Full View File inputbarang.php .....	139
Source Code 5.27 Membuat Flashdata Pada View Edit .....	140
Source Code 5.28 Membuat Button Kembali Pada View Edit.....	140
Source Code 5.29 Membuat Form Input Dengan Value Pada Edit .....	142
Source Code 6.1 Merubah Controllers Welcome .....	156
Source Code 6.2 Merubah File Header.php Pada Controllers Welcome .....	158
Source Code 6.3 Merubah Code Footer.php.....	159
Source Code 6.4 Memanggil Library Form Validation .....	162
Source Code 6.5 Penerapan Form Validation Pada Field Name .....	162
Source Code 6.6 Penerapan Form Validation Pada Field Email .....	164
Source Code 6.7 Penerapan Form Validation Pada Field Password ....	165
Source Code 6.8 Contoh Pemanggilan Fungsi Form Validation.....	167
Source Code 6.9 Dokumentasi Add Item Pada Shopping Cart .....	168
Source Code 6.10 Melakukan Load Library Cart .....	169
Source Code 6.11 Menambahkan Item Database Ke Cart.....	170
Source Code 6.12 Membuat Models m_cart.php.....	171
Source Code 6.13 Function Delete Pada Controllers Cart.....	173
Source Code 6.14 Memanggil Function Delete Pada Button Delete....	174
Source Code 6.15 Membuat Fungsi Destroy .....	176
Source Code 6.16 Menambahkan Button Dan Memanggil Fungsi Destroy Pada Button Tersebut.....	177
Source Code 6.17 Membuat Function Uodate Items Pada Controllers Cart .....	178
Source Code 6.18 Code Pemanggilan View Pada Halaman Keranjang Belanja .....	181
Source Code 6.19 Membuat Function Seacrh Pada Controllers.....	184

Source Code 6.20 Membuat Models Dengan Function get_keyword ...	185
Source Code 6.21 Memanggil Controllers Pada Forn Input-Nya.....	186

## **BAB I**

### **PENDAHULUAN**

Pada abad ke 21 dimana para pembisnis usaha kecil menengah atau yang biasa kita sebut dengan UKM telah menyadari mengenai penerapan sistem *E-Commerce* dalam meningkatkan daya saing bisnis usaha mereka [1]. *E-Commerce* merupakan salah satu sebuah sistem bisnis yang sangat populer untuk saat ini, dimana pada umumnya mengacu pada komunikasi bisnis dan transaksi, menjual, serta membulu produk melalui media internet (*online*) [2]. Dengan adanya penerapan *E-Commerce* kedalam dunia binis para pengusaha akan lebih terbantu dan mudah dalam melayani konsumen sehingga menimbulkan bisnis tersebut dapat mempertahankan para *customer*-nya [3]. Disamping itu efek dari penggunaan sistem *e-commerce* ini dapat menimbulkan hubungan atau kerjasama antara mitra bisnis dan pemasok [3]. Negara Indonesia merupakan salah satu negara yang sebagian besar masyarakatnya telah menggeluti usaha di bidang UKM. Hal ini menunjukkan dimana masyarakat negara Indonesia memiliki tingkat kreatifitas yang cukup tinggi. Terdapat 3 sektor kreatif di negara Indonesia yang pertama adalah kuliner dimana berupa usaha yang mengeluarkan produk berupa makan dan minuman. Adapun kota – kota yang dijadikan destinasi kuliner di Indonesia yaitu Kota Bandung, Jakarta, Bali, Yogyakarta, Solo, Semarang, dan masih banyak lagi. Sektor yang kedua adalah *fashion* dimana merupakan sebuah usaha yang mengeluarkan produk berupa pakaian, sepatu, aksesoris dan sesuatu yang dapat menunjang gaya hidup. Sektor terakhir yaitu kerajinan dimana merupakan sebuah usaha yang menghasilkan produk – produk kreatif dari buatan

tangan atau keterampilan tangan sehingga menciptakan sebuah seni atau karya.

Tidak heran apabila saat ini semua kegiatan bisnis sudah memasuki jejaringan sosial. Banyak sekali para pengusaha yang memanfaat media sosial untuk melakukan promosi produknya. Hal ini dikarenakan negara Indonesia sudah memasuki RI (Revolusi Industri) 4.0. Di era RI 4.0 ini dimana sebuah pekerjaan sudah dilakukan secara modern seperti halnya sistem *e-commerce* yang membantu kegiatan bisnis. Namun pada kenyataanya penerapan sistem *e-commerce* ini masih jarang sekali digunakan oleh negara – negara berkembang [4]. Padahal studi telah mengatakan penerapan sistem *e-commerce* ini tidak hanya dapat diterapkan oleh industri – industri yang memiliki tingkatan tinggi, namun pada kenyataannya industri dengan tingkatan rendah pun dapat menerapkannya secara optimal [5]. Dengan penerapan sistem *e-commerce* tersebut terbukti dimana industri tersebut secara perlahan akan tumbuh [5]. Dengan adanya penerapan sistem *e-commerce* tersebut dimana dapat meningkatkan produktivitas tenaga kerja signifikan terkait positif [5]. Sistem *e-commerce* juga dapat meningkatkan daya saing bagi pengusaha yang menerapkannya [6].

Berdasarkan teori – teori tersebut dimana penulis akan menciptakan sebuah sistem *e-commerce* yang mampu menyatukan para pengusaha UMKM, UKM, dan PKL. *E-Commerce* tersebut akan diberi nama JURAGAN yang dimana merupakan arti dari “Jualan Rakyat Gabungan Online”. JURAGAN dirancang untuk membantu para pengusaha – pengusaha bersaing di era RI 4.0 ini. JURAGAN akan menggabungkan para pengusaha dengan status UMKN, UKM, hingga PKL kedalam satu

wadah dengan sistem *e-commerce* dimana para pengusaha tersebut dapat mempromosikan produknya melalui media internet atau secara *online*. JURAGAN juga berfungsi sebagai media komunikasi dan transaksi melalui media internet (*online*). JURAGAN tersebut merupakan sistem *e-commerce* dengan bentuk *website* dengan berbasis komunitas yang dimana dirancang dengan menggunakan *framework codeigniter* dengan bahasa pemrograman PHP dan *MariaDB* sebagai basis datanya.

## **BAB II**

## **LANDASAN TEORI**

### **2.1 Usaha Kecil Menengah (UKM)**

Usaha Kecil Menengah (UKM) merupakan salah satu kegiatan bisnis atau usaha yang didirikan berdasarkan dari inisiatif sendiri [7]. UKM merupakan sekelompok usaha paling banyak dan terbesar di negara Indonesia [7]. Banyaknya UKM yang berdiri di negara Indonesia ini dikarenakan produk – produk yang dimilikinya sangat diminati oleh masyarakat [7]. Salah satu UKM yang sangat diminati oleh masyarakat Indonesia yaitu salah satu sektor di bidang kuliner. Hal ini dikarenakan makanan merupakan salah satu kebutuhan pokok bagi kelangsungan hidup makhluk hidup, sehingga berpeluang besar dalam pengambilan keuntungan. Sebelum melanjut ke pembahasan berikutnya, apakah kalian tahu apa perbedaan UKM dan UMKM ?. Berikut akan dijelaskan apa itu perbedaan UKM dan UMKM menurut UU No 28 Tahun 2008.

#### **2.1.1 Usaha Mikro**

Dimana suatu perusahaan dikategorikan menjadi usaha mikro apabila memiliki kriteria sebagai berikut :

1. Memiliki kekayaan bersih sebesar Rp. 50.000.000 tidak termasuk bangunan tempat usaha dan tanah.
2. Memiliki penghasilan tahunan sebesar Rp. 300.000.000.

#### **2.1.2 Usaha Kecil**

Dimana suatu perusahaan dikategorikan menjadi usaha kecil apabila memiliki kriteria sebagai berikut :

1. Memiliki kekayaan bersih dimana lebih dari Rp. 50.000.000 dan paling banyak Rp.500.000.000 bukan termasuk tanah dan bangunan usaha.
2. Memiliki penghasilan tahunan dimana lebih dari Rp. 300.000.000 dan paling banyak sebesar Rp. 2.500.000.000.

### **2.1.3 Usaha Menengah**

Dimana suatu perusahaan dikategorikan menjadi usaha menengah apabila memiliki kriteria sebagai berikut :

1. Memiliki kekayaan bersih dimana lebih dari Rp. 500.000.000 dan paling banyak sebesar Rp. 10.000.000.000 bukan termasuk tanah dan bangunan usaha.
2. Memiliki penghasilan tahunan dimana lebih dari Rp. 300.000.000 dan paling banyak sebesar Rp. 50.000.000.000.

### **2.2 Pedagang Kaki Lima (PKL)**

Pedagang Kaki Lima (PKL) dapat diartikan dimana seseorang yang melakukan usaha dengan menggunakan gerobak. Istilah ini muncul dari persepsi masyarakat yang dimana dua kaki milik pengusaha dan ditambah tiga kaki dari roda gerobak pengusaha sehingga disebut dengan pedagang kaki lima.

### **2.3 Promosi**

Promosi merupakan salah satu kegiatan dalam proses bisnis yang dimana dengan tujuan memperkenalkan suatu produk adan jasa kepada konsumen [7]. Promosi dapat diartikan juga sebagai suatu komunikasi informasi antara penjual dengan pembeli yang bertujuan untuk memberikan sebuah pengenal produk sehingga membuat pelanggan yang

asalnya tidak tahu menjadi tahu dan menarik minat untuk membeli produk tersebut.

## **2.4 Internet**

Internet merupakan sistem informasi global yang terhubung secara logika oleh *address* yang unik secara global dan berbasis pada *internet protocol* (IP), mendukung komunikasi dengan menggunakan TCP/IP sehingga membuatnya dapat diakses baik secara umum atau khusus [8].

## **2.5 E-Commerce**

*E-Commerce* didefinisikan sebagai proses pembelian, penjualan, mentransfer atau bertukar produk, jasa atau informasi dengan menggunakan jaringan komputer [9]. Dengan menggunakan bentuk-bentuk cara tradisional dari proses bisnis dan memanfaatkan jejaring sosial melalui internet, strategi bisnis dapat berhasil apabila dilakukan dengan benar, yang akhirnya dapat menghasilkan peningkatan *customer* [9]. Dengan adanya *e-commerce* tersebut dimana proses bisnis yang dilakukan akan semakin membaik dan dapat meningkatkan daya saing antar sesama industri [9]. *E-Commerce* merupakan sebuah bisnis yang populer untuk saat ini, dimana umumnya mengacu pada komunikasi bisnis dan transaksi melalui internet, menjual, dan membeli produk secara online [2].

## **2.6 Komunitas**

Komunitas pada umumnya diartikan sebagai sebuah perkumpulan sosial dari beberapa organisme yang dimana saling berbagi lingkungan dan umumnya memiliki ketertarikan serta habitat yang sama [10]. Komunitas juga dapat diartikan sebagai identifikasi serta interaksi sosial yang dibentuk dengan berbagai dimensi kebutuhan fungsional.

## **2.7 Aplikasi**

Aplikasi adalah penggunaan atau penerapan dalam suatu konsep yang menjadi sebuah pokok pembahasan, dimana aplikasi juga bisa diartikan sebagai program komputer yang diciptakan dengan tujuan membantu dan mempermudah kegiatan manusia untuk melaksanakan sebuah tugas tertentu [11].

## **2.8 Framework**

*Framework* berfungsi dalam memfasilitasi pemrograman web dan membuatnya menjadi lebih teratur [12]. Dimana *framework* akan meningkatkan produktivitas pemrograman karena menuliskan sepotong *source code* yang biasanya bersifat panjang dan membutuhkan waktu yang cukup lama kini bisa dikerjakan dalam hitungan menit [12]. *Framework* juga memiliki keunggulan dalam hal keamanan, hal ini dikarenakan *user* menggunakannya dalam jangka panjang [12]. *Framework* juga bersifat *free* sehingga banyak diminati oleh para developer karena dapat membantu developer bekerja lebih cepat [12].

## **2.9 CodeIgniter**

*Codeigniter* merupakan *application development framework* untuk membangun aplikasi menggunakan PHP [13]. Tujuan dari *codeigniter* tersebut yaitu untuk memungkinkan mengembangkan proyek lebih cepat daripada menulis kode dari awal, dengan menyediakan serangkaian *library* yang biasanya dibutuhkan oleh para pengembang, antarmuka yang sederhana, dan struktur logis untuk mengakses *library* tersebut [13]. *Codeigniter* didasarkan pada pola pengembangan MVC (*Model View Controller*) [13]. MVC adalah pendekatan perangkat lunak yang memisahkan logika aplikasi dari presentasi [13]. Hal ini memungkinkan

halaman web yang dibuat akan mengandung *source code* yang minim dikarenakan skrip PHP yang terpisah [13].

## **2.10 Hypertext Preprocessor (PHP)**

*Hypertext Preprocessor* (PHP) merupakan bahasa pemrograman *open source* yang dibuat oleh Rasmus Lerdorf pada tahun 1995 sebagai serangkaian skrip *Perl Commn Gateway Interface* (CGI). PHP memiliki perkembangan yang signifikan, sejak versi 3 dimana PHP merupakan bahasa pemrogramman yang berorientasi objek dan pada versi 5 dimana PHP memiliki tujuan untuk menjadi bahasa pemrograman yang umum dalam pengembangan web. Seperti java, bahasa PHP menggabungkan antarmuka dan pewarisan tunggal. Namun pada versi 7, kinerja PHP menjadi dua kali lebih cepat dari PHP 5. Hingga saat ini dimana PHP 7 telah digunakan untuk mengembangkan sistem manajemen dan pelatihan [14].

## **2.11 Database (Basis Data)**

*Database* secara sederhana dapat kita artikan sebagai data. Secara teori dimana *database* adalah sekumpulan data atau informasi yang kompleks, data – data tersebut disusun menjadi beberapa kelompok dengan tipe data yang sejenis. Dimana data tersebut akan saling berhubungan satu sama lain atau berdiri sendiri sehingga dapat dengan mudah untuk di akses [15].

## **2.12 MariaDB**

*MariaDB* merupakan sistem manajemen basis data *relasional* yang dikembangkan oleh *MySQL*. *MariaDB* dikembangkan oleh komunitas pengembang yang dimana sebelumnya telah berkontribusi untuk basis data *MySQL*. Alasan dimana pengembang *MySQL* membangun *MariaDB* yaitu

telah diakuiinya *MySQL* oleh pihak *oracle* sehingga membuat *MySQL* menjadi sebuah produk yang berlisensi *proprietary* [16].

## **BAB III** **PERANCANGAN**

Perancangan sistem merupakan salah satu tahap yang sangat penting dilakukan apabila kalian sedang dalam proses tahap menganalisis sistem tersebut. Apa itu analisis ?, analisis menurut penulis adalah sebuah kegiatan untuk memecahkan suatu masalah guna menghindari kesalahan – kesalahan pada sistem yang akan dibangun. Analisis sistem juga merupakan sebuah fondasi awal dalam keberhasilan sistem tersebut.

Perancangan adalah suatu kegiatan dalam merancang dan juga mendesain suatu sistem secara jelas yang biasanya berisi mengenai *desain user interface*, peroses dalam pengolahan sebuah data dan prosedur – prosedur lainnya dalam mendukung kinerja operasi sistem tersebut. Pada dasarnya perancangan sistem dibuat dengan tujuan untuk membantu para programmer atau pihak – pihak yang berperan dalam sistem tersebut dalam membangun sistem tersebut.

Untuk membangun aplikasi dengan konsep *e-commerce* seperti JURAGAN (Jualan Rakyat Gabungan *Online*) dimana perlu dilakukannya tahap analisis dan perancangan terlebih dahulu. Dalam buku ini dimana penulis telah menyediakan perancangan dalam membuat sistem JURAGAN tersebut yang terdiri dari perancangan *use case* dan *desain user interface* yang akan digunakan.

### **3.1 Use Case JURAGAN**

*Use case* adalah teknik yang digunakan oleh seorang analisis dalam melakukan pengembangan sebuah *software* dan sistem informasi guna menangkap kebutuhan fungsional dari sistem yang akan dibangun. *Use case* berfungsi dalam menjelaskan interaksi – interaksi yang terjadi antara aktor dan juga inisiator dengan sistem tersebut.

Manfaat dalam membuat *use case* antara lain yaitu digunakan dalam melakukan komunikasi dengan *end user* dan *domain expert*. *Use case* juga dapat memastikan pemahaman secara tepat mengenai *requirement* atau kebutuhan sistem. *Use case* digunakan dalam mengidentifikasi siapa yang akan berinteraksi dengan sistem dan apa yang harus sistem perbuat atau kerjakan.

Dalam membuat *use case* dimana terdapat komponen – komponen yang perlu kita ketahui terlebih dahulu. Apa saja yah komponen tersebut ? mari simak penjelasan mengenai komponen – komponen *use case* dibawah ini.



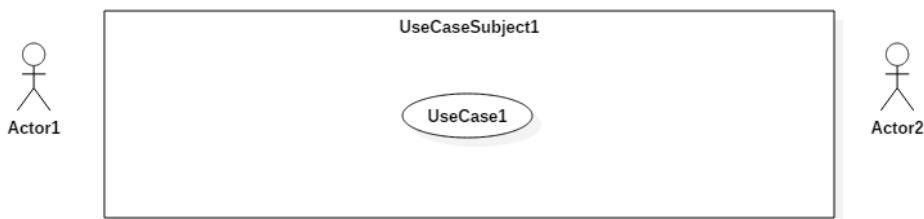
*Gambar 3.1 Komponen Aktor Use Case*

Aktor merupakan komponen yang menggambarkan *user* atau seseorang yang akan berinteraksi dengan sistem tersebut. Aktor hanya bisa melakukan interaksi terhadap sistem yang dimana menandakan aktor bukan pemegang kendali penuh pada *use case*.



Gambar 3.2 Komponen UseCase

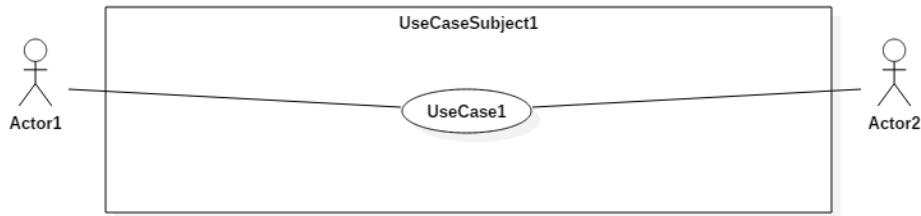
*UseCase* merupakan komponen yang berfungsi sebagai memberikan gambaran fungsional sistem yang akan dibangun yang dimana dapat membuat pengguna lebih mengerti dalam mengaplikasikan sistem tersebut.



Gambar 3.3 Komponen Use Case Subject

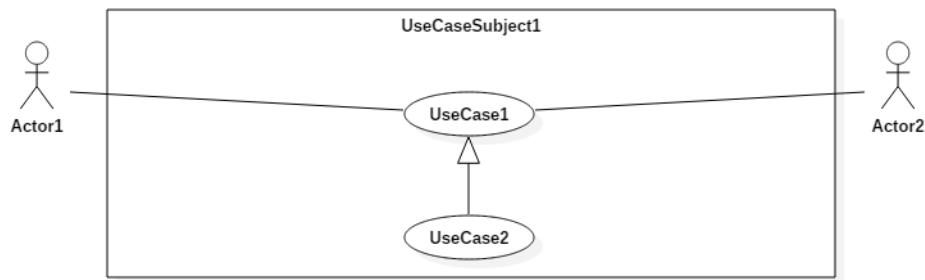
*Use case subject* merupakan komponen yang berfungsi sebagai tempat menampungnya *subject – subject use case* atau sebagai sebuah *frame* dari *user case* tersebut.

Selain komponen – komponen yang dimiliki oleh *use case* dimana kita juga perlu memahami mengenai garis – garis relasi pada *use case*. Relasi adalah sebuah hubungan yang dimana berfungsi untuk menghubungkan komponen aktor dengan *usecase*. Adapun relasi – relasi yang dimiliki oleh *use case* adalah sebagai berikut.



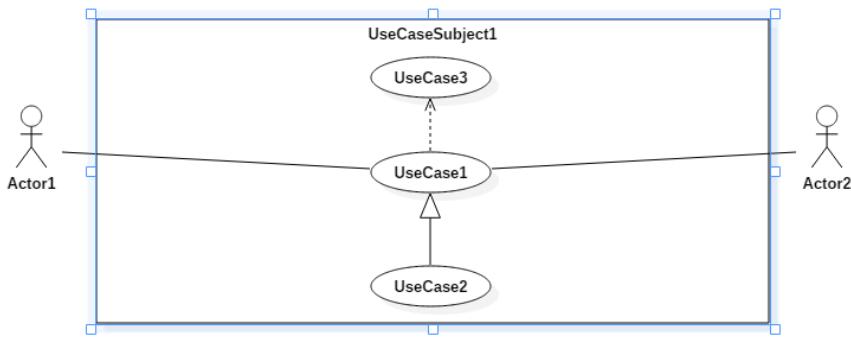
*Gambar 3.4 Relasi Association*

Relasi *association* berfungsi untuk menghubungkan *link* antar *element* pada *use case*. *Association* berbentuk garis lurus tanpa disertai anak panah di setiap ujungnya.



*Gambar 3.5 Relasi Generalization*

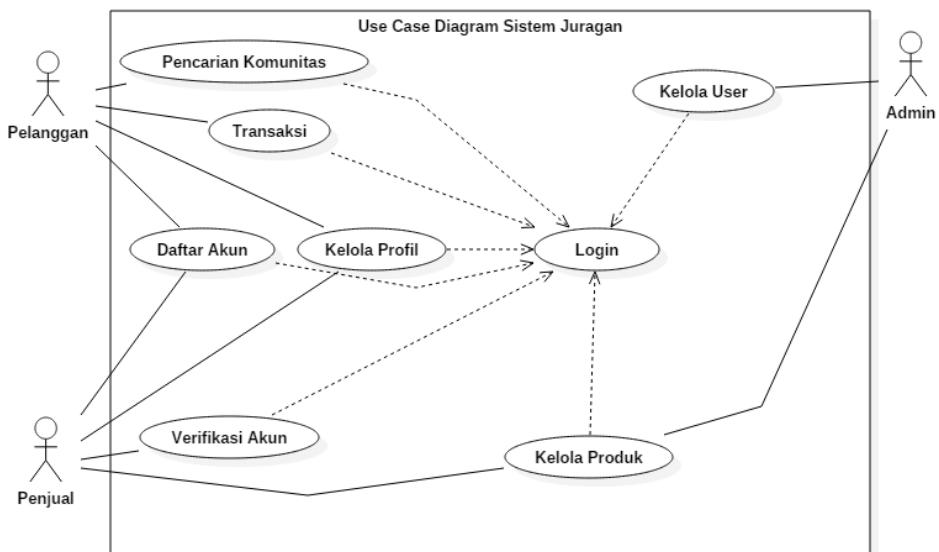
Relasi *generalization* berfungsi sebagai penghubung sebuah elemen yang menjadi spesialisasi terhadap elemen yang lain. *Generalization* biasa berbentuk garis lurus panjang disertai anak panah pada ujungnya.



Gambar 3.6 Relasi Despendency

Relasi *despendency* yaitu merupakan sebuah elemen yang bergantung beberapa cara dengan elemen yang lainnya. *Despendency* biasanya berbentuk garis putus – putus dengan anak panah di ujungnya.

Dalam membangun sistem JURAGAN kali ini dimana penulis sudah memabngun *use case* yang terdiri dari tiga aktor yaitu pelanggan, penjual, dan juga admin.



Gambar 3.7 Use Case Sistem JURAGAN

Dari *use case* tersebut dimana penulis akan memberikan beberapa penjelasan seperti definisi aktor, definisi *use case*, dan *use case scenario* agar *use case* yang diberikan dapat dibaca dengan jelas.

### 3.1.1 Definisi Aktor

Definisi aktor merupakan penjasalan dari aktor – aktor yang terdapat pada *use case* JURAGAN yang dimana aktor dapat melakukan apa saja dan berelasi terhadap *use case* apa saja. Penjelasan mengenai aktor – aktor tersebut dapat dilihat pada tabel 3.1

*Tabel 3. 1 Definisi Aktor Pada Use Case JURAGAN*

No.	Aktor	Deskripsi
1.	Pelanggan	a. Daftar Akun b. <i>Login</i> c. Kelola Profil d. Transaksi e. Pencarian Komunitas
2.	Penjual	a. Daftar Akun b. <i>Login</i> c. Kelola Profil d. Verifikasi Akun e. Kelola Produk
3	Admin	a. Kelola User b. Kelola Produk

### **3.1.2 Definisi Use Case**

Definisi *use case* ini menjelaskan mengenai pekerjaan apa yang dilakukan terhadap komponen *use case* pada *use case* JURAGAN tersebut. Untuk lebih jelasnya diamana dapat dilihat pada tabel 3.2.

*Tabel 3. 2 Definisi Komponen Use Case Pada Use Case JURAGAN*

No.	Use Case	Deskripsi
1.	Daftar Akun	a. Melakukan proses pendaftaran atau <i>registration</i> akun pada sistem JURAGAN.
2.	<i>Login</i>	a. Melakukan proses <i>login</i> terhadap akun yang sudah melakukan <i>registration</i> atau sudah terdaftar di dalam sistem JURAGAN.
3.	Kelola Profil	a. Melakukan pengelolaan profil terhadap akun <i>user</i> .
4.	Pencarian Komunitas	a. Melakukan pencarian produk berdasarkan komunitas.
5.	Transaksi	a. Melakukan penambahan produk kedalam keranjang. b. Melakukan proses transaksi.
6.	Verifikasi Akun	a. Mengaktifkan akun toko yang sudah terdaftar dengan menginputkan No KTP dan <i>upload</i> KTP

7	Kelola Produk	<ul style="list-style-type: none"> <li>a. Melakukan pengelolaan produk terhadap akun level toko.</li> <li>b. Melakukan <i>view</i> produk terhadap akun level pelanggan.</li> </ul>
8	Kelola User	<ul style="list-style-type: none"> <li>a. Melakukan pengelolaan akun <i>user</i> oleh admin.</li> </ul>

### 3.3.3 Scenario Use Case JURAGAN

*Scenario use case* merupakan penjelasan mengenai jalannya *use case* yang dibuat. Pada tabel 3.3 merupakan penjelasan *scenario* dari *use case* JURAGAN.

*Tabel 3.3 Scenario Use Case Daftar Akun*

Identifikasi	
No.	JR1
Nama	Daftar Akun
Tujuan	Mendaftarkan akun pada sistem JURAGAN.
Deskripsi	Melakukan proses pendaftaran akun guna dapat melakukan proses <i>login</i> pada sistem JURAGAN.
Aktor	Pelanggan Dan Penjual

<b>Skenario</b>	
Kondisi Awal	Halaman registrasi
<b>Aksi Aktor</b>	<b>Reaksi Sistem</b>
1. Melakukan <i>input</i> data registrasi yang dibutuhkan sistem.	a. -
2. Menekan <i>button</i> daftar akun.	b. Melakukan validasi terhadap data yang dikirim.
3. -	c. Memberikan notifikasi akun berhasil terdaftar dan harus di aktivasi melalui <i>email</i> .
4. Membuka <i>email</i> yang digunakan untuk mendaftar.	d. Mengirimkan <i>email</i> kepada <i>user</i> yang mendaftar.
5. Membuka <i>email</i> dan melakukan aktivasi.	e. Mengirimkan <i>token</i> kepada tabel <i>user_token</i> .
6. -	f. Memberikan notifikasi akun sudah aktif.

Tabel 3.4 Scenario Use Case Login

<b>Identifikasi</b>	
No.	JR2

Nama	<i>Login</i>
Tujuan	Membuka akses lebih luas pada sistem JURAGAN.
Deskripsi	Mengakses sistem JURAGAN secara menyeluruh dengan kategori <i>user</i> tertentu.
Aktor	Pelanggan Dan Penjual
<b>Skenario</b>	
Kondisi Awal	Halaman login
<b>Aksi Aktor</b>	<b>Reaksi Sistem</b>
1. Melakukan <i>input email</i> dan <i>password</i> yang sudah terdaftar pada sistem	a. -
2. Memberikan <i>action</i> pada <i>button login</i>	b. Memulai proses validasi
3. -	c. Menampilkan halaman utama sistem JURAGAN.

*Tabel 3.5 Scenario Use Case Kelola Profil*

Identifikasi	
No.	JR3
Nama	Kelola Profil

Tujuan	Melengkapi data profil.
Deskripsi	Melakukan pengelolaan terhadap data profil <i>user</i> .
Aktor	Pelanggan dan Penjual
<b>Skenario</b>	
Kondisi Awal	Halaman Edit Profil
<b>Aksi Aktor</b>	
1. Melakukan <i>input</i> data profil secara lengkap.	a. -
2. Memberikan <i>action</i> terhadap <i>button edit profile</i>	b. Melakukan validasi terhadap data yang dikirim.
3. -	c. Memberikan notifikasi edit profil berhasil.

*Tabel 3.6 Scenario Use Case Transaksi*

Identifikasi	
No.	JR5
Nama	Transaksi
Tujuan	Menambahkan produk ke dalam keranjang belanja.

Deskripsi	Menambahkan produk ke dalam keranjang belanja untuk melakukan proses transaksi.
Aktor	Pelanggan
<b>Skenario</b>	
Kondisi Awal	Halaman Produk
<b>Aksi Aktor</b>	
1. Memberikan aksi terhadap <i>button tambah</i> ke keranjang	a. Mengirimkan <i>id</i> produk yang ditambahkan.
2. -	b. Melakukan validasi
3. -	c. Menambahkan produk ke dalam keranjang.

Tabel 3.7 Scenario Use Case Verifikasi Akun

Identifikasi	
No.	JR6
Nama	Verifikasi Akun
Tujuan	Mengaktifkan akun toko.
Deskripsi	Mengaktifkan akun toko dengan menginputkan No KTP dan <i>upload</i> foto KTP.
Aktor	Penjual

<b>Skenario</b>	
Kondisi Awal	Halaman Verifikasi Akun
<b>Aksi Aktor</b>	<b>Reaksi Sistem</b>
1. Menginputkan no KTP	a. -
2. <i>Upload</i> foto KTP	b. -
3. Memberikan aksi pada <i>button</i> verifikasi akun	c. Melakukan validasi terhadap data yang dikirim
4. -	d. Memberikan notifikasi akun toko sudah aktif.

Tabel 3.8 Scenario Use Case Kelola Produk (Penjual)

<b>Identifikasi</b>	
No.	JR7
Nama	Kelola Produk
Tujuan	Mengelola data produk
Deskripsi	Melakukan pengelolaan terhadap data produk.
Aktor	Penjual
<b>Skenario</b>	
Kondisi Awal	Halaman Inventori

Aksi Aktor	Reaksi Sistem
1. Memberikan aksi terhadap <i>button tambah produk</i>	a. Memunculkan <i>pop up form input</i> produk
2. Melakukan <i>input</i> data produk yang dibutuhkan oleh sistem	b. -
3. Memberikan aksi pada <i>button simpan produk</i>	c. Melakukan validasi terhadap data yang dikirim
4. -	d. Menyimpan data ke dalam tabel barang
5. -	e. Melakukan <i>view</i> produk
6. Memberikan aksi pada <i>button edit produk</i>	f. Menampilkan <i>form edit produk</i>
7. Melakukan edit data produk.	g. -
8. Memberikan aksi pada <i>button edit produk</i>	h. Melakukan validasi terhadap data yang dikirim
9. -	i. Memberikan notifikasi produk berhasil di edit

*Tabel 3.9 Scenario Use Case Kelola Produk (Admin)*

<b>Identifikasi</b>	
No.	JR8
Nama	Kelola Produk
Tujuan	Mengelola data produk
Deskripsi	Melakukan pengelolaan terhadap data produk.
Aktor	Admin
<b>Skenario</b>	
Kondisi Awal	Halaman Admin
<b>Aksi Aktor</b>	
1. Memilih menu data produk pada sidebar	a. Menampilkan halaman data produk
2. -	b. Mengambil data produk dari tabel barang
3.	c. Menampilkan produk

*Tabel 3.10 Scenario Use Case Kelola User*

<b>Identifikasi</b>	
No.	JR9
Nama	Kelola User

Tujuan	Mengelola user yang telah terdaftar
Deskripsi	Melakukan pengelolaan data terhadap <i>user</i> yang terdaftar
Aktor	Admin
<b>Skenario</b>	
Kondisi Awal	Halaman Admin
<b>Aksi Aktor</b>	
1. Memilih menu <i>user</i> pada <i>sidebar</i>	a. Menampilkan halaman <i>user</i>
2. -	b. Mengambil data dari tabel <i>user</i>
3. -	c. Menampilkan data <i>user</i>

### 3.2 Perancangan Basis Data (*Database*)

Basis Data atau *database* merupakan sekumpulan berbagai informasi atau data yang dimana tersimpan pada suatu media komputer dan tersusun secara sistematik sehingga dapat diolah dan diperiksa dengan menggunakan program komputer. Pada umumnya pengolahan *database* pada suatu media komputer biasa dikenal dengan sebutan DBMS (*Database Management System*) yang dimana memiliki tujuan untuk mempermudah proses pengelolaannya. *Database* juga dapat diartikan sebagai sekumpulan tabel – tabel yang dapat menyimpan suatu data dan informasi.

Pada perancangan pembuatan sistem JURAGAN, *database* sangatlah berperan untuk menyimpan data – data seperti data barang untuk menyimpan produk – produk bagi penjual dan data *user* yang berguna untuk menyimpan informasi – informasi *user*. Dalam pembuatan sistem JURAGAN ini dimana penulis telah menggunakan *MariaDB* sebagai *databasenya*. Mengapa sih harus menggunakan *database MariaDB* ?, menurut pengetahuan penulis dimana pengembangan *database MariaDB* lebih terbuka dan cepat, memiliki performa yang lebih baik, merupakan salah satu *database* yang sangat populer dikalangan para programming, sangat kompetibel dan juga bersifat *open source* yang merupakan keunggulan – keunggulan pada *database MariaDB* tersebut. Pada pembuatan sistem JURAGAN ini dimana penulis menggunakan tiga tabel dengan rancangan sebagai berikut.

### 3.2.1 Tabel *User*

Tabel *user* dibuat untuk menampung data atau informasi yang berkaitan dengan *user*. *User* disini merupakan seseorang yang terlibat dalam pengelolaan sistem JURAGAN. *User* tersebut dibagi kedalam 3 level yang dimana level 1 merupakan *admin*, level 2 *user* pelanggan dan level 3 *user* penjual. Pada tabel 3.11 dimana teman – teman dapat melihat susunan struktur pada tabel *user* tersebut.

*Tabel 3.11 Struktur Tabel User*

Nama	Tipe	Keterangan
Id	<i>Integer (11)</i>	Merupakan <i>id</i> dari <i>user</i> dan juga <i>primary key</i> dari tabel <i>user</i> .
Name	<i>Varchar (30)</i>	<i>Field</i> yang berguna untuk menampung nama dari <i>user</i> .

<i>Email</i>	<i>Varchar (25)</i>	<i>Field</i> yang berguna untuk menampung <i>email</i> dari <i>user</i> .
<i>Password</i>	<i>Varchar (12)</i>	<i>Field</i> yang berguna untuk menampung <i>password</i> dari akun <i>user</i> dan bersifat md5.
<i>Image</i>	<i>Varchar (15)</i>	<i>Field</i> yang berguna untuk menampung foto profil <i>user</i> .
<i>Role_id</i>	<i>Integer (11)</i>	<i>Field</i> yang berguna untuk menampung <i>id</i> level <i>user</i> tersebut.
<i>Is_active</i>	<i>Integer (11)</i>	<i>Field</i> yang berguna untuk menampung <i>id active user</i> . Dimana angka “1” menandakan akun <i>user</i> tersebut sudah aktif dan angka “0” menandakan akun <i>user</i> tersebut belum aktif.
<i>Date_created</i>	<i>Integer (11)</i>	<i>Field</i> yang berguna untuk menampung tanggal kapan <i>user</i> tersebut daftar.
Kebijakan	<i>Varchar (15)</i>	<i>Field</i> yang berguna untuk menampung kebijakan bagi <i>user</i> pelanggan. Berupa setuju dan tidak setuju.
Tlpn	<i>Varchar (15)</i>	<i>Field</i> tersebut berguna untuk menampung nomor telepon dari <i>user</i> tersebut.
Alamat	<i>Varchar (60)</i>	<i>Field</i> tersebut berguna untuk menampung alamat berupa jalan, RT/RW, dan nomor rumah.

Kecamatan	Varchar (25)	<i>Field</i> tersebut berguna untuk menampung alamat berupa kecamatan dari <i>user</i> tersebut.
Kelurahan	Varchar (25)	<i>Field</i> tersebut berguna untuk menampung alamat berupa kelurahan dari <i>user</i> tersebut.
Kota	Varchar (25)	<i>Field</i> tersebut berguna untuk menampung alamat berupa kota dari <i>user</i> tersebut.
Kode_pos	Varchar (15)	<i>Field</i> tersebut berguna untuk menampung alamat berupa kode pos dari <i>user</i> tersebut.
Status_toko	Varchar (25)	<i>Field</i> tersebut berguna untuk menampung informasi mengenai toko tersebut sudah terverifikasi atau belum.
Cat_toko	Text	<i>Field</i> tersebut berguna untuk menampung deskripsi atau catatan toko bagi <i>user</i> level penjual.
No_ktp	Varchar(20)	<i>Field</i> tersebut berguna untuk menampung nomor KTP <i>user</i> .
Img_ktp	Varchar(25)	<i>Field</i> tersebut berguna untuk menyimpan foto ktp <i>user</i> .

### 3.2.2 Tabel *User\_token*

Tabel *user\_token* tersebut dibuat untuk menampung data token dari *user* tersebut. Karena sistem JURAGAN menggunakan validasi melalui *email* yang dimana nantinya setiap *user* akan memimiliki

token. Token tersebut dapat digunakan untuk melakukan fungsi reset *password* apabila *user* mengalami lupa *password*. Adapun perancangan struktur pada tabel *user\_token* dapat dilihat pada tabel 3.12.

*Tabel 3.12 Struktur Tabel User\_token*

Nama	Tipe	Keterangan
Id	<i>Integer (11)</i>	Merupakan <i>primary key</i> dari tabel tersebut.
Email	<i>Varchar (25)</i>	Merupakan <i>email</i> dari <i>user</i> yang di daftarkan.
Token	<i>Varchar (128)</i>	Merupakan <i>field</i> yang berfungsi untuk menampung token tersebut.
Date_created	<i>Integer (11)</i>	Merupakan waktu <i>deadline</i> untuk melakukan validasi.

### 3.2.3 Tabel tb\_barang

Tabel tb\_barang tersebut dibuat untuk menampung data atau informasi mengenai produk – produk yang akan di *input* kan oleh *user* dengan level penjual. Mengenai struktur dari tb\_barang tersebut dimana teman – teman dapat melihatnya pada tabel 3.13.

*Tabel 3.13 Struktur Tabel tb\_barang*

Nama	Tipe	Keterangan
Id_brg	<i>Integer (11)</i>	Merupakan <i>id</i> dari barang tersebut dan sebagai <i>primary key</i> .

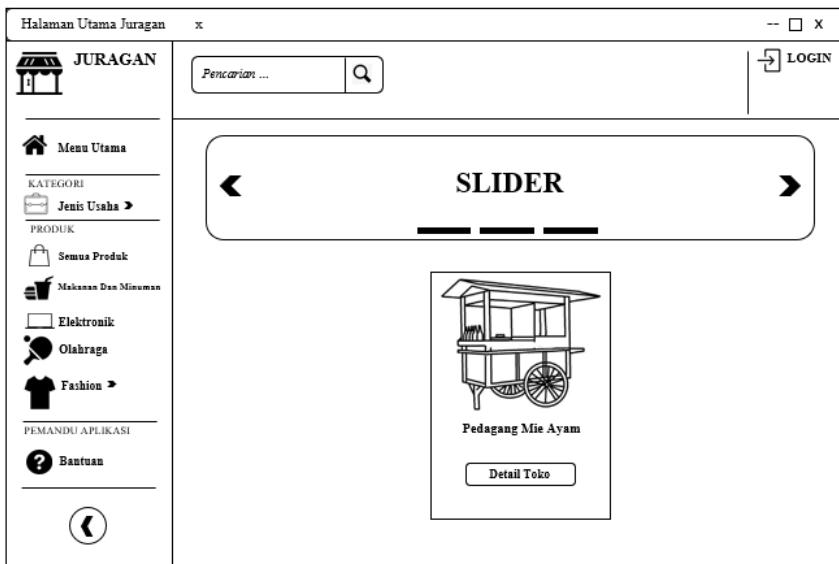
<i>Id</i>	<i>Integer (11)</i>	Merupakan <i>id</i> dari <i>user</i> dengan level penjual.
<i>Email</i>	<i>Varchar (15)</i>	Merupakan email dari <i>user</i> dengan level <i>penjual</i> .
<i>Nama_brg</i>	<i>Varchar (25)</i>	Merupakan nama dari produk yang akan di <i>input</i> .
<i>Keterangan</i>	<i>Varchar (60)</i>	Merupakan keterangan dari produk yang akan di <i>input</i> .
<i>Detail</i>	<i>Text</i>	Merupakan deskripsi dari produk tersebut.
<i>Kategori</i>	<i>Varchar (20)</i>	Merupakan kategori dari produk yang akan di <i>input</i> . Terdapat 4 kategori yaitu makanan dan minuman, elektronik, olahraga, dan <i>fashion</i> .
<i>Harga</i>	<i>Integer (11)</i>	Merupakan <i>field</i> untuk menampung harga dari produk tersebut.
<i>Stok</i>	<i>Integer (4)</i>	Merupakan <i>field</i> untuk menampung jumlah stok pada produk tersebut.
<i>Tlpn</i>	<i>Varchar (15)</i>	Merupakan <i>field</i> untuk menampung nomor telpon <i>user</i> dengan level penjual.
<i>Gambar</i>	<i>Text</i>	Merupakan <i>field</i> untuk menampung gambar dari produk tersebut.

### **3.3 User Interface Sistem JURAGAN**

Pernahkah kalian melihat tampilan suatu sistem ? Template – template dari suatu sistem ? apa teman – teman tahu yang dimaksud dengan *user interface* ? *User interface* atau biasanya disingkat dengan *UI* merupakan suatu bentuk visual, template, atau tampilan pada sebuah *software* atau *website* yang dibuat dengan tujuan memberikan desain interaksi didalamnya. Lalu apa itu desain interaksi ?, desain interaksi merupakan struktur dan juga prilaku antara suatu produk berupa *webiste* atau *software* terhadap penggunanya. Desain interaksi dapat disebut juga sebagai jembatan komunikasi anatara *user* dan produk tersebut.

Dalam sistem baik itu *website* maupun *software user interface* sangatlah berperan penting dikarenakan *user interface* merupakan penghubung langsung antara sistem dengan *user* itu sendiri. Dalam dunia *e-commerce* dimana *user interface* dapat meningkat kesuksesan, mengapa ?, hal ini dikarenaka dengan adanya *user interface* yang mudah digunakan dan mudah dimengerti atau *user friendly* dapat menyebabkan aplikasi tersebut akan sering digunakan. Dengan *user interface* yang menarik juga dapat mengundang akan terus menggunakan aplikasi tersebut. Menarik disini bukan dalam atian sistem kita harus yang mewah namun dengan tampilan dan desain interaksi yang nyaman digunakan akan membuat *user* betah menggunakan aplikasi tersebut.

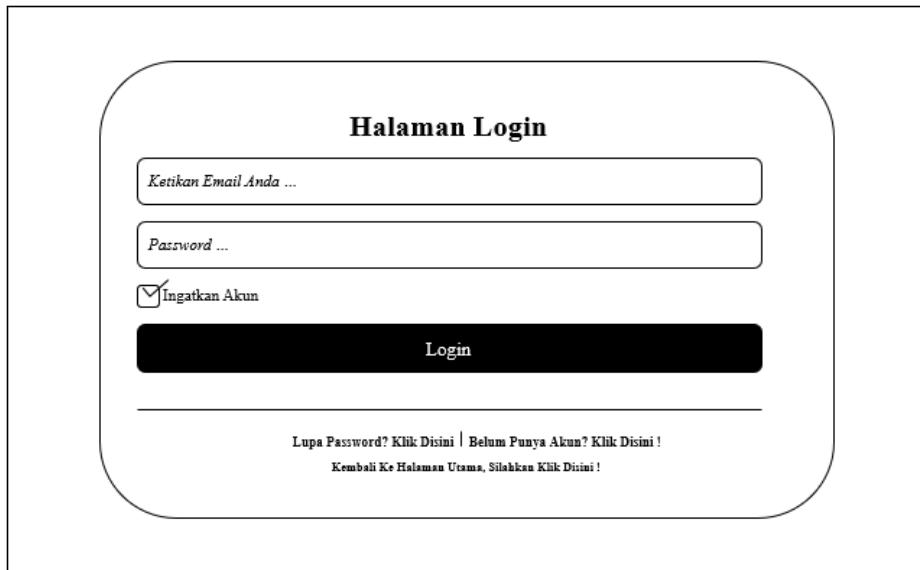
Dalam sistem JURAGAN ini dimana *user interface* akan di desain dengan cukup nyaman dan bersifat *user friendly*. Adapun perancangan yang dilakukan untuk membangun sistem JURAGAN dapat teman – teman lihat pada gambar 3.8 – 3.20.



Gambar 3.8 Perancangan UI Halaman Utama

Pada gambar 3.8 dimana teman – teman dapat melihat perancangan *user interface* pada halaman utama atau tampilan awal saat *user* mengakses sistem JURAGAN. Dimana *user interface* tersebut di bagi menjadi lima halaman *template* yaitu *header*, *sidebar*, *topbar*, *main*, dan *footer*. Sebelum membahas lebih jauh dimana teman – teman harus mengetahui terlebih dahulu apa itu *template header*, *sidebar*, *topbar*, *main* dan *footer*. Mari kita bahas satu per satu, *template header* merupakan suatu *template* yang letaknya berada di bagian atas sistem tersebut, biasanya *header* berisikan judul halaman dari sistem tersebut. Pada perancangan sistem JURAGAN tersebut dimana bagian *header* terdapat pada bagian paling atas yang merupakan judul dari halaman yang sedang di akses. *Template sidebar* merupakan sebuah *template* yang biasanya di letakkan pada bagian sebelah kiri sistem yang diberikan kolom – kolom dengan fungsi *link navigasi*. Coba teman – teman perhatikan bagian sebelah kiri

pada perancangan *user interface* JURAGAN tersebut, dimana terdapat *icon – icon* dengan fungsi *link navigasi*. *Icon – icon* tersebut juga bisa disebut sebagai *desain interaksi* mengapa ?, karena pada dasarnya *icon* berbentuk gambar sehingga membuat *user* lebih mengerti kegunaan dari tiap *icon* tersebut. Dengan adanya *icon* tersebut dimana membuat tampilan sistem menjadi lebih menarik. Pada tiap *icon* dimana terdapat keterangan singkat untuk mempermudah *user* mengetahui fungsi dari *icon* tersebut. *Template* selanjutnya adalah *topbar* yang merupakan sebuah *template* yang terdapat pada bagian atas sistem namun berbeda dengan *header*. Apa perbedaannya ? dimana *header* biasanya berisikan judul namun *topbar* diisikan dengan fungsi – fungsi atau fitur dari sistem tersebut seperti pencarian atau *icon* keranjang. Selanjutnya adalah *template main* yang merupakan bagian penting dari setiap *template*, mengapa demikian ? hal ini dikarenakan *main* bagian utama dari sistem tersebut, semua informasi akan di tampilkan pada *main*. Pada sistem JURAGAN dimana *template* akan diisikan dengan *slider* papan iklan dan foto profil dari pedagang – pedagang yang di ambil langsung dari *database*. *Template* yang terakhir merupakan *footer* yang biasanya ditampilkan pada bagian bawah sistem. *Footer* biasanya berisikan *copyright* dari sistem tersebut, namun *footer* juga dapat diisi dengan informasi – informasi seperti kontak atau hal lainnya.



Gambar 3.9 Perancangan UI Halaman Login

Pada gambar 3.9 merupakan perancangan pada halaman *login* pada sistem JURAGAN. Halaman *login* merupakan sebuah halaman untuk memulai aktivitas lebih jauh pada suatu sistem, contohnya pada sistem *e-commerce* dimana sebelum melakukan transaksi dengan keadaan belum *login* pasti teman – teman akan diaragkan kembali ke halaman *login*. Pada perancangan halaman *login* tersebut dimana terdapat dua buah *textbox* yang berfungsi untuk menampung *inputan email* dan *password*. Apa itu *textbox* ? *textbox* merupakan sebuah *field* yang digunakan untuk memasukkan *text* dan lain sebagainya. Pada perancangan halaman *login* sistem JURAGAN tersebut dimana *textbox* telah diberikan fungsi *placeholder* yang dimana bertujuan untuk membuat *desain* tersebut begitu minimalis namun mudah dimengerti. Lalu apa sih fungsi *placeholder* itu ?, *placeholder* merupakan sebuah fitur yang diberikan oleh *html 5*, dimana *placeholder* berfungsi untuk membuat keterangan atau penamaan pada *form* dan letaknya berada pada halaman *form* tersebut. Selain *textbox* terdapat juga *button* pada perancangan sistem JURAGAN tersebut. *Button* merupakan sebuah perangkat *user interface* dengan bentuk tombol. Pada *button* tersebut diberikan

nama *login*, hal seperti ini dapat kita contohkan sebagai desain interaksi. *Button* berfungsi sebagai memberikan aksi terhadap sistem tersebut, sebagai contoh apabila *button login* tersebut di klik maka aksi *login* akan dijalankan. Di bagian bawah *buttoh* dimana terdapat fungsi “hr” yang merupakan fitur dari HTML yang apabila diterapkan akan memunculkan garis lurus horizontal. Pemberian fungsi “hr” ini diberikan sebagai pembatas antara *button* dengan *text* dibawahnya. Pada perancangan halam *login* sistem JURAGAN tersebut juga terdapat beberapa *text* dengan fungsi *link* yang dimana apabila di klik akan menuju ke suatu halaman tertentu, contohnya apabila teman – teman mengklik *text* “Kembali ke halaman utama, klik disini !” yang dimana teman – teman akan berpindah halaman ke halaman utama sistem JURAGAN.

The diagram shows a wireframe of a user interface for account registration. The main title is "Daftarkan Akun Anda!". There are four input fields: "Nama Lengkap ...", "Email Anda ...", "Password Anda ...", and "Ulangi Password Anda ...". A large black button labeled "Daftar Akun" is positioned below the password fields. At the bottom of the form, there are two links: "Lupa Password ? Klik Disini !" and "Sudah Punya Akun ? Silahkan Login !".

Gambar 3.10 Perancangan UI Halaman Daftar Akun

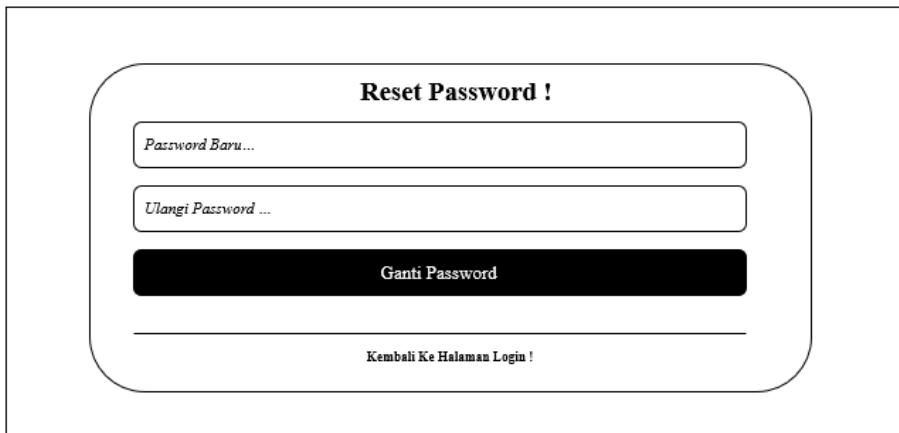
Pada gambar 3.10 merupakan perancangan *user interface* pada halaman daftar akun pada sistem JURAGAN. Halaman Daftar Akun ini berfungsi sebagai *form* untuk melakukan registrasi data diri *user*. Pada halaman daftar akun tersebut dimana komponen – komponen *template* yang

digunakan sama dengan komponen – komponen yang berada pada halaman *login*. Komponen – komponen tersebut antara lain *textbox*, *text*, *button*, dan fungsi “hr” dimana yang bedakan adalah jumlah dan fungsi dari komponen – komponen tersebut. Pada halaman daftar akun dapat dilihat jumlah *textbox* lebih banyak dari pada halaman *login*. Untuk komponen – komponen lainnya seperti *button* dan juga fungsi “hr” sama – sama terdapat satu buah.



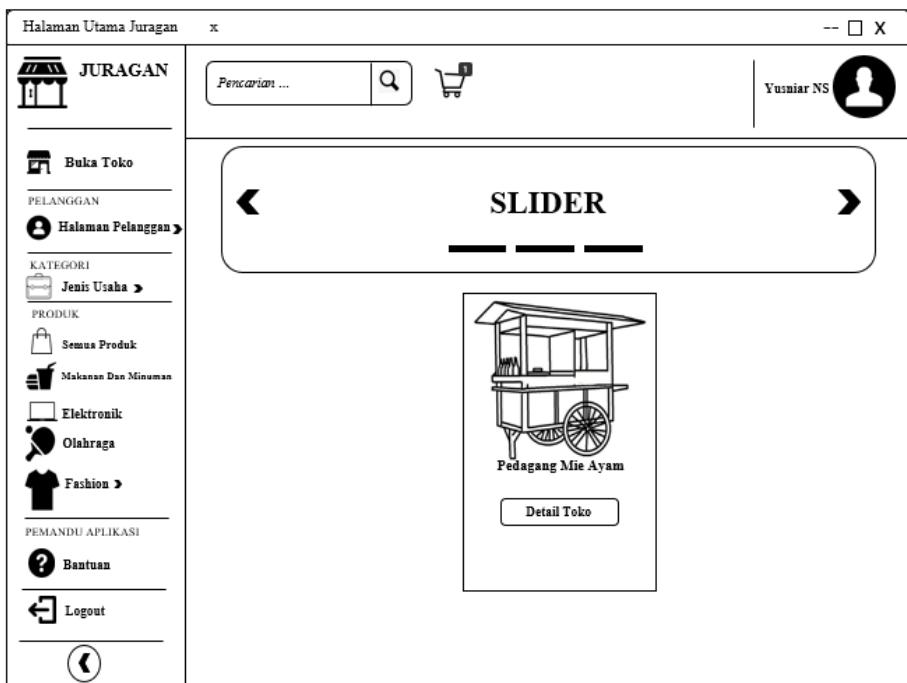
*Gambar 3.11 Perancangan UI Halaman Lupa Password*

Pada gambar 3.11 merupakan perancangan *user interface* halaman lupa *password* pada sistem JURAGAN. Halaman Lupa *Password* tersebut dibuat untuk melakukan reset *password* apabila *user* mengalami lupa terhadap *password* yang sudah dibuat. Pada halaman *reset password* dimana komponen – komponen yang digunakan berupa *textbox*, *button*, fungsi “hr” dan *text* yang dimana masing – masing berjumlah satu buah.



*Gambar 3.12 Perancangan UI Halaman Reset Password*

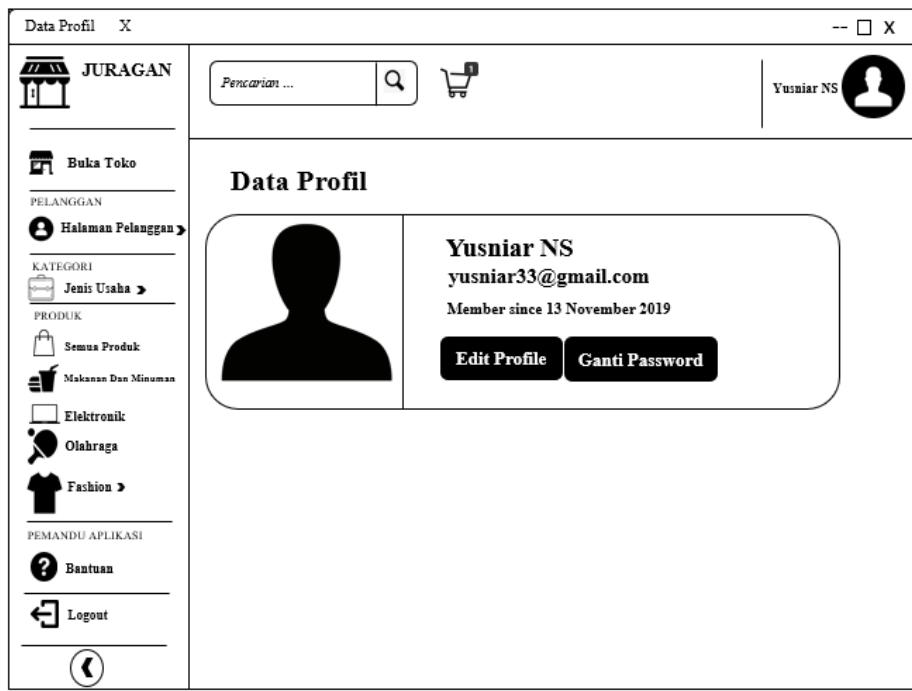
Pada gambar 3.12 merupakan perancangan *user interface* halaman reset *password* pada sistem JURAGAN. Pada halaman reset *password* tersebut dimana lanjutan dari halaman lupa *password*. Halaman reset *password* tersebut berfungsi untuk melakukan pembuatan *password* baru terhadap *user* tersebut. Pada halaman reset *password* komponen – komponen yang digunakan adalah 2 *textbox*, 1 *button*, fungsi “hr”, dan 1 *text* dengan fungsi *link*. Pada tiap – tiap *textbox* dimana terdapat fungsi *placeholder* yang merupakan sebuah *text* bayangan. Fungsi *placeholder* merupakan salah satu jenis desain interaksi dari sistem JURAGAN pada halaman *reset password*. *Placeholder* juga digunakan untuk membuat desain pada suatu sistem terlihat lebih minimalis dan mudah digunakan. Pada *button* dimana diberikan keterangan Ganti Password yang merupakan salah satu jenis dari desain interaksi juga. *Button* tersebut dibuat untuk memulai proses atau aksi ganti *password*.



Gambar 3.13 Perancangan UI Halaman Utama Kondisi Login

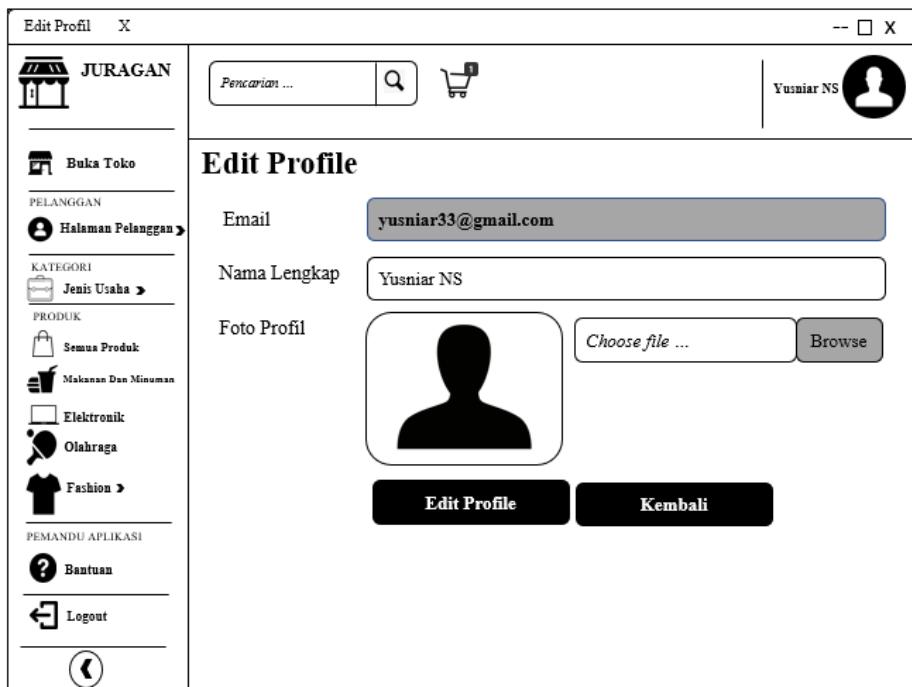
Pada gambar 3.13 merupakan perancangan *user interface* halaman utama pada sistem JURAGAN. Pada sistem JURAGAN ini dimana terdapat dua halaman utama, yang membedakan dari kedua halaman tersebut adalah kondisi dari tiap – tiap *user*. Pada gambar 3.8 dimana merupakan halaman utama dengan kondisi *user* belum melakukan *login*. Halaman tersebut merupakan tampilan awal saat pertama kali *user* mengakses sistem JURAGAN sedangkan pada gambar 3.13 merupakan halaman utama dengan kondisi *user* sudah melakukan login dimana akases yang diberikan lebih banyak. Pada gambar 3.13 tersebut dimana halaman tersebut dibagi menjadi 5 bagian *template* yaitu *header* yang merupakan bagian paling atas sistem dan biasanya di isikan dengan judul atau *tittle*, lalu ada *sidebar* yang dimana letaknya tepat pada sisi sebelah kiri dan terdapat beberapa kolom dengan fungsi *navigasi link*. Pada halaman

*sidebar* tersebut terdapat *icon – icon* yang dimana dapat menambah estetika pada sistem JURAGAN tersebut. Terdapat bagian *topbar* yang letaknya berada pada bagian atas sistem JURAGAN dan di bawah *template header*. Pada bagian *topbar* tersebut dimana penulis memberikan *textbox* yang berfungsi sebagai kolom pencarian, lalu terdapat juga *icon* keranjang yang dimana berfungsi sebagai penghubung ke halaman keranjang belanja. Pada bagian sisi kanan *topbar* dimana terdapat foto profil dan nama *user* yang sedang *login*. Lalu terdapat *template main* yang merupakan bagian inti dari halaman tersebut. *Template main* tersebut akan menampilkan data – data pedagang yang terdaftar pada sistem JURAGAN dan terdapat juga *slider* yang berfungsi sebagai papan iklan dari sistem JURAGAN tersebut. Selanjutnya *template footer* yang dimana bagian terakhir dari *template* tersebut. *Footer* biasanya berada di bagian bawah dari sistem tersebut. Pada sistem JURAGAN akan diberikan tanda *copyright* dari sistem tersebut.



Gambar 3.14 Perancangan UI Halaman Data Profil

Pada gambar 3.14 merupakan perancangan *user interface* halaman data profil pada sistem JURAGAN. Halaman data profil dibuat untuk melihat informasi – informasi singkat dari *user* yang sedang *login*. Pada halaman data profil tersebut dimana bagian *template header*, *sidebar*, *topbar*, dan *footer* akan terlihat sama seperti gambar 3.13 yang membedakan hanya bagian *template main* nya saja. Pada *template main* halaman data profil sistem JURAGAN tersebut diberikan sebuah fungsi *card* dari *bootstrap* yang berfungsi untuk menampung data – data profil *user*. Pada *card* tersebut diberikan dua buah *button* yang dimana berfungsi untuk menuju ke halaman *edit profil* dan *ganti password*.

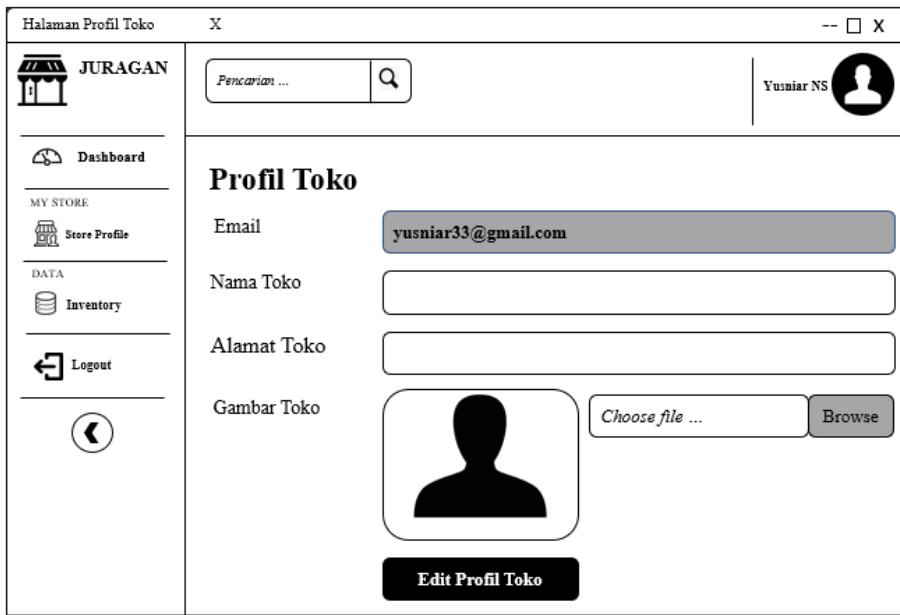


Gambar 3.15 Perancangan UI Halaman Edit Profil

Pada gambar 3.15 merupakan perancangan *user interface* halaman *edit profil* pada sistem JURAGAN. Pada bagian *template header*, *sidebar*, *topbar* dan *footer* akan terlihat sama seperti pada gambar 3.13. Dimana yang membedakan adalah *template* bagian *main*. Pada bagain *main* halaman *edit* profil sistem JURAGAN terdapat tiga buah *textbox* yang dimana satu *textbox* bersifat *readonly*. Apa itu *readonly* ?, *readonly* merupakan fungsi *fzze* pada sebuah *textbox* atau isi pada *textbox* tersebut tidak dapat dirubah dan hanya di tampilkan saja. Pada halaman *main edit* profil sistem JURAGAN tersebut juga dimana terdapat kinerja *upload* foto profil. Foto profil yang biasa di *upload* hanya dengan *file* dalm bentuk PNG, JPG, dan JPEG.

*Gambar 3.16 Perancangan UI Halaman Ganti Password*

Pada gambar 3.17 merupakan perancangan *user interface* halaman ganti *password* pada sistem JURAGAN. Halaman ganti *password* dibuat dengan tujuan untuk mengubah *password* akun dari *user* tersebut, namun dengan syarat *password* lama harus di *input* kan kembali. Halaman ganti *password* sistem JURAGAN dimana terbagi menjadi lima bagian *template* yang dimana *template header*, *sidebar*, *topbar* dan *footer* akan terlihat sama seperti pada gambar 3.13 yang membedakan hanyalah *template* bagian *main*. Pada bagian *main* dimana terdapat sebuah *form* guna melakukan pergantian *password* tersebut. Komponen – komponen yang digunakan berupa *textbox* sebagai penampung *input*-tan yang diberikan, *text* sebagai pemberian label atau keterangan dari setiap *textbox*, dan *button* yang berfungsi untuk memulai aksi ganti *password* tersebut.



Gambar 3.17 Perancangan UI Halaman Profil Toko

Pada gambar 3.17 merupakan perancangan *user interface* halaman profil toko pada sistem JURAGAN. Halaman profil toko dibuat untuk memberikan informasi – informasi dan melakukan pengeditan profil toko pada *user* dengan level penjual. Pada halaman profil toko tersebut dimana terbagi menjadi lima bagian *template* yaitu *header* yang dimana merupakan bagian paling atas dari suatu sistem tersebut dan biasanya berupa judul atau *tittle*, *sidebar* yang dimana letaknya berada pada sisi sebalah kiri dari sistem JURAGAN dan *icon* yang ditampilkan nampak lebih sedikit dari pada gambar 3.13 yang merupakan halaman utama *user* level pelanggan, *topbar* yang letaknya berada pada bagian atas sistem JURAGAN dan dibawah *template header* yang dimana hanya menyediakan *form* pencarian dan juga tampilan *foto profil* dan nama dari *user* level penjual yang *login*, bagian *footer* yang nampak sama dengan halaman – halaman sebelumnya, lalu bagian *main* yang dimana terdapat

beberapa komponen – komponen antara lain *textbox*, *text*, dan juga *button*. Pada bagian tersebut dimana *user* level penjual tersebut juga dapat melakukan proses *upload* foto profil.

No.	Nama Barang	Keterangan	Stok	Harga	Aksi
1	Laptop Asus X 555 U	Intel core i5 - 6200	10	Rp. 6.400.000	
2	Samsung Galaxy S10	8 GB / 128 GB	20	Rp. 10.000.000	
3	Canon 80D Kit EF-S	24,2 MP APS-C	15	Rp. 17.890.000	

Gambar 3.18 Perancangan UI Halaman Produk Toko

Pada gambar 3.18 merupakan perancangan *user interface* halaman produk toko pada sistem JURAGAN. Halaman produk toko tersebut dibuat untuk menampilkan produk – produk yang sudah di *input* kan oleh *user* tersebut dan hanya dapat di akses oleh *user* dengan level penjual. Pada halaman tersebut dimana terbagi menjadi lima bagian *template* yaitu *header*, *sidebar*, *topbar*, *footer* yang terlihat sama seperti pada gambar 3.17 namun bagian *main* yang berbeda. Pada bagian *main* tersebut akan ditampilkan data – data produk yang di berikan fungsi *table* pada *bootstrap* agar terlihat lebih rapih. Selain *table* dimana terdapat juga *button* di bagian atas *table* tersebut yang berfungsi untuk menampilkan halaman *input* produk dengan menggunakan fungsi modal pada *bootstrap*.

FORM INPUT PRODUK	
Email	<input type="text" value="yusniar33@gmail.com"/>
Nama Produk	<input type="text"/>
Detail Produk	<input type="text"/>
Kategori	<input type="text"/>
Harga	<input type="text"/>
Stok	<input type="text"/>
Gambar Produk	<input type="text"/>
<b>Kembali</b> <b>Simpan Produk</b>	

*Gambar 3.19 Perancangan UI Halaman Input Produk*

Pada gambar 3.19 merupakan perancangan *user interface* halaman *input* produk pada sistem JURAGAN. Halaman *input* produk tersebut dibuat untuk melakukan kegiatan pengimputan produk – produk bagi *user* dengan level penjual. Dimana *form* tersebut menggunakan fungsi *modal* pada *bootstrap*. Apa itu fungsi *modal* ?, fungsi *modal* tersebut hampir mirip dengan sebuah notifikasi yang dimana apabila diklik pada suatu *button* yang sudah di aplikasikan maka *form* tersebut akan muncul tanpa berpindah ke halaman lain. Pada halaman *input* produk tersebut dimana menggunakan beberapa komponen – komponen seperti *textbox* yang

dimana salah satunya menggunakan fungsi *readonly*, *text* yang dimana berfungsi sebagai pemberian label atau keterangan dari tiap – tiap *texbox*, dan *button* yang berfungsi sebagai pemberian aksi terhadap halaman tersebut.

NO	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Laptop	1	Rp. 6.4000.000	Rp. 6.4000.000	
2	Sepatu	1	Rp. 500.000	Rp. 500.000	
					Rp. 6.900.000

**Bersihkan Keranjang** **Lanjut Belanja** **Pembayaran**

Gambar 3.20 Perancangan UI Halaman Keranjang Belanja

Pada gambar 3.20 merupakan perancangan *user interface* halaman keranjang belanja pada sistem JURAGAN. Halaman keranjang belanja tersebut dibuat dengan tujuan untuk menampung produk – produk yang akan dibeli oleh *user* dengan level pelanggan dan hanya dapat diakses oleh *user* dengan level pelanggan. Pada halaman keranjang belanja dimana terbagi menjadi lima bagian *template* yang dimana pada bagian *header*, *sidebar*, *topbar*, dan *footer* terlihat sama seperti pada gambar 3.13 namun bagian *main* yang berbeda. Pada bagian *main* tersebut dimana akan

diberikan *form* dengan bentuk *table* agar produk – produk dapat tersusun dengan rapih. Komponen – komponen yang digunakan berupa *button* yang berfungsi untuk memberikan aksi pada proses transaksi. Pada halaman tersebut *user* dapat melakukan pengelolaan terhadap produk – produk yang akan dibeli oleh *user*.

## **BAB IV**

## **SOFTWARE PENDUKUNG**

Sebelum teman – teman membuat suatu sistem dimana diperlukan *software – software* pendukung untuk membantu proses pengerjaan teman – teman dalam melakukan pemrograman. Apa saja sih yang perlu dipersiapkan ? bagaimana sih cara *download* nya ? yuk kita bahas pada sub bab berikut.

### **3.1 Xampp**

*Xampp* merupakan sebuah *software* yang memiliki fungsi sebagai server lokal dimana berguna sebagai membuat sebuah *website* yang sifatnya masih dikembangkan. *Xampp* bekerja tanpa menggunakan koneksi internet atau secara *offline* yang dimana layaknya *web hosting* namun tidak dapat diakses oleh banyak orang. Dikarenakan JURAGAN merupakan sebuah *website* yang sifatnya masih dikembangkan maka sebelum dilakukannya *web hosting* kita perlu mempersiapkannya terlebih dahulu. *Xampp* sangat berperan penting untuk membantu kinerja pengembangan JURAGAN. Bagaimana cara melakukan *install software Xampp* tersebut ?, ikuti langkah – langkah nya sebagai berikut :

1. Bagi teman – teman yang belum mempunyai *software Xampp* tersebut silahkan *download* telebih dahulu dengan cara mengunjungi *website* resminya. Teman – teman dapat mengunjungi *link* <https://www.apachefriends.org/index.html> yang merupakan *website* resmi dari *Xampp*.



Gambar 4.1 Website Resmi Xampp

2. Setelah terbuka coba perhatikan pada bagian *download*. Terdapat sebuah *text* “*Click here for other versions*”, silahkan klik *text* tersebut.



Gambar 4.2 Tahap Dua Dalam Mendownload Xampp

3. Teman – teman akan di bawa ke halaman pemilihan versi yang akan di *download*. Sebelum melakukan *download* perlu teman – teman ketahui dimana dalam *website* resmi *Xampp* merupakan veris terbaru dan sudah menggunakan PHP 7. Apabila teman – teman tidak terbiasa dengan PHP 7 dapat menggunakan PHP 5 dengan cara men *download software* *Xampp* dengan versi yang lebih rendah. Lantas apa sih perbedaan PHP 7 dan PHP 5 ?, dimana akan di bahas pada bagian akhir sub bab *Xampp*.

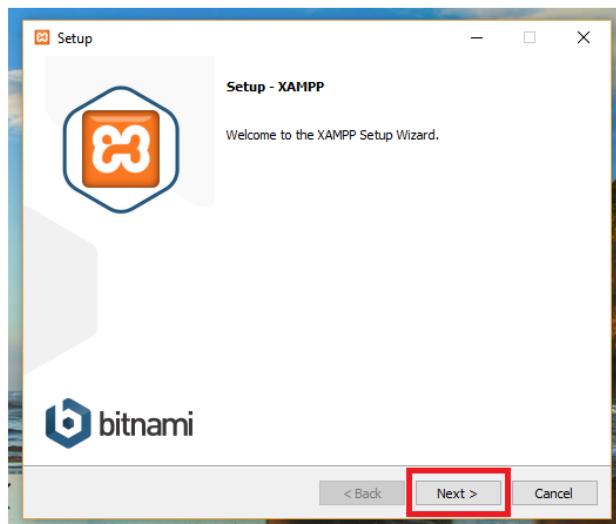
## Download

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

The screenshot shows the XAMPP for Windows download page. It features a table with three rows, each representing a different version of XAMPP. The columns are 'Version', 'Checksum', and 'Size'. The first row is for '7.2.26 / PHP 7.2.26' with checksums md5 and sha1, and a size of 145 Mb. The second row is for '7.3.13 / PHP 7.3.13' with the same checksums and size. The third row is for '7.4.1 / PHP 7.4.1' with the same checksums and size. Each row has a 'Download (64 bit)' button. Below the table, there are links for 'Requirements', 'Add-ons', and 'More Downloads'. A note states that Windows XP or 2003 are not supported and links to a compatible version. To the right, there are sections for 'Documentation/FAQs' (with a note about no real manual), 'Add-ons and Themes' (with icons for Bitnami tools like MySQL, PHP, Apache, and WordPress), and a note about Bitnami providing an all-in-one tool.

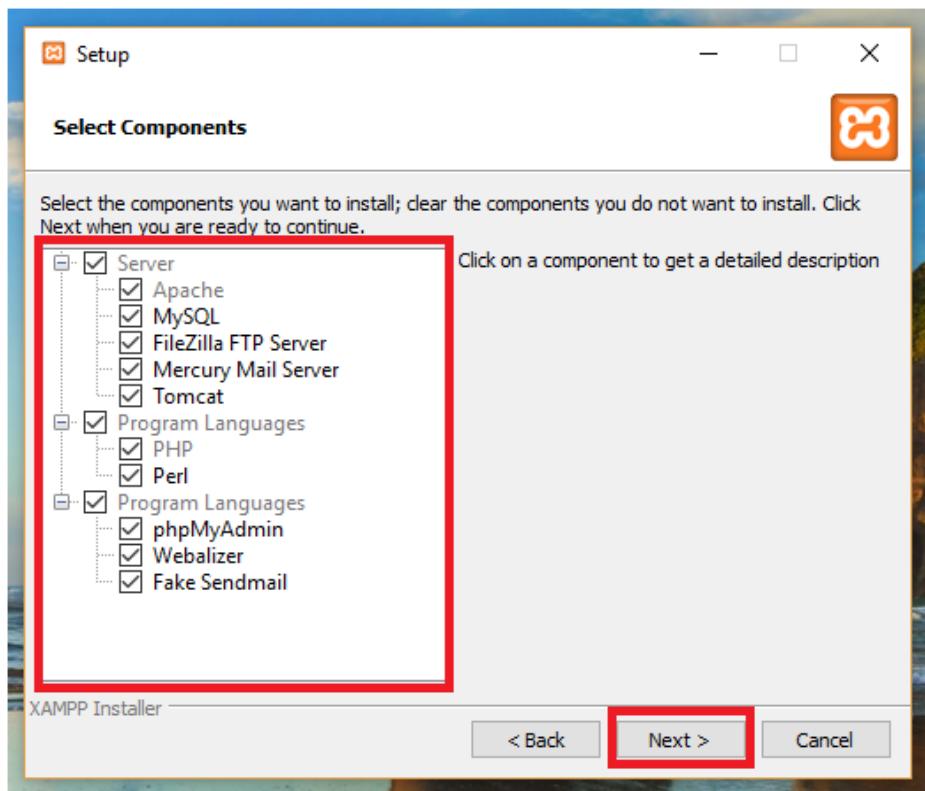
Gambar 4.3 Halaman Versi Xampp

4. Silahkan teman – teman *download software Xampp* tersebut. Tunggu beberapa menit hingga proses *download* selesai.
5. Apabila proses *download* telah selesai, silahkan teman – teman buka *file* tersebut, jalankan dengan *run administrator*. Akan muncul sebuah jendela baru yang menandakan proses *instalasi* akan dimulai, pilih *next*.



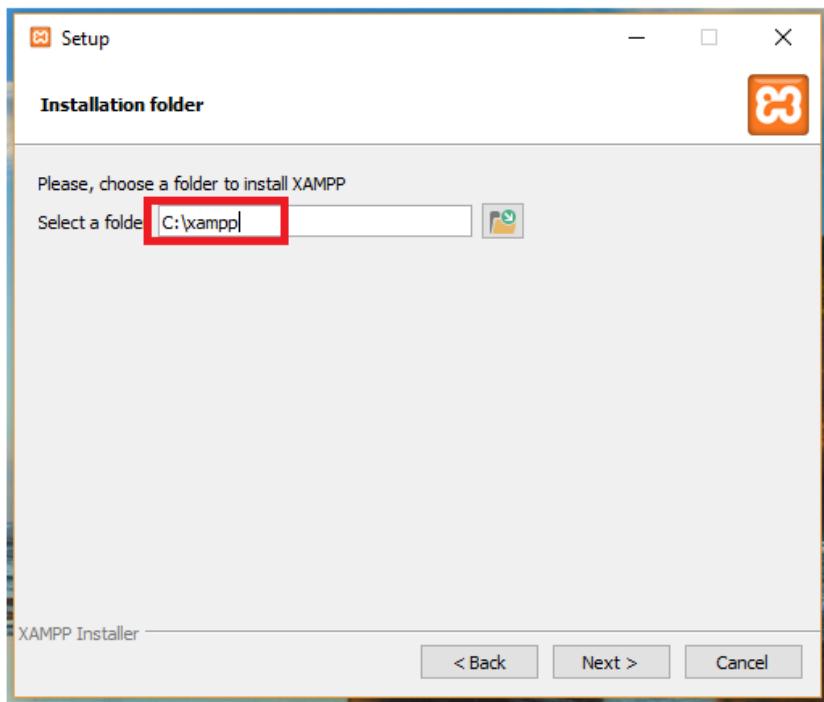
Gambar 4.4 Proses Awal Installasi Xampp

6. Pada tahap berikutnya dimana teman – teman akan diminta untuk memilih komponen – komponen yang akan digunakan. Perhatikan gambar 4.5, pilih *next* untuk melanjutkan proses *installasi software Xampp* tersebut.



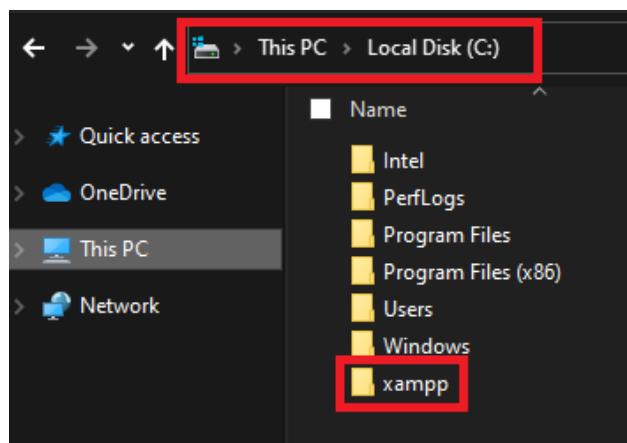
Gambar 4.5 Pemilihan Komponen - Komponen Pada Software Xampp

7. Pada tahap berikutnya dimana teman – teman akan diminta untuk memilih dimana *software xampp* tersebut disimpan. Saya sarankan simpan pada direktori C:, pilih *next* untuk melanjutkannya.



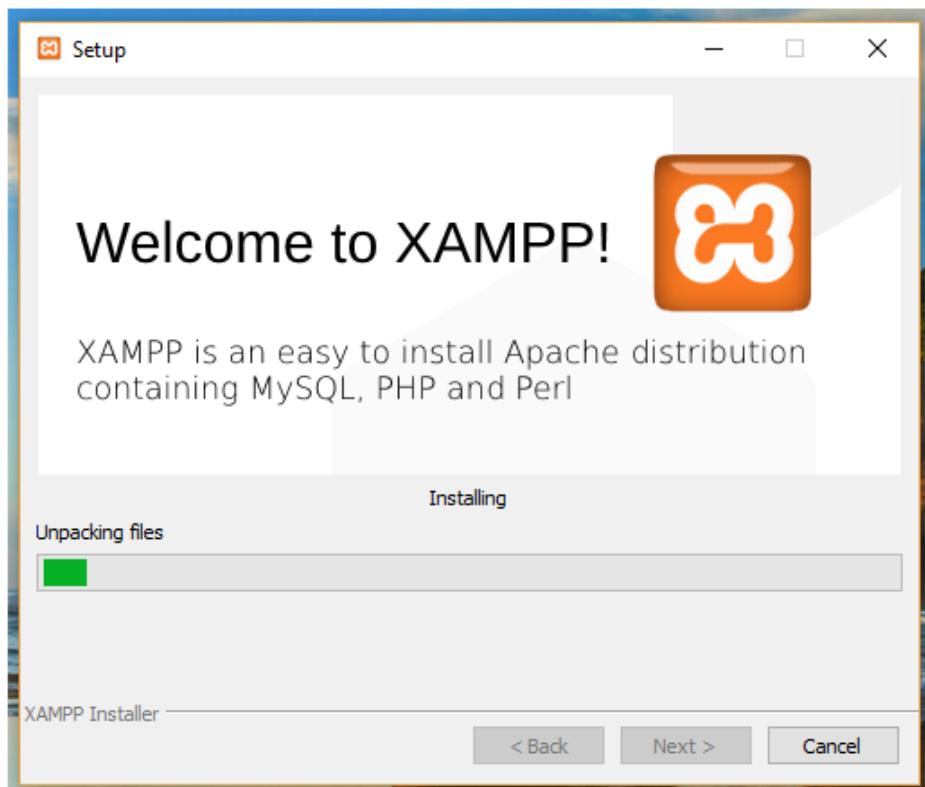
Gambar 4.6 Pemilihan Tempat Penyimpanan Software Xampp

8. Untuk lebih jelasnya perhatikan gambar 4.7, dimana *software Xampp* saya disimpan pada direktori C.



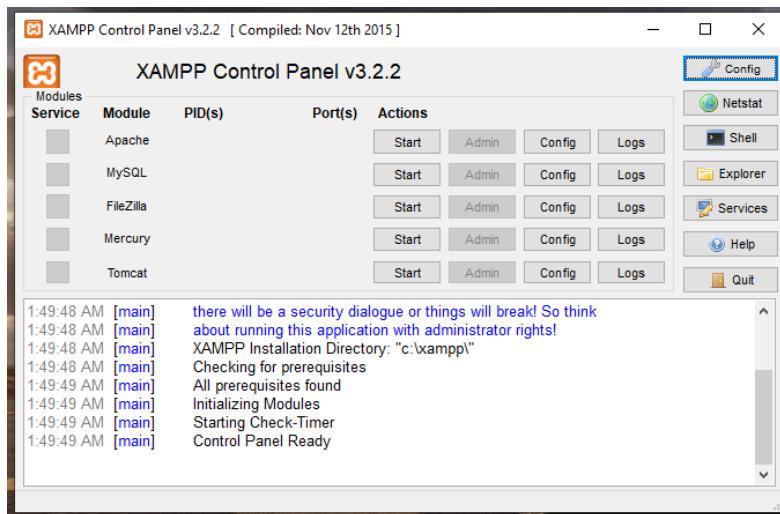
Gambar 4.7 Tempat Penyimpanan Xampp

9. Saat teman – teman memilih *next* pada tahap 7 maka akan muncul progres bar yang menandakan proses *installasi* sedang berjalan, tunggu beberapa menit hingga selesai.



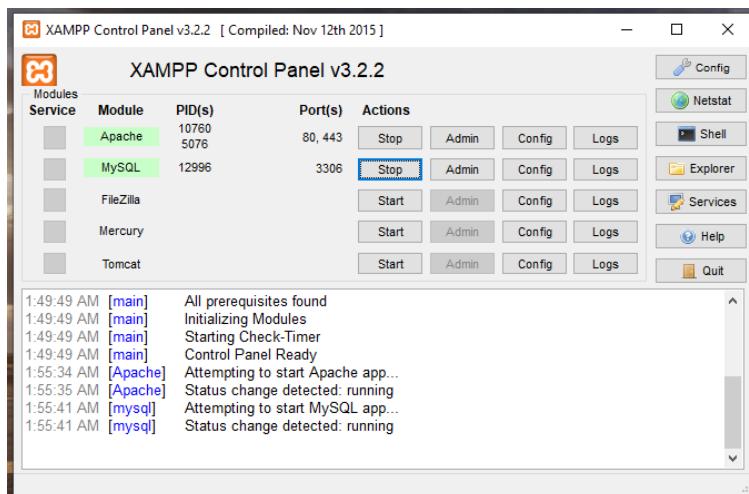
*Gambar 4.8 Proses Installasi Xampp Sedang Berjalan*

10. Apabila progres bar sudah terisi penuh maka proses *installasi software Xampp* teman – teman sudah berhasil. Pilih *next* untuk melanjutkannya.
11. Buka *software Xampp* tersebut pada laptop/PC teman – teman. Maka akan telihat seperti pada gambar 4.9.



Gambar 4.9 Tampilan Software Xampp

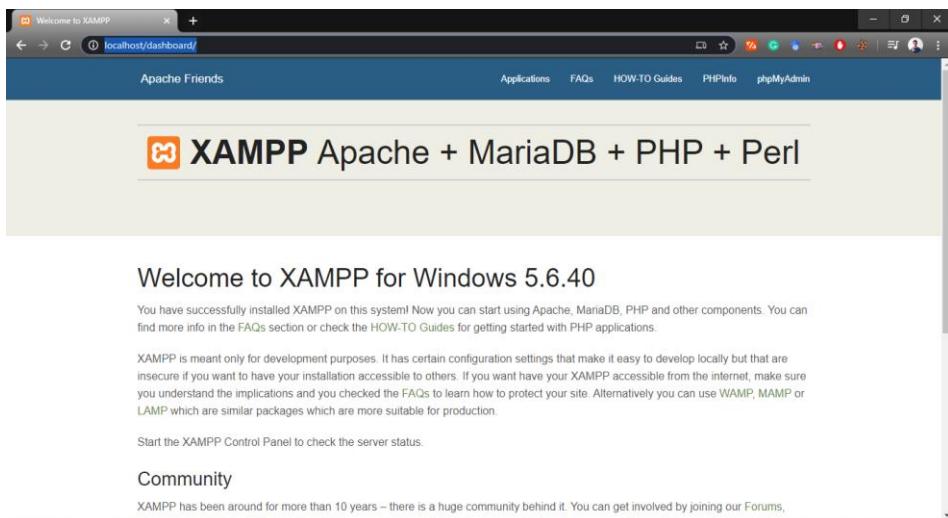
12. Ada beberapa hal yang perlu teman – teman perhatikan. Silahkan teman – teman jalankan *module Apache* dan *MySQL* dengan cara memberikan *actions start*. Apabila *module Apache* dan *MySQL* berubah warna menjadi hijau menandakan *software xampp* teman – teman sudah siap digunakan.



Gambar 4.10 Menjalankan Module Apache Dan MySQL

13. Silahkan teman – teman buka *web browser* yang sedang digunakan.

Coba teman – teman kunjungi halaman *localhost* dengan cara memasukkan *link* <http://localhost/dashboard/>. Apabila berhasil akan nampak seperti pada gambar 4.11. Perlu diingat *software xampp* dapat berjalan tanpa adanya koneksi internet atau bersifat *offline*.



*Gambar 4.11 Halaman Localhost*

#### **4.1.1 Perbedaan PHP 7 Dan PHP 5**

Pada dasarnya dimana PHP 5 adalah sebuah evolusi yang berjalan pada PHP. PHP 5 telah menawarkan peningkatan dari segi fungsionalitas dan penambahan fitur baru yang dimana yaitu seperti dukungan terhadap XML dan juga *Web Service* yang menggunakan libxml2, dukungan terhadap basis data SQLite serta membuat *file swf* dan *applet java*.

Sedangkan untuk PHP 7 memiliki PHPNG (*PHP-Next-Gen*) dimana berfungsi untuk memberikan performa yang maksimal.

Peningkatan performa pada PHP 7 ini dikarenakan sebuah *framework* Zend telah melakukan peningkatan kinerja yang sangat besar, dan dimana para developer dapat menggunakan patokan terhadap HHVM.

## 4.2 Visual Studio Code

*Programmer* merupakan sebuah pekerjaan yang dimana bertugas dalam menerapkan atau menulis *script code* kepada sistem yang akan dibuat atau dikembangkan. Dalam melakukan aktivitasnya dimana *programmer* memerlukan beberapa *tools* yang dapat mempermudah pekerjaannya, seperti sebuah *software* yang dapat menampung penulisan *script codenya*. Ada banyak sekali *tools* untuk membantu *programmer* dalam mengetikkan *script code*-nya seperti *notepad++*, *sublime*, *visual studio code*, dan masih banyak lagi.

Pada pembahasan kali ini dimana penulis menggunakan *visual studio code* atau *Vscode* untuk membangun sistem JURAGAN. *Vscode* merupakan salah satu *text editor* yang paling populer digunakan oleh para *programmer*. Hal ini dikarenakan *Vscode* memiliki beberapa fitur yang dapat mempermudah *programmer* dalam melakukan pengkodean. Salah satunya adalah fitur yang dimana dapat melakukan *copy paste code* secara instan dengan cara menekan tombol kombinasi “*CTRL + SHIFT + DOWN*” untuk meng *copy paster code* ke arah bawah dan “*CTRL + SHIFT + UP*” untuk ke arah atas.

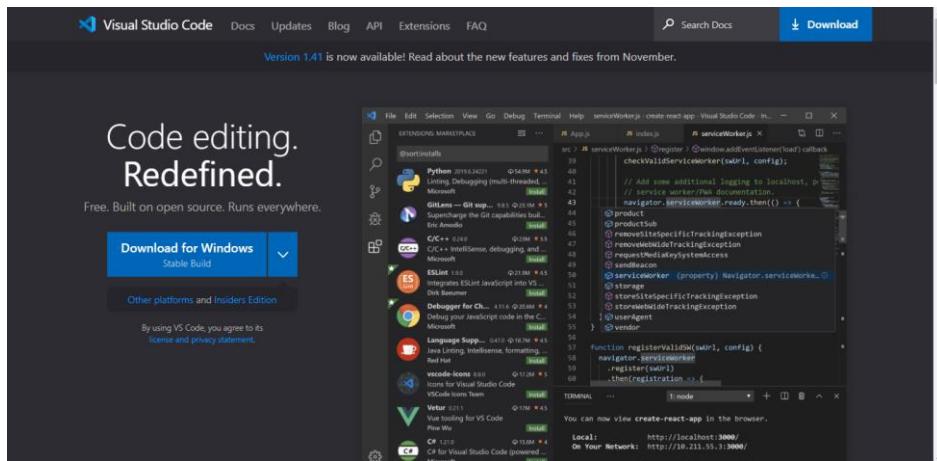
Sebelum melakukan penginstallan *software Vscode* dimana kita harus mengetahui spesifikasi atau *requirements* yang dibutuhkan. Spesifikasi yang dibutuhkan dapat dilihat pada tabel 4.1.

*Tabel 4.1 Spesifikasi Software VS Code*

<b><i>Hardware</i></b>	<b><i>Operation System</i></b>	<b><i>Sarat</i></b>
<i>Processor 1.6 GHz 1 GB RAM</i>	<i>Windows 7, 8.0, 8.1, 10</i>	<i>32/64 Bit</i>
	<i>Linux Debian: Ubuntu 14.04, Debian 7</i>	<ul style="list-style-type: none"> <li>• GLIBCXX <i>Version 4.4.15 Or later</i></li> </ul>
	<i>Linux Red Hat: Enterprise Linux 7, CentOS 7, Fedora 23</i>	<ul style="list-style-type: none"> <li>• GLIBC Version <i>2.15 Or later</i></li> </ul>

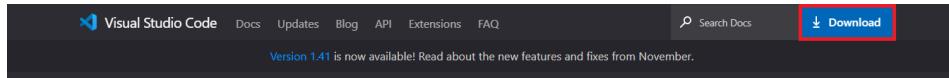
Apakah teman – teman tertarik ingin menggunakan *software visual studio code* ? Bagaimana sih cara melakukan *installasi* terhadap *software visual studio code* ? Pada pembahasan kali ini dimana teman – teman akan belajar bagaimana cara melakukan *installasi* terhadap *software visual studio code* tersebut. Yuk ikuti langkah – langkah berikut ini :

1. Dimana teman – teman perlu mengunjungi *website* resmi dari *visual studio code* untuk melakukan proses *download*. Silahkan kunjungi *link* berikut, <https://code.visualstudio.com/>.



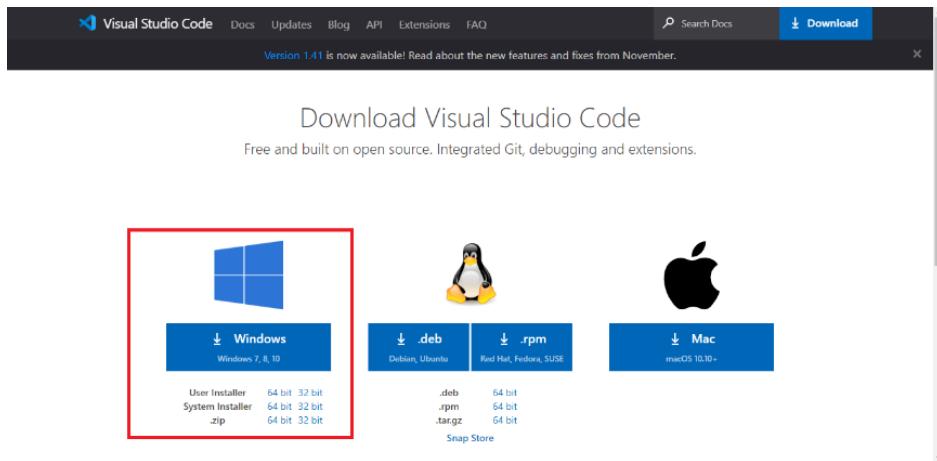
Gambar 4.12 Halaman Website Resmi VSCode

2. Perhatikan bagian sisi kanan pada *website* tersebut dan lihat pada bagian paling atas terdapat tulisan *download*, silahkan teman – teman klik.



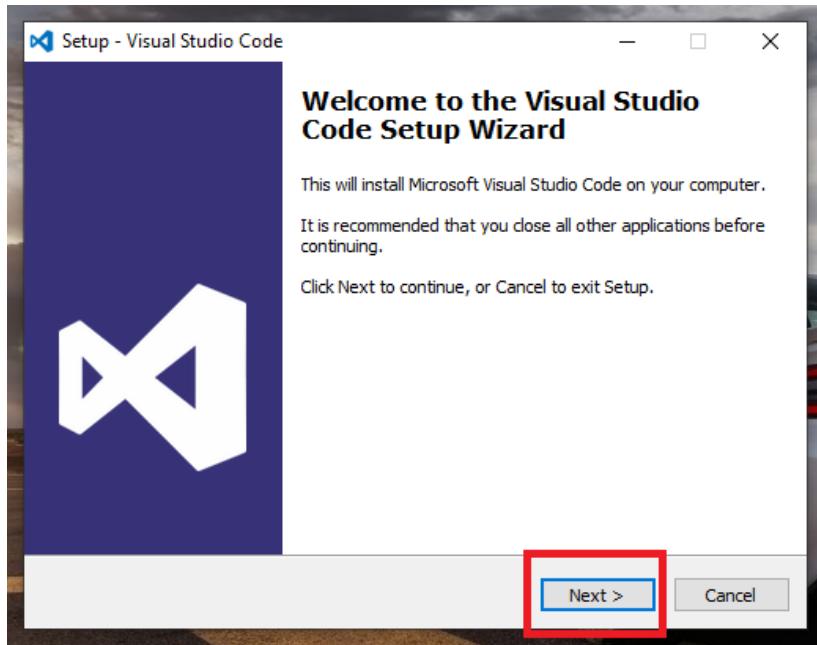
Gambar 4.13 Memilih Halaman Download VSCode

3. Dimana teman – teman akan dibawa kehalman *donwload*, silihkan *download* sesuai sistem operasi yang sedang teman – teman gunakan. Karena saya menggunakan *windows* jadi saya akan memilih *windows*. Sistem akan melakukan proses *download*, tunggu beberapa menit hingga proses *download* tersebut selasi.



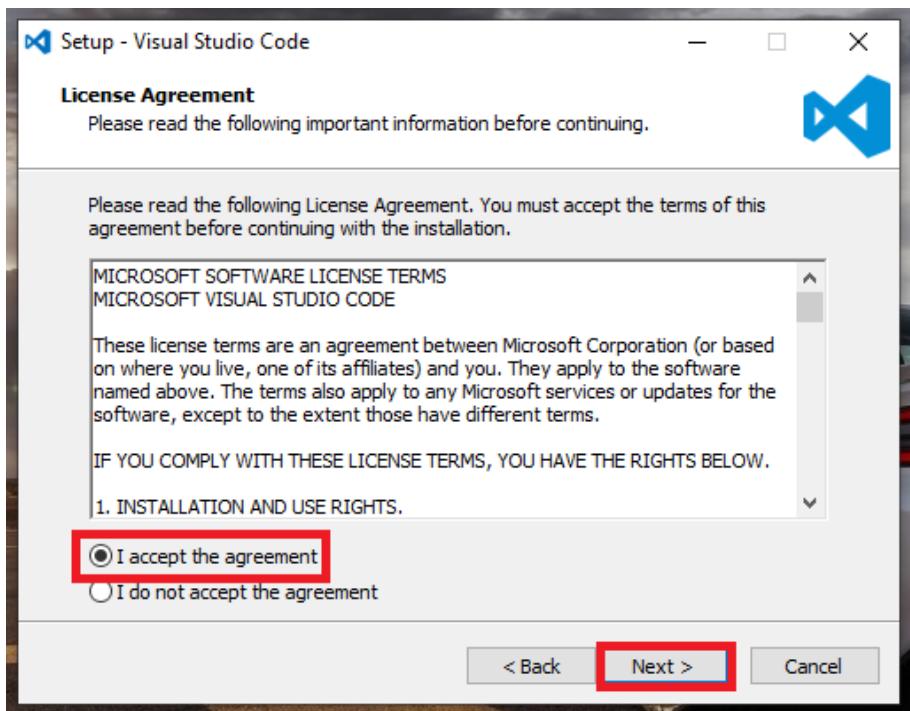
Gambar 4.14 Pemilihan OS Saat Download VSCode

4. Apa proses *download* telah selesai, silahkan teman – teman jalankan *file* tersebut dengan cara *run administrator*. Akan terbuka halaman baru, pilih *next* untuk melanjutkannya.



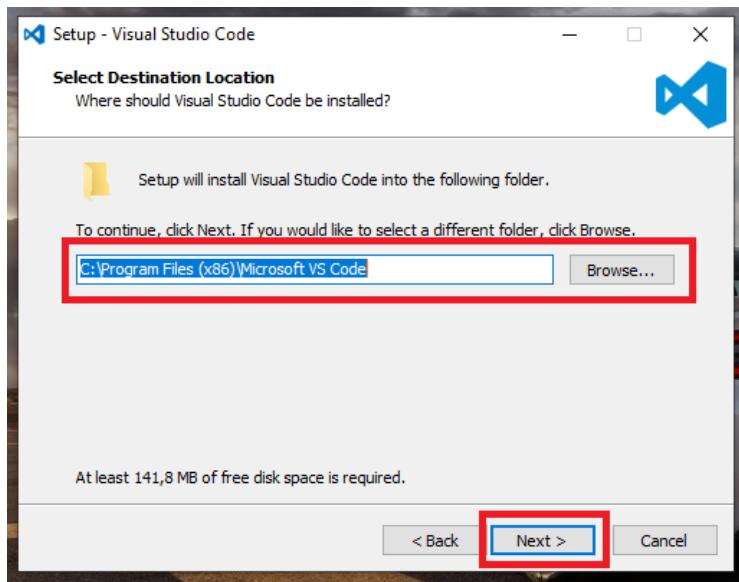
Gambar 4.15 Proses Awal Installasi VSCode

5. Silahkan teman – teman pilih “*I accept the agreement*”, lalu pilih *next* untuk melanjutkan ke tahap berikutnya.



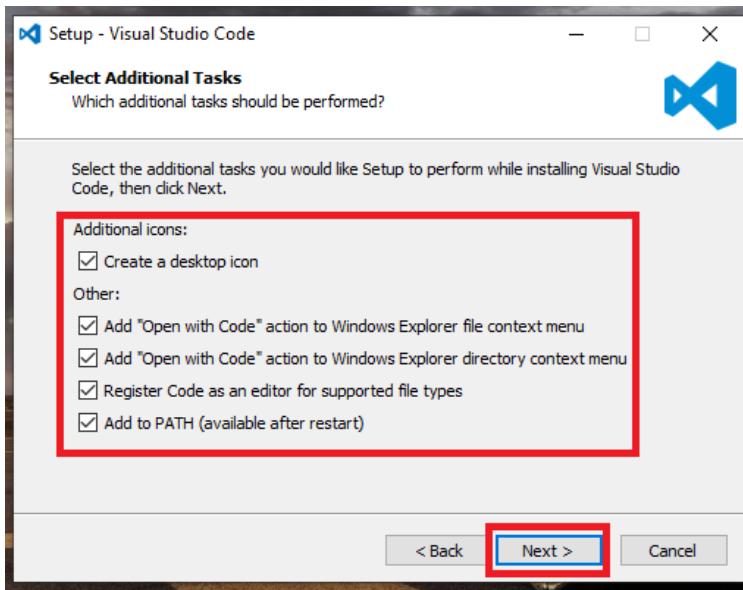
Gambar 4.16 Proses Installasi VSCode Tahap Dua

6. Pada tahap berikutnya dimana teman – teman akan diperintahkan memilih tempat penyimpanan *software Visual Studio Code* tersebut. Silahkan teman – teman pilih dimanapun untuk menyimpan *file* tersebut, lalu pilih *next* untuk melanjutkan ketahap berikutnya.



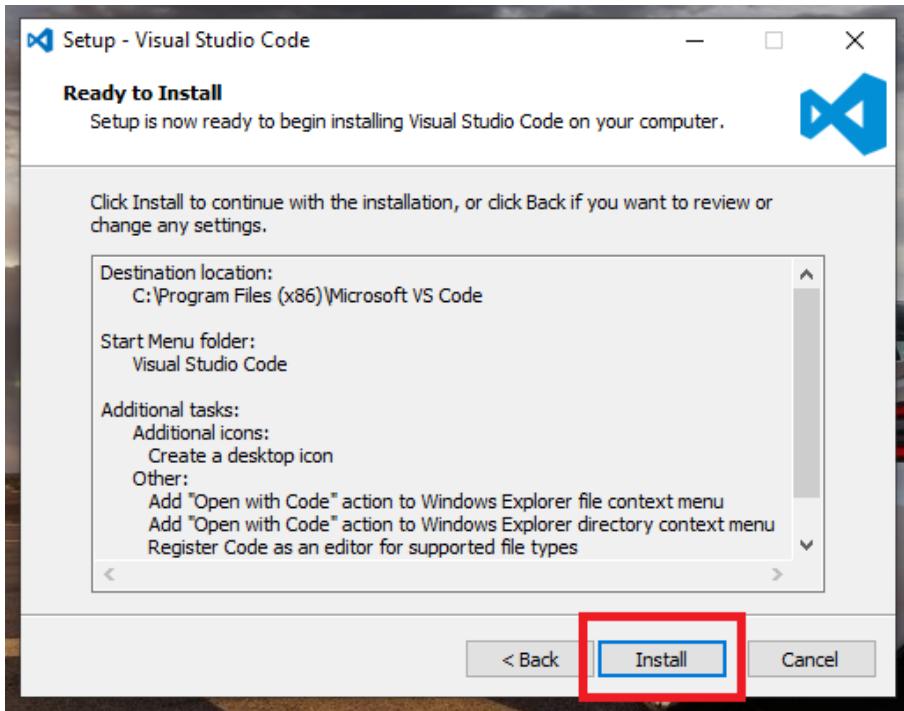
Gambar 4.17 Pemilihan Tempat Penyimpanan VSCode

7. Pada tahap berikutnya silahkan teman – teman centang semua pilihannya. Pilih *next* untuk melanjutkan ke tahap berikutnya.



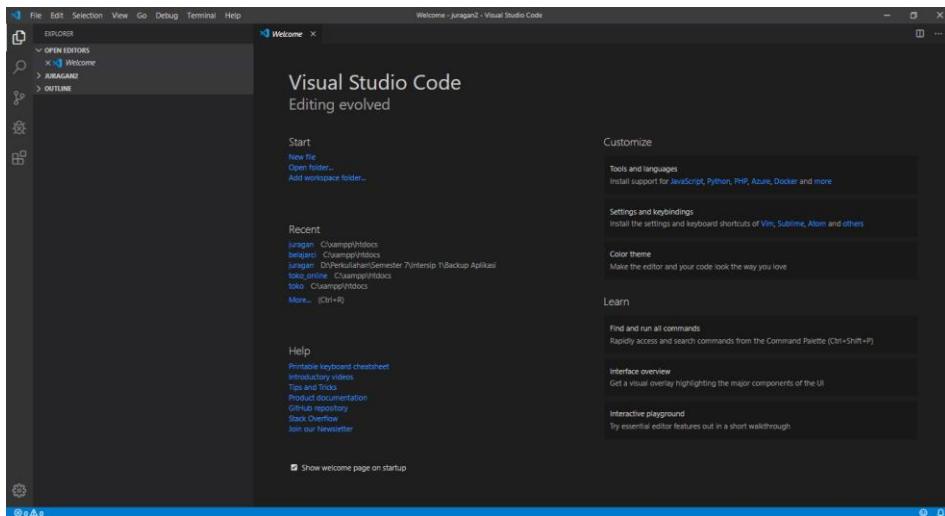
Gambar 4.18 Pemilihan Additional Tasks VSCode

8. Pada tahap berikutnya dimana teman – teman akan diperintahkan melakukan proses *install*. Silahkan teman – teman pilih *install*, dimana akan muncul progres bar yang menunjukkan proses *installasi* sedang berjalan. Tunggu beberapa menit hingga *progress bar* penuh.



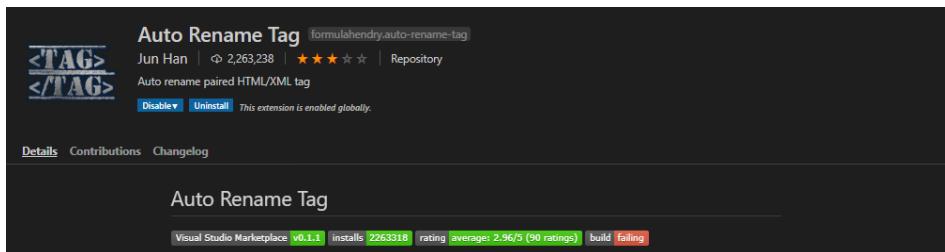
Gambar 4.19 Proses Installasi VSCode

9. Apabila *progress bar* sudah terisi penuh menandakan proses *installasi software Visual Studio Code* teman – teman sudah selesai. Silahkan teman – teman buka *software Visual Studio Code* maka tampilan awalnya akan terlihat seperti pada gambar 4.20.



Gambar 4.20 Tampilan Visual Studio Code

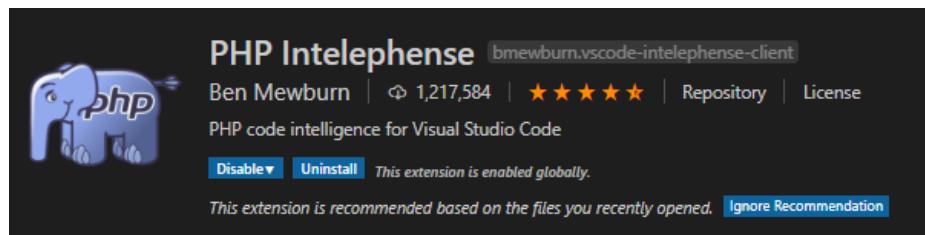
Apabila proses *install software Vscode* sudah selesai dimana teman - teman perlu melakukan *install* beberapa *extensions* untuk membantu proses pengkodean. Apa saja *extensions* tersebut ?, yang perlu teman - teman *install* adalah “*Auto Rename Tag*”.



Gambar 4.21 Extension Auto Rename Tag

*Extensions* tersebut berfungsi untuk merapihkan *script code* kalian saat melakukan *save*. Ketika teman - teman melakukan *save* dengan cara menekan tombol kombinasi “*CTRL + S*” dimana *script code* yang telah kalian ketik akan otomatis dirapihkan. Hal tersebut sangat bermanfaat karena *script code* yang teman - teman ketikan akan terlihat lebih rapih.

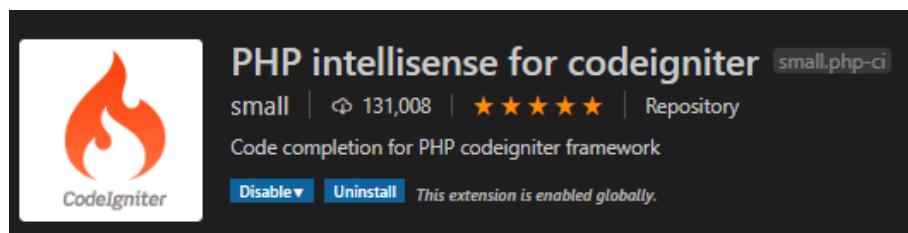
*Extensions* yang kedua dimana teman - teman perlu meng-*install* “*PHP Intelephense*”.



Gambar 4.22 Extension *PHP Intelephense*

*Extensions* tersebut dimana memiliki fungsi sebagai memberikan fitur lengkap terhadap *PHP* dengan saran yang sangat detail dan dukungan ”*Go To*” langsung kepada sumbernya. Dengan adanya *extensions* tersebut dimana teman - teman tidak perlu mengingat semua *sintaks* perintah yang akan teman - teman ketik, hal ini dikarenakan fitur *intelephense* dapat bekerja tanpa harus melakukan konfigurasi.

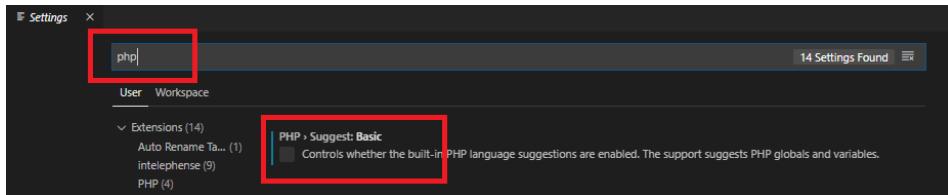
*Ektensions* yang terakhir dimana teman - teman perlu melakuakan *install* “*PHP Instellisense for codeigniter*”.



Gambar 4.23 Extension *PHP Intellisese For CI*

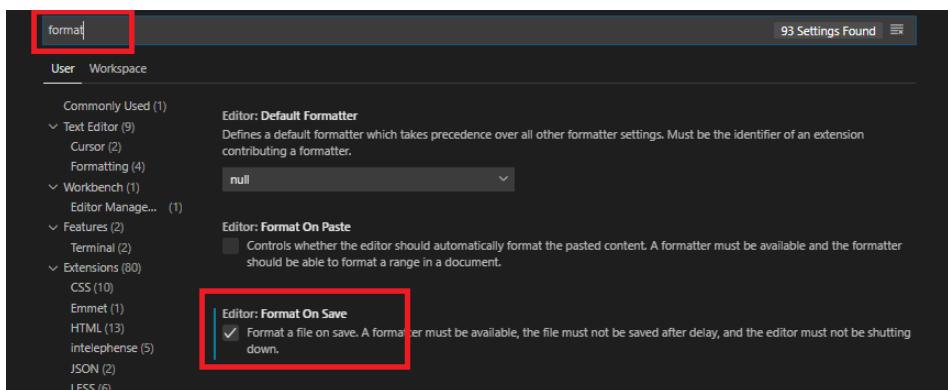
*Extensions* tersebut memiliki fungsi untuk memudahkan teman - teman dalam melakukan *script code* pemanggilan *PHP* yang berada pada *codeigniter*. Untuk menggunakan semua *extensions* yang telah teman - teman *install* dimana perlu dilakukan *setting* kembali pada *software*

Vscode. Pilih *preferences* → *Setting*, maka akan terbuka seperti pada gambar 3.24. Ketikkan "PHP" pada kolom pencarian, lalu hilangkan centang pada bagian "*PHP Suggest Basic*" agar proses *extensions PHP Intelephense* yang akan berjalan.



Gambar 4.24 Setting Awal VSCode

Langkah yang kedua silahkan ketik "format" pada kolom pencarian, berikan centang pada bagian "*format on save*" agar saat teman - teman melakukan *save script code*-nya akan otomatis dirapihkan.



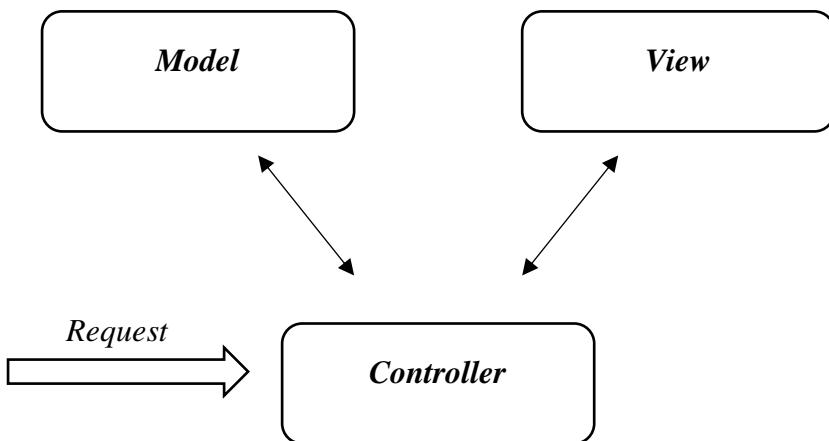
Gambar 4.25 Setting Kedua VSCode

### 4.3 Codeigniter

*Codeigniter* merupakan *application development framework* untuk membangun aplikasi menggunakan PHP [13]. Tujuan dari *codeigniter* tersebut yaitu untuk memungkinkan mengembangkan proyek lebih cepat daripada menulis kode dari awal, dengan menyediakan serangkaian *library*

yang biasanya dibutuhkan oleh para pengembang, antarmuka yang sederhana, dan struktur logis untuk mengakses *library* tersebut [13]. *Codeigniter* didasarkan pada pola pengembangan MVC [13]. MVC adalah pendekatan perangkat lunak yang memisahkan logika aplikasi dari presentasi [13]. Hal ini memungkinkan halaman web yang dibuat akan mengandung *source code* yang minim dikarenakan skrip PHP yang terpisah [13]. MVC akan melakukan pembagian aplikasi menjadi tiga bagian yang fungsional yaitu *model*, *view*, dan *controller* dimana memiliki pengertian sebagai berikut :

- ***Model*** merupakan tempat berkumpulnya *script code* untuk memulai proses bisnis dan data. Dimana *model* akan berhubungan langsung dengan basis data (*database*) untuk melaukan eksekusi sebuah *query insert, update, delete*. *Model* juga akan menangani proses validasi pada bagian *controller* akan tetapi *model* tidak dapat berhubungan langsung dengan bagian *view*.
- ***View*** adalah tempat untuk menangani logika presentasi yang didalamnya terdapat *script code* untuk membuat desain interaksi atau tampilan dari sistem yang akan dibuat.
- ***Controller*** adalah penghubung antara *model* dan *view* yang dimana akan menerima sebuah *request – request* dan data dari pengguna (*user*). Dimana *controller* akan menentukan apa yang akan di proses oleh sistem tersebut.



*Gambar 4.26 Konsep MVC*

#### **4.3.1 Installasi Framework Codeigniter**

Apakah teman – teman tertarik dan ingin belajar mengenai *framework codeigniter* ?. Apabila teman – teman ingin belajar mengenai *framework codeigniter*, silahkan teman – teman lakukan *installasi* terhadap *framework codeigniter* tersebut. Bagaimana sih cara melakukan *installasi framework codeigniter* ?, yuk simak *tutorial* berikut ini :

1. Untuk men-*download* *framwork codeigniter* tersebut silahkan teman – teman kunjungi *website* resminya. Silahkan kunjungi *link* berikut, <https://codeigniter.com/>.
2. Dimana teman – teman akan dibawa ke *website* resmi dari *framework codeigniter* tersebut.



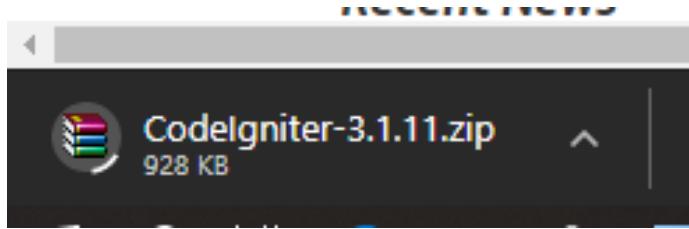
Gambar 4.27 Website Remi Codeigniter

3. Pada halaman websiter tersebut terdapat pilihan *download*, silahkan teman – teman klik.



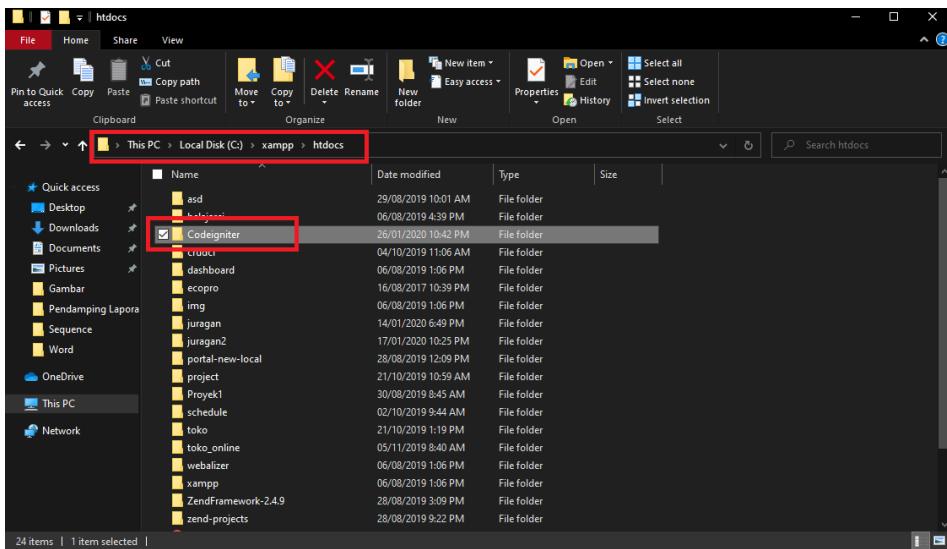
Gambar 4.28 Pilihan Download CI

4. Sistem akan melakukan proses *download*, tunggu beberapa menit hingga proses *download* selesai.



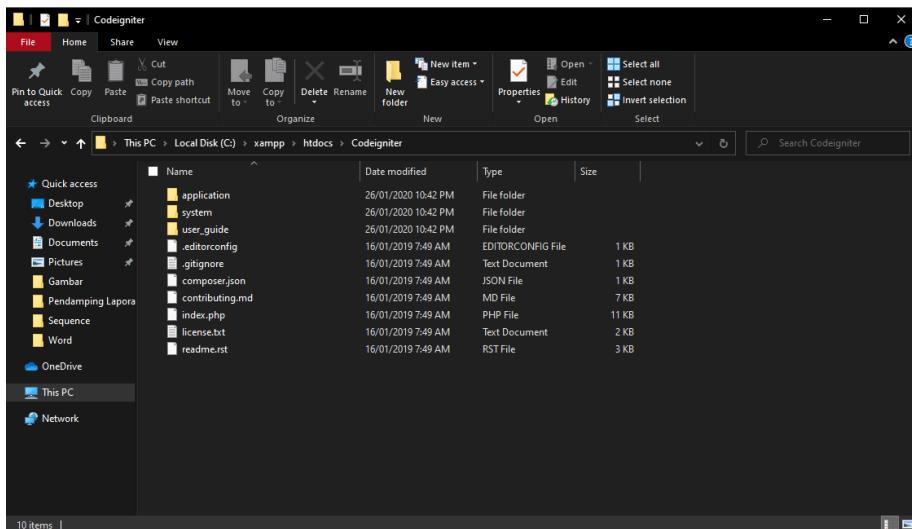
Gambar 4.29 Proses Download Codeigniter

5. Lakukan *extract* pada file zip yang sudah teman - teman *download* tersebut dan simpan pada folder “Xampp/htdocs/”. *Rename* saja namanya menjadi *Codeigniter*.



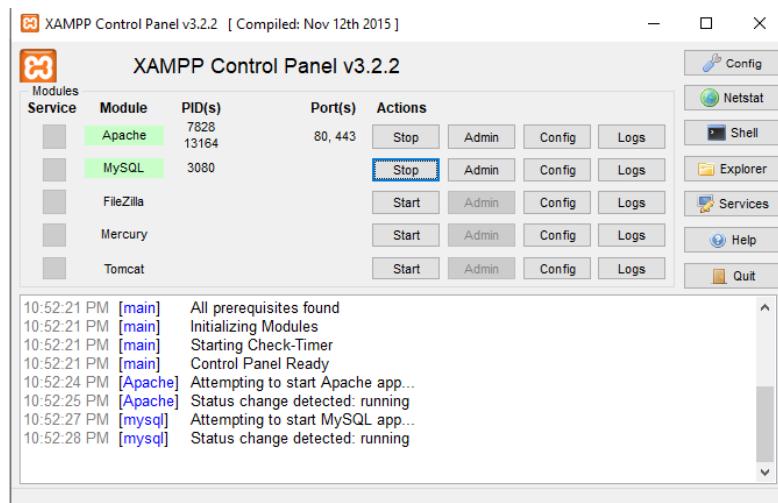
Gambar 4.30 Memindahkan File Codeigniter

6. Silahkan teman – teman cek folder *codeigniter* yang sudah di *download*, pastikan isi di dalamnya sama persis dengan gambar 4.31.



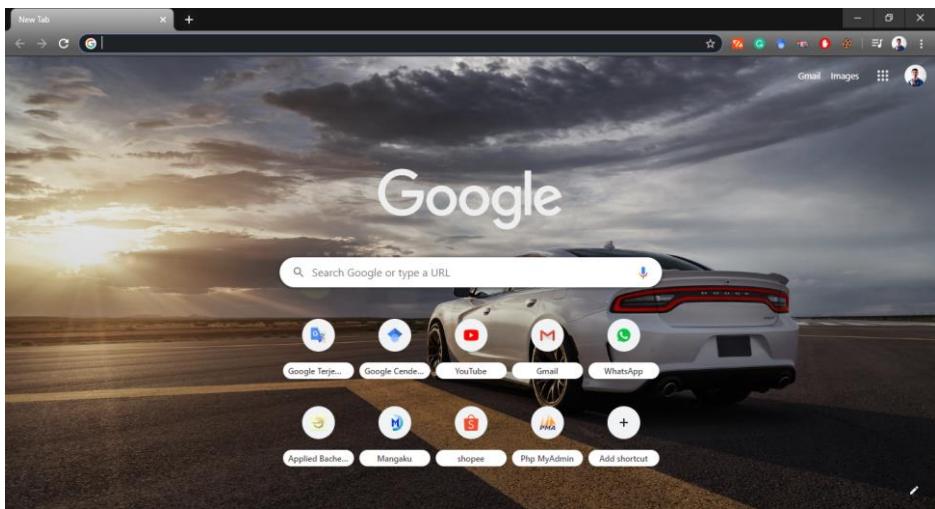
Gambar 4.31 Isi File Codeigniter

7. Silahkan teman – teman lakukan pengujian terhadap *framework codeigniter* tersebut. Dimana diperlukan *software Xampp* dalam melakukan pengujian *framework codeigniter* tersebut. Silahkan teman – teman jalankan *software xampp* tersebut.



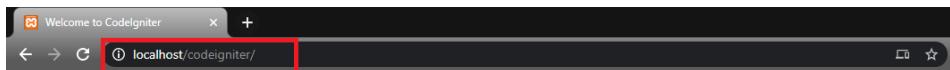
Gambar 4.32 Menjalankan Software Xampp

8. Silahkan teman – teman buka *web browser* yang sedang digunakan, disini saya menggunakan *web browser chrome*.



Gambar 4.33 Membuka Web Browser

9. Untuk melakukan pengujian dimana teman – teman perlu memasukkan nama *folder* tersebut, silahkan teman – teman ketik <http://localhost/codeigniter/> pada kolom pencarian.



Gambar 4.34 Menguji Framework Codeigniter

10. Apabila teman – teman berhasil dalam melakukan *installasi framewrok codeigniter*, maka akan muncul tampilan seperti pada gambar 4.35.



*Gambar 4.35 Pengujian CI Berhasil*

## **BAB V**

## **FRAMEWORK CODEIGNITER**

### **5.1 Sejarah Codeigniter**

Rick Ellis yang merupakan seorang musisi musik dengan genre rock yang dimana profesiannya beralih menjadi seorang *programmer* yang dimana pada sebuah riset kecil – kecilan telah menghasilkan sebuah *framework PHP* dengan ukuran yang cukup kecil dan ringan serta dapat memenuhi fitur umum pada aplikasi *PHP*. Rick Ellis dialah seseorang yang pertama kali menuliskan *codeigniter*. Pada tanggal 28 Februari 2016 dimana *codeigniter* pertama kali dirilis, akan tetapi pada tahun 2014 dimana *framework* tersebut telah dimiliki oleh BCIT (*British Columbia Institute Of Technology*).

### **5.2 Kelebihan Codeigniter**

Mengapa *codeigniter* ini termasuk salah satu *framework* yang di favoritkan para *programmer* ?, ternyata *codeigniter* memiliki kelebihan sendiri loh dibandingkan dengan *framework* yang lain. Apa saja sih kelebihan *framework codeigniter* ?, berikut ini merupakan kelebihan yang dimiliki oleh *fremework codeigniter* adalah sebagai berikut :

- *Codeigniter* merupakan salah satu *framework* yang memiliki performa paling cepat dibandingkan dengan *framework* – *framework* yang lain.
- *Codeigniter* merupakan salah satu *framework* dengan konfigurasi yang minim sehingga mudah untuk dipahami oleh seseorang yang baru belajar pemrogramman. Akan tetapi *codeigniter* mengizinkan untuk melakukan konfigurasi yang dimana

tujuannya untuk menyesuaikan dengan *database* dan kebebasan *routing*.

- *Codeigniter* memiliki dokumentasi yang sangat lengkap. Teman – taman dapat membukanya di website resmi *codeigniter* untuk melihat dokumentasi yang dimiliki oleh *codeigniter*.

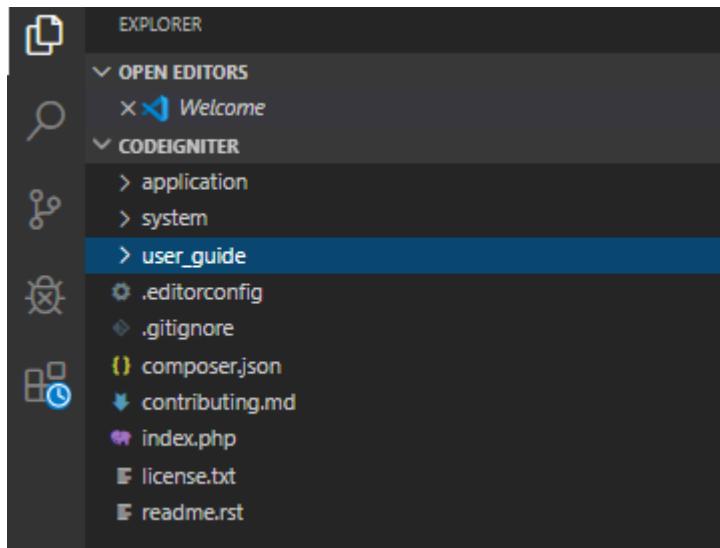


Gambar 5.1 Dokumentasi Codeigniter

- *Codeigniter* sangat cocok dan mudah dipelajari bagi pemula, hal ini dikarenakan *framework* tersebut tidak telalu bergantung pada *tool* tambahan seperti *ORM*, *composer* dan lainnya.
- *Codeigniter* memiliki banyak sekali komunitas yang tersebar di Indonesia sehingga mudah untuk mencari referensi atau *tutorial pemrograman*.

### 5.3 Struktur Direktori Codeigniter

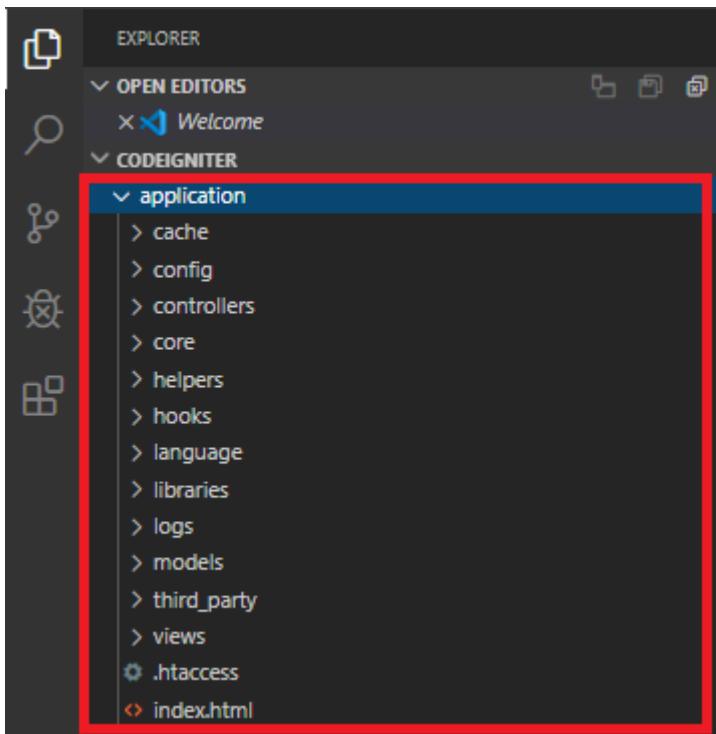
Sebelum teman – teman menggunakan *framewrok codeigniter* tersebut alangkah baiknya apabila teman – teman belajar dan memahami terlebih dahulu struktur direktori dari *framwork codeigniter* tersebut. Pada *framework Codeigniter* dimana struktur yang dimilikinya terlihat seperti pada gambar 5.2.



Gambar 5.2 Struktur Direktori CI

Dalam melakukan kegiatan pemrograman dimana teman – teman akan melakukannya pada direktori *application*. Pada direktori tersebut teman – teman dapat melakukan konfigurasi serta membuat tampilan untuk *website* yang akan dibangun. Untuk memahami lebih lanjut mengenai direktori *application* teman – teman dapat simak penjelasan berikut ini.

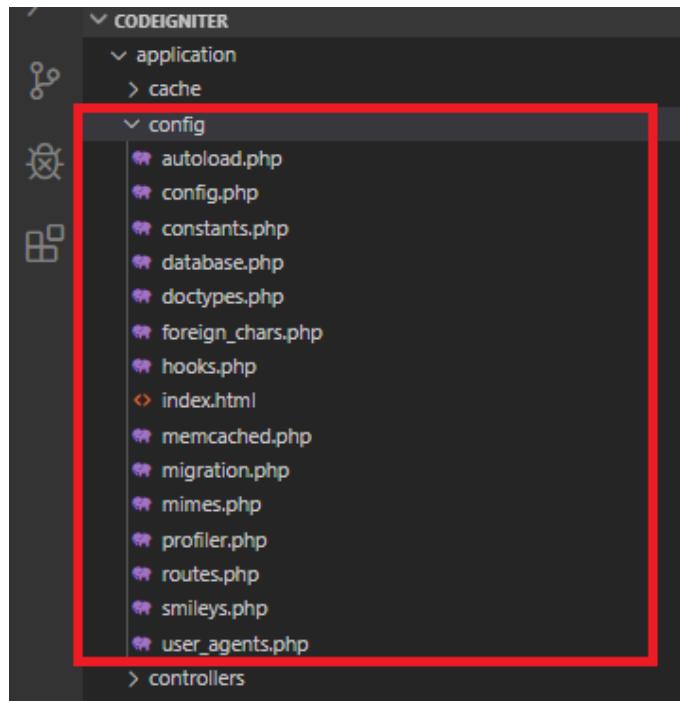
Direktori *application* merupakan direktori yang akan kita gunakan dalam pembuatan aplikasi kita nanti. Coba teman - teman klik pada direktori *application* tersebut dan lihat isinya, akan terlihat seperti pada gambar 5.3.



Gambar 5.3 Isi Dari Direktori Application

Terdapat banyak sekali *folder – folder* yang berada pada direktori *application* tersebut, mari kita perjelas satu – persatu berdasarkan penjelasan berikut :

- **Cache** yang dimana isinya merupakan *cache* dari aplikasi itu sendiri.
- **Config** yang dimana berfungsi sebagai tempat untuk melakukan konfigurasi aplikasi yang akan kita buat. Dimana *config* memiliki *file – file* tersendiri.



Gambar 5.4 File Dalam Folder Config

- **Controller** yang dimana untuk memberikan *control* terhadap aplikasi yang akan kita buat.

```
file: Welcome.php
application > controllers > Welcome.php > PHP Intelephense > Welcome
1  <?php
2  defined('BASEPATH') OR exit('No direct script access allowed');
3
4  class Welcome extends CI_Controller {
5
6      public function index()
7      {
8          $this->load->view('welcome_message');
9      }
10 }
11
```

Gambar 5.5 Contoh Script Code Controller

- *Core* yang dimana berfungsi untuk melakukan *custome core*.
- *Helpers* yang dimana berfungsi untuk menampung *script code* fungsi *helpers*. *Helpers* dapat membantu dalam melakukan pengembangan aplikasi menjadi lebih cepat dan juga efisien. Dimana pada sebuah *helper* dapat terdiri dari beberapa fungsi. Dalam pemanggilan *helper* terdapat dua cara yaitu melalui *autoload.php* dan melakukan *load* dalam setiap *controller*.

```
$autoload['helper'] = array('url','form','file');
Pemanggilan melalui file autoload.php

$this->load->helper('nama_helper');
Pemanggilan dengan load di setiap controller
```

Gambar 5.6 Pemanggilan Fungsi Helper

- *Hooks* yang dimana berfungsi untuk menampung *script hook*.
- *Language* yang dimana berfungsi sebagai menyimpan *script string* untuk bahasa. Hal ini berfungsi apabila aplikasi yang kita buat mendukung multibahasa.
- *Libraries* yang dimana berfungsi untuk menampung *library*.
- *Logs* yang dimana berfungsi untuk menampung *logs* dari aplikasi tersebut
- *Models* yang dimana berfungsi untuk menampung *script code model*, biasanya berhubungan langsung dengan *database*. Perhatikan gambar 4.7 yang merupakan salah satu contoh *script code* model dalam pemanggilan *record* yang berada pada tabel “*tb\_barang*”.

```
<?php  
  
class M_user extends CI_Model  
{  
    public function t_toko()  
    {  
        return $this->db->get_where('user', array('status_toko' => 'Sudah Terverifikasi'));  
    }  
  
    public function t_barang()  
    {  
        return $this->db->get('tb_barang');  
    }  
}
```

Gambar 5.7 Contoh Script Code Model

Untuk menggunakan *model* tersebut dimana teman - teman harus melakukan pemanggilan terlebih dahulu pada *file autoload.php* seperti pada gambar 5.8.

```
/*  
$autoload['model'] = array('m_barang', 'm_user', 'm_store', 'm_cart', 'm_kategori', 'm_admin');  
|
```

Gambar 5.8 Pemanggilang Fungsi Model Pada Autoload

Apabila sudah di panggil melalui *file autoload.php* maka fungsi *model* tersebut sudah dapat digunakan, namun untuk menampilkannya pada *view* dimana teman - teman harus melakukan pemanggilan kembali pada *file controller*, perhatikan gambar 5.9.

```
public function halaman_produk()  
{  
    $data['barang'] = $this->m_user->t_barang()->result();  
    $data['user'] = $this->db->get_where('user', ['email' => $this->session->userdata('email')])->row_array();  
    $data['title'] = 'Semua Produk';  
  
    $this->load->view('user/templates/header', $data);  
    $this->load->view('user/templates/sidebar');  
    $this->load->view('user/templates/topbar', $data);  
    $this->load->view('user/halaman_produk', $data);  
    $this->load->view('user/templates/footer');  
}
```

\**m\_user* merupakan nama dari model yang kita buat.  
\**t\_barang* merupakan *function* dari *m\_user*.

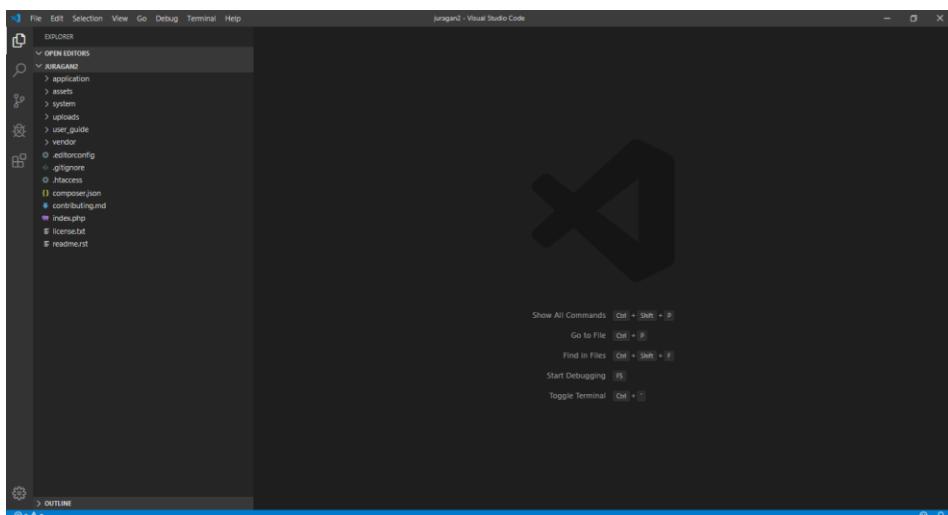
Gambar 5.9 Pemanggilan Fungsi Model Dari Controller

- ***Third\_party*** yaitu direktori yang berikan *library* dari pihak ketiga.
- ***Views*** yang dimana berfungsi sebagai menampung *script code* dari tampilan sistem yang akan kita buat.

#### 5.4 Konfigurasi Framework Codeigniter

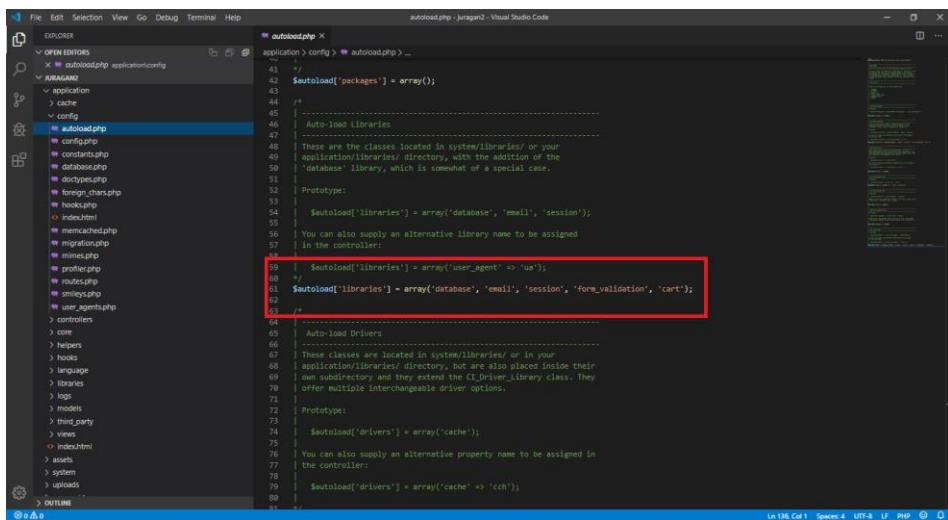
Sebelum teman – teman menggunakan *framework codeigniter* tersebut dimana teman – teman perlu melakukan konfigurasi terlebih dahulu. Bagaimana cara melakukan konfigurasi tersebut ?, berikut akan saya berikan konfigurasi apa saja yang dilakukan dalam membuat sistem JURAGAN.

Pada tahap pertama, silahkan teman – teman buka terlebih dahulu *folder framework codeigniter* yang sudah di *dowanload* dan dilakukan pengujian. Karena kali ini saya akan memberikan konfigurasi pada sistem JURAGAN, dimana saya akan membuka *folder framewok codeigniter* tersebut dan sudah saya *rename* menjadi juragandua.



Gambar 5.10 Folder CI JURAGAN

Silahkan teman – teman masuk ke direktori *application/config* dan bukan file *autoload.php*. Dimana teman – teman perlu melakukan konfigurasi pada bagian *libraries*, silahkan teman – teman scroll sampe baris 61. Lakukan konfigurasi terhadap *libraries* yang diperlukan. Pada sistem JURAGAN dimana *libraries* yang diperlukan adalah *database* yang dimana akan menggunakan *database* sebagai tempat penyimpanan datanya, *email* yang dimana akan digunakan sebagai fungsi *validasi*, *session*, *form\_validation* yang dimana digunakan untuk pemberian desain interaksi pada sistem, dan yang terakhir adalah *cart* dimana digunakan untuk membangun keranjang belanja.



```

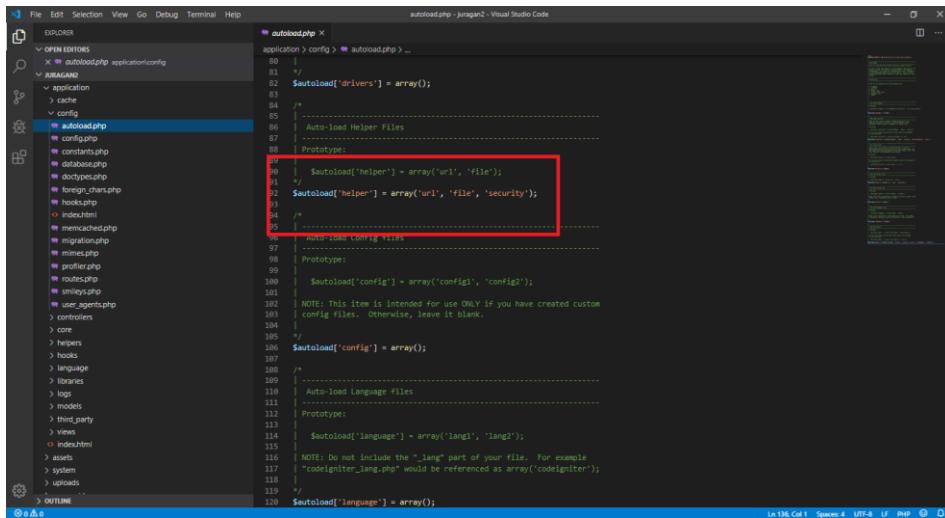
File Edit Selection View Go Debug Terminal Help
application > config > autoload.php > ...
41 /**
42  * Auto-load Libraries
43  */
44 /**
45  * These are the classes located in system/libraries/ or your
46  * application/libraries/ directory, with the addition of the
47  * 'database' library, which is somewhat of a special case.
48  */
49 /**
50  * Prototype:
51  */
52 /**
53  * $autoload['libraries'] = array('database', 'email', 'session');
54  */
55 /**
56  * You can also supply an alternative library name to be assigned
57  * in the controller:
58  */
59 /**
60  * $autoload['libraries'] = array('user_agent' => 'ua');
61  */
62 /**
63  * Auto-load Drivers
64  */
65 /**
66  * These classes are located in system/libraries/ or in your
67  * application/libraries/ directory, but are also placed inside their
68  * own subdirectory and they extend the CI_Driver_Library class. They
69  * offer multiple interchangeable driver options.
70  */
71 /**
72  * Prototype:
73  */
74 /**
75  * $autoload['drivers'] = array('cache');
76  */
77 /**
78  * You can also supply an alternative property name to be assigned in
79  * the controller:
80  */
81 /**
82  * $autoload['drivers'] = array('cache' => 'cch');
83  */
84 /**
85  */

```

Gambar 5.11 Konfigurasi Libraries Sistem JURAGAN

Langkah berikutnya masih pada file *autoload.php*, dimana teman – teman perlu melakukan konfigurasi terhadap *helpers* agar dapat menjalankan fungsi *helpers*, contoh seperti *script code* “ *base\_url()* ”. Silahkan teman – teman scroll kembali sampe pada baris 92. Pada sistem JURAGAN dimana fungsi *helpers* yang diperlukan adalah *url* yang

dimana berfungsi sebagai pemanggilan “*base\_url( )*”, file yang dimana berfungsi sebagai *upload* gambar, dan *security*.



```
autoload.php
application > config > autoload.php ...
80 |
81 |
82 | $autoload['drivers'] = array();
83 |
84 | /*
85 |  * Auto-load helper files
86 |  */
87 | Prototype:
88 |
89 |     | $autoload['Helper'] = array('url', 'file');
90 |     |
91 |     | $autoload['helper'] = array('url', 'file', 'security');
92 |
93 | }
94 |
95 |
96 |
97 | /**
98 |  * Auto-load Config files
99 |  */
100 | Prototype:
101 |
102 |     | $autoload['config'] = array('config', 'config');
103 |
104 |     | NOTE: This item is intended for use ONLY if you have created custom
105 |     | config files. Otherwise, leave it blank.
106 |
107 |     |
108 |     | $autoload['language'] = array();
109 |
110 |     | Prototype:
111 |
112 |         | $autoload['language'] = array('lang1', 'lang2');
113 |
114 |         |
115 |         | NOTE: Do not include the ".lang" part of your file. For example
116 |         | "codeigniter.lang.php" would be referenced as array('codeigniter');
117 |
118 |
119 |
120 | $autoload['language'] = array();
```

Gambar 5.12 Konfigurasi Fungsi Helpers Sistem JURAGAN

Langkah berikutnya masih tetap pada file *autoload.php*. Apabila teman – teman ingin menggunakan fungsi *model* dimana teman – teman perlu melakukan konfigurasi terlebih dahulu pada file *autoload.php*. Silahkan teman – teman lakukan scroll hingga baris 135. Konfigurasi fungsi *model* pada sistem JURAGAN adalah m\_barang yang dimana akan mengatur proses pemanggilan *database* yang berkaitan dengan produk, m\_user yang dimana akan mengelola proses ke *database* yang berhubungan dengan *user*, m\_store dimana akan mengelola proses ke *database* yang berhubungan langsung dengan toko, m\_cart dimana akan mengelola proses ke *database* yang berkaitan langsung dengan keranjang belanja, m\_kategori dimana akan mengelola proses ke *database* yang berkaitan langsung dengan kategori, dan yang terakhir adalah m\_admin dimana akan mengelola proses ke *database* yang berkaitan langsung dengan *admin*.

```

application > config > autoload.php > ...
109 | Auto-load Language files
110 |
111 | Prototype:
112 |
113 |     $autoload['language'] = array('lang1', 'lang2');
114 |
115 |     NOTE: Do not include the ".lang" part of your file. For example
116 |     "codeigniter.lang.php" would be referenced as array('codeigniter');
117 |
118 | */
119 |
120 | $autoload['language'] = array();
121 |
122 | /**
123 |  * Auto-load Models
124 |  */
125 |
126 | Prototype:
127 |
128 |     $autoload['model'] = array('first_model', 'second_model');
129 |
130 |     You can also supply an alternative model name to be assigned
131 |     in the controller:
132 |
133 |     $autoload['model1'] = array('first_model' => 'first');
134 |
135 |     $autoload['model'] = array('m_barang', 'm_user', 'm_stone', 'm_cart', 'm_kategori', 'm_admin');
136

```

Gambar 5.13 Konfigurasi Models Pada Sistem JURAGAN

Langkah berikutnya teman – teman masih tetap berada pada direktori yang sama, buka file config.php, dan jangan lupa lakukan save pada file autolod.php. Pada file config.php dimana teman – teman perlu melakukan konfigurasi pada bagian base\_url nya. Isikan dengan link website teman – teman. Pada sistem JURAGAN dimana link untuk mengakses tersebut adalah <http://localhost/juragan2/>, scroll hingga baris 26.

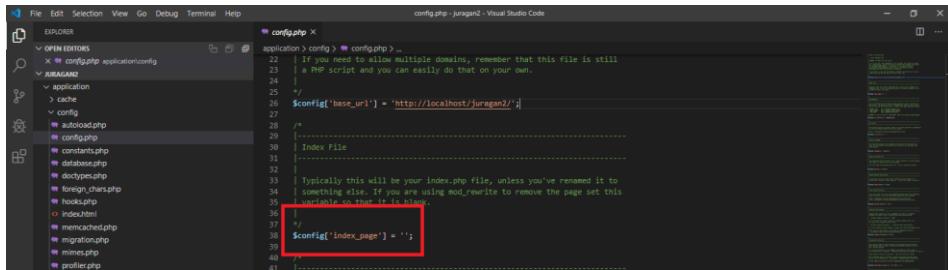
```

application > config > config.php > ...
1 | Define('BASEPATH', '');
2 | defined('BASEPATH') OR exit('No direct script access allowed');
3 |
4 |
5 | -----
6 | Base Site URL
7 |
8 |
9 | URL to your CodeIgniter root. Typically this will be your base URL,
10 | with a trailing slash:
11 |
12 | http://example.com/
13 |
14 | WARNING: You MUST set this value!
15 |
16 | If it is not set, then CodeIgniter will try guess the protocol and path
17 | to your installation, but due to security concerns the hostname will be set
18 | to $_SERVER['SERVER_NAME'] if available, or localhost otherwise.
19 | The auto-selection of host is only for convenience during
20 | development and MUST NOT be used in production.
21 |
22 | If you need to allow multiple domains, remember that this file is still
23 |
24 | -----
25 | config['base_url'] = 'http://localhost/juragan2/';
26 |
27 | -----
28 | Index File
29 |
30 |
31 | Typically this will be your index.php file, unless you've renamed it to
32 | something else. If you are using mod_rewrite to remove the page set this
33 | variable so that it is blank.
34 |
35 | config['index_page'] = '';
36 |
37 |
38 | config['index_page'] = '';
39 |
40 | /*
41 | -----

```

Gambar 5.14 Konfigurasi File Config Sistem JURAGAN

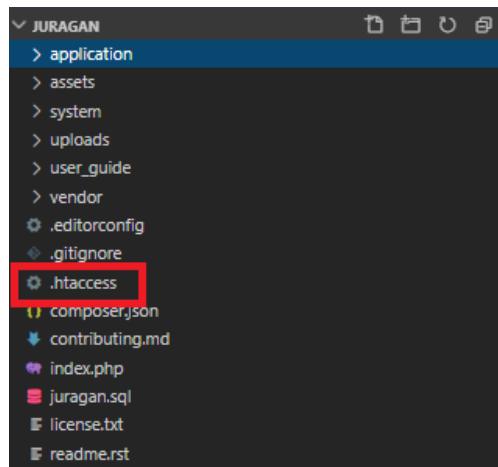
Langkah berikutnya masih dalam *file* yang sama yaitu *config.php* dimana teman – teman perlu menghapus isi pada bagian *index\_page* pada baris 38. Hal ini berfungsi agar teman – teman tidak perlu mengetikkan kembali *index.php*.



```
application > config > config.php ...
...
| If you need to allow multiple domains, remember that this file is still
| a PHP script and you can easily do that on your own.
...
25 /**
26 | $config['base_url'] = 'http://localhost/juragan2/';
27 |
28 |-----
29 | Index File
30 |
31 | -----
32 |
33 | Typically this will be your index.php file, unless you've renamed it to
34 | something else. If you are using mod_rewrite to remove the page set this
35 | variable to the name of your file.
36 |
37 |
38 | $config['index_page'] = '';
39 |
40 |-----
```

Gambar 5.15 Konfigurasi *index\_page* Sistem JURAGAN

Apabila teman – teman sudah mengkosongkan isi dari bagian *index\_page* yang berada pada *file config.php*, jangan lupa teman – teman perlu melakukan penyettingan pada *mod\_rewrite* nya. Silahkan teman – teman membuat *file* baru di luar direktori *application* dan berikan “*.htaccess*”.



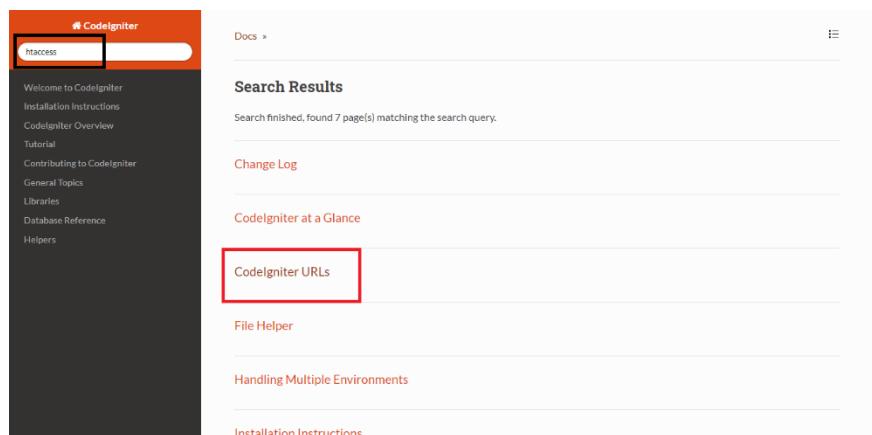
Gambar 5.16 Membuat File Baru *.htaccess*

Karena file tersebut baru dibuat dimana isinya masih kosong, lalu apa sih isi dari file `.htaccess` tersebut ?, gimana kalau kita lihat saja pada *user guide* yang merupakan dokumentasi dari *framework codeigniter* tersebut. Buka website resminya dan pilih *read the manual*.



Gambar 5.17 Membuka Dokumentasi CI

Dimana akan terbuka sebuah halaman baru, pada kolom pencarian silahkan teman – teman ketika “`htaccess`” lalu enter. Akan muncul beberapa pilihan, silahkan teman – teman pilih menu *CodeigniterURL*.



Gambar 5.18 Mencari Dokumentasi htaccess

Silahkan teman – teman scroll kebawah hingga menemukan bagian “*Removing the index.php*”, terdapat *script code* seperti yang ditandai oleh kotak merah, *copy script code* tersebut.

### Removing the index.php file

By default, the `index.php` file will be included in your URLs:

```
example.com/index.php/news/article/my_article
```

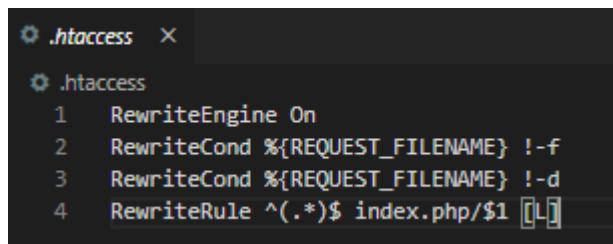
If your Apache server has `mod_rewrite` enabled, you can easily remove this file by using a `.htaccess` file with some simple rules. Here is an example of such a file, using the “negative” method in which everything is redirected except the specified items:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]
```

In the above example, any HTTP request other than those for existing directories and existing files is treated as a request for your `index.php` file.

Gambar 5.19 Script Code htaccess

Paste *script code* tersebut pada *file “.htaccess”* yang sudah teman – teman buat sebelumnya, lalu *save* dan *setting mod\_rewrite* telah selesai.

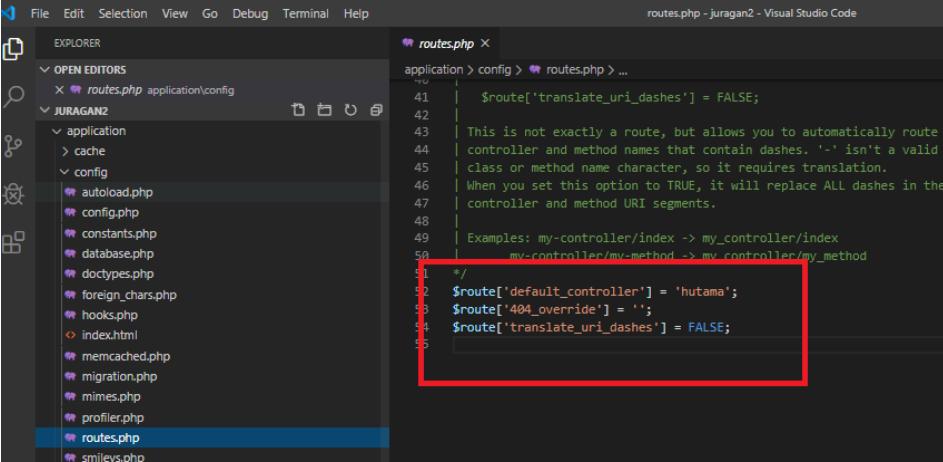


```
❶ .htaccess ❷
❸ .htaccess
❹ 1 RewriteEngine On
❺ 2 RewriteCond %{REQUEST_FILENAME} !-f
❻ 3 RewriteCond %{REQUEST_FILENAME} !-d
❼ 4 RewriteRule ^(.*)$ index.php/$1 [L]
```

Gambar 5.20 Mengisikan File .htaccess

Langkah berikutnya dimana teman – teman perlu melakukan konfigurasi terhadap *default controller*nya. Silahkan teman – teman masuk ke direktori *application/config* dan buka *file routes.php*. Pada *file routes.php* silahkan teman – teman scroll paling bawah dan lihat bagian *default\_controller* yang dimana apabila belum dirubah masih

menggunakan *wellcome*. Teman – teman perlu merubah *default\_controller* tersebut, pada sistem JURAGAN dimana *default\_controller* yang digunakan adalah “*hutama*” yang merupakan halaman awal dari sistem JURAGAN. Dikarenakan kita sudah melakukan konfigurasi pada *default controller*nya dimana saat mengakses sistem tersebut yang akan berjalan pertama kali adalah *default controller*nya.



```
routes.php ×
application > config > routes.php ...
40 |     $route['translate_uri_dashes'] = FALSE;
41 |
42 |     This is not exactly a route, but allows you to automatically route
43 |     controller and method names that contain dashes. '-' isn't a valid
44 |     class or method name character, so it requires translation.
45 |     When you set this option to TRUE, it will replace ALL dashes in the
46 |     controller and method URI segments.
47 |
48 |     Examples: my-controller/index -> my_controller/index
49 |               mv-controller/mv-method -> my_controller/my_method
50 |
51 */
52 $route['default_controller'] = 'hutama';
53 $route['404_override'] = '';
54 $route['translate_uri_dashes'] = FALSE;
```

Gambar 5.21 Konfigurasi Default Controller Sistem JURAGAN

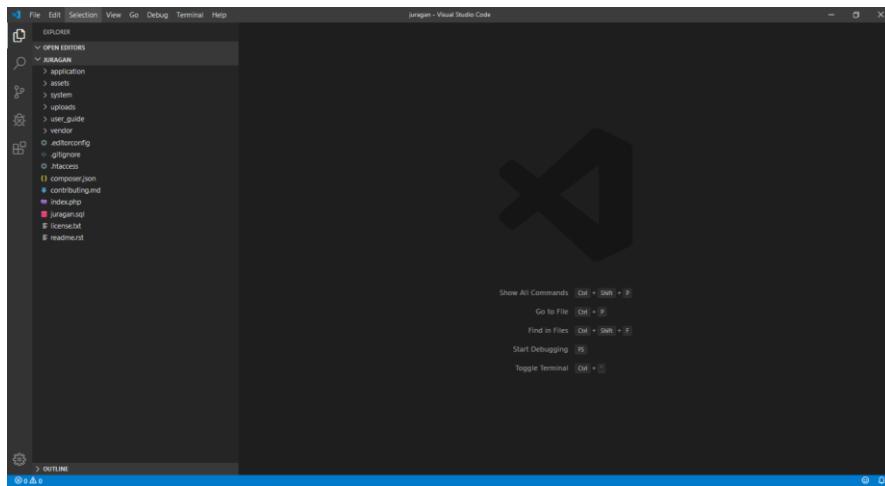
Langkah selanjutnya dimana teman – teman perlu melakukan *setting* terhadap *database* yang akan digunakan. Masih pada direktori yang sama, silahkan teman – teman buka file *database.php*. Konfigurasi *database* yang digunakan oleh sistem JURAGAN adalah sebagai berikut.

```
72     '
73     $active_group = 'default';
74     $query_builder = TRUE;
75
76     $db['default'] = array(
77         'dsn' => '',
78         'hostname' => 'localhost', Nama Server Xampp
79         'username' => 'root', Username Xampp
80         'password' => '', Password Xampp, Diisi Jika Ada
81         'database' => 'juragandua', Nama Database
82         'dbdriver' => 'mysqli',
83         'dbprefix' => '',
84         'pconnect' => FALSE,
85         'db_debug' => (ENVIRONMENT !== 'production'),
86         'cache_on' => FALSE,
87         'cachedir' => '',
88         'char_set' => 'utf8',
89         'dbcollat' => 'utf8_general_ci',
90         'swap_pre' => '',
91         'encrypt' => FALSE,
92         'compress' => FALSE,
93         'stricton' => FALSE,
94         'failover' => array(),
95         'save_queries' => TRUE
96     );
97 
```

Gambar 5.22 Konfigurasi Database Sistem JURAGAN

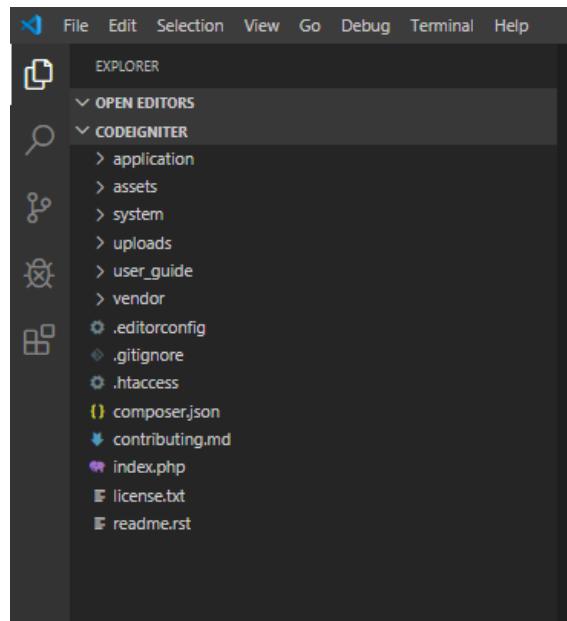
## 5.5 Membuat CRUD Dengan Codeigniter

Setelah diberikannya teori – teori tersebut, dimana kali ini teman – teman akan mencoba membuat sebuah *website* sederhana dengan fungsi CRUD. Apa itu *website* CRUD ?, *web site* CRUD adalah dimana suatu sistem yang memiliki fungsi atau *action* “*cretae, read, update, dan delete*”. Untuk membuat *website* CRUD dimana teman – teman memerlukan *software* pendukung yang sudah teman – teman siapkan pada bab 4. Silahkan jalankan *software Xampp*, lalu buka *software Visual Studio Code*.



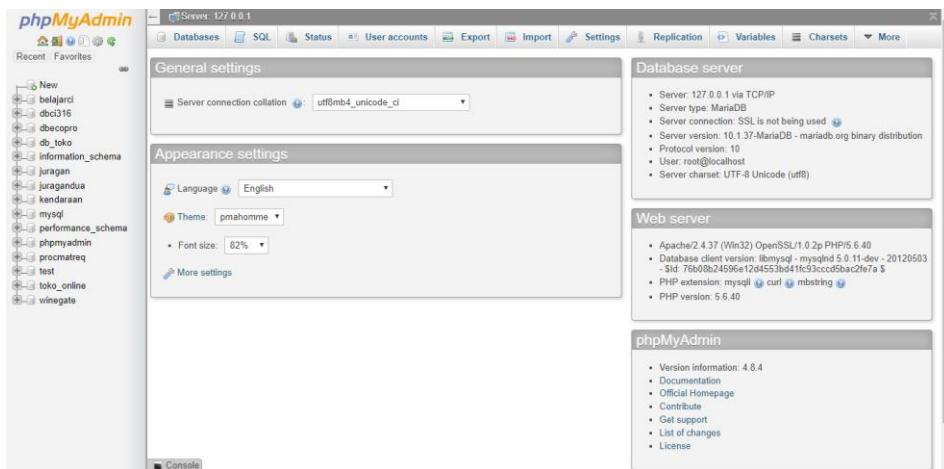
Gambar 5.23 Mempersiapkan Software VSCode

Dalam membuat aplikasi CRUD bagaimana kita pakai saja *folder codeigniter* yang sudah teman – teman *download* pada bab 4. Buka *folder codeigniter* tersebut pada *software visual studio code*.



Gambar 5.24 Open Folder CI Pada VSCode

Hal yang pertama teman – teman lakukan dalam membuat aplikasi CRUD ini adalah membuat *database* terlebih dahulu. Silahkan teman – teman masuk ke halaman *PhpMyAdmin* dengan cara mengunjungi *link* berikut, <http://localhost/phpmyadmin/>.



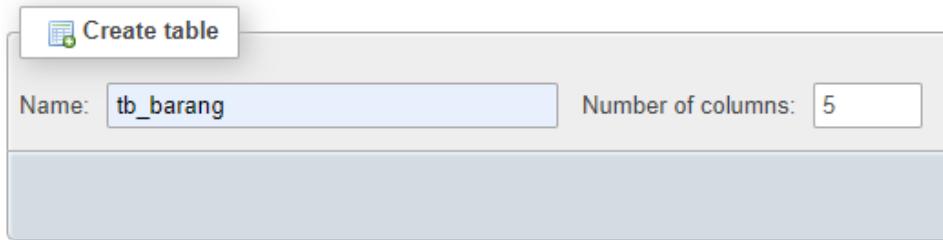
Gambar 5.25 Halaman PHP MyAdmin

Pada halaman *PHP MyAdmin* silahkan teman – teman pilih menu *database* untuk membuat *database*. Silahkan teman – teman memberikan nama pada *database* yang akan dibuat, pada pembelajaran kali ini dimana saya akan menamai *database* dengan “crud\_brg” lalu pilih *create*.

A screenshot of a 'Create database' dialog box. It has a 'Create database' button with a gear icon, a text input field containing 'crud\_brg', a dropdown menu set to 'latin1\_swedish\_ci', and a 'Create' button.

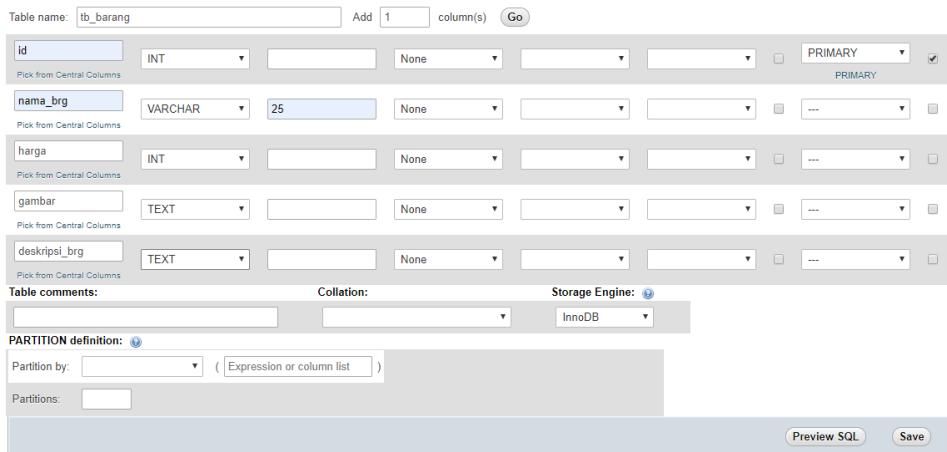
Gambar 5.26 Membuat Nama Database

*Database* teman – teman sudah terbuat, dimana langkah selanjutnya teman – teman perlu membuat sebuah *table* pada *database*. Pada pembelajaran kali ini dimana saya akan membuat *table* *tb\_barang* dengan 5 kolom.



Gambar 5.27 Membuat Tabel tb\_barang

Pada bagian sisi sebelah kanan terdapat button “Go” silahkan teman – teman klik dimana teman – teman akan dibawa kedalam membuat struktur tabel “tb\_barang”. Perhatikan gambar 5.28 untuk membuat struktur *table* tb\_barang.



Gambar 5.28 Membuat Struktur Tabel tb\_barang

Setelah itu pilih save dimana struktur *table* tb\_barang teman – teman sudah siap. Perhatikan gambar 5.29 merupakan struktur *table* “tb\_barang” pada *database* teman – teman.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id</b> 	int(11)			No	None		AUTO_INCREMENT	 Change  Drop ▾ More
2	<b>nama_brg</b>	varchar(25)	latin1_swedish_ci		No	None			 Change  Drop ▾ More
3	<b>harga</b>	int(11)			No	None			 Change  Drop ▾ More
4	<b>gambar</b>	text	latin1_swedish_ci		No	None			 Change  Drop ▾ More
5	<b>deskripsi_brg</b>	text	latin1_swedish_ci		No	None			 Change  Drop ▾ More

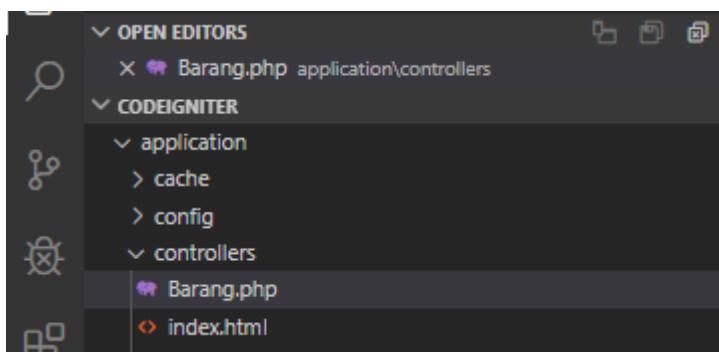
Gambar 5.29 Struktur Tabel tb\_barang

Pada struktur tabel “tb\_barang” tersebut dimana terdapat *id* yang berfungsi untuk memberikan *id* pada barang yang akan di *input*. *Id* tersebut merupakan *primary key* dari tabel “tb\_barang”. *Primary key* adalah sebuah *code* unik yang dimiliki pada suatu barang tersebut dan umumnya tidak sama. *Id* tersebut diberikan fungsi *auto increment* yang dimana akan secara otomatis bertambah. Karena *id* tersebut sifatnya angka maka tipe data yang digunakan adalah *integer*. Lalu terdapat *name\_brg* yang dimana berfungsi untuk menampung nama barang yang akan di *input* kan, karena sifatnya dapat berupa angka dan *text* maka tipe data yang digunakan adalah *varchar* dengan panjang 25. Lalu terdapat harga yang dimana berfungsi untuk menampung harga dari barang tersebut, dikarenakan sifatnya adalah angka maka tipe data yang digunakan adalah *integer*. Lalu ada gambar yang dimana berfungsi untuk menampung gambar dari barang tersebut dan berikan tipe data *text* dan yang terakhir adalah *deskripsi\_brg* yang dimana berfungsi untuk menampung deskripsi barang tersebut, tipe data yang digunakan adalah *text*. Apabila dibentuk dengan *source code* SQL akan terlihat seperti pada *source code* 5.1.

```
CREATE TABLE `tb_barang` (
    `id` int(11) NOT NULL,
    `nama_brg` varchar(25) NOT NULL,
    `harga` int(11) NOT NULL,
    `gambar` text NOT NULL,
    `deskripsi_brg` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Source Code 5.1 SQL Create Table

Langkah selanjutnya silahkan teman – teman lakukan kofingurasi terhadap *folder codeigniter*. Untuk langkah – langkah konfigurasi silahkan teman – teman buka kembali sub bab 5.4. Apabila teman – teman sudah selesai melakukan konfigurasi dimana langkah selanjutnya teman – teman akan membuat *controller*. *Controller* terdapat pada direktori *application* yang dimana bertugas dalam menangani *HTTP request*. *Controller* juga akan menghubungkan antara *view* dan juga *models*. Silahkan teman – teman buat *file* baru dengan nama “Barang.php” pada direktori *application/controllers*.



Gambar 5.30 Membuat Controllers Barang.php

Bagaimana apakah teman – teman sudah membuat *controllers* Barang.php ?, jika sudah dimana teman – teman perlu memasukkan *source code* 5.2 untuk pertama kalinya.

```
<?php

defined('BASEPATH') or exit('No direct script
access allowed');

class Barang extends CI_Controller
{



}
```

*Source Code 5.2 Code Pertama Controllers*

Dalam *controllers* dimana teman – teman perlu membuat *function – function* yang berguna untuk memisahkan *request* yang akan diberikan. Dimana teman – teman perlu membuat *function – construct* untuk yang pertama. *Function* tersebut adalah *function* yang akan dikerjakan atau dieksekusi terlebih dahulu saat *file controller* Barang.php di akses. Pada *function* tersebut akan memanggil fungsi *models* dan juga *library form\_validation*, perhatikan *source code* 5.3.

```
<?php

defined('BASEPATH') or exit('No direct script
access allowed');

class Barang extends CI_Controller
{

    public function __construct()
    {
        parent::__construct();
        $this->load->model("m_barang");
```

```
    $this->load->library('form_validation');
}

}
```

*Source Code 5.3 Pemanggilan Fungsi Models Dan Form\_validation*

Setelah membuat *function \_\_construct* dimana teman – teman akan membuat *function default* dari *controllers* Barang.php tersebut. Silahkan teman – teman membuat *function* baru dengan nama *index*. *Function index* tersebut merupakan *default function* dari *controller* Barang.php yang artinya saat teman – teman menggail *controller* Barang.php tanpa *function* maka yang akan di akases adalah *function index* tersebut. Pada *function index* kita gunakan untuk membuat fungsi *read/view*. Silahkan teman – teman buat *function index* tersebut di bawah *function \_\_construct*, dengan mengisikan *source code 5.4*.

```
public function index()
{
    $data["products"] = $this->m_barang->
                        tampil_barang();
    $this->load->view("barang/v_barang", $data);
}
```

*Source Code 5.4 Membuat Function Index*

Jika teman – teman akses *controllers* Barang.php pada *web browser* teman – teman pasti akan terjadi *error*, hal ini dikarenakan kita belum membuat fungsi *models m\_barang* dan *view v\_barang*. Untuk membuat fungsi tersebut silahkan teman – teman masuk ke bagian pembuatan *models* dan *view*.

Langkah selanjutnya masih pada *file controllers* kita dimana teman – teman perlu membuat *function* tambah yang berfungsi untuk melakukan *fungsi create*. Buat *function* tambah di bawah *function index*, perhatikan *source code* 5.5.

```
public function tambah()
{
    $barang = $this->m_barang;
    $validation = $this->form_validation;

    $validation->set_rules($barang->rules());

    if($validation->run())
    {
        $barang->simpan();
        $this->session->set_flashdata(
            'success', 'Berhasil disimpan');
    }

    $this->load->view("barang/inputbarang");
}
```

*Source Code 5.5 Membuat Function Tambah Pada Controllers*

Pada *function* tersebut dimana teman – teman akan belajar dalam menerapkan fungsi *models* dan *form\_validation*. Perhatikan pada *script code* “*simpan( )*”, dimana merupakan sebuah *function* pada *models m\_barang* yang dipanggil menggunakan variabel “*barang*”. Dalam *codeigniter* untuk membuat sebuah variabel taman – taman perlu menambahkan logo “\$” disetiap awal nama variabel tersebut. Pada *function* tersebut dimana akan memanggil sebuah *form inputbarang* pada *folder view/barang*.

Langkah selanjutnya masih dalam pembuatan *controllers*, dimana teman – teman perlu menambahkan *function* baru dengan nama edit. Coba perhatikan *source code* 5.6 berikut.

```
public function edit($id)
{
    if (!isset($id)) redirect('barang/v_barang');

    $barang = $this->m_barang;
    $validation = $this->form_validation;
    $validation->set_rules($barang->rules());

    if ($validation->run())
    {
        $barang->update();
        $this->session->set_flashdata('success',
                                         'Berhasil disimpan');
    }

    $data["barang"] = $barang->ambil_id($id);
    if (!$data["barang"]) show_404();

    $this->load->view("barang/edit", $data);
}
```

*Source Code 5.6 Membuat Function Edit Pada Controllers*

Pada *function* tersebut dimana akan membaca *id* pada barang yang akan di edit. Jika *id* nya kosong maka sistem akan melakukan *redirect* ke *route* barang. Perhatikan *source code* “\$barang->update();” yang dimana akan menjalankan *function update* pada *file m\_barang*. *Function update* tersebut akan menyimpan data yang sudah di rubah ke dalam *database*. Apabila data berhasil di *update* maka akan muncul sebuah notifikasi dari *form\_validation* berupaka “Berhasil disimpan”. Pada *function edit* tersebut

dimana kita juga akan menampilkan *form edit* sekaligus mengambil data berdasarkan *id* yang diberikan.

Langkah selanjutnya merupakan langkah terakhir dalam pembuatan *controllers*, dimana teman – teman perlu membuat *function delete* untuk melakukan *request* pada fungsi *delete*. *Function delete* merupakan *function* terakhir pada file *Barang.php*. Silahkan teman – teman perhatikan *source code* 5.7 berikut.

```
public function delete($id)
{
    if (!isset($id)) show_404();

    if ($this->m_barang->delete($id))
    {
        redirect(site_url('barang/v_barang'));
    }
}
```

*Source Code 5.7 Membuat Function Delete Pada Controllers*

Pada *function* cukup sederhana dimana *function* tersebut akan mengambil *id* dari *tb\_barang*. Apabila *id* tersebut ada dan tidak kosong maka sistem akan menjalankan fungsi *models m\_barang* dan menjalankan *function delete* lalu melakukan *redirect* ke halaman *v\_barang*. Selamat teman – teman sudah berhasil membuat *file controllers*. Teman – teman dapat melihat *source code full* dari *file controllers* pada *source code* 5.8.

```
<?php

defined('BASEPATH') or exit('No direct script
access allowed');

class Barang extends CI_Controller
{
```

```
    public function __construct()
{
    parent::__construct();
    $this->load->model("m_barang");
    $this->load->library('form_validation');
}

    public function index()
{
    $data["products"] = $this->m_barang->
        tampil_barang();
    $this->load->view("barang/v_barang", $data);
}

    public function tambah()
{
    $barang = $this->m_barang;
    $validation = $this->form_validation;

    $validation->set_rules($barang->rules());

    if($validation->run())
    {
        $barang->simpan();
        $this->session->set_flashdata(
            'success', 'Berhasil disimpan');
    }

    $this->load->view("barang/inputbarang");
}

    public function edit($id)
{
    if (!isset($id)) redirect('barang/v_barang');

    $barang = $this->m_barang;
    $validation = $this->form_validation;
    $validation->set_rules($barang->rules());
```

```

if ($validation->run())
{
    $barang->update();
    $this->session->set_flashdata('success',
                                    'Berhasil disimpan');
}

$data["barang"] = $barang->ambil_id($id);
if (!$data["barang"]) show_404();

$this->load->view("barang/edit", $data);
}

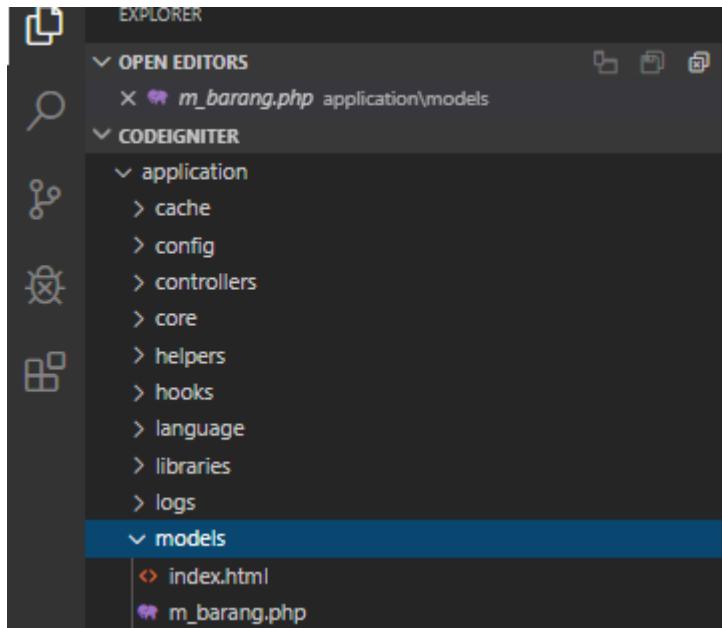
public function delete($id)
{
    if (!isset($id)) show_404();

    if ($this->m_barang->delete($id))
    {
        redirect(site_url('barang/v_barang'));
    }
}

```

*Source Code 5.8 Code Full Controllers Barang.php*

Langkah selanjutnya dimana teman – teman akan belajar membuat fungsi *models*. Silahkan teman – teman buat *file* baru dengan nama *m\_barang.php* pada direktori *application/models*.



Gambar 5.31 Membuat File m\_barang Pada Models

Silahkan teman – teman berikan *source code* 5.9 untuk langkah awal dalam membuat file *m\_barang.php*.

```
<?php  
  
class M_barang extends CI_Model  
{  
  
}
```

Source Code 5.9 Code Awal Pada *m\_barang.php*

Langkah selanjutnya dimana teman – teman perlu mendeskripsikan *field* yang ada pada *database* teman – teman. Perlu diperhatikan nama *field* harus sama persis dengan yang ada pada *database*. Perhatikan *soure code* 5.10 berikut.

```

private $_table = "tb_barang";

public $id;
public $nama_brg;
public $harga;
public $gambar = "default.jpg";
public $deskripsi_brg;

public function rules()
{
    return [
        ['field' => 'nama_brg',
         'label' => 'Nama',
         'rules' => 'required'],

        ['field' => 'harga',
         'label' => 'Harga',
         'rules' => 'numeric'],

        ['field' => 'deskripsi_brg',
         'label' => 'Deskripsi',
         'rules' => 'required']
    ];
}

```

*Source Code 5.10 Mendeskripsikan Field Pada Models*

Langkah selanjutnya dimana teman – teman perlu membuat variabel *private* yang dimana hanya dapat di akses pada *class* tersebut. Perhatikan *source code* *private \$\_table = “tb\_barang”;* yang merupakan variabel *private* untuk membaca tabel *tb\_barang* pada *database*. Selanjutnya teman – teman perlu mendeskripsikan *field* yang berada pada *table tb\_barang* tersebut dengan diawali *method public* agar dapat diakses secara *public*. Karena kita menggunakan fungsi *form\_validation* silahkan teman – teman buat *rules* nya atau aturan. Untuk membuat *rules* silahkan teman – teman membuat *function* baru dengan nama *rules* pada *file m\_barang.php*.

Langkah selanjutnya dimana teman – teman perlu membuat *function* `tampil_barang` yang berfungsi untuk mengambil semua isi pada tabel `tb_barang`. Perhatikan *source code* 5.11 berikut ini.

```
public function tampil_barang()
{
    return $this->db->get(
        $this->_table)->result();
}
```

*Source Code 5.11 Function Fungsi Read*

Pada *souce code* 5.11 tersebut dimana merupakan *function* yang berfungsi untuk mengambil semua isi data yang ada pada *table* `tb_barang`. *Source code* tersebut sama hal nya dengan “ **SELECT \* FROM tb\_barang** ” pada PHP. Pada bagian `_table` dimana akan membaca tabel yang sudah teman – temas deskripsikan diatas. **Result()** berifungsi untuk mengambil semua data yang berada pada *table* `tb_barang`.

Langkah selanjutnya dimana teman – teman perlu membuat fungsi *read* ke dalam *database* namun berdasarkan *id* yang diberikan. Hal ini digunakan saat teman – teman akan mengambil data berdasarkan *id*, sehingga data yang akan ditampilkan seuai dengan *id* yang diberikan. Perhatikan *souce code* 5.12 berikut.

```
public function ambil_id($id)
{
    return $this->db->get_where($this->_table,
        ["id" => $id])->row();
}
```

*Source Code 5.12 Fungsi Read Berdasarkan Id*

Silahkan teman – teman perhatikan *source code* 5.12 tersebut dimana teman – teman akan mendeskripsikan *id* pada *function* ambil\_id. Teman – teman akan mengambil data dari *database* dengan fungsi ***get\_whare***, langkah berikutnya teman – teman akan memanggil tabel dengan variabel ***\_table***. Setelah itu teman – teman akan mendeskripsikan *id* tersebut, ambil sebaris saja tambahkankan *script code* ***row( )***.

Langkah berikutnya dimana teman – teman perlu membuat fungsi simpan. *Function* tersebut berfungsi untuk menyimpan data ke dalam *table* yang dimana akan di dikirim dengan fungsi ***\$this->input->post( )***. Perhatikan *source code* 5.13.

```
public function simpan()
{
    $post = $this->input->post();
    $this->id = uniqid();
    $this->nama_brg = $post["nama_brg"];
    $this->harga = $post["harga"];
    $this->deskripsi_brg = $post["deskripsi_brg"];

    return $this->db->insert($this->_table, $this);
}
```

*Source Code 5.13 Membuat Function Simpan Pada Models*

Perhatikan *script code* tersebut dimana kita akan membuat *function* pada *models* dengan nama simpan. Pada *script code* ***\$post = \$this->input->post( )***; dimana berfungsi untuk membuat variabel *input*. Pada *script code* ***\$this->id = Uniqid( )***; dimana sistem akan membuat *id* unik untuk di *input* kan karena *field* tersebut berupa *primary key*. Selanjutnya pada bagian *script code* ***\$this->nama\_brg = \$post["nama\_brg"];*** sampai ***\$this->deskripsi\_brg =***

`$post["deskripsi_brg"];` dimana berfungsi untuk mengirim data yang di *input* kan ke dalam *field* yang dituju. Setelah itu dimana teman – teman perlu memasukkan data tersebut kedalam *database* dengan fungsi *insert* seperti *script code return \$this->db->insert->(\$this->\_table, \$this);*

Langkah selanjutnya mari kita belajar mengenai fungsi *edit* atau *update* pada *framework codeigniter*. Fungsi tersebut berfungsi apabila teman – teman ingin merubah data pada isi *field* yang berada dalam *database*. Silahkan teman – teman perhatikan *source code 5.14*.

```
public function update()
{
    $post = $this->input->post();

    $this->id = $post["id"];
    $this->nama_brg = $post["nama_brg"];
    $this->harga = $post["harga"];
    $this->deskripsi_brg = $post["deskripsi_brg"];

    return $this->db->update($this->_table, $this,
        array('id' => $post['id']));
}
```

*Source Code 5.14 Membuat Fungsi Update Pada Models*

Silahkan teman – teman ketik terlebih dahulu *source code 5.14* tersebut pada *software visual studio code*. Apabila sudah yuk kita bahas pada tiap – tiap barisnya. Pada baris pertama yaitu **public function update( )** yang dimana teman – teman akan membuat *function* dengan sifatnya *public* dan diberi nama *update*. Pada baris selanjutnya yaitu `$post = $this->input->post( );` dimana teman – teman akan membuat variabel *post* dan diberikan fungsi untuk menyimpan data yang telah di *input* kan. Pada baris selanjutnya yaitu `$this->id = $post["id"]` dimana teman – teman akan

memanggil variabel *post* terhadap *field id* pada *tabel tb\_barang*. Pada baris selanjutnya yaitu ***\$this->nama\_brg = \$post[“nama\_brg”];*** dimana teman – taman akan memanggil *variabel post* dan menerapkannya kepada *field nama\_brg* pada *table tb\_barang*. Pada baris selanjutnya yaitu ***\$this->harga = \$post[“harga”];*** dimana teman – teman akan memanggil *variabel post* dan menerapkannya kepada *field harga* pada *table tb\_barang*. Pada baris selanjutnya yaitu ***\$this->deskripsi\_brg = \$post[“deskripsi\_brg”];*** dimana teman – teman akan memanggil *variabel post* dan menerapkannya pada *field deskripsi\_brg* yang berada pada *table tb\_barang*. Pada baris selanjutnya yaitu ***return \$this->db->update*** dimana teman – teman akan membuat fungsi *update* terhadap *database*, lalu disambung dengan ***(\$this->\_table, \$this, array(‘id’ => \$post[‘id’]));*** yang dimana teman – teman akan menyimpan data tersebut berdasarkan *id* yang dikirim ke dalam *table tb\_barang* dengan memanggil fungsi *\_table*.

Langkah selanjutnya dimana teman – teman akan membuat fungsi *delete* atau hapus pada *framework codeigniter*. Fungsi *delete* tersebut hampir sama dengan fungsi *update* yang sudah teman – teman buat, namun lebih simpel, silahkan teman – teman perhatikan *source code 5.15* berikut ini.

```
public function delete($id)
{
    return $this->db->delete($this->_table,
        array("id" => $id));
}
```

*Source Code 5.15 Membuat Fungsi Delete Pada Models*

Silahkan teman – teman ketik *source code 5.15* tersebut kedalam *software visula studio code* teman – teman. Gimana apa sudah di ketik ? kalau sudah

yuk kita bahas tiap – tiap barisnya. Pada baris pertama yaitu ***public function delete(\$id)*** dimana teman – teman akan membuat *function* yang bersifat ***public*** dengan nama *delete* dan memberikan variabel *id* di dalamnya. Pada baris berikutnya yaitu ***return \$this → db → delete(\$this → \_table, array("id" => \$id));*** yang dimana teman – teman akan membuat fungsi *delete* terhadap data yang berada pada *database* dengan *table tb\_barang* yang dimana data tersebut diambil dari *id* yang dikirimkan. Selamat teman – teman sudah berhasil membuat fungsi *models* yang di dalamnya menerapkan fungsi CRUD. Untuk *script code* secara *full* pada *file m\_barang.php* teman – teman dapat melihatnya pada *source code* 5.16 berikut.

```
<?php

class M_barang extends CI_Model
{

    private $_table = "tb_barang";

    public $id;
    public $nama_brg;
    public $harga;
    public $gambar = "default.jpg";
    public $deskripsi_brg;

    public function rules()
    {
        return [
            ['field' => 'nama_brg',
            'label' => 'Nama',
            'rules' => 'required'],

            ['field' => 'harga',
            'label' => 'Harga',
            'rules' => 'numeric'],
        ];
    }
}
```

```
        ['field' => 'deskripsi_brg',
         'label' => 'Deskripsi',
         'rules' => 'required']
    ];
}

public function tampil_barang()
{
    return $this->db->get(
        $this->_table)->result();
}

public function ambil_id($id)
{
    return $this->db->get_where($this->_table,
        ["id" => $id])->row();
}

public function simpan()
{
    $post = $this->input->post();

    $this->id = uniqid();
    $this->nama_brg = $post["nama_brg"];
    $this->harga = $post["harga"];
    $this->deskripsi_brg =
        $post["deskripsi_brg"];

    return $this->db->insert(
        $this->_table, $this);
}

public function update()
{
    $post = $this->input->post();
```

```

    $this->id = $post["id"];
    $this->nama_brg = $post["nama_brg"];
    $this->harga = $post["harga"];
    $this->deskripsi_brg =
        $post["deskripsi_brg"];

    return $this->db->update(
        $this->_table, $this,
        array('id' => $post['id']));
}

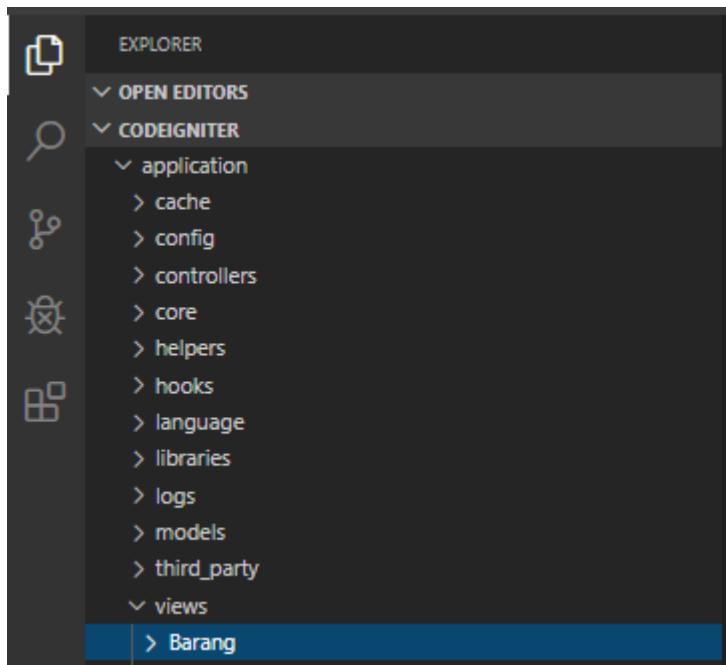
public function delete($id)
{
    return $this->db->delete($this->_table,
        array("id" => $id));
}

}

```

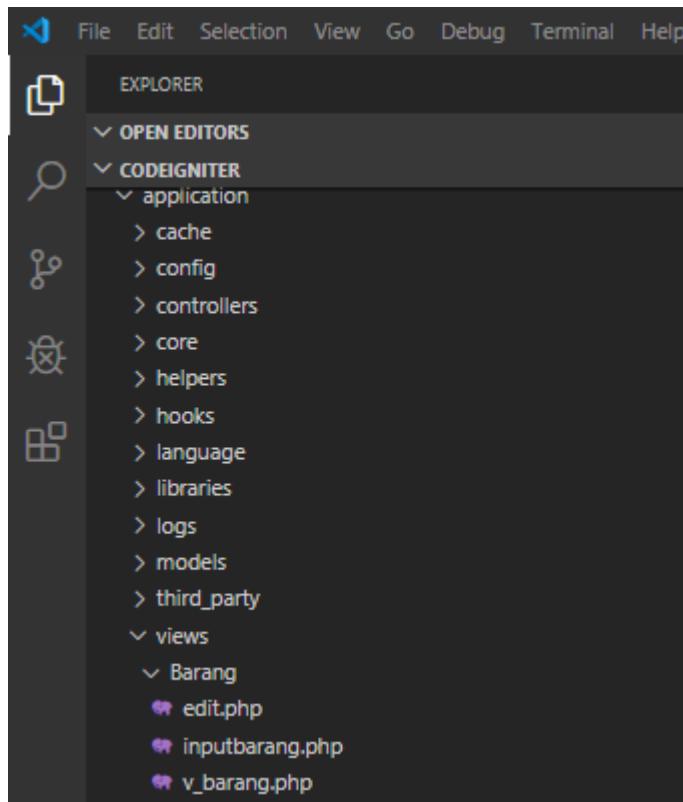
*Source Code 5.16 Code Full Models File m\_barang.php*

Pada pembahasan sebelumnya dimana teman – teman sudah belajar membuat *database* dengan menggunakan server *localhost* pada *software Xampp*, membuat *controllers* dengan nama *Barang.php* pada *framework codeigniter*, dan membuat fungsi *models* dengan menerapkan fungsi *CRUD (Create, Read, Update, dan Delete)* pada *framework codeigniter*. Langkah selanjutnya dimana teman – teman akan membuat *view* yang berguna untuk mengelola data – data pada *database* tersebut. Untuk membuat *view* tersebut dimana teman – teman perlu membuat *folder* dengan nama *barang* pada direktori *applications/views*.



Gambar 5.32 Membuat Folder Barang Pada Views

Terdapat tiga *view* yang perlu teman – teman buat, yaitu v\_barang dimana berfungsi untuk menampilkan barang – barang dari *database* yang telah di *input*, inputbarang yang dimana berfungsi untuk melakukan aktivitas *input* atau menambah data, dan edit yang dimana berfungsi untuk melakukan aktivitas edit atau *update* pada data barang. Lahkah dalam membuat *view* silahkan teman – teman buat terlebih dahulu tiga *view* tersebut di dalam *folder* barang dengan format PHP, perhatikan gambar 3.33 berikut.



Gambar 5.33 Membuat File PHP Pada Folder Barang

Langkah selanjutnya gimana kalau kita lakukan pengkodean pada *file v\_barang.php* agar data yang berada pada *database* dapat di tampilkan pada layar. Langkah pertama dalam pengkodean *file v\_barang.php* dimana teman – teman perlu memanggil *template bootstrap*. Bagaimana sih cara memanggil *template* pada *bootstrap* pada *framework codeigniter* ?, pertanyaan tersebut akan di bahas pada sub bab menghubugkan *bootstrap* pada *codeigniter*. *Template* yang perlu teman – teman panggil adalah *header*, hal ini karena *header* merupakan *template* pada bagian atas dari suatu sistem. Untuk memanggil *template header*, silahkan teman – teman perhatikan pada *source code 5.17* berikut ini.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <?php
        $this->load->view("templates/header.php")
    ?>
</head>
```

*Source Code 5.17 Memanggil Header Pada View v\_barang*

Pada *script code* tersebut dimana teman – teman perlu memanggil fungsi PHP yang diawali dengan **<?php** dan ditutup dengan **?>**. Untuk melakukan pemanggilan *header* dimana teman – teman perlu melakukan *load view* dengan cara menambahkan *code \$this->load->view("templates/header.php")* di dalam fungsi PHP tersebut. Pada *code* tersebut dapat dibaca dimana teman – teman akan melakukan memanggil *file header.php* pada *folder templates* yang berada pada direktori *views*.

Langkah selanjutnya dimana teman – teman perlu memanggil *templates topbar* dan juga *sidebar* pada bagian *body* dari *view* tersebut. Silahkan teman – teman perhatikan *source code* 5.18 berikut.

```
<body id="page-top">

    <?php
        $this->load->view("templates/topbar.php")
    ?>
    <div id="wrapper">

        <?php
            $this->load->view("templates/sidebar.php")
        ?>
```

*Source Code 5.18 Memanggil Template Topbar Dan Sidebar Pada View v\_barang*

Pada *script code* tersebut dimana teman – teman perlu memanggil fungsi PHP yang diawali dengan `<?php` dan ditutup dengan `?>`. Untuk melakukan pemanggilan *template topbar* dimana teman – teman perlu menambahkan `$this->load->view("templates/topbar.php")` di dalam fungsi PHP tersebut. *Code* tersebut dapat dibaca dimana teman – teman akan melakukan *load view* dengan *file topbar.php* yang berada pada *folder templates*. Sedangkan untuk memanggil *template sidebar* dimana teman – teman perlu menambahkan `$this->load->view("templates/sidebar.php")` di dalam fungsi PHP tersebut. *Code* tersebut dapat dibaca dimana teman – teman akan melakukan *load view* pada *file sidebar* yang berada pada *folder templates*.

Langkah selanjutnya dimana teman – teman perlu membuat fungsi *table* agar data yang berada pada *database* tersusun di dalam tabel. Silahkan teman – teman perhatikan *source code 5.19* berikut.

```
<div id="content-wrapper">

    <div class="container-fluid">

        <!-- Menampilkan Data -->
        <div class="card mb-3">
            <div class="card-header">
                <a href="<?php=
                    base_url('barang/tambah')
                    ?>">
                    <i class="fas fa-plus"></i>
                    Tambah Data
                </a>
            </div>

            <div class="card-body">
                <div class="table-responsive">
                    <table class="table table-hover">
```

```
id="dataTable" width="100%" cellspacing="0">
<thead>
  <tr>
    <th>Nama Barang</th>
    <th>Harga</th>
    <th>Gambar Barang</th>
    <th>Deskripsi Barang</th>
    <th>Aksi</th>
  </tr>
</thead>
```

*Source Code 5.19 Membuat Tabel Pada View v\_barang*

Pada *script code* tersebut dimana teman – teman perlu menggunakan fungsi *div*, teman – teman perlu memanggil *class container-fluid* yang dimana membuat tampilan pada sistem kita terlihat sama rata. Perhatikan *script code* pemanggilan *link* tersebut yang dimana dibuka dengan *tag <a href* dan ditutup dengan *</a>*. Pemanggilan *link* tersebut berfungsi untuk memamringgil *form input* barang dengan menerapkan fungsi *helpers* pada *codeigniter*. Dimana kita juga akan memanggil *icon plus* secara yang didapat pada pemanggil *file font awesome*. Selanjutnya teman – teman akan membuat *table* dengan *header* nya berupa “Nama Barang” untuk menampung data nama dari barang tersebut, “Harga” untuk menampung data harga dari barang tersebut, “Gambar Barang” untuk menampung data gambar dari gambar barang tersebut, “Deskripsi Barang” untuk menampung data deskripsi dari barang tersebut, dan yang terakhir adalah aksi untuk memberikan aksi pada data barang tersebut.

Langkah selanjutnya dimana teman – teman perlu membuat *table body* yang berfungsi untuk memanggil isi data barang tersebut dari *table tb\_barang* yang berada pada *database*. Silahkan teman – teman perhatikan *source code 5.20*.

```

<tbody>
    <?php foreach ($barang as $brg): ?>

        <tr>
            <td width="150">
                <?php= $brg->nama_brg ?>
            </td>

            <td>
                <?php= $brg->harga ?>
            </td>

            <td>
                
            </td>

            <td class="small">
                <?php= substr($brg->deskripsi_brg, 0, 120) ?>
            </td>

            <td width="250">
                <a href="<?php= base_url('barang/edit/'.
                    $brg->id) ?>" 
                    class="btn btn-small"><i class="fas fa-edit">
                </i> Edit </a>

                <a onclick="deleteConfirm('<?php=
                    base_url('barang/delete/'.{$barang-> id}) ?>')"
                    href="#" class="btn btn-small text-danger">
                    <i class="fas fa-trash"></i> Hapus</a>
                </td>
            </tr>
        <?php endforeach; ?>
    </tbody>
</table>
</div>
</div>
</div>

```

*Source Code 5.20 Membuat Table Body Pada View v\_barang*

Pada *script code* tersebut dimana teman – teman perlu menambahkan variabel barang dengan fungsi ***foreach***, agar penamaan tidak panjang ubah nama variabel barang tersebut dengan fungsi ***as*** menjadi ***brg***. Untuk menampilkan datanya dimana teman – teman perlu memanggilnya contoh silahkan teman – teman perhatikan pada baris **<?php= \$brg →nama\_brg ?>** yang dimana memerintahkan sistem untuk mengambil data pada *field nama\_brg* yang berada *table tb\_barang*, akan tetapi untuk memanggil data gambar dimana teman – teman perlu mendeskripsikan terlebih dahulu dimana teman – teman menyimpan gambar tersebut dengan menggunakan fungsi ***base\_url***. Karena kita telah memanggil ***foreach*** pada bagian dari *table body* jang lupa untuk menutupkannya dengan *code* **<?php endforeach; ?>** sebelum *table body* ditutup.

Langkah selanjutnya dimana teman – teman perlu memanggil *template footer* di akhir *code*. Perhatikan *source code* 5.21 berikut.

```
</div>
<!-- /.container-fluid -->
<!-- Memanggil Footer -->
<?php $this->load->view("templates/footer.php") ?>
</div>
<!-- /.content-wrapper -->
</div>
<!-- /#wrapper -->

<?php $this->load->view(
"templates(scrolltop.php") ?>
<?php $this->load->view("templates/modal.php") ?>
<?php $this->load->view("templates/js.php") ?>

</body>

</html>
```

*Source Code 5.21 Membuat Footer Pada View v\_barang*

Selamat *table* untuk menampilkan data barang dari *database* pada *file v\_barang* sudah selesai teman – teman buat, dimana *script code full* akan terlihat pada *source code 5.22*.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <?php
        $this->load->view("templates/header.php")
    ?>
</head>
<body id="page-top">

    <?php
        $this->load->view("templates/topbar.php")
    ?>
    <div id="wrapper">

        <?php
            $this->load->view("templates/sidebar.php")
        ?>
<div id="content-wrapper">

    <div class="container-fluid">

        <!-- Menampilkan Data -->
        <div class="card mb-3">
            <div class="card-header">
                <a href="<?php=
                    base_url('barang/tambah')
                ?>">
                    <i class="fas fa-plus"></i>
                    Tambah Data
                </a>
            </div>
            <div class="card-body">
                <div class="table-responsive">
```

```
<table class="table table-hover"
id="dataTable" width="100%" cellspacing="0">
<thead>
<tr>
<th>Nama Barang</th>
<th>Harga</th>
<th>Gambar Barang</th>
<th>Deskripsi Barang</th>
<th>Aksi</th>
</tr>
</thead>

<tbody>
<?php foreach ($barang as $brg): ?>

<tr>
<td width="150">
<?php= $brg->nama_brg ?>
</td>

<td>
<?php= $brg->harga ?>
</td>

<td>

</td>

<td class="small">
<?php= substr($brg->deskripsi_brg, 0, 120) ?>
</td>

<td width="250">
<a href="<?php= base_url('barang/edit/'.
$brg->id) ?>"><i class="fas fa-edit">
</i> Edit </a>

<a onclick="deleteConfirm('<?php=
```

```

base_url('barang/delete/'.$barang-> id) ?>')"
    href="#" class="btn btn-small text-danger">
        <i class="fas fa-trash"></i> Hapus</a>
    </td>
</tr>
<?php endforeach; ?>
</tbody>
</table>
</div>
</div>
</div>
</div>
<!-- /.container-fluid -->
<!-- Memanggil Footer -->
<?php $this->load->view("templates/footer.php") ?>
</div>
<!-- /.content-wrapper -->
</div>
<!--/#wrapper -->

<?php $this->load->view(
"templates(scrolltop.php") ?>
<?php $this->load->view("templates/modal.php") ?>
<?php $this->load->view("templates/js.php") ?>

</body>

</html>

```

*Source Code 5.22 Code Full File v\_barang*

Pada pembahasan sebelumnya dimana teman – teman sudah membuat file view *v\_barang* yang berfungsi untuk menampilkan data barang dan menampungnya kedalam *table* menggunakan *template bootstrap*. Langkah selanjutnya dimana teman – teman perlu membuat *form inputbarang*. *Form* tersebut berfungsi untuk menambahkan barang atau data baru dan menyimpannya kedalam *database*. Untuk membuat *form inputbarang*

dimana kita akan menggunakan kembali *template bootstrap*. Untuk memanggil *template* tersebut silahkan teman – teman gunakan cara yang sama pada saat teman – teman membuat halaman *v\_barang*. Silahkan teman – teman perhatikan kembali *source code 5.5* yang dimana merupakan *source code controllers* dengan *function tambah*. Pada *source code controllers* tersebut dimana terdapat fungsi *form\_validation* yang berguna sebagai *alert*. Agar fungsi pada *library form\_validation* tersebut dapat terpanggil pada *view* dimana teman – teman perlu memanggil *flashdata*-nya, perhatikan *source code 5.23*.

```
<?php
    if ($this->session->flashdata('success')):
?>
    <div class="alert alert-success" role="alert">

<?php=
    $this->session->flashdata('success');
?>

</div>
<?php endif; ?>
```

*Source Code 5.23 Memanggil Flashdata Pada View inputbarang*

Bagaimana caranya apabila *user* tidak jadi melakukan *input* barang ?, mari kita buat sebuah *button* kembali pada sistem kita. Kali kita akan menggunakan kembali *icon* yang terdapat pada *font awesome*. Silahkan teman – teman perhatikan *source code 5.24* berikut.

```
<div class="card mb-3">

<div class="card-header">

<a href=<?php= base_url('barang') ?>>
  <i class="fas fa-arrow-left"></i> Kembali
</a>

</div>
```

Source Code 5.24 Membuat Button Kembali Pada Form inputbarang

Pada *source code* tersebut dimana kita akan menggunakan fungsi *card* pada *bootstrap* dengan *margin button* 3. Perhatikan *script* *<a href* merupakan awal untuk membuat *link* dan ditutup dengan *tag* *</a>*. Buka *tag* PHP untuk menggunakan fungsi *helpers* dan panggil *controllers* “Barang”. Kok tidak pake *function* sih ?, karena kita akan membuat *user* ke halaman *v\_barang* maka cukup memanggil *function default* dari *controllers* yaitu *function index* yang dimana akan melakukan *load view* terhadap *file v\_barang.php*. Agar tampilan pada sistem kita lebih menarik gunakan pemanggilan *icon font awesome* dengan menggunakan *tag* *<i>* dan masukkan *class font awesome* nya lalu tutup dengan *tag* *</i>*, beri keterangan “Kembali” yang merupakan salah satu fungsi desain interaksi.

Langkah selanjutnya dimana teman – teman perlu membuat beberapa *textbox* dengan cara menambahkan *script input*, silahkan teman – teman perhatikan *source code* 5.25.

```
<div class="card-body">

<form action=<?php= base_url('barang/tambah') ?>">
  method="post" enctype="multipart/form-data" >
```

```
<div class="form-group">
    <label for="name">Nama Barang</label>
    <input class="form-control" <?php=
        form_error('nama_brg') ? 'is-invalid':'' ?>" type="text" name="nama_brg" />
    <div class="invalid-feedback">
        <?php echo form_error('nama_brg') ?>
    </div>
</div>

<div class="form-group">
    <label for="name">Harga Barang</label>
    <input class="form-control" <?php=
        form_error('harga') ? 'is-invalid':'' ?>" type="number" name="harga" min="0" />
    <div class="invalid-feedback">
        <?php echo form_error('harga') ?>
    </div>
</div>

<div class="form-group">
    <label for="name">Gambar Barang</label>
    <input class="form-control-file" <?php=
        form_error('gambar') ? 'is-invalid':'' ?>" type="file" name="gambar_brg" />
    <div class="invalid-feedback">
        <?php echo form_error('gambar_brg') ?>
    </div>
</div>

<div class="form-group">
    <label for="name">Deskripsi Barang</label>
    <textarea class="form-control" <?php=
        form_error('deskripsi_brg') ? 'is-invalid':'' ?>" name="deskripsi_brg"></textarea>
    <div class="invalid-feedback">
        <?php echo form_error('deskripsi_brg') ?>
    </div>
</div>
```

```
<input class="btn btn-success" type="submit"
      name="btn" value="Simpan Barang" />
</form>
</div>
```

*Source Code 5.25 Membuat Textbox Pada View inputbarang*

Dalam membuat *form inputbarang* tersebut coba teman – teman perhatikan pada *script* `<form action='<?php= base_url('barang/tambah') ?>' method='post' enctype='multipart/form-data'>` yang dimana merupakan *action* pada *form inputbarang*, silahkan teman – teman arahkan *action* tersebut ke pada *controllers tambah* agar saat *user* mengakses halaman tersebut dapat menjalankan fungsi *models simpan* yang sudah kita panggil pada *controllers tambah*. Pada setiap *inputan* dimana terdapat fungsi *form\_error* yang merupakan salah satu *library* dari *form validation*. Fungsi tersebut akan menampilkan *alert error* terhadap *user* apabila saat melakukan *input user* mengalami kesalahan. Apabila teman – teman telah selesai membuat *view form input barang* dalam *file inputbarang.php* maka *source code full* pada *file inputbarang.php* akan nampak seperti pada *source code 5.26*.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <?php
        $this->load->view("templates/header.php")
    ?>
</head>
<body id="page-top">

    <?php
        $this->load->view("templates/topbar.php")
    ?>
```

```
<div id="wrapper">

    <?php
        $this->load->view("templates/sidebar.php")
    ?>
<div id="content-wrapper">

    <div class="container-fluid">

<?php
    if ($this->session->flashdata('success')):
?>
    <div class="alert alert-success" role="alert">

<?php=
    $this->session->flashdata('success');
?>

</div>
<?php endif; ?>

<div class="card mb-3">

    <div class="card-header">

        <a href=<?php= base_url('barang') ?>">
            <i class="fas fa-arrow-left"></i> Kembali
        </a>

    </div>

    <div class="card-body">

        <form action=<?php= base_url('barang/tambah') ?>">
            method="post" enctype="multipart/form-data" >

<div class="form-group">
    <label for="name">Nama Barang</label>
    <input class="form-control" type="text" value=<?php=
```

```
form_error('nama_brg') ? 'is-invalid':'' ?>"  
    type="text" name="nama_brg" />  
<div class="invalid-feedback">  
    <?php echo form_error('nama_brg') ?>  
</div>  
</div>  
  
<div class="form-group">  
    <label for="name">Harga Barang</label>  
    <input class="form-control" <?php=  
        form_error('harga') ? 'is-invalid':'' ?>"  
        type="number" name="harga" min="0" />  
<div class="invalid-feedback">  
    <?php echo form_error('harga') ?>  
</div>  
</div>  
  
<div class="form-group">  
    <label for="name">Gambar Barang</label>  
    <input class="form-control-file" <?php=  
        form_error('price') ? 'is-invalid':'' ?>"  
        type="file" name="gambar_brg" />  
<div class="invalid-feedback">  
    <?php echo form_error('gambar_brg') ?>  
</div>  
</div>  
  
<div class="form-group">  
    <label for="name">Deskripsi Barang</label>  
    <textarea class="form-control" <?php=  
        form_error('deskripsi_brg') ? 'is-invalid':'' ?>"  
        name="deskripsi_brg"></textarea>  
<div class="invalid-feedback">  
    <?php echo form_error('deskripsi_brg') ?>  
</div>  
</div>  
  
<input class="btn btn-success" type="submit"  
    name="btn" value="Simpan Barang" />  
</form>
```

```

</div>

<!-- /.container-fluid -->
<!-- Memanggil Footer -->
<?php $this->load->view("templates/footer.php") ?>
</div>
<!-- /.content-wrapper -->
</div>
<!-- /#wrapper -->

<?php $this->load->view(
"templates(scrolltop.php") ?>
<?php $this->load->view("templates/modal.php") ?>
<?php $this->load->view("templates/js.php") ?>

</body>

</html>

```

*Source Code 5.26 Code Full View File inputbarang.php*

Pada pembahasan sebelumnya dimana teman – teman sudah belajar membuat *view* untuk menampilkan barang dan juga menambahkan barang. Langkah berikutnya dimana teman – teman membutuh *form* untuk melakukan pengeditan pada barang tersebut. Silahkan teman – teman buka file *edit.php* pada direktori *application/views/barang*. Untuk *form edit* tersebut sebenarnya mirip dengan cara membuat *form input* barang hanya saja yang membedakan adalah dimana teman – teman perlu mengirimkan *id* yang akan di *edit* dan menampilkan *value*-nya. Karena kita akan membuatnya dengan bantuan *bootstrap* dimana teman – teman perlu memanggil kembali *file template* tersebut dengan menggunakan fungsi *helpers “url”*. Untuk pemanggilan *template* dimana terlihat sama dengan cara membuat halaman *view* pada file *v\_barang.php*. Silahkan teman – teman buka kembali *source code 5.17, 5.18, dan 5.21*. Karena kita

menggunakan fungsi *library form validation* dimana teman – teman perlu membuat *flashdata* di dalam fungsi PHP, perhatikan teman – teman perhatikan *source code* 5.27 berikut.

```
<?php if ($this->session->flashdata('success')): ?>
    <div class="alert alert-success" role="alert">
        <?php echo $this->session->flashdata('success'); ?>
    </div>
<?php endif; ?>
```

*Source Code 5.27 Membuat Flashdata Pada View Edit*

Langkah selanjutnya dimana teman – teman perlu memberikan *button kembali* pada *form edit* tersebut. Untuk membuat *button kembali* tersebut silahkan teman – teman perhatikan *source code* 5.28 berikut.

```
<div class="card mb-3">
    <div class="card-header">
        <a href="<?php echo site_url('barang') ?>">
            <i class="fas fa-arrow-left"></i>Kembali
        </a>
    </div>
```

*Source Code 5.28 Membuat Button Kembali Pada View Edit*

*Button* tersebut berfungsi untuk mengembalikan *user* ke halaman *v\_barang.php*, hal tersebut berguna apabila *user* tidak jadi melakukan edit barang tersebut.

Langkah selanjutnya dimana teman – teman perlu membuat *form* tersebut yang terdiri dari beberapa *inputan* dengan memanggil isi dari *table* tersebut menggunakan fungsi *value*, perhatikan *script code* 5.29 berikut.

```
<div class="card-body">

<form action="=base_url('barang/edit') ?&gt;" method="post" enctype="multipart/form-data"&gt;

&lt;?php foreach ($barang as $brg): ?&gt;

    &lt;input type="hidden" name="id" value="<?= $brg-&gt;id?&gt;" /&gt;

    &lt;div class="form-group"&gt;
        &lt;label for="nama_brg"&gt;Nama Barang&lt;/label&gt;
        &lt;input class="form-control" type="text" name="nama_brg" value="<?= $brg-&gt;nama_brg ?&gt;" /&gt;
        &lt;div class="invalid-feedback"&gt;
            &lt;?php= form_error('nama_brg') ?&gt;
        &lt;/div&gt;
    &lt;/div&gt;

    &lt;div class="form-group"&gt;
        &lt;label for="harga"&gt;Harga Barang&lt;/label&gt;
        &lt;input class="form-control" type="number" name="harga" min="0" value="<?= $brg-&gt;harga ?&gt;" /&gt;
        &lt;div class="invalid-feedback"&gt;
            &lt;?php= form_error('price') ?&gt;
        &lt;/div&gt;
    &lt;/div&gt;

    &lt;div class="form-group"&gt;
        &lt;label for="gambar_brg"&gt;Gambar Barang&lt;/label&gt;
        &lt;input class="form-control-file" type="file" name="image" /&gt;
        &lt;?php= form_error('gambar_brg') ?&gt;
        &lt;?php= form_error('image') ?&gt;
    &lt;/div&gt;
&lt;/div&gt;</pre
```

```

<input type="hidden" name="old_image"
       value="<?php= $brg->gambar_brg ?>" />
<div class="invalid-feedback">
    <?php= form_error('gambar_brg') ?>
</div>
</div>

<div class="form-group">
    <label for="deskripsi_brg">Deskripsi Barang
    </label>
    <textarea class="form-control"
              <?php= form_error('deskripsi_brg') ?
              'is-invalid': '' ?>
              name="deskripsi_brg">
        <?php= $brg->deskripsi_brg ?>
    </textarea>
    <div class="invalid-feedback">
        <?php= form_error('deskripsi_brg') ?>
    </div>
</div>

<input class="btn btn-success" type="submit"
       name="btn" value="Edit Barang" />

<?php endforeach; ?>

</form>
</div>

```

*Source Code 5.29 Membuat Form Input Dengan Value Pada Edit*

Apabila teman – teman sudah membuat *form edit* tersebut langkah terakhir silahkan teman – teman panggil *template footer*. *Source Code* pada file *edit.php* secara *full* akan namapak seperti pada *source code 5.30*.

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <?php
$this->load->view("templates/header.php")
?>
    </head>

    <body id="page-top">

        <?php
$this->load->view("templates/topbar.php")
?>
        <div id="wrapper">

            <?php
$this->load->view("templates/sidebar.php")
?>

<?php if ($this->session->flashdata('success')): ?>
    <div class="alert alert-success" role="alert">
<?php echo $this->session->flashdata('success'); ?>
    </div>
    <?php endif; ?>

            <div class="card mb-3">
                <div class="card-header">
<a href="<?php echo site_url('barang') ?>">
    <i class="fas fa-arrow-left"></i>Kembali
    </a>
                </div>

                <div class="card-body">

<form action="<?php= base_url('barang/edit') ?>" method="post" enctype="multipart/form-data">

    <?php foreach ($barang as $brg): ?>
```

```
<input type="hidden" name="id"
      value="<?php= $brg->id?>" />

      <div class="form-group">
<label for="nama_brg">Nama Barang</label>
      <input class="form-control"
<?php echo form_error('nama_brg') ?
      'is-invalid': '' ?>" type="text" name="nama_brg"
      value="<?php= $brg->nama_brg ?>" />
      <div class="invalid-feedback">
<?php= form_error('nama_brg') ?>
      </div>
      </div>

      <div class="form-group">
<label for="harga">Harga Barang</label>
      <input class="form-control"
<?php= form_error('harga') ?
      'is-invalid': '' ?>" type="number" name="harga" min="0"
      value="<?php= $brg->harga ?>" />
      <div class="invalid-feedback">
<?php= form_error('price') ?>
      </div>
      </div>

      <div class="form-group">
<label for="gambar_brg">Gambar Barang</label>
      <input class="form-control-file"
<?php= form_error('gambar_brg') ?
      'is-invalid': '' ?>" type="file" name="image" />
<input type="hidden" name="old_image"
      value="<?php= $brg->gambar_brg ?>" />
      <div class="invalid-feedback">
<?php= form_error('gambar_brg') ?>
      </div>
      </div>
```

```

        <div class="form-group">
<label for="deskripsi_brg">Deskripsi Barang
    </label>
    <textarea class="form-control"
<?php= form_error('deskripsi_brg') ?
    'is-invalid':'' ?>
        name="deskripsi_brg">
<?php= $brg->deskripsi_brg ?>
    </textarea>
    <div class="invalid-feedback">
<?php= form_error('deskripsi_brg') ?>
    </div>
    </div>

<input class="btn btn-success" type="submit"
    name="btn" value="Edit Barang" />

    <?php endforeach; ?>

    </form>
    </div>

    <!-- /.container-fluid -->
    <!-- Memanggil Footer -->
<?php $this->load->view("templates/footer.php") ?>
    </div>
    <!-- /.content-wrapper -->
    </div>
    <!-- #wrapper -->

    <?php $this->load->view(
        "templates(scrolltop.php") ?>
<?php $this->load->view("templates/modal.php") ?>
<?php $this->load->view("templates/js.php") ?>

    </body>

    </html>

```

*Source Code 5.30 Code Full File Edit.php*

## **BAB VI**

### **E-COMMERCE DAN CODEIGNITER**

Seperti yang sudah kita bahas pada bab pendahuluan dimana *e-commerce* sangatlah penting di dunia RI 4.0 ini. Banyak pengusaha yang sudah menerapkan konsep *e-commerce* guna memperbesar daya saing mereka dalam dunia bisnis. *E-Commerce* sangatlah membantu baik bagi pelanggan maupun bagi penjual atau pengusaha. Dengan menerapkan *e-commerce* dimana adanya keuntungan – keuntungan yang dapat diambil oleh kedua belah pihak, contohnya bagi pelanggan dimana dapat mempermudah dalam proses transaksi dan pencarian suatu produk dan bagi penjual dimana dapat membantu proses penjualan seperti melakukan promosi dan lain sebagainya.

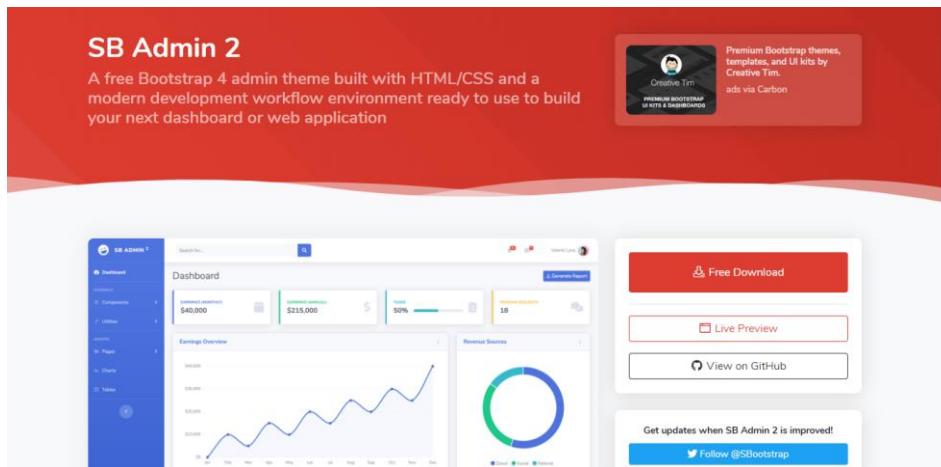
Apakah teman – teman tertarik belajar membangun sebuah *website* dengan konsep *e-commerce* ?, banyak sekali *website* – *website* yang bertebaran dan dibuat dengan *framework codeigniter*. Sistem JURAGAN adalah salah satunya yang menerapkan konsep *e-commerce* dan dibangun dengan menggunakan *framework codeigniter*. Apa saja sih yang perlu dipelajari untuk membangun sebuah *website* dengan konsep *e-commerce* menggunakan *framework codeigniter* ?, mari kita belajar pada sub – sub bab berikut.

#### **6.1 Mengubungkan Bootstrap Dengan CI**

*Bootstrap* merupakan kerangkan atau *library framework* CSS yang khusus dibuat untuk pengembangan *font end* pada sebuah *website*. CSS merupakan sebuah *template* pada sebuah *website*, yang dimana dengan adanya CSS dapat memperindah tampilan dari *website* yang akan kita bangun. Penampilan merupakan kunci utama dalam membangun sebuah

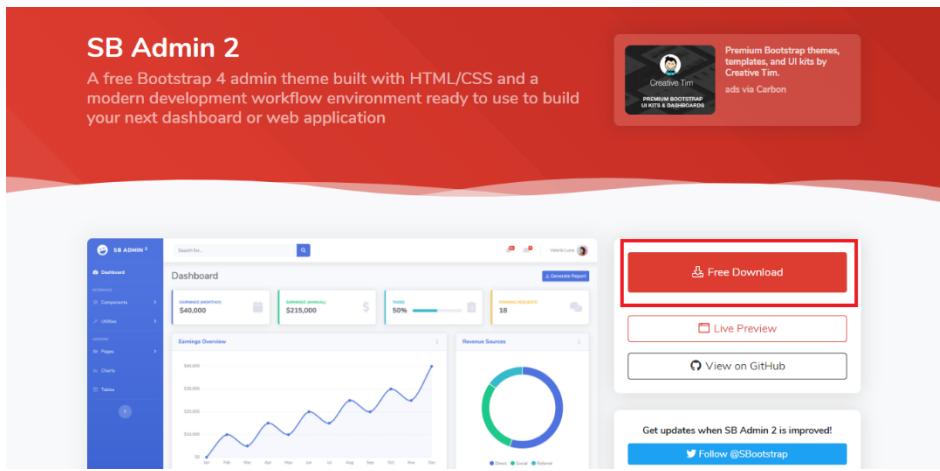
*website* dengan konsep *e-commerce*. Dengan tampilan yang menarik dan mudah digunakan dimana *user* akan terlihat lebih nyaman dan betah untuk mengakses dan menggunakan *website* tersebut. Dengan adanya *bootstrap* tersebut dimana dapat mempermudah dan mempercepat para developers dalam melakukan pengembangan terhadap suatu *website*.

Dalam sub bab berikut ini dimana teman – teman akan mempelajari mengenai *template* dan tentunya menggunakan *bootstrap*. Langkah pertama dalam melakukan pembelajaran pada sub bab ini dimana teman – teman perlu men-*download* salah satu *template bootstrap*. Penulis menyarankan teman – teman menggunakan *template SB ADMIN 2*, mengapa ?, karena *template* tersebut sudah menggunakan *bootstrap* versi 4. Untuk men-*download* *template* tersebut silahkan teman – teman kunjungi *link* <https://startbootstrap.com/themes/sb-admin-2/>. Teman – teman akan di bawa ke halaman seperti pada gambar 6.1.



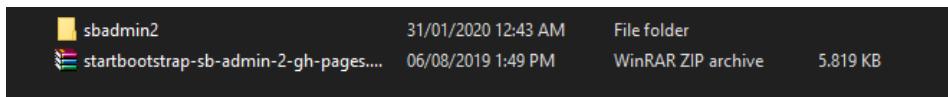
*Gambar 6.1 Halaman Start Bootstrap SB ADMIN 2*

Silahkan teman – teman *download template bootstrap* SB ADMIN 2 tersebut dengan memilih menu *download* seperti pada gambar 6.2.



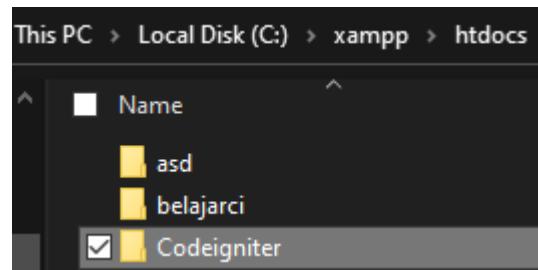
Gambar 6.2 Menu Download Template Bootstrap SB ADMIN 2

Sistem akan melakukan proses *download*, tunggu beberapa menit hingga proses *download* selesai. Hasil *download* tersebut berupa *file* dengan bentuk zip. Silahkan teman – teman lakukan *extract* terlebih dahulu dan *rename* menjadi sadmin2.



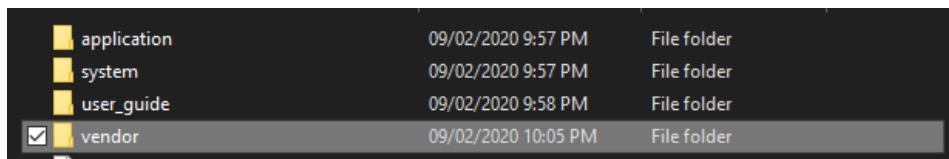
Gambar 6.3 Meng Extract File Zip Boostrap

Apabila teman – teman sudah selesai melakukan *download* terhadap *template bootstrap* SB ADMIN 2 tersebut tentunya teman – teman memerlukan *file codeigniter*. Disini saya sudah memiliki *file codeigniter* yang masih kosong dan sudah berapda pada direktori xampp/htdocs.



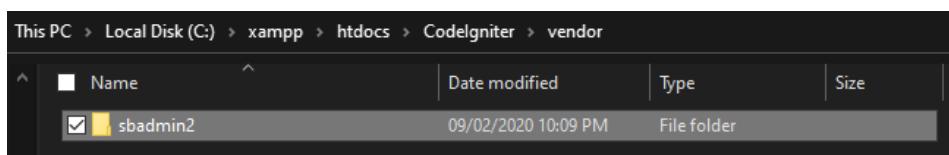
Gambar 6.4 File Codeigniter Pada Direktori Xampp/htdocs

Silahkan teman – teman buat *folder* dengan nama *vendor* terlebih dahulu pada *file codeigniter* teman – teman seperti pada gambar 6.5.



Gambar 6.5 Membuat Folder Vendor

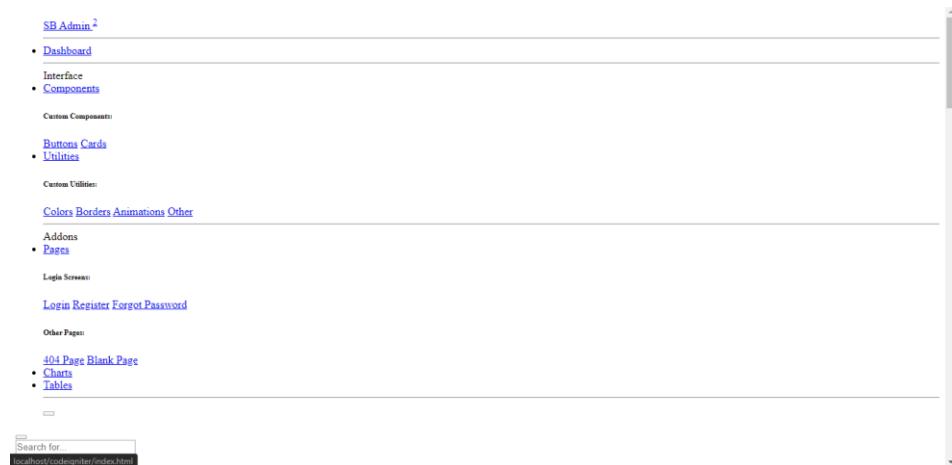
Folder *vendor* tersebut berfungsi untuk menampung referensi *template bootstrap* SB ADMIN 2 tersebut, silahkan teman – teman pindahkan *file* SB ADMIN 2 tersebut kedalam *folder vendor* seperti pada gambar 6.6.



Gambar 6.6 Memindahkan File SB ADMIN 2 Ke Dalam Vendor

Untuk mencoba *template bootstrap* SB ADMIN 2 tersebut gimana kalau kita rubah halaman *welcome* pada *framework codigniter* tersebut. Pada direktori *vendor/sbadmin2* silahkan teman – teman buka *file* dengan nama *blank.html* dan *copy* semua *script code* tersebut lalu pindahkan ke dalam

*file welcome\_message* yang berada pada direktori *application/views*. Apabila sudah dipindahkan jalankan *controllers welcome* dan lihat hasilnya.



Gambar 6.7 Tampilan View Welcome\_message.php

Apabila tampilan *welocome\_message.php* teman – teman berubah menjadi seperti pada gambar 6.7, menandakan bahwa *template bootstrap* teman – teman sudah dapat digunakan, namun *template css* pada *bootstrap* tersebut belum terpanggil. Bagaimana cara memanggil *template css* tersebut ?, silahkan ikuti pembalajaran lebih lanjut.

Langkah selanjutnya dimana teman – teman perlu membuat *folder* baru pada *file codeigniter* dan beri nama menjadi *assets*, seperti pada gambar 6.8 berikut.

Name	Date modified	Type	Size
application	09/02/2020 9:57 PM	File folder	
<input checked="" type="checkbox"/> assets	09/02/2020 10:25 PM	File folder	
system	09/02/2020 9:57 PM	File folder	
user_guide	09/02/2020 9:58 PM	File folder	
vendor	09/02/2020 10:09 PM	File folder	

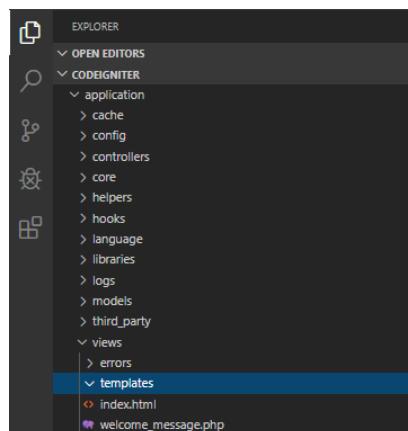
Gambar 6.8 Membuat Folder Assets

Folder *assets* tersebut berfungsi untuk menampung beberapa fungsi dari *templates bootstrap* SB ADMIN 2 tersebut, seperti *css*, *js*, dan *jquery*. Silahkan teman – teman pindahkan *css*, *js*, *img* dan *vendor* yang terdapat pada *file* SB ADMIN 2 tersebut kedalam *folder assets*, seperti pada gambar 6.9.

Name	Date modified	Type	Size
css	09/02/2020 10:30 PM	File folder	
img	09/02/2020 10:30 PM	File folder	
js	09/02/2020 10:30 PM	File folder	
vendor	09/02/2020 10:30 PM	File folder	

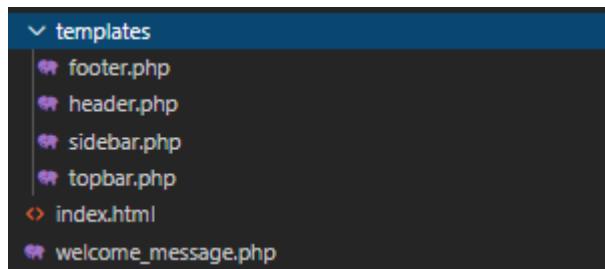
Gambar 6.9 Mengisi Folder Assets

Langkah selanjutnya dimana teman – teman perlu memisahkan bagian – bagian *template* tersebut menjadi satu *file* terpisah, hal ini dilakukan agar *template* yang kita buat dapat dipergunakan oleh berbagai *file* dan untuk mempermudah dalam proses pengeditan. Silahkan teman – teman buat *folder* baru pada direktori *application/views* dan beri nama *templates* untuk menampung bagian – bagian *template* tersebut.



Gambar 6.10 Membuat Folder templates Pada Views

Dalam *folder templates* tersebut silahkan teman – teman buat empat *file php* dengan nama *header.php*, *sidebar.php*, *topbar.php* dan *footer.php*, seperti pada gambar 6.11 berikut.



Gambar 6.11 Membuat 4 File PHP Pada Folder Templates

Setelah teman – teman sudah selesai mempersiapkan *folder – folder* untuk menampung *templates bootstrap* SB ADMIN 2 tersebut langkah selanjutnya kita akan belajar bagaimana caranya menghubungkan *css* tersebut dengan *framework codeigniter*. *Template* yang pertama adalah bagian *header*, yang dimana merupakan bagian awal pada suatu sistem dan biasanya berisi dengan judul atau *tittle* dari sistem tersebut. Silahkan teman – teman buka kembali *file welcome\_message.php* yang sudah diberikan *script code blank.html*, mari kita pisahkan terlebih dahulu bagian – bagian *template* tersebut. Silahkan teman – teman *cut* baris 1 – 26 pada *file welcome\_message.php* tersebut yang merupakan bagian *header* kedalam *file header.php*.

```
header.php x
application > views > templates > header.php > html > body#page-top > div#wrapper
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5
6    <meta charset="utf-8">
7    <meta http-equiv="X-UA-Compatible" content="IE=edge">
8    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9    <meta name="description" content="">
10   <meta name="author" content="">
11
12  <title>SB Admin 2 - Blank</title>
13
14  <!-- Custom fonts for this template-->
15  <link href="vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">
16  <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">
17
18  <!-- Custom styles for this template-->
19  <link href="css/sb-admin-2.min.css" rel="stylesheet">
20
21 </head>
22
23 <body id="page-top">
24
25  <!-- Page Wrapper -->
26  <div id="wrapper">
```

Gambar 6.12 Isi File Header.php

*Sidebar* merupakan bagian *template* kedua pada sebuah sistem yang dimana biasanya terletak pada sisi sebelah kiri sistem. *Sidebar* biasanya diisi oleh beberapa kolom yang dimana mengandung fungsi *link navigasi*. Silahkan teman – teman *cut code* baris 28 - 140 pada file *welcome\_message.php* yang merupakan bagian *sidebar* kedalam file *sidebar.php*.

```
<!-- Sidebar -->
<ul class="navbar-nav bg-gradient-primary sidebar sidebar-dark accordion" id="accordionSidebar">
  <!-- Sidebar - Brand -->
  <a class="sidebar-brand d-flex align-items-center justify-content-center" href="index.html">
    <div class="sidebar-brand-icon rotate-n-15">
      <i class="fas fa-laugh-wink"></i>
    </div>
    <div class="sidebar-brand-text mx-3">SB Admin <sup>2</sup></div>
  </a>
  <!-- Divider -->
  <hr class="sidebar-divider my-0">
  <!-- Nav Item - Dashboard -->
  <li class="nav-item">
    <a class="nav-link" href="index.html">
      <i class="fas fa-fw fa-tachometer-alt"></i>
      <span>Dashboard</span>
    </a>
  </li>
  <!-- Divider -->
  <hr class="sidebar-divider">
  <!-- Heading -->
  <div class="sidebar-heading">
    Interface
  </div>
  <!-- Nav Item - Pages Collapse Menu -->
  <li class="nav-item">
    <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseTwo" aria-expanded="true" aria-controls="collapseTwo">
      <i class="fas fa-fw fa-cog"></i>
      <span>Components</span>
    </a>
    <div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-parent="#accordionSidebar">
      <div class="bg-white py-2 collapse-inner rounded">
        <h6 class="collapse-header">Custom Components:</h6>
        <a class="collapse-item" href="buttons.html">Buttons</a>
        <a class="collapse-item" href="cards.html">Cards</a>
      </div>
    </div>
  </li>
</ul>
```

Gambar 6.13 Isi File Sidebar.php

*Topbar* merupakan bagian ketiga pada *template* tersebut dimana biasanya terletak pada posisi paling atas dari sebuah sistem dan berikan fungsi – fungsi seperti *form* pecarian salah satu contohnya. Silahkan teman – teman *cut code* baris 142 – 329 pada *file welcome\_message.php* yang merupakan bagian *topbar* kedalam *file topbar.php*

```

⌚ topbar.php ×
application > views > templates > ⌚ topbar.php > ⌘ div#content-wrapper.d-flex.flex-column > ⌘ div#content
1   <!-- Content Wrapper -->
2   <div id="content-wrapper" class="d-flex flex-column">
3
4   <!-- Main Content -->
5   <div id="content">
6
7   <!-- Topbar -->
8   <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-top shadow">
9
10   <!-- Sidebar Toggle (Topbar) -->
11   <button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-circle mr-3">
12   | <i class="fa fa-bars"></i>
13   </button>
14
15   <!-- Topbar Search -->
16   <form class="d-none d-sm-inline-block form-inline mr-auto ml-md-3 my-2 my-md-0 mw-100 navbar-search">
17   | <div class="input-group">
18   | | <input type="text" class="form-control bg-light border-0 small" placeholder="Search for..." aria-label="Search" aria-describedby="basic-addon2">
19   | | <div class="input-group-append">
20   | | | <button class="btn btn-primary" type="button">
21   | | | | <i class="fas fa-search fa-sm"></i>
22   | | </button>
23   | </div>
24   | </div>
25   </form>
26
27   <!-- Topbar Navbar -->
28   <ul class="navbar-nav ml-auto">
29
30   <!-- Nav Item - Search Dropdown (Visible Only XS) -->
31   <li class="nav-item dropdown no-arrow d-sm-none">
32   | <a class="nav-link dropdown-toggle" href="#" id="searchDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
33   | | <i class="fas fa-search fa-fw"></i>
34   | </a>
35   <!-- Dropdown - Messages -->
36   <div class="dropdown-menu dropdown-menu-right p-3 shadow animated--grow-in" aria-labelledby="searchDropdown">
37   | <form class="form-inline mr-auto w-100 navbar-search">
38   | | <div class="input-group">
39   | | | <input type="text" class="form-control bg-light border-0 small" placeholder="Search for..." aria-label="Search">
40   | | | <div class="input-group-append">
41   | | | | <button class="btn btn-primary" type="button">
```

*Gambar 6.14 Isi File Topbar.php*

*Footer* merupakan bagian keempat pada *templates* tersebut yang dimana biasanya terletak pada bagian bawah sistem. Pada *template bootstrap* SB ADMIN 2 tersebut dimana berisi tentang *copyright* dari suatu sistem. Silahkan teman – teman *cut code* pada baris 331 – 395 pada *file welcome\_mesaage.php* yang merupakan bagian *footer* dealam *file footer.php*.

```
application > views > templates > footer.php > div#logoutModal.modal.fade > div.modal-dialog > div.modal-content > div.modal-header > button.close
1  <!-- Footer -->
2  <footer class="sticky-footer bg-white">
3      <div class="container my-auto">
4          <div class="copyright text-center my-auto">
5              | <span>Copyright © Your Website 2019</span>
6          </div>
7      </div>
8  </footer>
9  <!-- End of Footer -->
10 </div>
11 <!-- End of Content Wrapper -->
12 </div>
13 <!-- End of Page Wrapper -->
14
15 <!-- Scroll to Top Button-->
16 <a class="scroll-to-top rounded" href="#page-top">
17     | <i class="fas fa-angle-up"></i>
18 </a>
19
20 <!-- Logout Modal-->
21 <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
22     <div class="modal-dialog" role="document">
23         <div class="modal-content">
24             <div class="modal-header">
25                 <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
26                 | <button class="close" type="button" data-dismiss="modal" aria-label="Close">
27                     | <span aria-hidden="true">×</span>
28                 </button>
29             </div>
30             <div class="modal-body">Select "Logout" below if you are ready to end your current session.</div>
31             <div class="modal-footer">
32                 <button class="btn btn-secondary" type="button" data-dismiss="modal">Cancel</button>
33                 | <a class="btn btn-primary" href="login.html">Logout</a>
34             </div>
35         </div>
36     </div>
37 </div>
38 </div>
39 <!-- Bootstrap core JavaScript-->
```

Gambar 6.15 Isi File Footer.php

Apabila semua *file* sudah dipisahkan dan teman – teman jalankan kembali *file* welcome\_mesaage.php tersebut maka akan semua *template* akan menghilang. Untuk memunculkannya dimana teman – teman perlu merubah *file controllers* welcome menjadi seperti pada *source code 6.1*.

```
public function index()
{
    $this->load->view('templates/header');
    $this->load->view('templates/sidebar');
    $this->load->view('templates/topbar');
    $this->load->view('welcome_message');
    $this->load->view('templates/footer');
```

Source Code 6.1 Merubah Controllers Welcome

Coba teman – teman perhatikan *source code* 6.1 tersebut yang dimana akan melakukan *load* terhadap *view* dan memanggil *file* pada *folder templates*. Apabila teman – teman jalankan *controllers* tersebut maka *css* pada *bootstrap* tersebut masih belum muncul, hal ini dikarenakan kita belum memanggil *file css* tersebut pada setiap *templates*. Untuk memanggil *file templates* tersebut silahkan teman – teman buka *file header.php* dan tambahkan sedikit *code* seperti pada *source code* 6.2 berikut.

```
!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>SB Admin 2 - Blank</title>

    <!-- Custom fonts for this template-->
    <link href="= base_url(); ?assets/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">
    <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

    <!-- Custom styles for this template-->
    <link href="= base_url(); ?assets/css/sb-admin-2.min.css" rel="stylesheet">
```

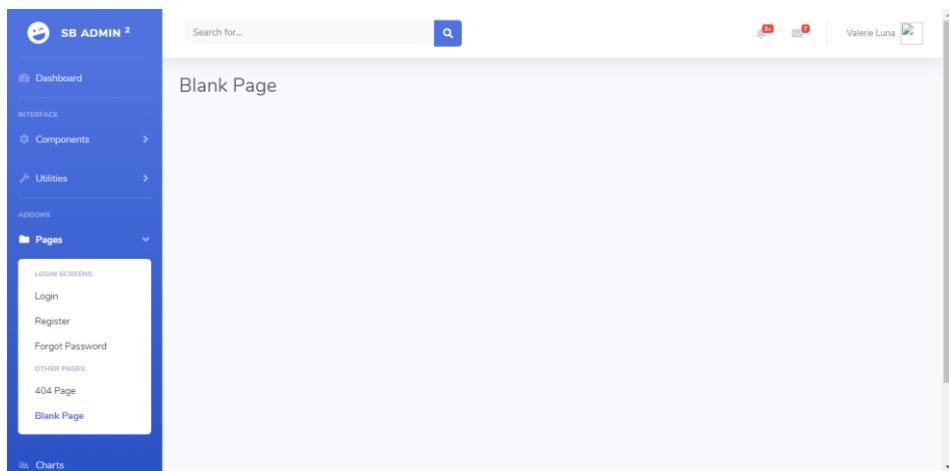
```
</head>

<body id="page-top">

    <!-- Page Wrapper -->
    <div id="wrapper">
```

Source Code 6.2 Merubah File Header.php Pada Controllers Welcome

Dimana teman – teman perlu menambahkan fungsi *helper “url”* untuk memanggil *file css* tersebut. Script code `<?= base_url(); ?>assets/` berguna untuk memanggil fungsi – fungsi yang berada pada *folder assets*. Silahkan teman – teman jalankan *file controllers* tersebut pada *web browser* teman – teman.



Gambar 6.16 Perubahan Sidebar Dan Topbar

Selamat dimana teman – teman sudah berhasil memunculkan bagian *sidebar* dan juga *topbar*, namun bagian *footer* masih belum muncul. Silahkan teman – teman buka *file footer.php* dan edit *code* tersebut menjadi seperti *source code* 6.3.

```
<!-- Bootstrap core JavaScript-->
<script src="= base_url(); ?&gt;assets/vendor/jquery/jquery.min.js"&gt;&lt;/script&gt;
&lt;script src="<?= base_url(); ?&gt;assets/vendor/bootstrap/js/bootstrap.bundle.min.js"&gt;&lt;/script&gt;

&lt;!-- Core plugin JavaScript--&gt;
&lt;script src="<?= base_url(); ?&gt;assets/vendor/jquery-easing/jquery.easing.min.js"&gt;&lt;/script&gt;

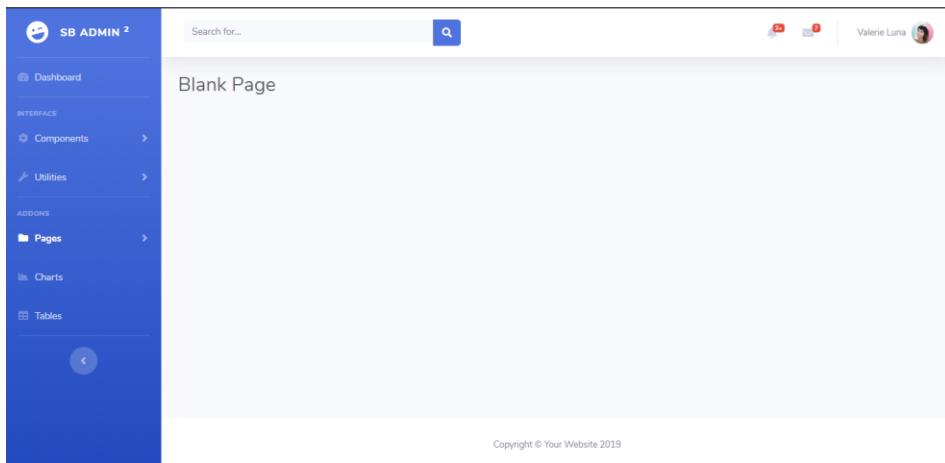
&lt;!-- Custom scripts for all pages--&gt;
&lt;script src="<?= base_url(); ?&gt;assets/js/sb-admin-2.min.js"&gt;&lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;</pre
```

Source Code 6.3 Merubah Code Footer.php

Pada file *footer.php* tersebut dimana teman – teman perlu memanggil fungsi *helpers “url”* pada *JavaScript* yang dimana fungsi java tersebut dapat digunakan. Selamat teman – teman sudah berhasil menghubungkan *template bootstrap* pada *framework codeigniter*, untuk melihat perubahannya silahkan teman – teman jalankan *file controllers welcome* tersebut.



Gambar 6.17 Menerapkan Templates Bootstrap Terhadap CI Berhasil

## 5.2 Fungsi Form Validation

Pernah gak sih kalian menemukan pesan *error* atau *success* saat melakukan *input* data ? Atau mendapatkan pesan harus melakukan *login* terlebih dahulu sebelum mengakses sistem lebih luas ?. Hal – hal yang teman – teman temukan berupa *alert – alert* seperti itu merupakan salah satu contoh dari fungsi *form validation*. Jadi apa sih *form validation* itu ?, *Form Validations* merupakan sebuah notifikasi atau pemberitahuan informasi yang digunakan oleh *form* yang sudah diberikan *rules*.

*Form Validations* sangatlah penting bagi kalian jika ingin membangun sebuah *website* dengan konsep *e-commerce*, hal ini diperlukan untuk memberikan beberapa aturan terhadap *website* teman – teman yang dibangun. Contohnya apabila seorang *user* ingin melakukan *transaksi* terhadap produk yang dibeli namun dalam kondisi belum melakukan *login*, hal seperti itu sudah jelas tidak dapat dilakukan bukan ?, sistem akan memperingati atau memberikan informasi kepada *user* tersebut harus

melakukan *login* terlebih dahulu sebelum melakukan transaksi. Hal seperti itu merupakan salah satu contoh dari fungsi *form validation*.

The screenshot shows a login form titled "Halaman Login". At the top, there is an orange error message box containing the text "Silaikan login terlebih dahulu !". Below the message are two input fields: one for email with the placeholder "Ketikkan Email Anda ..." and one for password with the placeholder "Ketikkan Password Anda ...". There is also a "Remember Account" checkbox labeled "Ingat Akun". A large blue "Login" button is at the bottom. Below the button, there are links for account creation ("Daftarkan Toko Anda ? Click Disini !") and password recovery ("Lupa Password ? Click Disini! | Kembali Ke Halaman Utama, Click Disini !").

Gambar 6.18 Contoh Form Validation Pada Sistem JURAGAN

Lalu bagaimana sih cara membuat fungsi *form validation* ?, pada pembelajaran kali ini dimana teman – teman akan belajar mengenai cara membuat fungsi *form validation* terhadap *form* pendaftaran akun. Disini saya akan memberikan menggunakan sistem JURAGAN. Hal pertama yang perlu teman – teman lakukan adalah memanggil *library form validation* tersebut pada *controllers* teman – teman dengan membuat *function \_\_construct*. perhatikan *source code* 6.4 berikut.

```
<?php  
defined('BASEPATH') or exit('No direct script access  
allowed');  
  
class Auth extends CI_Controller  
{  
    public function __construct()  
    {  
        parent::__construct();  
        $this->load->library('form_validation');  
    }  
}
```

*Source Code 6.4 Memanggil Library Form Validation*

Pada *source code* 6.4 tersebut dimana saya mempunya *controllers* yang dengan nama “Auth”. Pada *controllers* tersebut dimana saya telah membuat *function* baru dengan nama *\_\_construct*. *Function* tersebut merupakan *function* pertama yang akan dijalankan oleh *controllers* “Auth” saat *controllers* tersebut diakses. Pada *function* tersebut dimana saya melakukan *load library form\_validation* dengan mengetikkan *script* *\$this->load->library('form\_validation');*.

Pada sistem JURAGAN dimana saya mempunya halaman pendaftaran akun bagi kedua *user* baik pelanggan dan penjual. Untuk menjalankan fungsi *form validation* tersebut dimana kita membutuhkan sebuah *rules* atau aturan. Coba perhatikan *source code* 6.5 berikut.

```
public function registrasi() {  
    $data['title'] = 'Halaman Registrasi';  
    $this->form_validation->  
        set_rules('name', 'Name', 'required|trim');  
}
```

*Source Code 6.5 Penerapan Form Validation Pada Field Name*

Pada *source code* 6.5 tersebut dimana saya telah memberikan dua buah *rules* pada *field* “*name*” yang dimana tempat untuk menampung nama *user*. Dimana *rules* yang saya berikan adalah *required* atau nama tidak boleh kosong dan bersifat *trim*. Apabila kita praktekan terhadap sistem maka akan terlihat seperti pada gambar 6.19 yang dimana sistem akan memunculkan notifikasi atau informasi pemberitahuan terhadap *rules* yang sudah diberikan.

The screenshot shows a registration form titled "Daftarkan Akun Anda!". It has four input fields: "Nama Lengkap ...", "Email ...", "Password Anda ...", and "Ulangi Password Anda ...". The "Nama Lengkap ..." field contains an empty value and has a red error message below it: "The Name field is required.". The "Email ..." field contains the value "yusniarnss33@gmail.com". The "Password Anda ..." and "Ulangi Password Anda ..." fields are empty. A large blue "Daftar Akun" button is at the bottom. Below the button, there are two links: "Lupa Password? Click Disini!" and "Sudah Punya Akun? Silahkan Login!".

*Gambar 6.19 Praktek Form Validation Pada Field Name*

Langkah selanjutnya dimana saya akan memberikan *rules* terhadap *field* email. Perhatikan *source code* 6.6 yang merupakan *rules* pada *field* emial adalah sebagai berikut.

```
$this->form_validation->  
    set_rules('email', 'Email', 'required|trim|  
    valid_email|is_unique[user.email]',  
    [  
        'is_unique' => 'Email Sudah Terdaftar!'  
    ]);
```

Source Code 6.6 Penerapan Form Validation Pada Field Email

Dimana *rules* yang diberikan pada *field email* tersebut merupakan **required** yang artinya wajib untuk di isi. Terdapat *rules valid\_email* yang dimana apakah *format* yang diberikan wajib berbentuk *email*. *Rules* selanjutnya merupakan **is\_unique** yang dimana *email* yang sudah di daftarkan tidak dapat dipergunakan kembali.

The screenshot shows a registration form titled "Daftarkan Akun Anda!". It has two input fields: one for name ("Yusniar Nur Syarif Sidiq") and one for email ("yusniar33@gmail.com"). Below the email field, a red error message "Email Sudah Terdaftar!" is displayed. There are two password input fields labeled "Password Anda ..." and "Ulangi Password Anda ...". A large blue "Daftar Akun" button is at the bottom. At the bottom of the page, there are links for password recovery ("Lupa Password? Click Disini!") and logging in ("Sudah Punya Akun? Silahkan Login!").

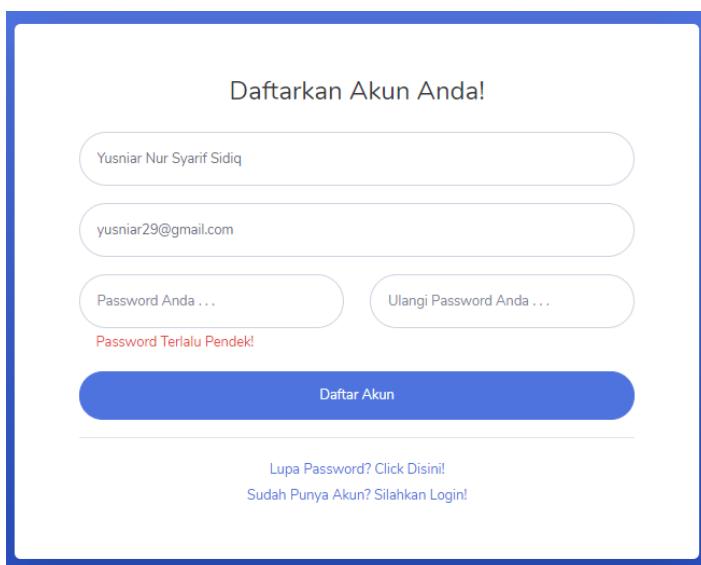
Gambar 6.20 Praktek Form Validation Terhadap Field Email

Langkah selanjutnya dimana terdapat *field* dengan nama *password* yang berfungsi untuk menampung data *password*. Perhatikan *source code* 6.7 berikut yang merupakan pembuatan *rules* terhadap *field password*.

```
$this->form_validation->  
set_rules('password1', 'Password',  
'required|trim|min_length[6]|matches[password2]',  
[  
    'matches' => 'Password Tidak Cocok!',  
    'min_length' => 'Password Terlalu Pendek!'  
]);
```

*Source Code 6.7 Penerapan Form Validation Pada Field Password*

Dimana *rules* yang diberikan adalah *required* yang merupakan bahwa *field* tersebut tidak boleh kosong. Untuk mengatur jumlah karakter pada *password* tersebut gunakan *rules min\_length[ ]*. Biasanya *password* terdapat dua *form input* dan untuk memberikan *password* 1 dan 2 wajib sama berikan *rules matches*.



*Gambar 6.21 Praktek Form Validation Pada Field Password*

Setelah memberikan *rules* pada tiap – tiap *field* langkah selanjutnya adalah menjalankan fungsi *form validation* tersebut pada *controllers* dengan cara menambahkan fungsi *if*. Perhatikan *source code* 6.8 yang merupakan contoh bagaimana menjalankan *source code form validation*.

```
public function registrasi()
{
    $data['title'] = 'Halaman Registrasi';
    $this->form_validation->
        set_rules('name', 'Name', 'required|trim');

    $this->form_validation->
        set_rules('email', 'Email', 'required|trim|
        valid_email|is_unique[user.email]',
        [
            'is_unique' => 'Email Sudah Terdaftar!'
        ]);

    $this->form_validation->
        set_rules('password1', 'Password',
        'required|trim|min_length[6]|
        matches[password2]',
        [
            'matches' => 'Password Tidak Cocok!',
            'min_length' => 'Password Terlalu Pendek!'
        ]);

    if ($this->form_validation->run() == false) {

        $this->load->view('auth/auth_header', $data);
        $this->load->view('auth/registrasi');
        $this->load->view('auth/auth_footer');

    } else {

        $email = $this->input->post('email', true);
        $data = [
            'name' => htmlspecialchars(
```

```

        $this->input->post('name', true),
        'email' => htmlspecialchars($email),
        'password' => password_hash(
            $this->input->post('password1'),
            PASSWORD_DEFAULT)
    ];

    $this->db->insert('user', $data);

    $this->session->set_flashdata('message',
        '<div class="alert alert-success" role="alert">
            Selamat akun anda telah terbuat!</div>');
}

}

```

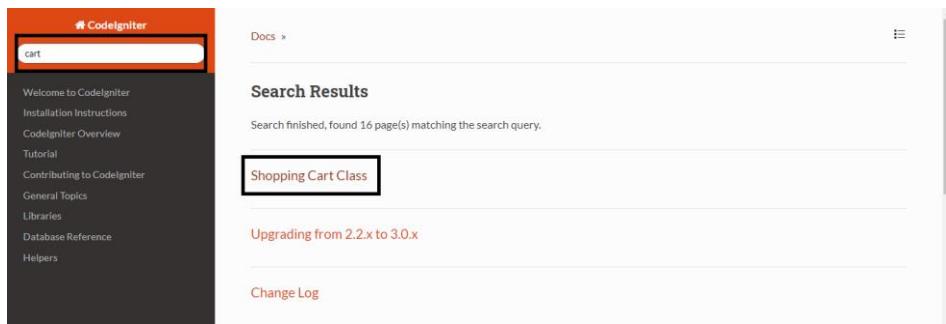
*Source Code 6.8 Contoh Pemanggilan Fungsi Form Validation*

### 5.3 Membuat Fitur Keranjang Belanja

*Shopping cart* atau bisa juga disebut dengan keranjang belanja yang dimana biasanya sering ditemukan pada *website – website e-commerce*. Keranjang belanja sangatlah berperan penting pada sistem *e-commerce* yang dimana memiliki *fungsi* untuk menampung *item – item* atau produk – produk yang akan *user* beli.

Dalam *framework codeigniter* dimana dokumentasi yang diberikan sangatlah lengkap dengan adanya *user guide*. Dalam *framework codeigniter* memiliki *library cart* yang dimana berfungsi untuk membuat fitur menambahkan sebuah *item* kedalam keranjang. Sebelum ke bagian praktikumnya mari kita bahas terlebih dahulu bagaimana cara menggunakan *library cart* tersebut. Silahkan teman – teman buka terlebih dahulu dokumentasi *user guide* pada *web site resmi codeigniter*. Pada kolol

pencarian silahkan teman – teman ketik *cart* maka akan muncul beberapa dokumentasi, pilih *shopping cart class*.



Gambar 6.22 Mengakses Dokumentasi Shopping Cart

Pada dokumentasi *shopping cart* tersebut dimana teman – teman diberikan langkah – langkah bagaimana cara menggunakan *library shopping cart* tersebut.

Pada langkah pertama dimana teman – teman akan belajar bagaimana cara menambahkan *item* dengan menggunakan bantuan *library shopping cart* tersebut. Silahkan teman – teman scroll kebawah dimana teman – teman akan menemukan “*Adding an Item to The Cart*” yang dimana terdapat *source code* 6.9 sebagai berikut.

```
$data = array(
    'id'      => 'sku_123ABC',
    'qty'     => 1,
    'price'   => 39.95,
    'name'    => 'T-Shirt',
    'options' => array('Size' => 'L', 'Color' => 'Red')
);

$this->cart->insert($data);
```

Source Code 6.9 Dokumentasi Add Item Pada Shopping Cart

Silahkan teman – teman perhatikan *source code* 6.9 yang di dapat pada dokumentasi *library shopping cart* pada *codeigniter*. Dimana teman – teman perlu mengirimkan *array* dengan informasi produk ke dalam *method* ***\$this->cart->insert(\$data);***.

Bagaimana apakah teman – teman sudah paham cara menambahkan *item* dengan *library cart* ?, Mari kita praktekan *source code* tersebut kedalam sistem yang akan kita bangun. Disini saya mempunya *controllers* dengan nama Cart.php. Pada langkah pertama dimana teman – teman perlu melakukan *load* terhadap *library cart*, perhatikan *script code* 6.10 berikut.

```
<?php
defined('BASEPATH') or exit('No direct script access
allowed');

class Cart extends CI_Controller
{
    public function __construct()
    {
        parent::__construct();
        $this->load->library('form_validation');
        $this->load->library(array('cart'));
    }
}
```

*Source Code 6.10 Melakukan Load Library Cart*

Sebelumnya sudah pernah kita bahas mengenai *function \_\_construct* merupakan *function* pertama yang akan di *run* oleh *controllers* saat pertama kali di akses. ***\$this->load->library('form\_validation');*** dimana teman – teman akan melakukan *load* terhadap *library form validation* yang berfungsi sebagai pemberian *alert*. ***\$this->load->library(array('cart'));*** dimana teman – teman akan melakukan *load* terhadap *library cart*.

Pada langkah selanjutnya dimana kita akan belajar menambahkan *item* yang berada pada *database* kedalam *library cart* dengan menerapkan *source code* 6.9. Perhatikan *source code* 6.11 sebagai berikut.

```
public function add($id)
{
    $barang = $this->m_cart->add('tb_barang',$id);

    $data = array(
        'id'      => $barang->id_brg,
        'qty'     => 1,
        'price'   => $barang->harga,
        'name'    => $barang->nama_brg,
    );

    $this->cart->insert($data);
    redirect('cart');
}
```

*Source Code 6.11 Menambahkan Item Database Ke Cart*

Dimana teman – teman perlu membuat *function add* di dalam *controllers* teman – teman dan memanggil *id* dari *database*. Pada baris selanjutnya dimana teman – teman perlu membuat variabel dengan nama *\$barang* dan diisikan fungsi *models*. Pada barisnya silahkan teman – teman terapkan *source code* 6.9 kedalam *function* tersebut. Karena kita akan menambahkan *item* yang ada dalam *database* maka *array* yang dikirim berupa pemanggilan nama *field* yang ada pada *database*. Kirim *array* tersebut dengan menggunakan *script* *\$this ->cart ->insert(\$data);* lalu lakukan *redirect* ke halaman *cart*.

Langkah selanjutnya dimana teman – teman perlu membuat *models* *m\_cart* tersebut pada diektori *application/models*. Perhatikan *source code* 6.12 berikut.

```
<?php

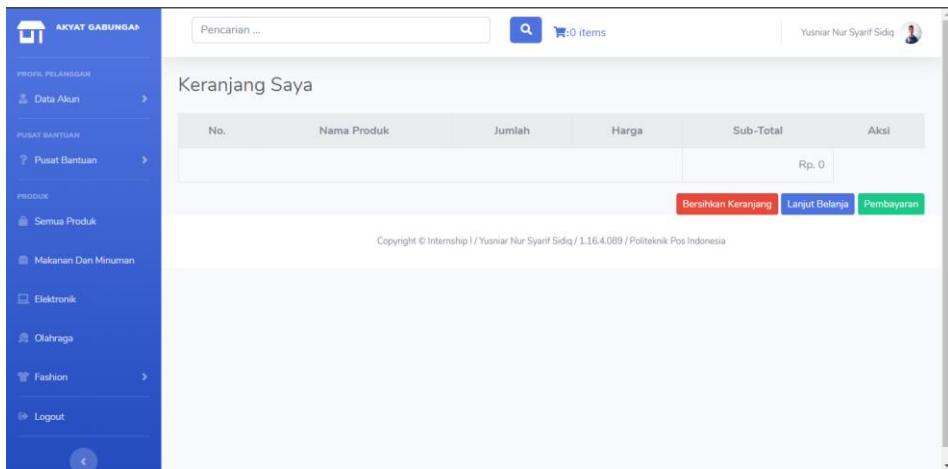
class M_cart extends CI_Model
{
    public function add($table_name, $id)
    {
        $this->db->where('id_brg', $id);
        $ambilData = $this->db->get($table_name);
        return $ambilData->row();
    }
}
```

*Source Code 6.12 Membuat Models m\_cart.php*

Pada *source code* tersebut dimana akan mengambil data dari *database* dengan menggunakan fungsi *where*. Mari kita bahas dari tiap – tiap barisnya, pada baris pertama dimana teman – teman perlu membuat *function add* pada *models m\_cart* dengan mendeskripsikan variabel *\$table\_name* dan *\$id*. Pada baris selanjutnya dimana teman – teman akan membuat pemanggilan data dari *database* berdasarkan *id\_brg* dengan memanggil variabel *\$id* menggunakan fungsi *where*. Pada baris berikutnya dimana teman – teman akan membuat variabel *\$ambilData* yang diberikan fungsi membaca data dari *database* dengan mengambil data pada *\$table\_name* menggunakan fungsi *get*. Pada baris selanjutnya dimana teman – teman perlu menambahkan fungsi *row agar* data yang ditambahkan diambil satu baris saja.

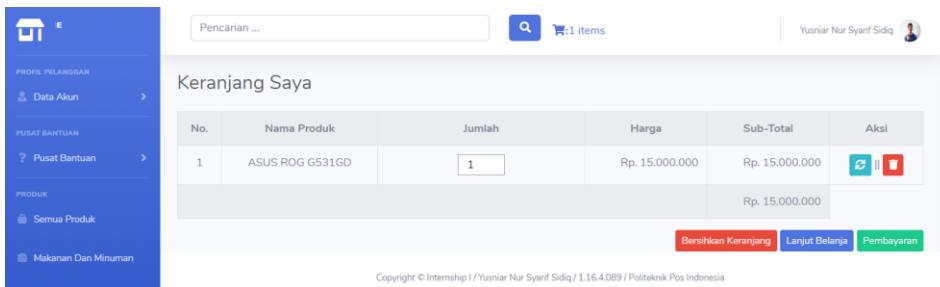
Selamat teman – teman sudah berhasil membuat fungsi *add* pada *library shopping cart*. Langkah selanjutnya dimana teman – teman

membutuhkan *view* untuk melakukan uji coba terhadap fungsi – fungsi *library shopping cart* tersebut. Silahkan teman – teman buat *view* halaman keranjang belanja sesuai keinginan teman – teman. Namun untuk menampilkan *item* yang ditambahkan dimana teman – teman perlu menambahkan *script foreach (\$this->cart->contents() as \$items)* pada *source code view* halaman keranjang belanja teman – teman. Dimana saya sudah membuat *view* halaman keranjang belanja seperti yang ditampilkan pada gambar 6.23.



Gambar 6.23 View Halaman Keranjang Belanja

Apabila teman – teman sudah membuat *view* halaman keranjang belanja mari kita coba apakah fungsi tambah *item* yang sudah kita buat. Pada gambar 6.23 dimana halaman keranjang belanja masih terlihat kosong. Apabila kita jalankan *function add* maka halaman keranjang belanja akan terisi seperti pada gambar 6.24.



*Gambar 6.24 Menjalankan Function Add Pada Controllers Cart*

Langkah selanjutnya bagaimana kalau kita belajar cara menghapus *items* yang pada halaman keranjang belanja tersebut. Silahkan teman – teman perhatikan *source code* 6.13 berikut.

```
public function delete($rowid)
{
    $data = array(
        'rowid' => $rowid,
        'qty' => 0
    );

    $this->cart->update($data);
    $this->session->set_flashdata(
        'message', '<div class="alert alert-danger" role="alert">
                    Item berhasil dihapus!</div>');
    redirect('cart/index');
}
```

*Source Code 6.13 Function Delete Pada Controllers Cart*

Coba teman – teman perhatikan *script code* yang diberikan tanda warna merah tersebut. Untuk membuat fungsi *delete* pada *library shopping cart* tersebut dimana teman – teman membutuhkan *\$rowid*. Lalu apa itu *\$rowid* ?, *\$rowid* adalah sebuah *code* unik yang secara otomatis terbuat saat kita

menambahkan suatu data menggunakan fungsi *add* pada *source code* 6.11. **\$rowid** ini dibutuhkan dalam pembuatan fungsi *delete* dan *update* pada *library shopping cart*.

Apabila teman – teman sudah paham mengenai **\$rowid** dimana saya akan menjelaskan baris per baris pada *source code* 6.13 tersebut. Pada baris pertama dimana teman – teman akan membuat *function* baru dengan nama *delete* dan mengambil **\$rowid**. Pada baris selanjutnya teman – teman akan membuat *array*, yang dimana akan memanggil *rowid* nya dan mengubah *qty* yang asalnya 1 menjadi 0. Pada baris selanjutnya dimana teman – teman perlu melakukan *update* data *array* nya menggunakan *script* **\$this →cart →update(\$data);**. Karena kita menggunakan *library form validation* maka kita buat *flash datanya* dan apabila berhasil lakukan *redirect* ke halaman keranjang belanja. Agar *function delete* tersebut dapat berjalan dimana teman – teman perlu memanggilnya pada halaman *view* pada bagian *button delete* seperti pada *source code* 6.26.

```
<?= anchor('cart/delete/' . $items['rowid'] , '  
<div class="btn btn-danger btn-sm">  
  <i class="fa fa-trash"></i></div>') ?>
```

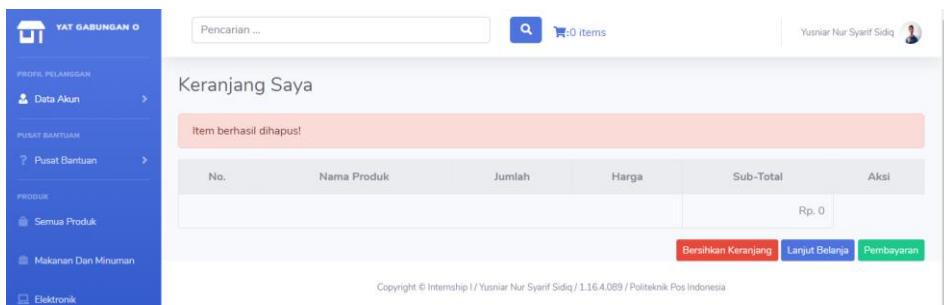
*Source Code 6.14 Memanggil Function Delete Pada Button Delete*

Apabila teman – teman sudah paham mengenai *function delete* silahkan teman – teman buka halaman keranjang belanja yang sudah teman – teman buat, arahkan krusor kepada *button delete* lalu perhatikan bagian sisi bawah sebelah kiri sistem dimana *rowid* dari *item* tersebut akan terpanggil dengan menggunakan *source code* 6.14 tersebut. Perhatikan gambar 6.25.



Gambar 6.25 Rowid Delete

Silahkan teman – teman aplikasi *button delete* teman – teman dan apabila *function delete* berjalan tanpa adanya *error* maka akan muncul *form validation* dan data *item* pada halaman keranjang akan berubah menjadi kosong seperti pada gambar 6.26.



Gambar 6.26 Function Delete Berhasil Diterapkan

*Function delete* tersebut dimana berlaku hanya untuk menghapus *item* secara satu per satu, namun bagaimana apabila *user* ingin menghapus semua *items* yang terdapat pada halaman keranjang belanja tersebut ? .

No.	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	ASUS ROG G531GD	1	Rp. 15.000.000	Rp. 15.000.000	
2	Bola Basket Spalding	1	Rp. 150.000	Rp. 150.000	
3	Cappuccino Cincau	1	Rp. 12.000	Rp. 12.000	
4	Paket Ayam Goreng	1	Rp. 12.000	Rp. 12.000	
				Rp. 15.174.000	

[Bersihkan Keranjang](#) [Lanjut Belanja](#) [Pembayaran](#)

Copyright © Internship I / Yusniar Nur Syarif Sidiq | 1.16.4.089 / Politeknik Pos Indonesia

Gambar 6.27 Banyaknya Items Pada Halaman Keranjang Belanja

Coba teman – teman perhatikan gambar 6.27 tersebut, dimana pada halaman keranjang belanja tersebut terdapat banyak *items* yang ditambahkan oleh *user*. Bagaimana apabila terjadi suatu kondisi dimana *user* ingin menghapus semua *items* tersebut ?, karena kita sudah membuat *function delete* dimana hal tersebut sudah bisa kita lakukan, namun dengan cara menghapus satu per satu, hal tersebut menggunakan waktu kerja yang relatif lama. Sebenarnya ada cara yang lebih cepat loh teman – teman, bagaimana caranya ?, dimana teman – teman dapat menggunakan fungsi *destroy*. *Destroy* adalah sebuah fungsi pada *codeigniter* yang berguna untuk menghapus data secara keseluruhan namun tidak menghapus data pada *database*. Mari kita terapkan fungsi *destroy* tersebut dengan membuat *function* baru pada *controllers cart* seperti pada *source code 6.15* berikut.

```
public function hapus_keranjang()
{
    $this->cart->destroy();
    redirect('user/halaman_produk');
}
```

Source Code 6.15 Membuat Fungsi Destroy

Apabila teman – teman sudah membuat *function* yang sudah menggunakan fungsi *destroy* tersebut jangan lupa di panggil ada halaman *view* nya dengan cara membuat *button* baru dan masukkan *source code* 6.16 berikut.

```
<div align="right">
    <a href="= base_url('cart/hapus_keranjang') ?"&gt;"&gt;
        &lt;div class="btn btn-sm btn-danger"&gt;
            Bersihkan Keranjang
        &lt;/div&gt;
    &lt;/a&gt;

    &lt;a href="<?= base_url('user/halaman_produk') ?"&gt;"&gt;
        &lt;div class="btn btn-sm btn-primary"&gt;
            Lanjut Belanja&lt;/div&gt;
    &lt;/a&gt;

    &lt;a href="<?= base_url('cart/pembayaran') ?"&gt;"&gt;
        &lt;div class="btn btn-sm btn-success"&gt;
            Pembayaran
        &lt;/div&gt;
    &lt;/a&gt;

&lt;/div&gt;</pre
```

*Source Code 6.16 Menambahkan Button Dan Memanggil Fungsi Destroy  
Pada Button Tersebut*

Perlu teman – teman ketahui bahwa fungsi *delete* yang terdapat pada *library shopping cart* tersebut hanya dapat menghapus *items* yang terdapat pada halaman keranjang belanja dan tidak menghapus data yang berada pada *database* meskipun dalam pemanggilannya menggunakan data pada *database*. Jika teman – teman ingin belajar bagaimana cara menghapus data pada *database* silahkan teman – teman buka kembali *source code* 5.7 pada sub bab membuat CRUD sederhana menggunakan *codeigniter*.

Pada pembahasan sebelumnya dimana teman – teman sudah belajar bagaimana cara menambahkan data yang berada pada *database* kedalam keranjang belanja menggunakan *library shopping cart* dan bagaimana cara menghapus data yang berada pada halaman keranjang belanja menggunakan *library shopping cart*. Pembahasan berikutnya dimana teman – teman akan belajar bagaimana cara melakukan *update items* pada *library shopping cart*. Untuk membuat *update items* tersebut dimana teman – teman akan menggunakan kembali *\$rowid*. Silahkan teman – teman perhatikan *source code* 6.17 berikut.

```
public function update($rowid)
{
    if ($rowid) {
        $data = array(
            'rowid' => $rowid,
            'qty'   => $this->input->post('qty')
        );

        $this->cart->update($data);
        $this->session->set_flashdata('message',
            '<div class="alert alert-success"
                role="alert">
                Keranjang berhasil diperbaharui!
            </div>');
    }

    redirect(base_url('cart/index'), 'refresh');

} else {

    redirect(base_url('cart/index'), 'refresh');
}
```

*Source Code 6.17 Membuat Function Uodate Items Pada Controllers*

*Cart*

Silahkan teman – teman perhatikan *source code* 6.17 tersebut dimana merupakan *function update* yang menerapkan *library shopping cart*. Mari kita bahas pada tiap – tiap baris dari *source code* 6.17 tersebut. Dimana pada baris pertama teman – teman akan membuat *function* dengan nama *update* serta memanggil variabel *\$rowid* dan bersifat *public*. Pada baris selanjutnya dimana teman – teman akan membuat fungsi *if* dan memanggil variabel *\$rowid*. Pada baris selanjutnya dimana teman – teman perlu membuat *array* yang disimpan pada variabel *\$data*. Perhatikan *array* tersebut dimana teman – teman akan memanggil *\$rowid* nya yang merupakan *code uniq* pada *row* yang akan dipanggil. Dimana data yang akan kita *update* adalah *quantity* atau jumlah barang yang akan di pesan maka dari itu pada *array qty* silahkan teman – teman berikan fungsi *input*. Pada baris selanjutnya dimana teman – teman perlu memanggil fungsi *update* dan masukkan *array* tersebut dengan cara memanggil variabel *\$data* menggunakan *script \$this →cart →update(\$data);*. Karena kita menggunakan fungsi *form validation* dimana pada baris selanjutnya silahkan teman – teman panggil *flashdata*-nya. Apabila berhasil maka lakukan *redirect* ke halaman keranjang belanja dengan menampilkan *alert* “Keranjang berhasil diperbaharui” namun apabila gagal lakukan *redirect* ke halaman yang sama akan tetapi dengan pesan *error*.

Apabila teman – teman sudah membuat *function update* tersebut dimana teman – teman perlu perhatikan juga bagian *view* pada halaman keranjang belanja teman – teman. Teman – teman perlu memanggil *rowid*. Silahkan teman – teman perhatikan *source code* 6.18 yang merupakan *source code* pemanggilan *items* pada *library shopping cart*.

```
<?php $i = 1; ?>
<?php $no = 1;

foreach ($this->cart->contents() as $items) :

    echo form_open(base_url('cart/update/' .
                           $items['rowid']));

?>
    <?php echo form_hidden($i.'[rowid]', $items['rowid']); ?>

<tr align="center">
    <td><?= $no++ ?></td>
    <td>
        <?php echo $items['name']; ?>
        <?php if ($this->cart->has_options($items['rowid']) == TRUE): ?>
<p>
        <?php foreach ($this->cart->product_options($items['rowid']) as $option_name =>
                           $option_value): ?>

            <strong><?php echo $option_name; ?></strong>
            <?php echo $option_value; ?><br />

            <?php endforeach; ?>
        </p>
        <?php endif; ?>
    </td>

    <td>
        <input class="col-sm-3" type="number" name="qty"
               value="<?= $items['qty'] ?>">
    </td>

    <td align="right">
        Rp. <?= number_format($items['price'], 0, ',', '.') ?>
    </td>
```

```

<td align="right">
    Rp. <?= number_format(
        $items['subtotal'], 0, ',', '.') ?>
</td>

<td>
    <button type="submit" class="btn btn-sm btn-info">
        <i class="fa fa-sync"></i>
    </button> ||
    <?= anchor('cart/delete/' . $items['rowid'] , '
        <div class="btn btn-danger btn-sm">
            <i class="fa fa-trash"></i></div>') ?>
</td>

</tr>
<?php
    echo form_close();
    endforeach;
?>

<tr>
    <td colspan="4"></td>
    <td align="right">Rp. <?= number_format(
        $this->cart->total(), 0, ',', '.') ?>
    </td>
</tr>

```

Source Code 6.18 Code Pemanggilan View Pada Halaman Keranjang  
*Belanja*

#### 6.4 Membuat Fitur Pencarian

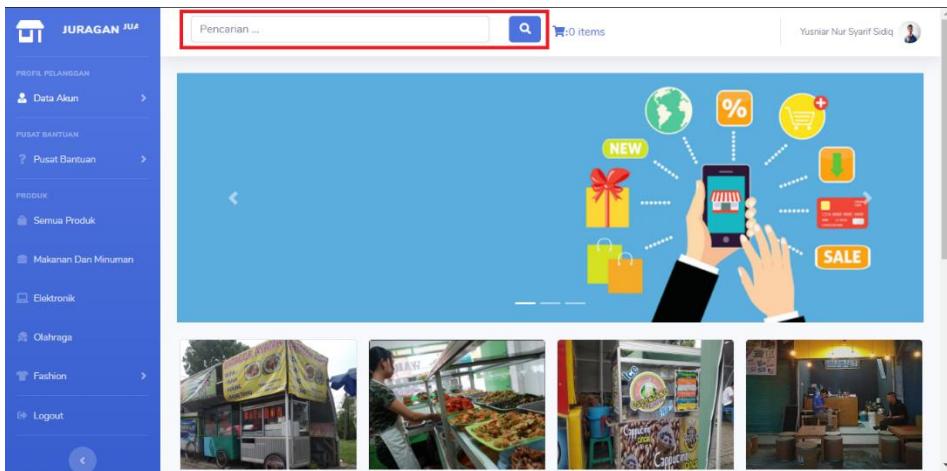
Apa kalian pernah mengakses sistem *e-commerce* ?, baik dalam bentuk *web site* ataupun *android* ?, apa saja sih *fitur – fitur* yang dimiliki oleh sistem *e-commerce* tersebut ?. Pada sistem yang mengandung konsep *e-commerce* dimana fitur pencarian sangatlah penting diterapkan dan tidak boleh dilewatkan. Hal ini dikarenakan dimana fitur pencarian tersebut

memiliki peran penting pada suatu sistem, dimana dapat mempermudah *user* dalam melakukan pencarian data secara cepat.

Apakah *framework codeigniter* dapat membuat fitur pencarian tersebut ?, Oh tentu saja bisa dong, apakah teman – teman tertarik untuk membuatnya ?, Yuk simak pembelajaran kali ini yaitu membuat fitur pencarian data pada sistem *e-commerce* menggunakan *framework codeigniter*.

Langkah awal yang perlu teman – teman lakukan adalah pastinya teman – teman memerlukan sebuah *website e-commerce* dalam tahap pengembangan dengan menggunakan *framework codeigniter*. Disini saya menggunakan sistem JURAGAN yang dimana merupakan salah sistem *e-commerce* yang sifatnya masih pada tahap pengembangan dan tentunya disusun dengan menggunakan *framework codeigniter*.

Langkah selanjutnya silahkan teman – teman membuat sebuah *form input* yang berfungsi sebagai kolom pencarian. Karena sistem JURAGAN menggunakan *template* pada *bootstrap SB ADMIN 2* yang dimana sudah disiapkan *form input* untuk fitur pencarian pada bagian *topbar*, perhatikan gambar 6.28 tersebut.



Gambar 6.28 Fitur Search Pada Topbar Dalam Sistem JURAGAN

Langkah selanjutnya silahkan teman – teman membuat *file controllers* untuk membuat *function search*. Pada sistem JURAGAN dimana saya akan membuat *function search* tersebut di dalam *controllers User*. Silahkan teman – teman buat *function* tersebut pada *file controllers* yang sudah dibuat dan masukkan *script* seperti pada *source code* 6.19.

```
public function search()
{
    $keyword = $this->input->post('keyword');

    $data['user'] = $this->db->get_where(
        'user', ['email' => $this->session->userdata(
            'email')])->row_array();

    $data['title'] = 'Search';

    $data['user_brg'] = $this->m_user->get_keyword(
        $keyword);

    $this->load->view('user/templates/header',
        $data);
    $this->load->view('user/templates/sidebar');
```

```
$this->load->view('user/templates/topbar',  
                      $data);  
$this->load->view('user/search', $data);  
$this->load->view('user/templates/footer');  
}
```

Source Code 6.19 Membuat Function Seacrh Pada Controllers

Silahkan teman – teman perhatikan *source code* 6.19 tersebut, Apakah teman – teman sudah mengerti ?, jika belum mari kita bedah pada tiap – tiap barisnya. Pada baris pertama dimana teman – teman akan membuat sebuah *function* dengan nama *search* dan bersifat *public*. Pada baris selanjutnya dimana teman – teman perlu membuat fungsi *input* terhadap *keyword* yang akan di cari dengan menggunakan *script* **\$this →input →post('keyword')** pada variabel **\$keyword**. Pada baris selanjutnya dimana bersifat *opsional*, apabila teman – teman menggunakan *session* pada sistem tersebut, dimana teman – teman perlu menggil *session* tersebut. Pada baris selanjutnya silahkan teman – teman beri judul pada halaman tersebut dengan menggunakan *script* **\$data['tittle']**. Pada baris selanjutnya dimana teman – teman perlu melakukan *load* terhadap *models* yang akan kita buat. Pada baris selanjutnya dimana teman – teman perlu melakukan *load view* pada *template* dan halaman *view* nya. Apabila teman – teman jalankan *controllers* tersebut dimana akan terjadi *erorr*, hal ini dikarenakan teman – teman belum membuat fungsi *models* untuk fitur pencarian tersebut.

Langkah selanjutnya dimana teman – teman perlu membuat fungsi *models* yang berguna untuk membaca data dan menampilkan data tersebut sesuai dengan *keyword* yang diberikan. Silahkan teman – teman perhatikan *source code* 6.20 berikut.

```
public function get_keyword($keyword)
{
    $this->db->select('*');
    $this->db->from('tb_barang');

    $this->db->like('nama_brg', $keyword);
    $this->db->or_like('gambar', $keyword);
    $this->db->or_like('detail', $keyword);

    return $this->db->get_where()->result();
}
```

Source Code 6.20 Membuat Models Dengan Function `get_keyword`

Perhatikan *source code* 6.20 tersebut, mari kita bahas pada tiap – tiap barisnya. Pada baris pertama dimana teman – teman akan membuat sebuah *function* pada *models* dengan naman `get_keyword` yang bersifat *public* dan memanggil `$keyword`. Pada baris berikutnya dimana teman – teman perlu memanggil semua isi yang ada pada *database* dengan cara menerapkan *script* `$this →db →select('*');`. Pada baris berikutnya dimana teman – teman perlu menentukan *table* mana yang akan di panggil dengan menggunakan *script* `$this →db →from('tb_barang');`. Jika baris kedua dan ketiga di gabungkan maka teman – teman dapat membacanya dimana kita akan memanggil semua data atau isi yang ada di dalam *record* dari *table tb\_barang*. Dalam fungsi *codeigniter* dimana teman – teman perlu menggunakan fungsi *like*. Fungsi *like* adalah untuk menyeleksi suau data berdasarkan nilai kemiripan yang terdapat pada suatu *field*. Perhatikan *script* `$this →db →like('nama_brg', $keyword)` yang dimana dapat dibaca bahwa sistem akan membaca data dari *field nama\_brg* pada *database* dan akan diseleksi berdasarkan kemiripan dari `$keyword` yang akan diberikan. Selanjutnya dimana terdapat fungsi *or like* yaitu berfungsi untuk

melakukan penyeleksian terhadap nilai *field* yang memiliki kesamaan dengan banyak sarat akan tetapi hanya salah satu yang dipenuhi. Pada baris terakhir dimana teman – teman perlu menampilkan data tersebut dengan menggunakan fungsi *get\_where*.

Apabila teman – teman sudah selesai membuat *file controllers* dan *models* nya dimana teman – teman perlu membuka memanggil *file controllers* tersebut pada bagian *template* nya, silahkan teman – teman perhatikan *source code* 6.21 berikut.

```
<!-- Topbar Search -->
<div class="form-group col-sm-5 mx-sm-3 mb-2">

    <?= form_open('user/search') ?>

    <input type="text" name="keyword"
        class="form-control" placeholder="Pencarian ...">
</div>
<button type="submit" class="btn btn-primary mb-2">
    <i class="fas fa-fw fa-search"></i>
</button>

    <?= form_close() ?>
```

*Source Code 6.21 Memanggil Controllers Pada Form Input-Nya*

Silahkan teman – teman jalankan *controllers* tersebut, dimana fungsi yang diberikan telah berhasil digunakan namun *view* nya tidak muncul, maka dari itu langkah selanjutnya dimana teman – teman perlu membuat *view* untuk memunculkan data berdasarkan *keyword* yang diberikan. Silahkan perhatikan *source code* 6.22 berikut.

```
<div class="container-fluid">
<div id="carouselExampleIndicators"
  class="carousel slide" data-ride="carousel">
<ol class="carousel-indicators">
  <li data-target="#carouselExampleIndicators"
    data-slide-to="0" class="active">
  </li>
  <li data-target="#carouselExampleIndicators"
    data-slide-to="1">
  </li>
  <li data-target="#carouselExampleIndicators"
    data-slide-to="2">
  </li>
</ol>
<div class="carousel-inner">
<div class="carousel-item active">
  
</div>
<div class="carousel-item">
  
</div>
<div class="carousel-item">
  
</div>
</div>
<a class="carousel-control-prev"
  href="#carouselExampleIndicators" role="button"
  data-slide="prev">
  <span class="carousel-control-prev-icon"
    aria-hidden="true"></span>
  <span class="sr-only">Previous</span>
</a>
<a class="carousel-control-next"
  href="#carouselExampleIndicators" role="button"
  data-slide="next">
  <span class="carousel-control-next-icon"
    aria-hidden="true"></span>
  <span class="sr-only">Next</span>
</a>
```

```
    data-slide="next">
<span class="carousel-control-next-icon"
aria-hidden="true"></span>
<span class="sr-only">Next</span>
</a>
</div>

<div class="row text-center mt-4">

<?php foreach ($user_brg as $us) { ?>

<div class="card ml-3 mb-3" style="width: 16rem;">
<img src=<?php echo base_url() . '/uploads/' .
$us->gambar ?> class="card-img-top">
<div class="card-body">
<h5 class="card-title mb=1"><?= $us->nama_brg ?>
</h5>
<small><?= $us->keterangan ?></small><br>
<span class="badge badge-pill badge-success mb-3">
Rp. <?= number_format(
$us->harga, 0, ',', '.') ?></span><br>
<?= anchor('cart/add/' . $us->id_brg, '
<div class="btn btn-sm btn-primary">
Tambah Ke Keranjang
</div>') ?>
<?= anchor('user/detail_barang/' . $us->id_brg, '
<div class="btn btn-sm btn-success">Detail</div>')
?>
</div>
</div>

<?php } ?>

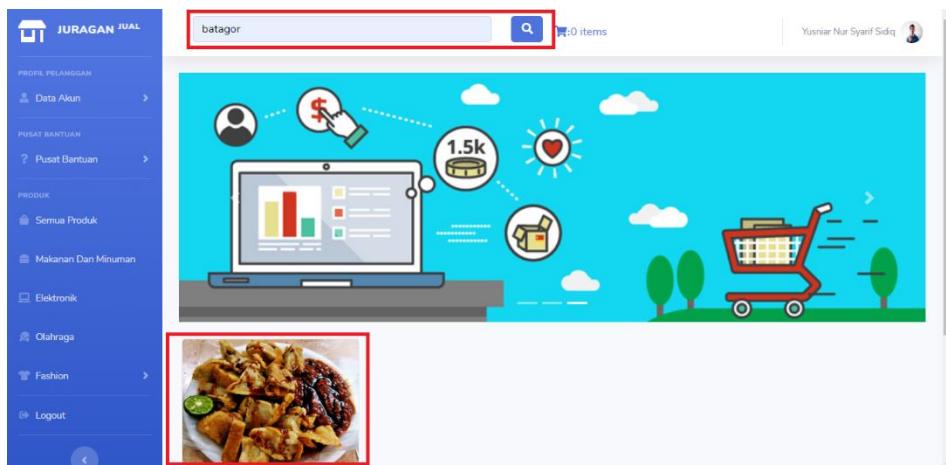
<!-- <table>

<?php foreach ($user as $us) { ?>
<tr>
<td><?php echo $us->nama_brg ?></td>
</tr>
<?php } ?>
```

```
</table> -->  
  
</div>  
</div>
```

Source Code 6.22 Membuat View Search

Apabila teman – teman sudah membuat *view* nya silahkan teman – teman jalankan kembali *fitur* pencarian tersebut. Contoh dimana saya akan mencari makanan dengan *keyword* “Batagor”. Maka saat dilakukan pencarian dimana hanya *record* yang memiliki kemeripan dengan batagor akan ditampilkan, seperti pada gambar 6.23.



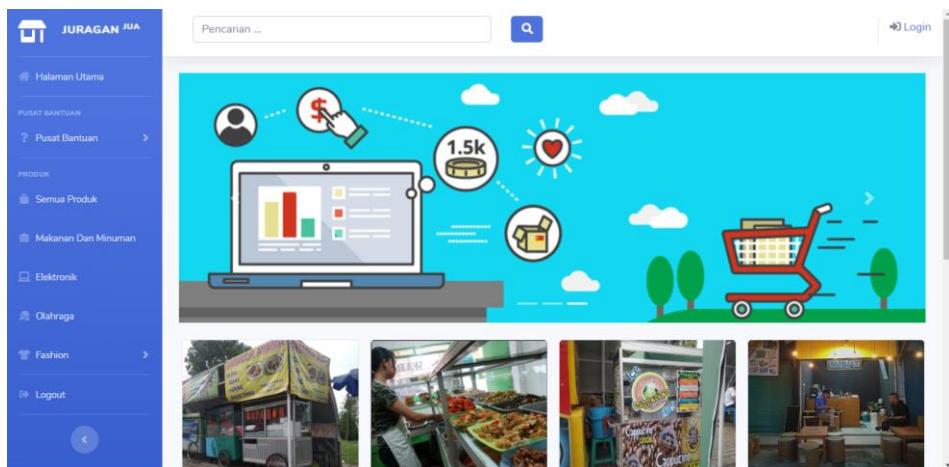
Source Code 6.23 Fitur Pencarian Berhasil Dibuat

## BAB VII

# PANDUAN MENGGUNAKAN SISTEM JURAGAN

Sistem JURAGAN merupakan sebuah sistem yang dirancang dengan *framework codeigniter* dan menggunakan *MariaDB* sebagai basis datanya. Sistem JURAGAN tersebut dibuat untuk menggabungkan para pengusaha baik dibidang kuliner, elektronik, fashion dan peralatan olahraga, mulai dari level UMKM sampai pedagang kaki lima. JURAGAN dirancang dengan konsep *e-commerce* dan berbasis komunitas. Apa yang dimaksud dengan komunitas disini ?, dimana *user* dapat mencari suatu produk berdasarkan komunitas daerah dan produk. Pada pembahasan kali ini dimana penulis akan membuat panduan bagaimana cara menggunakan sistem JURAGAN tersebut.

### 7.1 Halaman Awal Sistem JURAGAN

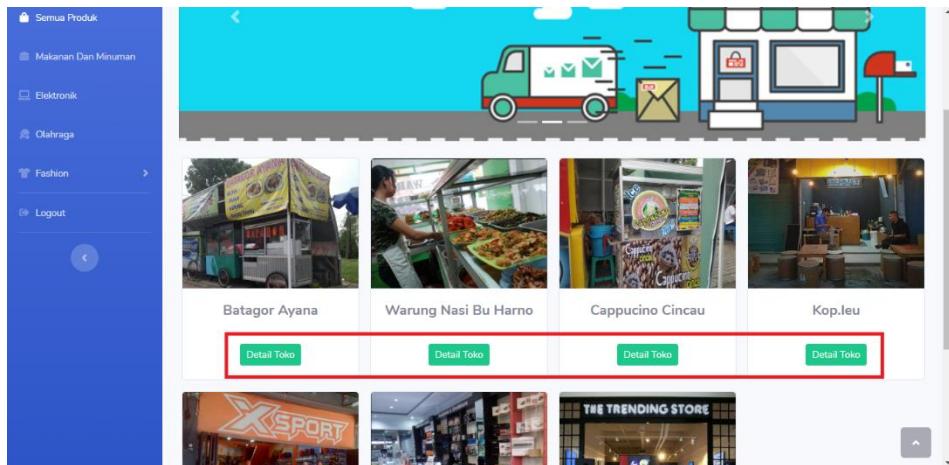


Gambar 7.1 Halaman Awal Sistem JURAGAN

Saat pertama kali *user* mengakses sistem JURAGAN dimana akan ditampilkan halaman awal dari sistem JURAGAN tersebut. JURAGAN

memiliki tampilan halaman awal seperti pada gambar 7.1 yang dimana menggunakan desain dengan warna latar dominan biru. Pada halaman awal tersebut dimana *user* dapat melakukan *explore* terhadap sistem JURAGAN tersebut, akan tetapi ada batasannya. Pada halaman awal tersebut dimana kondisi *user* belum melakukan *login* sehingga *user* tidak dapat melakukan proses transaksi. Namun *user* dapat melihat produk – produk apa saja yang sedang ditawarkan pada sistem JURAGAN tersebut dan informasi – informasi yang diberikan baik informasi produk maupun toko.

Pada halaman awal tersebut dimana *user* dapat melihat informasi – informasi toko yang sudah terdaftar dan terverifikasi pada sistem JURAGAN dengan cara melakukan klik pada *button* “**Detail Toko**” seperti pada gambar 7.2 berikut.



*Gambar 7.2 Mengaplikasikan Button Detail Toko Pada Halaman Awal*

Silahkan pilih salah satu toko yang ingin teman – teman lihat, sebagai contoh dimana saya akan memilih toko “**Warung Nasi Bu Harno**” untuk melihat informasi dari toko tersebut. Silahkan teman – teman klik *button*

“**Detail Toko**” tersebut maka teman – teman akan dibawa langsung ke halaman detail toko pada sistem JURAGAN, seperti yang ditampilkan pada gambar 7.2 berikut.

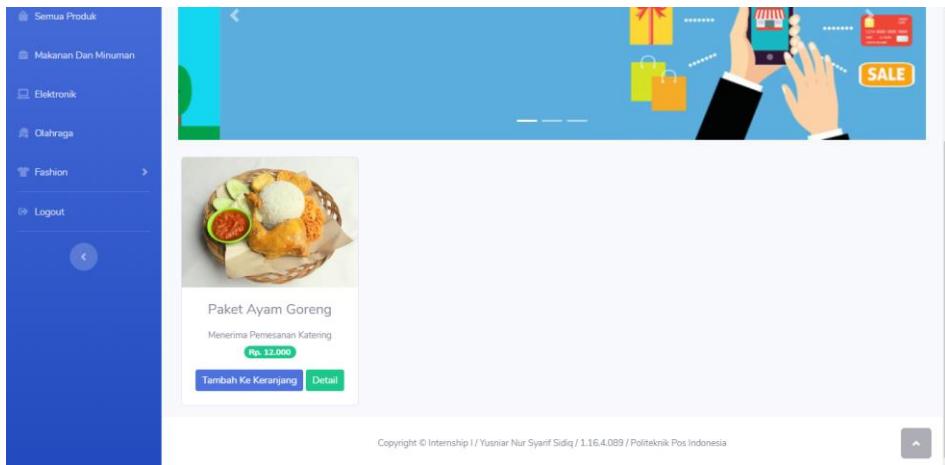
The screenshot shows the JURAGAN system's 'Detail Toko' (Store Detail) page. On the left is a sidebar with links like 'Halaman Utama', 'Pusat Bantuan', 'Produk', 'Logout', etc. The main area has a search bar and a 'Login' button. The 'Detail Toko' section features a photo of a food stall. Below it, there's a table with store details:

Nama Toko	Warung Nasi Bu Harno
Alamat Toko	Asrama Yon Zipur 9 Rt 06 Rw 02 No. 320 Kec. Cinambo Kel. Pakemitan Kota Bandung 40294
No. Telporn	+626285723438131
Catatan Toko	Warang Nasi Bu Harno menyediakan berbagai macam menu makanan setiap harinya. Menyediakan paket katering per bulan atau pemesanan dengan jumlah banyak. Harga paket dapat dilihat pada produk - produk kami yang sudah ter upload. Temukan kami di alamat yang sudah tertera.

At the bottom are 'Produk Toko' and 'Kembali' buttons.

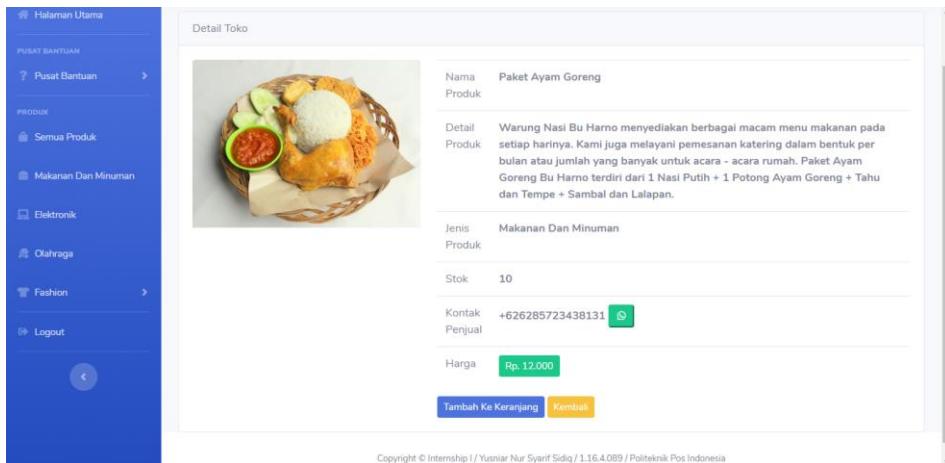
Gambar 7.3 Halaman Detail Toko Pada Sistem JURAGAN

Apabila teman – teman ingin kembali ke halaman awal tersebut dimana teman – teman memilih *button “Kembali”*, namun pada halaman detail toko tersebut dimana teman – teman juga dapat melihat produk apa saja sih yang ditawarkan oleh toko tersebut. Dimana teman – teman perlu mengakses halaman produk toko dengan cara memilih *button “Produk Toko”* pada halaman detail toko tersebut yang dimana nantinya teman – teman akan dibawa ke halaman “**Produk Toko**” seperti pada gambar 7.4 berikut.



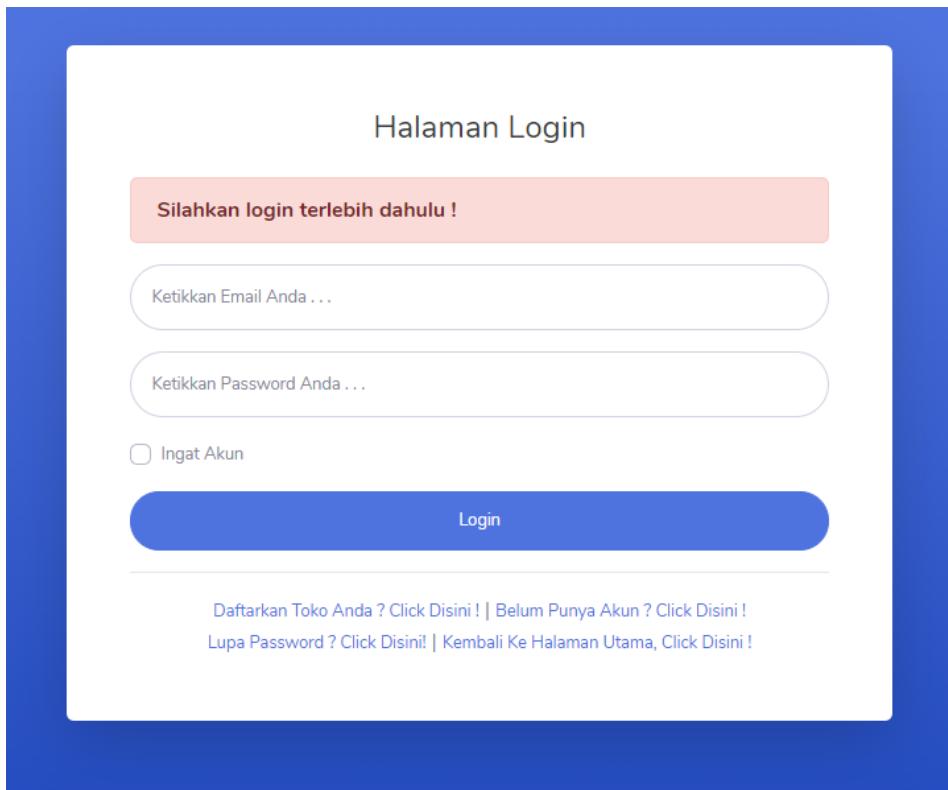
Gambar 7.4 Halaman Produk Toko

Silahkan teman – teman perhatikan pada halaman produk toko tersebut dimana saat di akses hanya menampilkan produk dari toko tersebut. Pada gambar 7.4 dimana kita telah memilih toko “**Warung Nasi Bu Harno**” sehingga yang di tampilkan adalah produk dari toko tersebut. Pada halaman tersebut dimana terdapat dua *button* “**Tambah Ke Keranjang**” dan “**Detail**”. Pada tiap – tiap *button* dimana memiliki fungsi masing – masing, mari kita bahas mengenai *button* “**Detail**” terlebih dahulu, dimana *button* tersebut berfungsi untuk melihat informasi mengenai produk tersebut. Apabila teman – teman klik *button* “**Detail**” tersebut dimana teman – teman akan dibawa ke halaman detail produk seperti yang ditampilkan pada gambar 7.5.



Gambar 7.5 Halaman Detail Produk

Pada halaman detail produk tersebut dimana teman – teman dapat mengetahui informasi – informasi seputar produk tersebut. Informasi yang teman – teman dapatkan berupa nama dari produk tersebut, detail dari produk tersebut, jenis produk dari produk tersebut, jumlah stok dari produk tersebut, kontak penjual produk tersebut, dan harga dari produk tersebut. Pada halaman detail produk tersebut dimana teman – teman menukan dua *button*, yaitu *button* “**Tambah Ke Keranjang**” yang dimana untuk melanjutkan proses ke tahap transaksi dan *button* “**Kembali**” yang dimana untuk mengembalikan *user* ke halaman produk. Karena kita sudah mengaplikasikan *button* “**Detail**”, sekarang coba teman – teman aplikasikan *button* “**Tambah Ke Keranjang**” pada halaman detail produk tersebut. Dimana respon yang terjadi pada sistem akan ditunjukkan pada gambar 7.6 berikut.



*Gambar 7.6 Peringatan Wajib Login Terlebih Dahulu*

Kenapa sistem malah memberikan pemberitahuan “**Silahkan login terlebih dahulu !**” ?, hal ini dikarena *user* masih dalam kondisi belum melakukan *login*. Untuk melanjutkan ke tahap transaksi dimana ada *validasi* yang perlu dipenuhi oleh *user*. Dimana *user* harus melakukan *login* terlebih dahulu agar dapat mengakses halaman transaksi tersebut.

## **7.2 Halaman Registrasi Dan Login Sistem JURAGAN**

Sebelum teman – teman melakukan proses *login* dimana teman – teman perlu melewati proses pendaftaran akun terlebih dahulu. Pada sistem JURAGAN dimana menggunakan *email* sebagai *username*-nya. Mengapa menggunakan *email* ? karena *email* pada umumnya tidak ada

yang sama. Selain itu dimana *email* lebih mudah di ingat dari pada *username*. Hal ini dikarenakan dalam kehidupan sehari – hari kita dimana lebih sering menggunakan *email* dari pada *username* sehingga menyebabkan daya ingat terhadap *email* lebih kuat dari pada *username*. Dengan menggunakan *email* sebagai *username* dimana dapat mengurangi jumlah *field* pada *table* dalam *database*. Hal ini menunjukkan bahwa *email* lebih efektif digunakan dari pada *username*.

Pada sistem JURAGAN dimana halaman daftar akun tersebut dibagi menjadi dua, yaitu halaman daftar akun khusus untuk *user* pelanggan dan halaman daftar akun khusus untuk *user* penjual. Dimana kita akan membahas mengenai halaman daftar akun bagi *user* dengan level pelanggan terlebih dahulu.

Daftarkan Akun Anda!

Nama Lengkap ...

Email Anda ...

Password Anda ...

Ulangi Password Anda ...

Daftar Akun

Lupa Password? Click Disini!

Sudah Punya Akun? Silahkan Login!

Gambar 7.7 Halaman Daftar Akun Pelanggan Pada Sistem JURAGAN

Perhatikan gambar 7.7 tersebut, dimana merupakan halaman pendaftaran akun bagi *user* dengan level pelanggan pada sistem JURAGAN. Untuk melakukan proses daftar akun tersebut dimana *user* perlu memberikan beberapa informasi data diri yang di butuhkan oleh sistem JURAGAN tersebut. Dimana *user* perlu memberikan data nama lengkap dan *email* yang aktif untuk melakukan pendaftaran. Pada halaman daftar akun tersebut dimana terdapat fungsi *form validation* yang dimana akan memberikan peringatan untuk kepada *user* tersebut apabila terjadi kesalahan saat meng-*input* data. Terdapat beberapa peraturan yang terdapat pada halaman registrasi tersebut, dimana semua *form* wajib di isi apabila kosong maka sistem akan memberikan peringatan kepada *user* bahwa kolom tersebut tidak boleh kosong, perhatikan gambar 7.8 berikut.

The screenshot shows a registration form titled "Daftarkan Akun Anda!" (Register Account). The form consists of three input fields: "Nama Lengkap ..." (Full Name), "Email Anda ..." (Your Email), and "Password Anda ..." (Your Password). Each field has a red error message below it: "The Name field is required.", "The Email field is required.", and "The Password field is required.". Below the fields is a large blue "Daftar Akun" (Register) button. At the bottom of the form, there are two links: "Lupa Password? Click Disini!" (Forgot Password? Click Here!) and "Sudah Punya Akun? Silahkan Login!" (Already have an account? Please log in!).

Gambar 7.8 Peringatan Kolom Tidak Boleh Kosong

Peraturan selanjutnya dimana *format* email yang diberikan harus benar – benar sesuai dan tidak asal. Hal ini sangatlah penting, karena pada saat *user* selesai mendaftar akun maka akun tersebut belum aktif, dan untuk mengaktifkan akun tersebut dimana *user* perlu melakukan proses aktivasi melalui *email*. Apabila *user* memberikan *email* dengan format yang salah maka sistem akan memberikan sebuah notifikasi seperti pada gambar 7.9 berikut.

The screenshot shows a registration form titled "Daftarkan Akun Anda!". It includes fields for Name, Email, Password, and Re-enter Password. The Email field contains "yusniarnss29", which is highlighted in red with the error message "The Email field must contain a valid email address." Below the form are links for password recovery and login.

Daftarkan Akun Anda!

Yusniar Nur Syarif Sidiq

yusniarnss29

The Email field must contain a valid email address.

Password Anda ...

Ulangi Password Anda ...

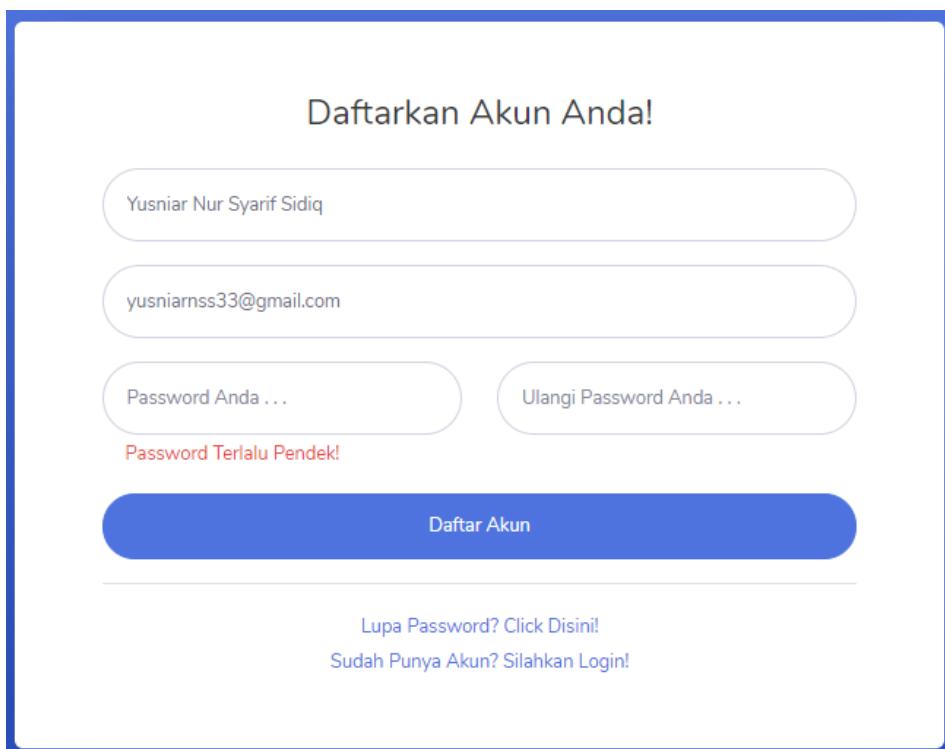
Daftar Akun

Lupa Password? Click Disini!  
Sudah Punya Akun? Silahkan Login!

Gambar 7.9 Notifikasi Format Email Tidak Sesuai

Peraturan berikutnya dimana terdapat pada bagian kolom *password*. Pada sistem JURAGAN *password* dapat dibuat dalam bentuk *text*, *angka*, dan campuran antara *text* dan angka, namun perlu di perhatikan bahwa *password* minimal berjumlah 6 huruf atau angka apabila kurang maka

sistem akan memberikan notifikasi bahwa *password* terlalu pendek seperti pada gambar 7.10.



Gambar 7.10 Notifikasi Password Terlalu Pendek

Coba teman – teman perhatikan gambar 7.10 tersebut dimana terdapat dua kolom *password*. Hal ini tersebut dibuat dengan tujuan supaya *user* lebih teliti lagi saat membuat *password* pada akunnya. Dimana terdapat aturan di dalamnya yaitu antara *password* satu dan dua harus sama atau *matches* apabila tidak sama maka sistem akan memberikan peringatan bahwa *password* tidak cocok seperti yang ditampilkan pada gambar 7.11 berikut.

The screenshot shows a registration form titled "Daftarkan Akun Anda!". It includes fields for name ("Yusniar Nur Syarif Sidig"), email ("yusniarnss33@gmail.com"), password ("Password Anda ..."), and password confirmation ("Ulangi Password Anda ..."). A red error message "Password Tidak Cocok!" (Passwords do not match) is displayed below the password fields. A blue "Daftar Akun" button is at the bottom, and links for password recovery and login are at the bottom right.

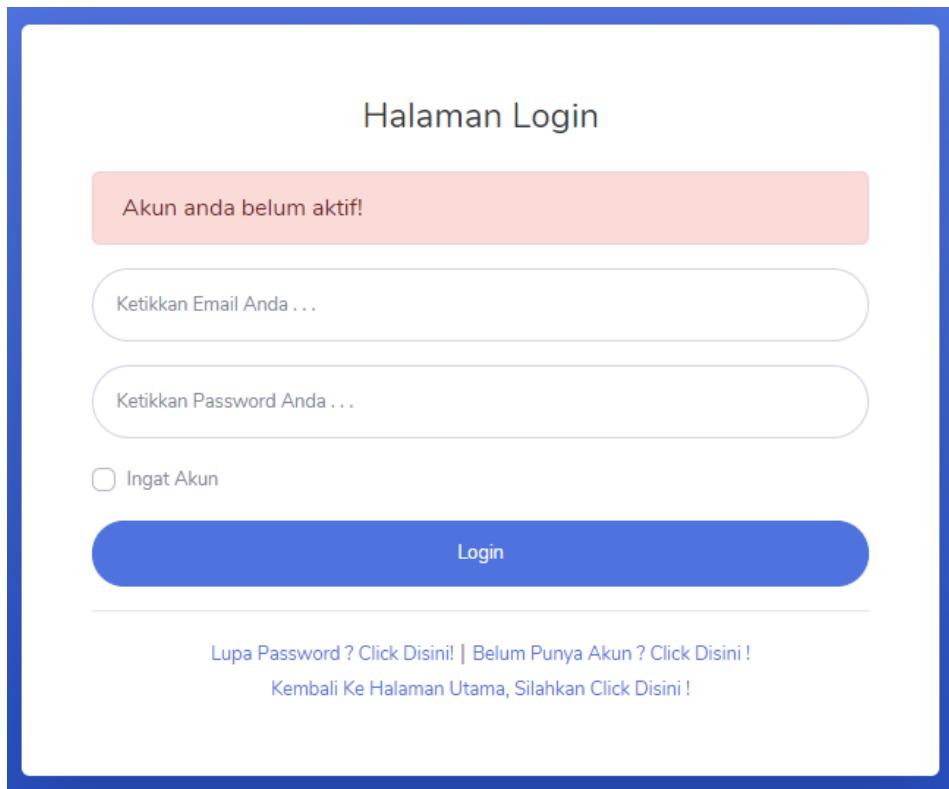
Gambar 7.11 Notifikasi Password Tidak Cocok

Apa yang akan terjadi bahwa data yang *user* kirimkan kepada sistem sudah benar ?, dimana sistem akan mengembalikan *user* ke halaman *login*, hal ini menandakan bahwa akun *user* tersebut sudah berhasil terdaftar pada sistem JURAGAN, perhatikan gambar 7.12 berikut.

The screenshot shows a login page titled "Halaman Login". A green success message box contains the text "Selamat akun anda telah terbuat, silahkan aktivasi akun anda!". Below it are fields for "Ketikkan Email Anda ..." and "Ketikkan Password Anda ...". There is an "Ingat Akun" checkbox and a blue "Login" button. At the bottom, there are links for password recovery and account activation.

Gambar 7.12 Akun User Berhasil Terdaftar

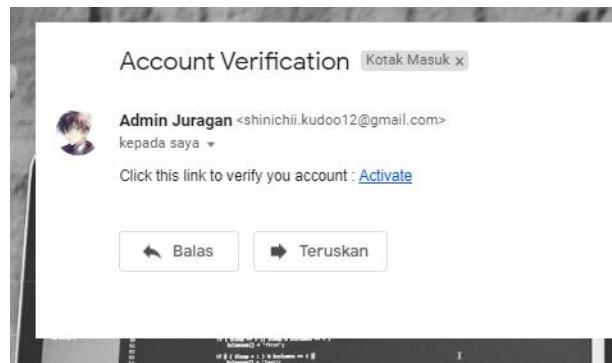
Selamat akun JURAGAN teman – teman sudah terdaftar, coba teman – teman lakukan *login* pada sistem JURAGAN dengan memberikan *email* dan *password* yang sudah terdaftar.



*Gambar 7.13 Notifikasi Akun Belum Aktif*

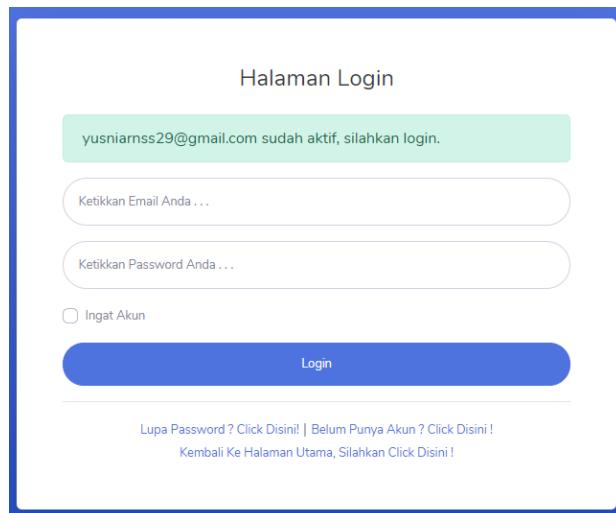
Kok malah muncul peringatan akun belum aktif sih, kenapa ?, coba teman – teman perhatikan kembali gambar 7.12, dimana terdapat peringatan bahwa teman – teman perlu melakukan aktivasi terlebih dahulu terhadap akun teman – teman. Lalu bagaimana cara aktivasi akun tersebut ?, seperti dijelaskan sebelumnya dimana teman – teman perlu melakukan aktivasi melalui *email* yang sudah di daftarkan tersebut agar akun teman – teman aktif dan dapat melakukan *login*. Coba teman – teman buka *email*

tersebut, dimana admin sistem JURAGAN akan mengirimkan pesan untuk melakukan aktivasi akun, seperti yang ditunjukkan pada gambar 7.14 berikut.



Gambar 7.14 Aktivasi Akun Melalui Email

Silahkan teman – teman klik *link activate* tersebut, dimana teman – teman akan dibawa kemabli menuju halaman *login* dan akun teman – teman sudah aktif sehingga dapat melakukan *login* pada sistem JURAGAN, perhatikan gambar 7.15 berikut.



Gambar 7.15 Akun User Telah Aktif

Perlu diingat bahwa sistem JURAGAN mempunyai dua halaman daftar akun yang dimana adalah halaman daftar akun untuk *user* dengan level pelanggan dan halaman daftar akun untuk *user* dengan level penjual. Halaman daftar akun bagi *user* penjual terlihat seperti pada gambar 7.16 sebagai berikut.

The screenshot shows a registration form titled "Daftarkan Toko Anda!" (Register Your Store!). It consists of several input fields and a checkbox. At the bottom, there is a large blue button labeled "Daftar Toko" (Register Store). Below the button, there are links for password recovery and login.

Daftarkan Toko Anda!

Nama Toko Anda ...

Email Toko Anda ...

Password Anda ...

Ulangi Password Anda ...

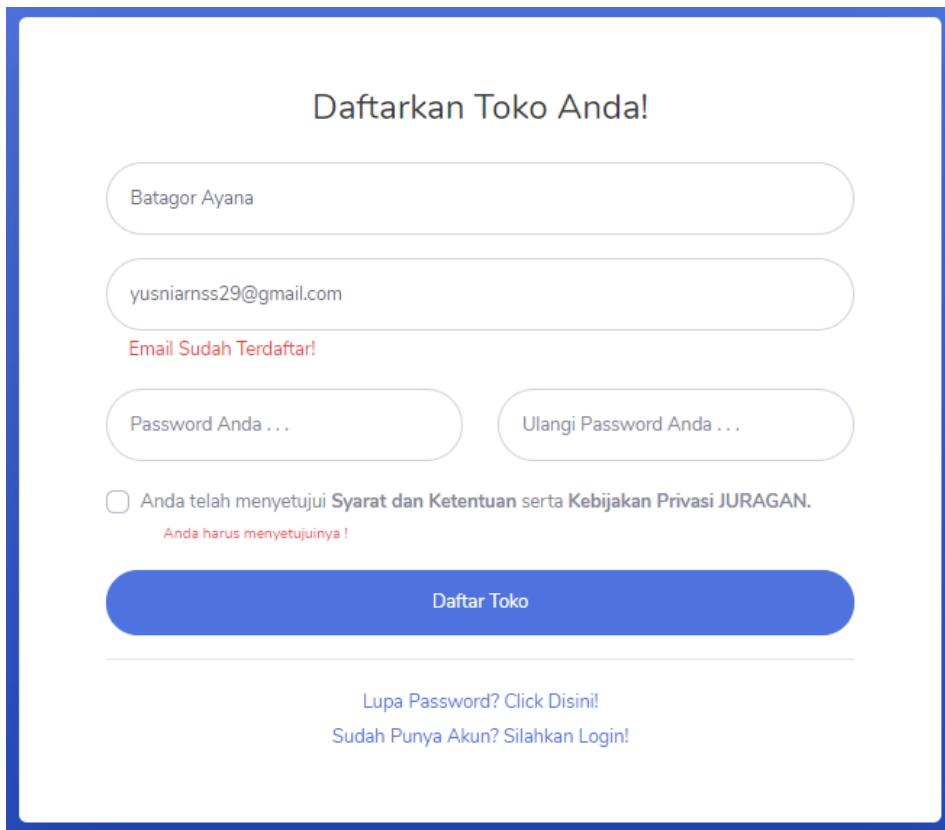
Anda telah menyetujui Syarat dan Ketentuan serta Kebijakan Privasi JURAGAN.

**Daftar Toko**

Lupa Password? Click Disini!  
Sudah Punya Akun? Silahkan Login!

Gambar 7.16 Halaman Daftar Akun Toko

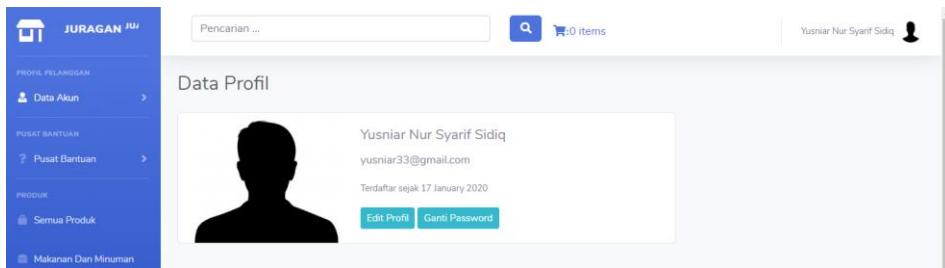
Mengenai prosedur pendaftaran akun toko tersebut terlihat sama seperti pendaftaran pada akun *user* pelanggan, hanya saja yang membedakan adalah dimana pada halaman daftar akun toko terdapat syarat dan ketentuan serta kebijakan privasi yang harus disetujui oleh *user* penjual. Dimana sistem JURAGAN hanya dapat menampung satu *email* sehingga *email* yang sudah terdaftar tidak dapat dipergunakan kembali.



Gambar 7.17 Peringatan Pada Pendaftaran Akun Toko

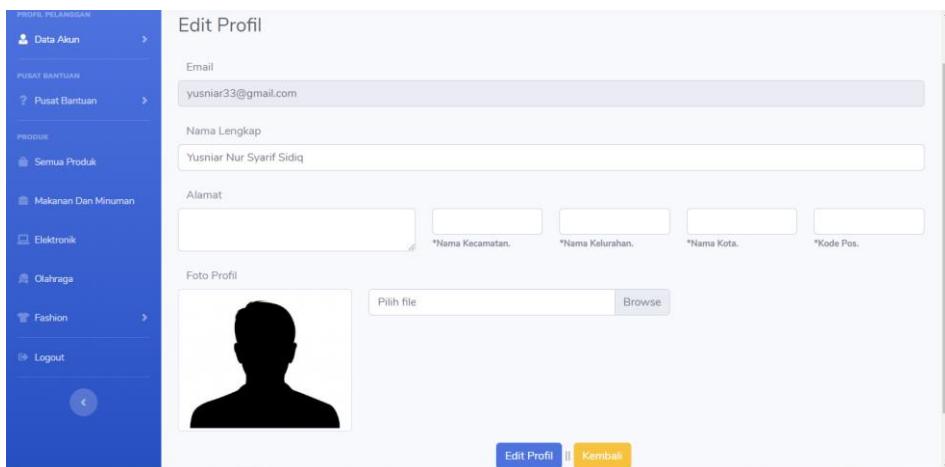
### 7.3 Mengelola Data Akun Pada Sistem JURAGAN

Halaman data akun tersebut dimana merupakan halaman untuk melakukan pengelolaan terhadap data *user*. Untuk mengelola akun *user* dimana terdapat halaman data profil yang berfungsi dalam menampilkan informasi dari *user* tersebut. Apabila teman – teman baru pertama kali mendaftar akun sistem JURAGAN maka informasi yang diberikan tidaklah seluruhnya, melainkan informasi yang teman – teman berikan saat melakukan proses daftar akun. Untuk mengakses halaman data profil dimana teman – teman perlu memilih menu “**Profil**” pada bagian data akun.



Gambar 7.18 Halaman Data Profil

Untuk melengkapi informasi – informasi tersebut dimana *user* perlu melakukan *edit profil*. Halaman *edit profil* dapat diakses dengan cara mengklik button “**Edit Profil**” pada halaman data profil atau melalui *sidebar* sistem JURAGAN.



Gambar 7.19 Halaman Edit Profil

Pada gambar 7.19 tersebut dimana merupakan halaman *edit profil*, dimana sistem akan melakukan *load* data dari *database* sehingga memunculkan informasi – informasi yang sudah diberikan pada sistem JURAGAN melalui pendaftaran akun. *User* dapat menambahkan informasi – informasi berupa alamat hingga foto profil *user* tersebut, akan

tetapi *email* tidak dapat diubah. Silahkan teman – teman tambahkan informasi – informasi yang dibutuhkan sistem untuk melengkapi data yang terdapat pada halaman data profil.

Edit Profil

Email  
yusniar33@gmail.com

Nama Lengkap  
Yusniar Nur Syarif Sidiq

Alamat  
Jl. A.H.Nasution Asrama Yon Zipur 9 Rt 05  
Rw 05 No. 247  Kec. Cinambo \*Nama Kecamatan.  
Kel. Pakemitan \*Nama Kelurahan.  
Kota Bandung \*Nama Kota.  
40294 \*Kode Pos.

Foto Profil  
 Screenshot\_2016-08-07-23-14-44-1.png Browse

**Edit Profil** || **Kembali**

*Gambar 7.20 Melakukan Pengeditan Data Profil*

Apabila teman – teman sudah selesai menambahkan informasi – informasi pada data akun teman – teman silahkan klik button “**Edit Profil**” yang dimana untuk mengirmkan data tersebut kepada sistem. Dimana teman – teman akan di bawa kembali ke halaman data profil, dimana yang datanya sudah bertambah dan lengkap, seperti yang ditampilkan pada gambar 7.21 berikut.

## Data Profil

Profil berhasil diedit!



Yusniar Nur Syarif Sidiq

yusniar33@gmail.com

Jl. A.H.Nasution Asrama Yon Zipur 9 Rt 05 Rw 05 No. 247  
Kec. Cinambo Kel. Pakemitan Kota Bandung 40294

Terdaftar sejak 17 January 2020

Edit Profil

Ganti Password

Gambar 7.21 Kelengkapan Data Profil User

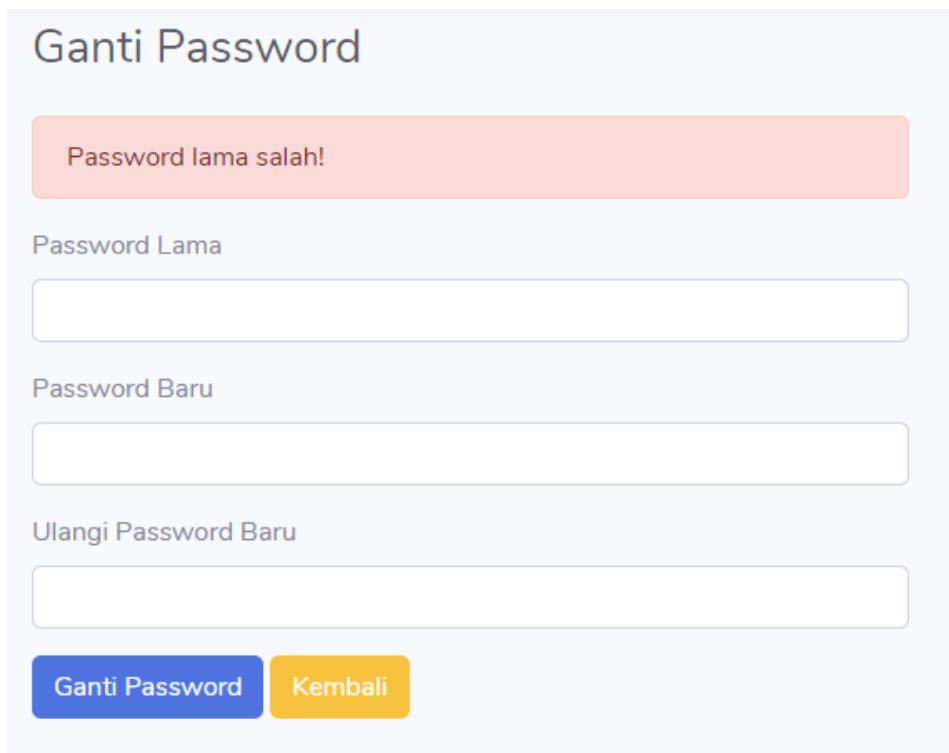
Teman – taman juga dapat merubah *password* yang sudah dibuat sebelumnya dengan cara megakses halaman ganti password. Untuk mengakses halaman ganti *password* tersebut dimana teman – teman bisa meng klik button “**Ganti Password**” yang terdapat pada halaman data profil atau melalui sidebar “**Data Akun**”.

The screenshot shows a user profile page for 'Yusniar Nur Syarif Sidiq'. On the left, there's a sidebar with navigation links: 'PROFILE PELANGGAN' (Data Akun), 'PUSAT BANTUAN' (Pusat Bantuan), 'PRODUK' (Semua Produk, Makanan Dan Minuman, Elektronik), and 'AKYAT GABUNGAN' (with a house icon). The main content area has a search bar and a cart icon showing '0 items'. Below the search bar, the title 'Ganti Password' is displayed. There are four input fields: 'Password Lama', 'Password Baru', 'Ulangi Password Baru', and a 'Kembali' button. At the bottom, there are two buttons: 'Ganti Password' (in blue) and 'Kembali' (in yellow).

Gambar 7.22 Halaman Ganti Password Pada Sistem JURAGAN

Untuk melakukan kegiatan ganti *password* tersebut dimana teman – teman perlu mengingat *password* pertama atau yang sedang teman – teman

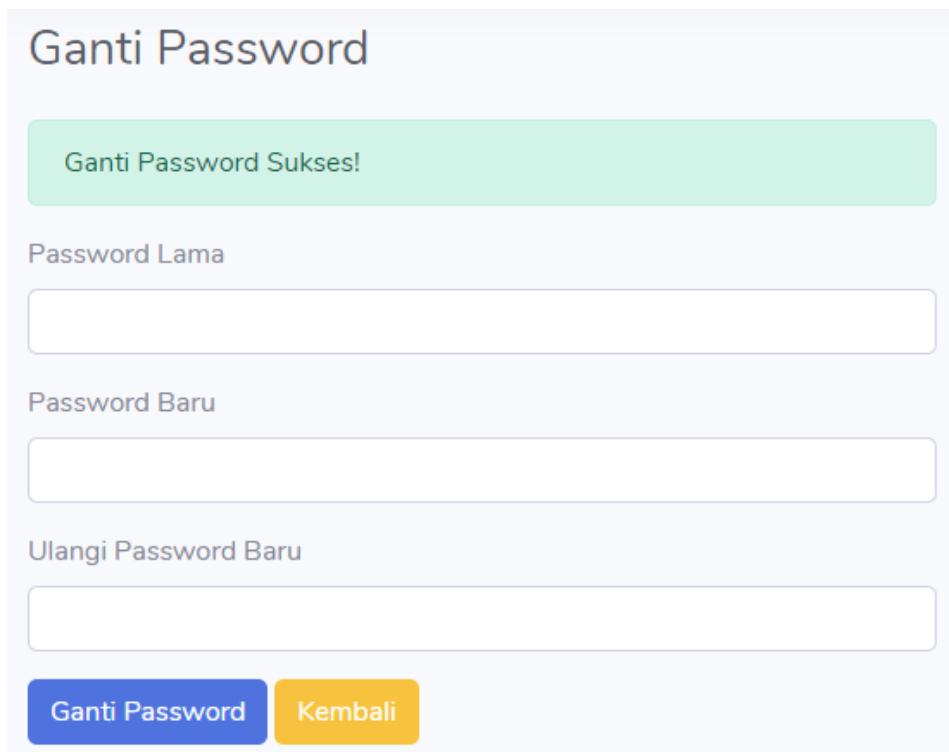
gunakan, namun apabila teman – teman mengalami lupa *password* dimana sistem JURAGAN sudah menyiapkan ftur **Lupa Password** tersebut yang akan kita bahas pada sub bab berikutnya. Terdapat *rules* yang diperlukan untuk melakukan kegiatan ganti *password* tersebut, dimana teman – teman harus mengingat *password* yang sedang digunakan. Apabila teman – teman salah memberikan *password* yang sedang digunakan maka sistem akan memberikan sebuah peringatan seperti yang ditampilkan pada gambar 7.23 berikut.



Gambar 7.23 Peringatan Password Lama Salah

Untuk *rules* pada kolom *password* baru dan ulangi *password* baru sama dengan saat melakukan daftar akun yaitu dimana kedua *password* tidak boleh kosong dan bersifat *matches*. Apabila teman – teman berhasil

melakukan proses ganti *password* dimana sistem akan memberikan respon tetep berada pada halaman ganti *password* tersebut, akan tetapi memunculkan notifikasi “**Ganti Password Sukses!**”. Silahkan teman – teman lakukan kegiatan ganti *password* tersebut dan klik button “**Ganti Password**”.

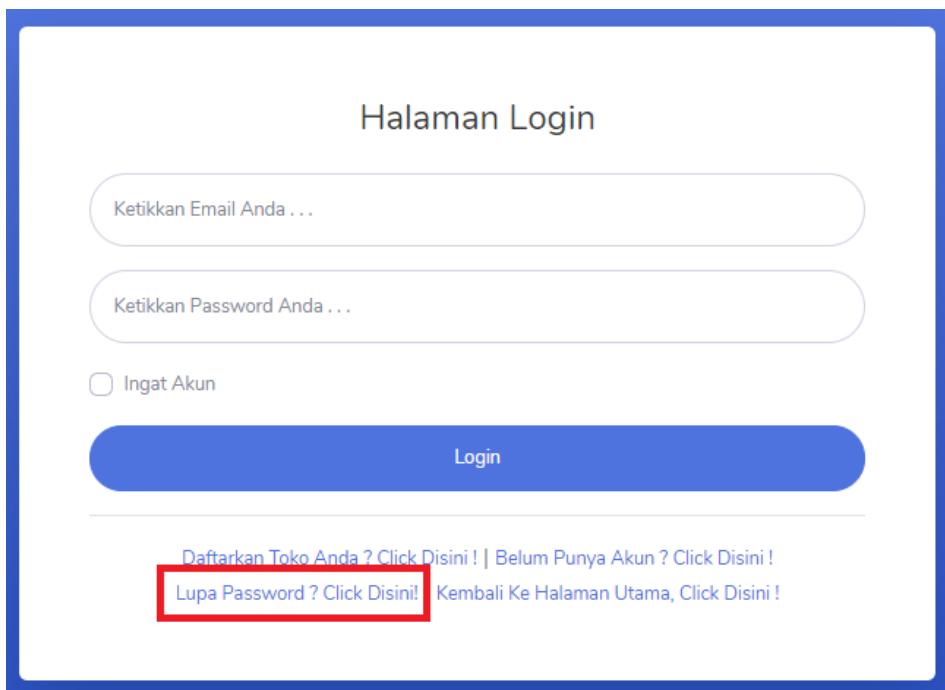


*Gambar 7.24 Proses Ganti Password Sukses*

#### **7.4 Proses Lupa Password Pada Sistem JURAGAN**

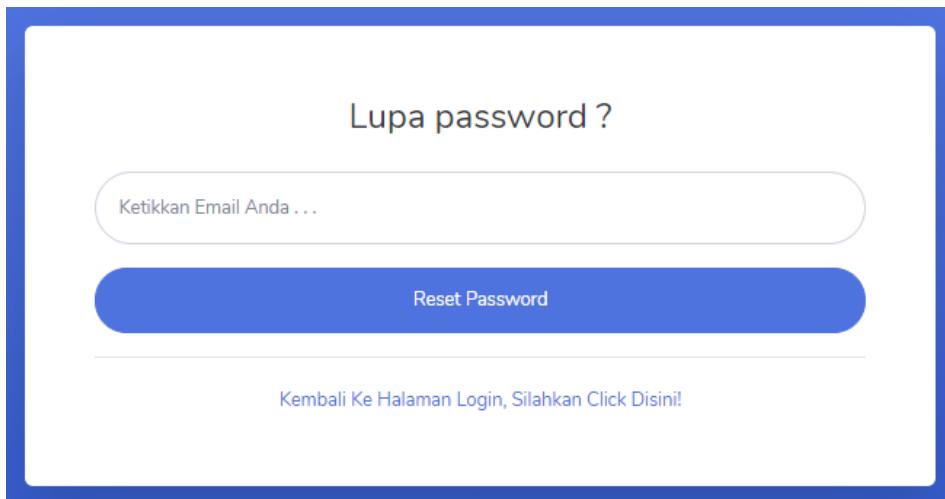
Pernahkah kalian mengalami kondisini dimana lupa terhadap *password* akun kalian ? Apa yang akan kalian lakukan jika hal tersebut terjadi ?. Sebagai orang pastinya pernah mengalami dimana kondisi lupa terhadap *password* akun yang sudah dibuat. Apabila teman – teman mengalami kondisi tersebut pada akun JURAGAN, apa yang harus teman

– teman lakukan ?. Pada sistem JURAGAN dimana sudah dilengkapi dengan fitur ganti *password* untuk mengantisipasi masalah tersebut. Untuk menggunakan *fitur* tersebut dimana teman – teman perlu mengakses halaman lupa *password* dengan cara mengklik *text* “**Lupa Password ? Click Disini!**” yang terdapat pada halaman *login*.



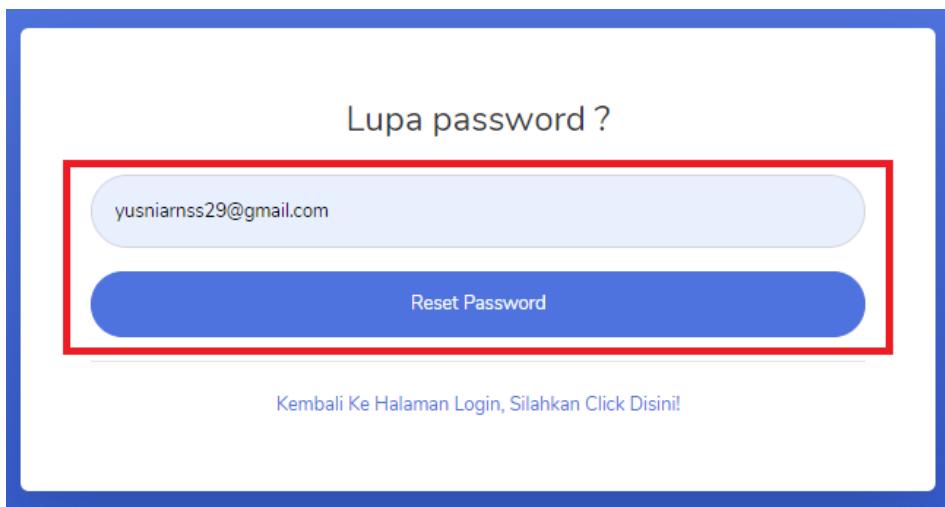
Gambar 7.25 Cara Mengakses Halaman Lupa Password

Perhatikan gambar 7.25 tersebut, apabila teman – teman mengklik *text* yang diberikan kotak merah tersebut, sistem akan membawa teman – teman kepada halaman lupa *password* yang terdapat pada sistem JURAGAN tersebut, perhatikan gambar 7.26 sebagai berikut.



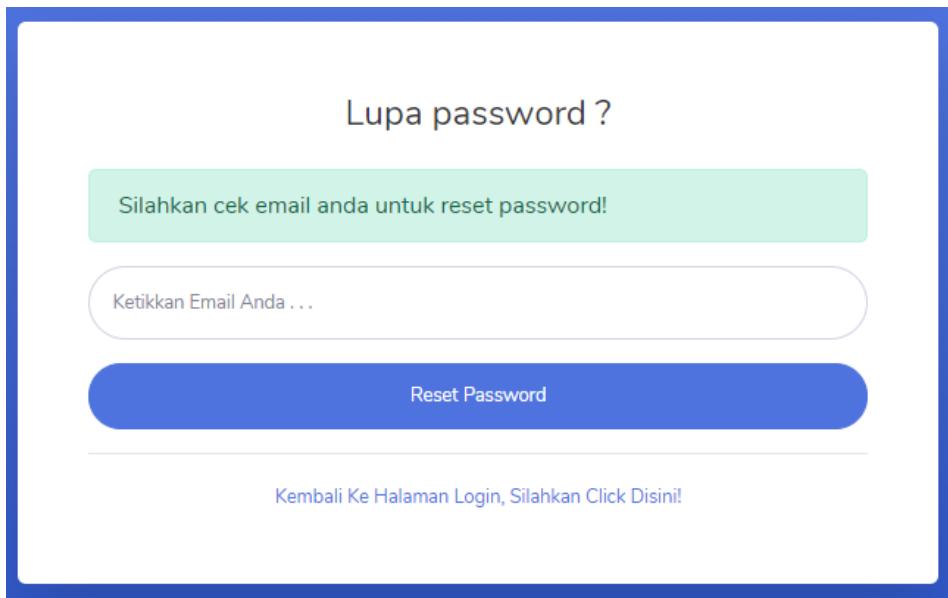
Gambar 7.26 Halaman Lupa Password Pada Sistem JURAGAN

Untuk menggunakan *fitur lupa password* tersebut dimana sistem membuatuhkan *email* yang sudah terdaftar dan aktif pada sistem JURAGAN. Silahkan teman – teman masukkan *email* yang akan di *reset password*, pada kolom *email* dan klik *button “Reset Password”*.



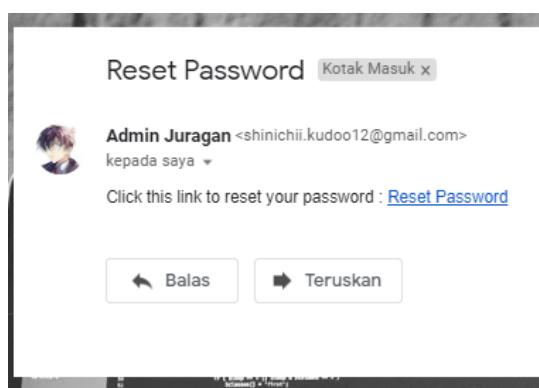
Gambar 7.27 Memberikan Email Yang Akan Di Reset Passwordnya

Saat **button “Reset Password”** di aplikasikan dimana sistem akan memberikan respon tetap berada pada halaman tersebut, namun memberikan informasi berupa notifikasi seperti pada gambar 7.28.



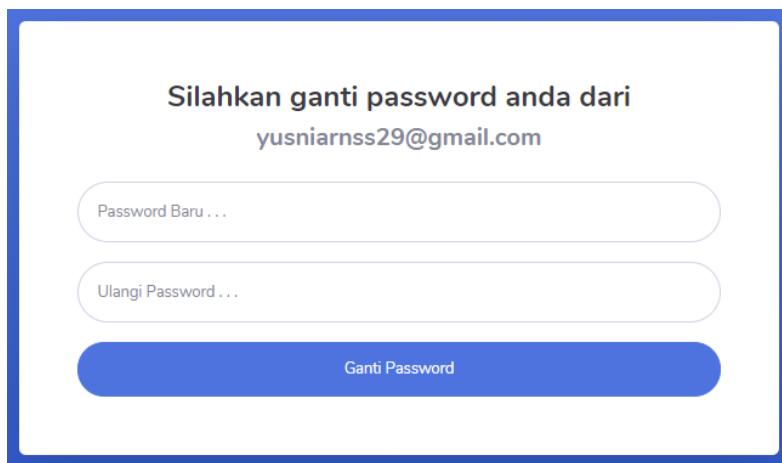
*Gambar 7.28 Notifikasi Cek Email Untuk Melakukan Reset Password*

Silahkan cek *email* teman – teman apakah sistem sudah mengirimkan *email* untuk melakukan *reset password* ?, perhatikan gambar 7.29 berikut.



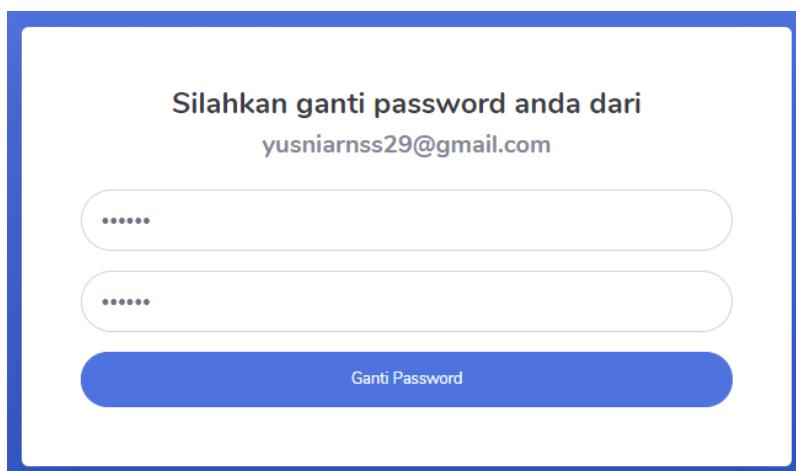
*Gambar 7.29 Email Untuk Melakukan Reset Password*

Silahkan teman – teman klik *text “Reset Password”*, dimana teman – teman akan dibawa ke halaman *reset password* pada sistem JURAGAN, seperti yang ditampilkan pada gambar 7.30 berikut.



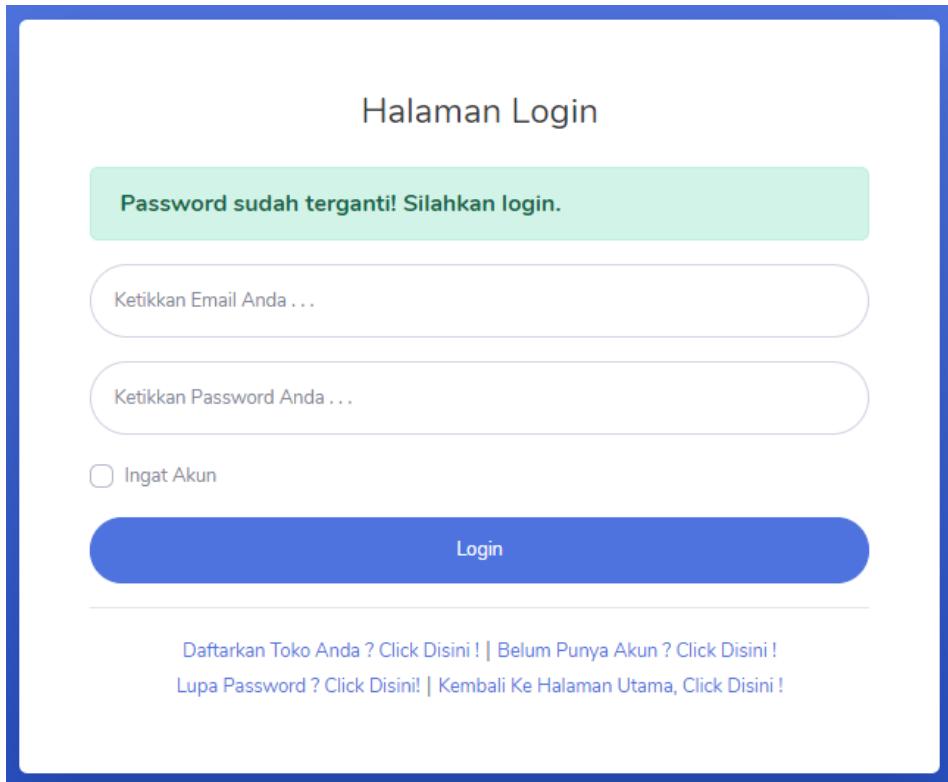
Gambar 7.30 Halaman Reset Password Sistem JURAGAN

Dimana *password* lama teman – teman sudah terhapus, silahkan teman – teman lakukan *input* terhadap *password* yang baru, lalu klik *button “Ganti Password”*.



Gambar 7.31 Membuat Password Baru

Dimana saat teman – teman mengaplikasikan *button* “**Ganti Password**” tersebut sistem akan membawa teman – teman kembali kehalaman *login* dan memberikan notifikasi bahwa *password* berhasil terganti. Selamat teman – teman dapat melakukan *login* kembali dengan *email* yang sama dan *password* yang baru.



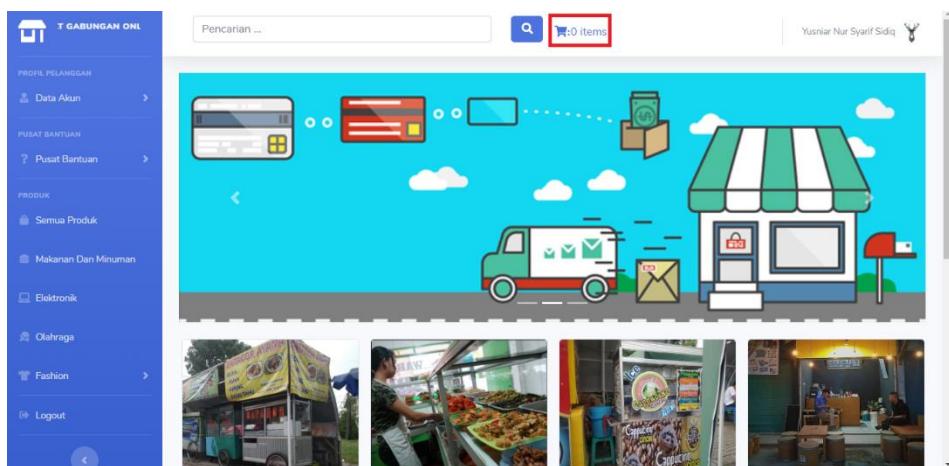
*Gambar 7.32 Fitur Lupa Password Berhasil Dilakukan*

## **7.5 Mengelola Keranjang Belanja Pada Sistem JURAGAN**

Keranjang belanja merupakan salah satu *fitur* utama pada sebuah *website e-commerce*, dimana fungsinya adalah menampung *items – items* yang akan di beli oleh *user*. Dalam sistem JURAGAN tersebut dimana sudah tersedia fitur keranjang belanja tersebut dan hanya dapat diakses

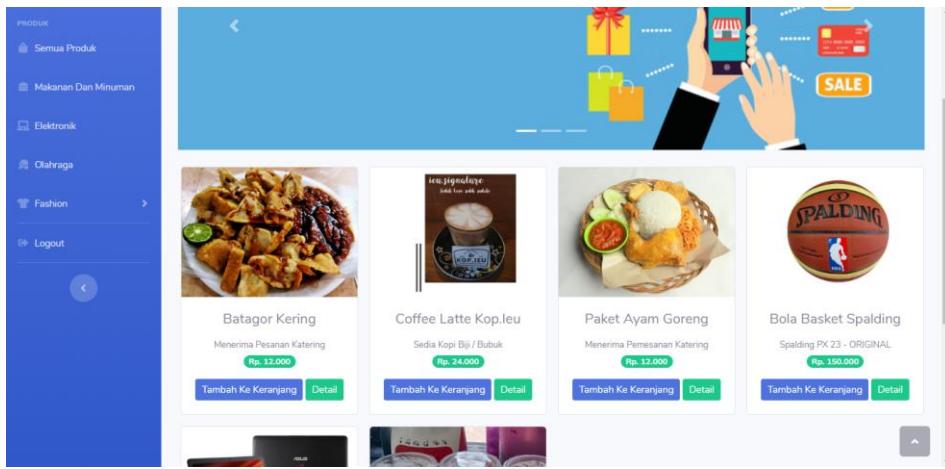
oleh *user* dengan level pelanggan. Untuk menggunakan fitur tersebut dimana teman – teman dapat mengikuti *tutorial* mengenai cara mengelola keranjang belanja teman – teman.

Langkah pertama dimana teman – teman perlu melakukan *login* terlebih dahulu terhadap sistem JURAGAN sehingga dapat masuk ke halaman utamanya, seperti pada gambar 7.33 berikut.



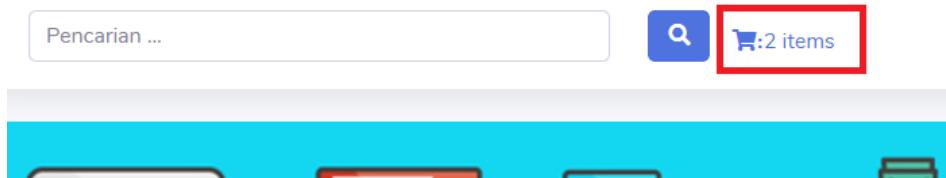
*Gambar 7.33 Keranjang Belanja Belum Terisi*

Perhatikan bagian *topbar* pada sistem JURAGAN tersebut, dimana terdapat *icon* keranjang dan masih belum terisi. Silahkan teman – teman tambahkan beberapa produk yang ditawarkan oleh sistem JURAGAN tersebut. Teman – teman dapat melihat semua produk yang terdapat pada sistem JURAGAN dengan cara mengakses halaman produk. Untuk mengakses halaman produk tersebut dimana teman – teman dapat menemukan menu “**Semua Produk**” pada bagian *sidebar* sistem JURAGAN.



Gambar 7.34 Halaman Produk Sistem JURAGAN

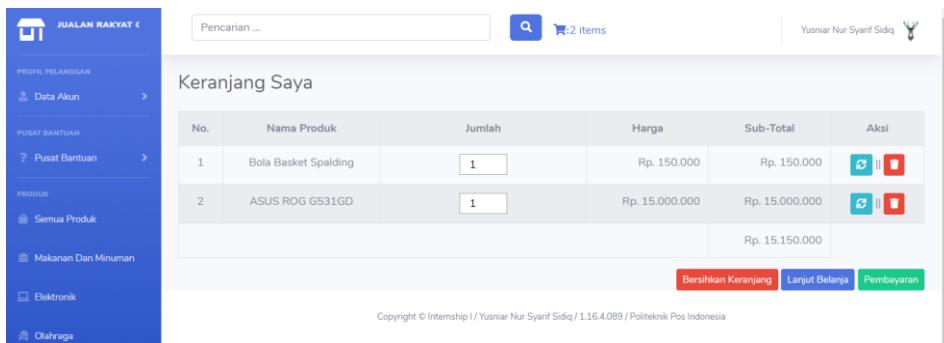
Perhatikan gambar 7.34 tersebut dimana merupakan tampilan dari halaman produk sistem JURAGAN. Pada halaman tersebut semua produk yang ditawarkan oleh para penjual akan ditampilkan. Terdapat dua *button* pada halaman tersebut yaitu *button* “Tambah Ke Keranjang” dan *button* “Detail” yang sebelumnya sudah kita bahas pada sub bab halaman awal sistem JUAGAN. Silahkan teman – teman tambahkan beberapa produk tersebut dengan mengaplikasikan *button* “Tambah Ke Keranjang”.



Gambar 7.35 Keranjang Belanja Terisi

Perhatikan gambar 7.35 tersebut dimana *icon* belanja kita sudah terisi 2 *items*. Untuk melihat isi dari keranjang belanja tersebut dimana teman – teman perlu mengakses halaman keranjang belanja dengan cara klik saja

*icon* keranjang tersebut dan sistem akan membawa teman – teman ke halaman keranjang belanja.



The screenshot shows a user interface for a shopping cart. At the top, there's a navigation bar with a house icon, the text 'JUALAN RAKYAT', a search bar containing 'Pencarian ...', a magnifying glass icon, a shopping cart icon with '2 items', and a user profile 'Yusniar Nur Syarif Sidiq' with a small profile picture. On the left, there's a sidebar with 'PROFIL PELANGGAN' and 'Data Akun'. Below that are sections for 'PURAT BANTUAN' (with 'Pusat Bantuan'), 'PRODUK' (with 'Semua Produk', 'Makanan Dan Minuman', 'Elektronik', and 'Olahraga'), and a 'Pencarian ...' input field. The main content area is titled 'Keranjang Saya' and displays a table of items:

No.	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Bola Basket Spalding	<input type="text" value="1"/>	Rp. 150.000	Rp. 150.000	
2	ASUS ROG G531GD	<input type="text" value="1"/>	Rp. 15.000.000	Rp. 15.000.000	
					Rp. 15.150.000

At the bottom right of the table are three buttons: 'Bersihkan Keranjang' (Red), 'Lanjut Belanja' (Blue), and 'Pembayaran' (Green). A copyright notice at the bottom center reads: 'Copyright © Internship I / Yusniar Nur Syarif Sidiq / 1.16.4.089 / Politeknik Pos Indonesia'.

*Gambar 7.36 Halaman Keranjang Belanja Sistem JURAGAN*

Pada halaman keranjang belanja tersebut dimana teman – teman dapat melihat informasi – informasi mengenai barang belanjaan teman – teman mulai dari nama *items*, jumlah *item*, harga satuan dari *item* tersebut, total harga dari *item* tersebut, dan total harga keseluruhan isi dari keranjang belanja teman – teman. Disini saya akan mencontohkan bagaimana mengelola keranjang belanjaan teman – teman. Apabila teman – teman ingin menambahkan dan mengurangkan jumlah *item* maka teman – teman dapat memanfaatkan *fitur scroll number* yang terdapat pada *field* jumlah. Apabila sudah dirubah silahkan teman – teman aplikasikan *button update* yang terdapat pada *field* aksi, maka data pada keranjang tersebut akan ter *update*.

Keranjang Saya					
Keranjang berhasil diperbarui!					
No.	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Bola Basket Spalding	<input type="text" value="2"/> 	Rp. 150.000	Rp. 300.000	
2	ASUS ROG G531GD	<input type="text" value="1"/> 	Rp. 15.000.000	Rp. 15.000.000	
				Rp. 15.300.000	

[Bersihkan Keranjang](#) [Lanjut Belanja](#) [Pembayaran](#)

Copyright © Internship I / Yusniar Nur Syarif Sidiq / 1.16.4.089 / Politeknik Pos Indonesia

*Gambar 7.37 Menambah Dan Mengurangkan QTY*

Apabila teman – teman ingin menghapus salah satu *items* pada keranjang belanja, teman – teman dapat mengaplikasikan *button* hapus yang terdapat pada *field* aksi.

Keranjang Saya					
Item berhasil dihapus!					
No.	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Bola Basket Spalding	<input type="text" value="2"/> 	Rp. 150.000	Rp. 300.000	
					Rp. 300.000

[Bersihkan Keranjang](#) [Lanjut Belanja](#) [Pembayaran](#)

Copyright © Internship I / Yusniar Nur Syarif Sidiq / 1.16.4.089 / Politeknik Pos Indonesia

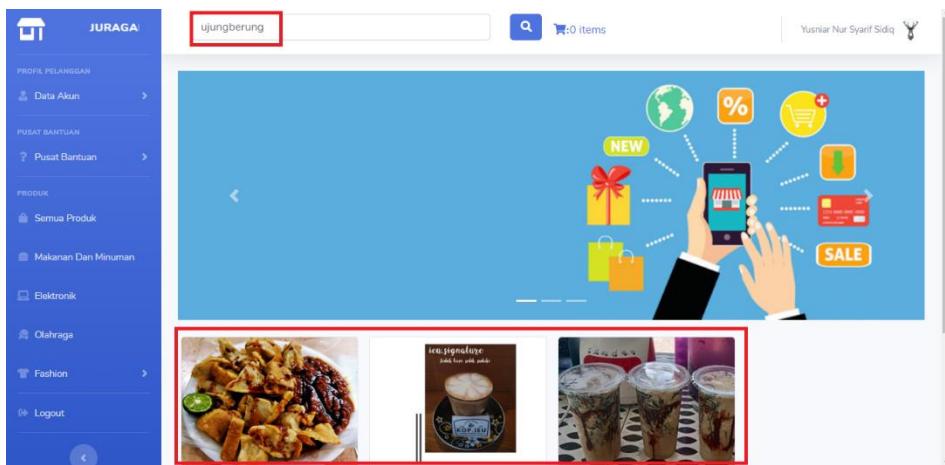
*Gambar 7.38 Menghapus Salah Satu Items Pada Halaman Keranjang Belanja*

Pada halaman keranjang belanja tersebut dimana terdapat 3 *button* yang letaknya diluar *table*. Dimana *button* “**Bersihkan Keranjang**” berfungsi untuk menghapus keseluruhan *items* yang berada pada halaman keranjang tersebut. *Button* “**Lanjut Belanja**” berfungsi apabila *user* ingin melakukan belanja lagi, dengan mengaplikasikan *button* tersebut dimana

sistem akan membawa *user* kembali ke halaman produk. *Button* “**Pembayaran**” berfungsi sebagai *button* untuk melanjutkan proses transaksi pada *items* yang berada pada keranjang belanja teman – teman.

## 7.6 Menerapkan Fitur Komunitas

Terdapat *fitur* unggulan yang dimiliki oleh sistem JURAGAN tersebut yaitu dimana sistem JURAGAN dibuat dengan konsep *e-commerce* berbasis komunitas. Fitur komunitas ini dimana memungkinkan *user* untuk mencari sesuatu berdasarkan komunitas yang diberikan. Contoh dimana *user* ingin mengetahui produk apa saja sih yang dijual pada daerah ujungberung. Pada kasus ini dimana *user* ingin mengetahui produk – produk apa saja yang berdapa pada komunitas daerah ujungberung tersebut. Dimana *user* dapat menggunakan *fitur* pencarian yang berada pada bagian *topbar* dari sistem JURAGAN dan masukkan kata kunci komunitasnya. Pada kasus ini *user* ingin mengetahui produk yang berada pada daerah ujungberung, maka ketikkan “**Ujungberung**” pada kolom pencarian tersebut lalu klik *button* pencarinya maka sistem akan memberikan informasi mengenai produk – produk apa saja yang dapat ditemukan oleh *user* tersebut di daerah ujungberung.



Gambar 7.39 Menampilkan Produk Berdasarkan Komunitas

## **DAFTAR PUSTAKA**

- [1] P. Chung, R. C. Yeh and Y. C. Chen, "Implementation Of E-Logistics Systems For Developing EC Capability In Small And," *International Journal of e-Education*, 2015.
- [2] N. Na Rahman, Z. Sulaiman, A. B. A Hamid and Z. Khalifah, "The Implementation of E-Commerce Application In Bumiputera Small and Medium," *International Journal of Advances in Management and Economics*, 2018.
- [3] M. Kartiwi, H. Hussin, M. A. Suhaimi, M. Razi and Mohamed, "Impact Of External Factors On Determining E-Commerce Benefits Among SMEs in," *Journal Of Global Entrepreneurship Research*, 2018.
- [4] R. Rahayu and J. Day, "Determinant Factors Of E-Commerce Adoption by SMEs in Developing Country: Evidence," *ScienceDirect*, vol. 195, no. 3, pp. 142-150, 2015.
- [5] M. Falk and E. Hagten, "E-Commerce Trend And Impacts Across Europe," *ScienceDirect*, vol. 170, pp. 357-369, 2015.
- [6] A. S. Tiruvenee and D. Ladkoo, "Wholesale And Retail E-Commerce In Mauritius: Views Of Customers And Employees," *Studies in Business and Economics*, vol. 10, no. 2, pp. 170-186, 2015.

- [7] I. and A. B. Pradipta, "Pemanfaatan Sosial Media Untuk Meningkatkan Market Share UKM," *TEKNOMATIKA*, vol. 8, no. 1, Juli 2015.
- [8] A. K. Sherlyanita and N. A. Rakhmawati, "Pengaruh Dan Pola Aktivitas Penggunaan Internet Serta Media Sosial Pada Siswa SMPN 52 Surabaya," *Journal of Information System Engineering and Business Intellogence*, vol. 2, no. 1, pp. 17-22, April 2016.
- [9] Pradana and M. , "Klasifikasi Bisnis E-Commerce Di Indonesia," *MODUS*, vol. 27, no. 2, pp. 163-174, 2015.
- [10] L. W. Santoso, K. A. Sahertian and D. H. Setiabudi, "Pembuatan Website Untuk Komunitas PPKM," *Jurnal Infra*, vol. 1, no. 5, pp. 266-270, 2017.
- [11] Nurcahyono and Fendi, "Pembangunan Aplikasi Penjualan Dan Stok Barang Pada Toko Nuansa Elektronik Pacitan," *Journal Speed - Sentra Penelitian Engineering Dan Edukasi*, vol. 3, p. 4, 2017.
- [12] N. Prokofyeva and V. Boltunova, "Analysis And Practical Application Of PHP Framework In Development Of Web Information System," *ScienceDirect*, vol. 104, pp. 51-56, December 2016.
- [13] O. Betari, M. Erramdani, S. Roubi, K. Arrhioui and S. Mbarki, "Model Transformation In The MOF Meta-Modeling Architecture: From UML to Codeigniter PHP Framework," *Springer*, pp. 227-234, 2017.

- [14] H. Yu, H. Xu, Y. Xu, Z. Li, P. Wang and P. Wang, "Construction and Evaluation of PHP-Based Management and Training System For Electrical Power Laboratory," pp. 371-381, 2016.
- [15] R. Sovia and J. Febio, "Membangun Aplikasi E-Library Menggunakan HTML, PHP Script, Dan MySQL Database," *Processor*, vol. 6, no. 2, 2017.
- [16] I. Warman and R. Ramdaniansyah, "Analisis Perbandingan Kinerha Query Database Management System (DBMS) Antara MySQL 5.7.16 Dan MariaDB 10.1," *Jurnal TEKNOIF*, vol. 6, no. 1, 1 April 2018.