

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

MEMBANGUN WEB SITE E-COMMERCE

Dengan Framework Codeigniter

Yusniar Nur Syarif Sidiq

Politeknik Pos Indonesia



Kreatif Industri Nusantara

‘Gunakan Waktu
Luangmu Bersama
Buku, Kelak Ilmu
Baru Akan Kau
Dapatkan’

CONTRIBUTORS

Yusniar Nur Syarif Sidiq, Noviana Riza, Politeknik Pos Indonesia, Bandung,
Indonesia

CONTENTS IN BRIEF

DAFTAR ISI

DAFTAR GAMBAR	x
DAFTAR TABEL.....	xiv
DAFTAR SOURCE CODE.....	xv
FOREWORD	xvii
KATA PENGANTAR	xviii
ACKNOWLEDGMENTS	xi
ACRONYM	xx
GLOSSARY	xxi
SYMBOLS.....	xxii
INTODUCTION.....	xxiii
BAB 1 MENGENAL SISTEM JURAGAN	1
BAB 2 LANDASAN TEORI.....	5
2.1 Usaha Kecil Menengah (UKM)	6
2.1.1 Usaha Mikro	6
2.1.2 Usaha Kecil	6
2.1.3 Usaha Menengah	7
2.2 Pedagang Kaki Lima (PKL).....	7
2.3 Promosi.....	7
2.4 Internet	7
2.5 E-Commerce.....	8
2.6 Komunitas	8
2.7 Aplikasi	8
2.8 Framework	9
2.9 CodeIgniter.....	9

2.10 Hypertext Preprocessor (PHP)	9
2.11 Database (Basis Data).....	10
2.12 MariaDB	10
BAB 3 PERANCANGAN SISTEM JURAGAN	11
3.1 <i>Use Case</i> JURAGAN	12
3.1.1 Definisi Aktor.....	16
3.1.2 Definisi <i>Use Case</i>	17
3.1.3 Scenario <i>Use Case</i> JURAGAN.....	18
3.2 Perancangan Basis Data (<i>Database</i>)	26
3.2.1 Tabel <i>User</i>	27
3.2.2 Tabel <i>User_token</i>	29
3.2.3 Tabel tb_barang.....	30
3.3 <i>User Interface</i> Sistem JURAGAN	31
BAB 4 SOFTWARE PENDUKUNG	47
3.1 Xampp	48
4.1.1 Perbedaan PHP 7 Dan PHP 5	54
4.2 Visual Studio Code.....	54
4.2.1 <i>Extensions Visual Studio Code</i>	60
4.3 <i>Codeigniter</i>	62
4.3.1 <i>Installasi Framework Codeigniter</i>	63
BAB 5 FRAMEWORK CODEIGNITER	69
5.1 Sejarah Codeigniter	70
5.2 Kelebihan Codeigniter.....	70
5.3 Struktur Direktori Codeigniter	71
5.4 Konfigurasi Framework Codeigniter	76
5.5 Membuat CRUD Dengan Codeigniter	84
5.5.1 Membuat <i>Database</i>	85
5.5.2 Membuat Controllers.....	88
5.5.3 Membuat <i>Models</i>	101
5.5.4 Membuat <i>View</i>	109
BAB 6 E – COMMERCE DAN CODEIGNITER	127

6.1 Mengubungkan Boostrap Dengan CI.....	128
6.2 Fungsi Form Validation	141
6.3 Membuat Fitur Keranjang Belanja.....	149
6.4 Membuat Fitur Pencarian	163
BAB 7 PANDUAN MENGGUNAKAN SISTEM JURAGAN	169
7.1 Halaman Awal Sistem JURAGAN	170
7.2 Halaman Registrasi Dan Login Sistem JURAGAN.....	175
7.3 Mengelola Data Akun Pada Sistem JURAGAN.....	183
7.4 Proses Lupa Password Pada Sistem JURAGAN.....	188
7.5 Mengelola Keranjang Belanja Pada Sistem JURAGAN.....	193
7.6 Menerapkan Fitur Komunitas.....	197
7.7 Melakukan Verifikasi Akun Toko.....	198
7.8 Melakukan Proses <i>Logout</i>	200

DAFTAR PUSTAKA

DAFTAR GAMBAR

Gambar 3.1 Komponen Aktor Use Case.....	13
Gambar 3.2 Komponen UseCase	13
Gambar 3.3 Komponen Use Case Subject.....	14
Gambar 3.4 Relasi Association.....	14
Gambar 3.5 Relasi Generalization	15
Gambar 3.6 Relasi Despendency	15
Gambar 3.7 Use Case Sistem JURAGAN	16
Gambar 3.8 Perancangan UI Halaman Utama.....	32
Gambar 3.9 Perancangan UI Halaman Login	34
Gambar 3.10 Perancangan UI Halaman Daftar Akun	35
Gambar 3.11 Perancangan UI Halaman Lupa Password	36
Gambar 3.12 Perancangan UI Halaman Reset Password	37
Gambar 3.13 Perancangan UI Halaman Utama Kondisi Login.....	38
Gambar 3.14 Perancangan UI Halaman Data Profil.....	39
Gambar 3.15 Perancangan UI Halaman Edit Profil	40
Gambar 3.16 Perancangan UI Halaman Ganti Password	41
Gambar 3.17 Perancangan UI Halaman Profil Toko	42
Gambar 3.18 Perancangan UI Halaman Produk Toko.....	43
<i>Gambar 3.19 Perancangan UI Halaman Input Produk</i>	44
Gambar 3.20 Perancangan UI Halaman Keranjang Belanja.....	45
Gambar 4.1 Website Resmi Xampp.....	48
Gambar 4.2 Tahap Dua Dalam Mendownload Xampp	49
Gambar 4.3 Halaman Versi Xampp.....	49
Gambar 4.4 Proses Awal Installasi Xampp	50
Gambar 4.5 Pemilihan Komponen - Komponen Pada Software Xampp.....	50
Gambar 4.6 Pemilihan Tempat Penyimpanan Software Xampp	51
Gambar 4.7 Tempat Penyimpanan Xampp	51
Gambar 4.8 Proses Installasi Xampp Sedang Berjalan.....	52
Gambar 4.9 Tampilan Sofware Xampp	52
<i>Gambar 4.10 Menjalankan Module Apache Dan MySQL.....</i>	53
Gambar 4.11 Halaman Localhost	53
Gambar 4.12 Halaman Website Resmi VSCode	56
Gambar 4.13 Memilih Halaman Download VSCode	56
Gambar 4.14 Pemilihan OS Saat Download VSCode	56
Gambar 4.15 Proses Awal Installasi VSCode.....	57

Gambar 4.16 Proses Installasi VSCode Tahap Dua.....	57
Gambar 4.17 Pemilihan Tempat Penyimpanan VSCode	58
Gambar 4.18 Pemilihan Additional Tasks VSCode	58
Gambar 4.19 Proses Installasi VSCode	59
Gambar 4.20 Tampilan Visual Studio Code	59
Gambar 4.21 Extension Auto Rename Tag	60
Gambar 4.22 Extension PHP Intelephense	60
Gambar 4.23 Extension PHP Intellisese For CI.....	61
Gambar 4.24 Setting Awal VSCode	61
Gambar 4.25 Setting Kedua VSCode	62
Gambar 4.26 Konsep MVC	63
Gambar 4.27 Website Remi Codeigniter	64
Gambar 4.28 Pilihan Download CI.....	64
Gambar 4.29 Proses Download Codeigniter.....	65
Gambar 4.30 Memindahkan File Codeigniter	65
Gambar 4.31 Isi File Codeigniter.....	66
Gambar 4.32 Menjalankan Software Xampp.....	66
Gambar 4.33 Membuka Web Browser	67
Gambar 4.34 Menguji Framework Codeigniter.....	67
Gambar 4.35 Pengujian CI Berhasil	68
Gambar 5.1 Dokumentasi Codeigniter	71
Gambar 5.2 Struktur Direktori CI.....	71
<i>Gambar 5.3 Isi Dari Direktori Application</i>	72
Gambar 5.4 File Dalam Folder Config	73
Gambar 5.5 Contoh Script Code Controller	74
Gambar 5.6 Pemanggilan Fungsi Helper	74
Gambar 5.7 Contoh Script Code Model	75
Gambar 5.8 Pemanggilang Fungsi Model Pada Autoload.....	75
Gambar 5.9 Pemanggilan Fungsi Model Dari Controller.....	76
Gambar 5.10 Folder CI JURAGAN.....	77
Gambar 5.11 Konfigurasi Libraries Sistem JURAGAN	77
Gambar 5.12 Konfigurasi Fungsi Helpers Sistem JURAGAN.....	78
Gambar 5.13 Konfigurasi Models Pada Sistem JURAGAN	79
Gambar 5.14 Konfigurasi File Config Sistem JURAGAN	79
Gambar 5.15 Konfigurasi index_page Sistem JURAGAN.....	80
Gambar 5.16 Membuat File Baru .htaccess	80
Gambar 5.17 Membuka Dokumentasi CI	81
Gambar 5.18 Mencari Dokumentasi htaccess	81
Gambar 5.19 Script Code htaccess	82
Gambar 5.20 Mengisikan File .htaccess	82
Gambar 5.21 Konfigurasi Default Controller Sistem JURAGAN.....	83
Gambar 5.22 Konfigurasi Database Sistem JURAGAN	84

Gambar 5.23 Mempersiapkan Software VSCode	85
Gambar 5.24 Dashboard Xampp.....	86
Gambar 5.25 Halaman phpMyAdmin.....	86
Gambar 5.26 Halaman Database Pada phpMyAdmin	87
Gambar 5.27 Membuat Tabel Pada Database.....	88
Gambar 5.28 Hasil View Inventory	113
Gambar 5.29 Halaman Input Barang	118
Gambar 5.30 Halaman Edit Barang	122
Gambar 5.31 Halaman Detail Barang	125
Gambar 6.1 Halaman Start Boostrap SB ADMIN 2	129
Gambar 6.2 Menu Download Template Boostrap SB ADMIN 2	130
Gambar 6.3 Meng Extract File Zip Boostrap.....	130
Gambar 6.4 File Codeigniter Pada Direktori Xampp/htdocs	130
Gambar 6.5 Membuat Folder Vendor	131
Gambar 6.6 Memindahkan File SB ADMIN 2 Ke Dalam Vendor.....	131
Gambar 6.7 Tampilan View Welcome_message.php	132
Gambar 6.8 Membuat Folder Assets	132
Gambar 6.9 Mengisi Folder Assets.....	133
Gambar 6.10 Membuat Folder templates Pada Views.....	133
Gambar 6.11 Membuat 4 File PHP Pada Folder Templates	134
Gambar 6.12 Isi File Header.php	134
Gambar 6.13 Isi File Sidebar.php	135
Gambar 6.14 Isi File Topbar.php	136
Gambar 6.15 Isi File Footer.php	137
Gambar 6.16 Hasil Yang Diberikan Source Code 6.2	139
Gambar 6.17 Hasil Dari Source Code 6.3.....	141
Gambar 6.18 Contoh Form Validation Pada Sistem JURAGAN	142
Gambar 6.19 Praktek Form Validation Pada Field Name	144
Gambar 6.20 Praktek Form Validation Terhadap Field Email	146
Gambar 6.21 Praktek Form Validation Pada Field Password.....	147
Gambar 6.22 Mengakses Dokumentasi Shopping Cart.....	150
Gambar 6.23 View Halaman Keranjang Belanja.....	154
Gambar 6.24 Menjalankan Function Add Pada Controllers Cart	154
Gambar 6.25 Rowid Delete.....	156
Gambar 6.26 Function Delete Berhasil Diterapkan	157
Gambar 6.27 Banyaknya Items Pada Halaman Keranjang Belanja.....	157
Gambar 6.28 Fitur Search Pada Topbar Dalam Sistem JURAGAN	164
Gambar 6.29 Halaman Pencarian Pada Sistem JURAGAN	168
Gambar 7.1 Halaman Awal Sistem JURAGAN	170
Gambar 7.2 Mengaplikasikan Button Detail Toko Pada Halaman Awal ...	171
Gambar 7.3 Halaman Detail Toko Pada Sistem JURAGAN	172
Gambar 7.4 Halaman Produk Toko	172

Gambar 7.5 Halaman Detail Produk	173
Gambar 7.6 Peringatan Wajib Login Terlebih Dahulu.....	174
Gambar 7.7 Halaman Daftar Akun Pelanggan Pada Sistem JURAGAN ...	175
Gambar 7.8 Peringatan Kolom Tidak Boleh Kosong	176
Gambar 7.9 Notifikasi Format Email Tidak Sesuai	177
Gambar 7.10 Notifikasi Password Terlalu Pendek	178
Gambar 7.11 Notifikasi Password Tidak Cocok.....	178
Gambar 7.12 Akun User Berhasil Terdaftar	179
Gambar 7.13 Notifikasi Akun Belum Aktif.....	180
Gambar 7.14 Aktivasi Akun Melalui Email	181
Gambar 7.15 Akun User Telah Aktif.....	181
Gambar 7.16 Halaman Daftar Akun Toko.....	182
Gambar 7.17 Peringatan Pada Pendaftaran Akun Toko	183
Gambar 7.18 Halaman Data Profil.....	184
Gambar 7.19 Halaman Edit Profil	184
Gambar 7.20 Melakukan Pengeditan Data Profil	185
Gambar 7.21 Kelengkapan Data Profil User	185
Gambar 7.22 Halaman Ganti Password Pada Sistem JURAGAN	186
Gambar 7.23 Peringatan Password Lama Salah	187
Gambar 7.24 Proses Ganti Password Sukses.....	188
Gambar 7.25 Cara Mengakses Halaman Lupa Password	189
Gambar 7.26 Halaman Lupa Password Pada Sistem JURAGAN	189
Gambar 7.27 Memberikan Email Yang Akan Di Reset Passwordnya	190
Gambar 7.28 Notifikasi Cek Email Untuk Melakukan Reset Password	190
Gambar 7.29 Email Untuk Melakukan Reset Password.....	191
Gambar 7.30 Halaman Reset Password Sistem JURAGAN	191
Gambar 7.31 Membuat Password Baru	192
Gambar 7.32 Fitur Lupa Password Berhasil Dilakukan	192
Gambar 7.33 Keranjang Belanja Belum Terisi.....	193
Gambar 7.34 Halaman Produk Sistem JURAGAN	194
Gambar 7.35 Keranjang Belanja Terisi	194
Gambar 7.36 Halaman Keranjang Belanja Sistem JURAGAN	195
Gambar 7.37 Menambah Dan Mengurangkan QTY.....	196
Gambar 7.38 Menghapus Salah Satu Items Pada Halaman Keranjang Belanja	196
Gambar 7.39 Menampilkan Produk Berdasarkan Komunitas	197
Gambar 7.40 Akun Toko Belum Terverifikasi	198
Gambar 7.41 Menu Verifikasi Akun	199
Gambar 7.42 Halaman Verifikasi Akun	199
Gambar 7.43 Akun Toko Sudah Terverifikasi.....	200
Gambar 7.44 Fungsi Modals Pada Logout	200
Gambar 7.45 Proses Logout Berhasil	201

DAFTAR TABEL

Tabel 3.1 Definisi Aktor Pada Use Case JURAGAN	16
Tabel 3. 2 Definisi Komponen Use Case Pada Use Case JURAGAN	17
Tabel 3.3 Scenario Use Case Daftar Akun	18
Tabel 3.4 Scenario Use Case Login.....	20
Tabel 3.5 Scenario Use Case Kelola Profil.....	21
Tabel 3.6 Scenario Use Case Transaksi.....	21
Tabel 3.7 Scenario Use Case Verifikasi Akun.....	22
Tabel 3.8 Scenario Use Case Kelola Produk (Penjual).....	23
<i>Tabel 3.9 Scenario Use Case Kelola Produk (Admin)</i>	24
Tabel 3.10 Scenario Use Case Kelola User	25
Tabel 3.11 Struktur Tabel User.....	27
Tabel 3.12 Struktur Tabel User_token.....	29
Tabel 3.13 Struktur Tabel tb_barang	30
Tabel 4.1 Spesifikasi Software VS Code	55

DAFTAR SOURCE CODE

Source Code 5.1 Tahap Awal Mengedit File Controllers	89
Source Code 5.2 Function Inventory Pada Controllers	90
Source Code 5.3 Function Input Pada Controllers.....	92
Source Code 5.4 Function Edit Pada Controllers	94
Source Code 5.5 Function Update Pada Controllers	95
Source Code 5.6 Function Delete Pada Controllers.....	96
Source Code 5.7 Code Full Controllers	101
Source Code 5.8 Membuat Class Pada File Models	102
Source Code 5.9 Membuat Function Inventory Pada File Models	103
Source Code 5.10 Membuat Fungsi Get_Where Dengan Session.....	104
Source Code 5.11 Membuat Fungsi Get_Where Tanpa Session	104
Source Code 5.12 Membuat Function Inputdata Pada File Models	105
Source Code 5.13 Membuat Function edit_barang Pada File Models.....	105
Source Code 5.14 Membuat Function update_data Pada File Models	107
Source Code 5.15 Membuat Function Delete Pada File Models	108
Source Code 5.16 Code Full File Models	109
Source Code 5.17 Membuat View Inventory	112
Source Code 5.18 Halaman Input Produk Dengan Fungsi Modals	117
Source Code 5.19 Membuat View Edit Pada Folder Store.....	121
Source Code 5.20 View Detail Barang Pada Folder Store	124
Source Code 6.1 Merubah Controllers Welcome	137
Source Code 6.2 Merubah File header.php Pada Folder Templates	139
Source Code 6.3 Merubah File footer.php Pada Folder Templates.....	140
Source Code 6.4 Memanggil Library Form Validation	143
Source Code 6.5 Membuat Rules Pada Kolom Nama	144
Source Code 6.6 Menambahkan Rules Pada Kolom email	145
Source Code 6.7 Membuat Rules Pada Kolom Password	146
Source Code 6.8 Code Full Function Registrasian JURAGAN	149
Source Code 6.9 Dokumentasi Add Item Pada Shopping Cart	150
Source Code 6.10 Melakukan Load Library Cart	151
Source Code 6.11 Menambahkan Item Database Ke Cart.....	152
Source Code 6.12 Membuat Models m_cart.php.....	153
Source Code 6.13 Function Delete Pada Controllers Cart.....	155
Source Code 6.14 Memanggil Function Delete Pada Button Delete.....	156
Source Code 6.15 Membuat Fungsi Destroy	158

Source Code 6.16 Menambahkan Button Dan Memanggil Fungsi Destroy Pada Button Tersebut.....	159
Source Code 6.17 Membuat Function Uodate Items Pada Controllers Cart	160
<i>Source Code 6.18 Code Pemanggilan View Pada Halaman Keranjang Belanja</i>	162
Source Code 6.19 Membuat Function Seacrh Pada Controllers	165
Source Code 6.20 Membuat Models Dengan Function get_keyword	166
Source Code 6.21 Memanggil Controllers Pada Forn Input-Nya.....	167

FOREWORD

KATA PENGANTAR

ACKNOWLEDGMENTS

ACRONYM

GLOSSARY

SYMBOLS

INTRODUCTION

BAB 1

MENGENAL SISTEM JURAGAN

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 membeli produk melalui media internet (*online*) [2]. Dengan adanya penerapan *E-Commerce* kedalam dunia bisnis 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 kenyataannya 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 dan berbasis komunitas. *E-Commerce* tersebut akan diberi nama JURAGAN yang dimana memiliki arti dari “Jualan Rakyat Gabungan Online”. JURAGAN dirancang untuk membantu para pengusaha – pengusaha dapat bersaing di era RI 4.0 ini. JURAGAN akan menggabungkan para pengusaha dengan status UMKM, UKM, hingga PKL kedalam satu wadah dengan konsep 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 salah satu sistem berbentuk *website* yang dimana dalam perancangannya menggunakan

4 | BAB I Mengenal Sistem JURAGAN

framework codeigniter dan bahasa pemrograman PHP. *Framework Codeigniter* merupakan salah satu *framework* yang cukup populer di kalangan para perogramer, hal ini dikarenakan *framework codeigniter* mempunyai kelebihan tersendiri seperti memiliki dokumentasi yang cukup lengkap sehingga dapat membantu proses pemrograman menjadi lebih mudah.

BAB 2

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 kgiatan 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 3

PERANCANGAN SISTEM JURAGAN

12 | BAB III Perancangan Sistem JURAGAN

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.

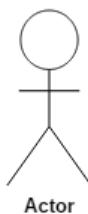
Untuk membangun aplikasi dengan konsep *e-commerce* seperti JURAGAN (Jualan Rakyat Gabungan *Online*) dimana perlu dilakukannya tahap analisis dan perancangan terlebih dahulu. Dalam buku ini dimana penulis telah menyediakan perancangan dalam membuat sistem JURAGAN tersebut yang terdiri dari perancangan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

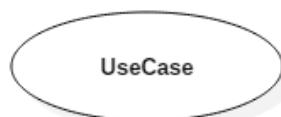
Manfaat dalam membuat *use case* antara lain yaitu digunakan dalam melakukan komunikasi dengan *end user* dan *domain expert*. *Use case* juga dapat memastikan pemahaman secara tepat mengenai *requirement* atau kebutuhan sistem. *Use case* digunakan dalam mengidentifikasi siapa yang akan berinteraksi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

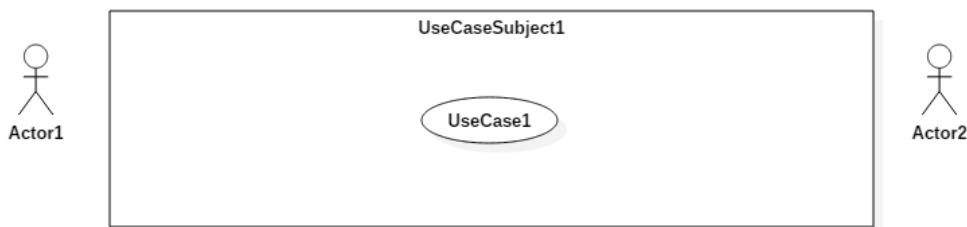
Aktor merupakan komponen yang menggambarkan *user* atau seseorang yang akan berinteraksi dengan sistem tersebut. Aktor hanya bisa melakukan interaksi terhadap sistem yang dimana menandakan aktor bukan pemegang kendali penuh pada *use case*.



Gambar 3.2 Komponen UseCase

14 | BAB III Perancangan Sistem JURAGAN

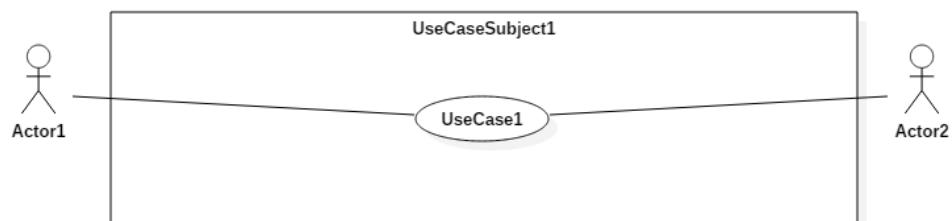
UseCase merupakan komponen yang berfungsi sebagai memberikan gambaran fungsional sistem yang akan dibangun yang dimana dapat membuat pengguna lebih mengerti dalam mengaplikasikan sistem tersebut.



Gambar 3.3 Komponen Use Case Subject

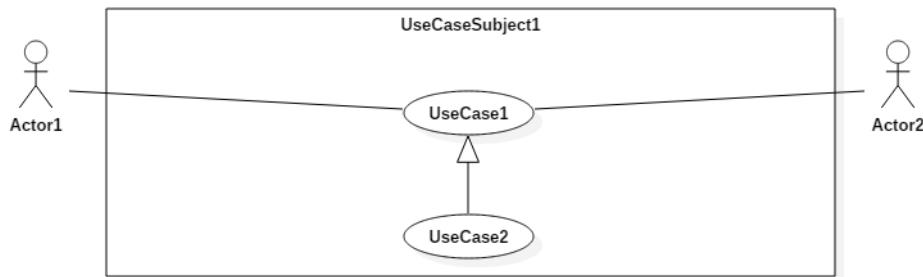
Use case subject merupakan komponen yang berfungsi sebagai tempat menampungnya *subject – subject use case* atau sebagai sebuah *frame* dari *user case* tersebut.

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



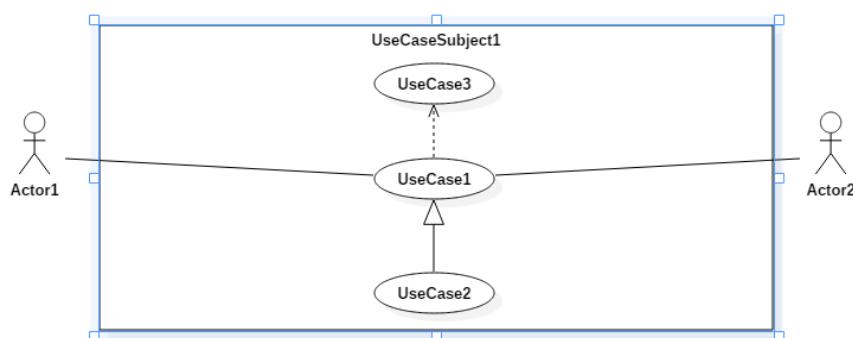
Gambar 3.4 Relasi Association

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.

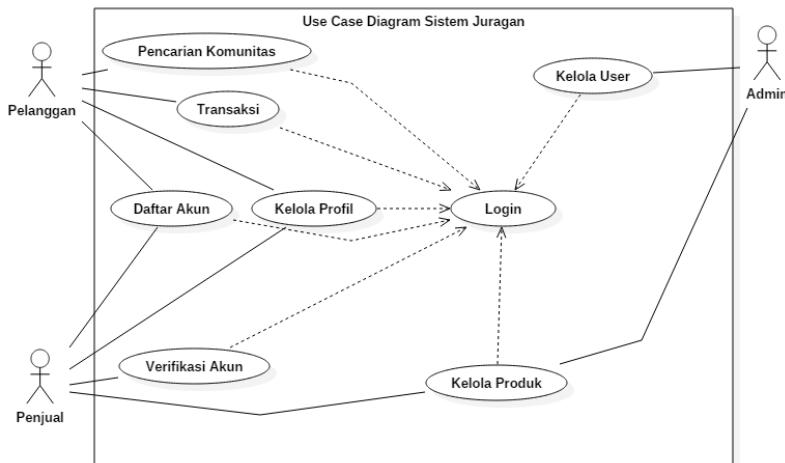


Gambar 3.6 Relasi Despendency

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.

16 | BAB III Perancangan Sistem JURAGAN

Dalam membangun sistem JURAGAN kali ini dimana penulis sudah membangun *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. Login

		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 di mana 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>registrasion</i> akun pada sistem JURAGAN.
2.	<i>Login</i>	a. Melakukan proses <i>login</i> terhadap akun yang sudah melakukan <i>registrasion</i> atau sudah terdaftar di dalam sistem JURAGAN.
3.	Kelola Profil	a. Melakukan pengelolaan profil terhadap akun <i>user</i> .

18 | BAB III Perancangan Sistem JURAGAN

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	a. Melakukan pengelolaan produk terhadap akun level toko. b. Melakukan <i>view</i> produk terhadap akun level pelanggan.
8	Kelola User	a. Melakukan pengelolaan akun <i>user</i> oleh admin.

3.1.3 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
Skenario	
Kondisi Awal	Halaman registrasi
Aksi Aktor	
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

Identifikasi	
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
Skenario	
Kondisi Awal	Halaman login
Aksi Aktor	
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
Skenario	
Kondisi Awal	Halaman Edit Profil
Aksi Aktor	
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
Skenario	
Kondisi Awal	Halaman Produk
Aksi Aktor	
1. Memberikan aksi terhadap <i>button tambah</i> ke keranjang	a. Mengirimkan <i>id</i> produk yang ditambahkan. b. Melakukan validasi
2. -	
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
Skenario	
Kondisi Awal	Halaman Verifikasi Akun
Aksi Aktor	Reaksi Sistem
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)

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

24 | BAB III Perancangan Sistem JURAGAN

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</i> produk
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)

Identifikasi	
No.	JR8

Nama	Kelola Produk
Tujuan	Mengelola data produk
Deskripsi	Melakukan pengelolaan terhadap data produk.
Aktor	Admin
Skenario	
Kondisi Awal	Halaman Admin
Aksi Aktor	
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

Identifikasi	
No.	JR9
Nama	Kelola User
Tujuan	Mengelola user yang telah terdaftar
Deskripsi	Melakukan pengelolaan data terhadap user yang terdaftar

Aktor	Admin
Skenario	
Kondisi Awal	Halaman Admin
Aksi Aktor	Reaksi Sistem
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
<i>Id</i>	<i>Integer (11)</i>	Merupakan <i>id</i> dari <i>user</i> dan juga <i>primary key</i> dari tabel <i>user</i> .
<i>Name</i>	<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	<i>Varchar (25)</i>	<i>Field</i> tersebut berguna untuk menampung alamat berupa kecamatan dari <i>user</i> tersebut.
Kelurahan	<i>Varchar (25)</i>	<i>Field</i> tersebut berguna untuk menampung alamat berupa kelurahan dari <i>user</i> tersebut.
Kota	<i>Varchar (25)</i>	<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	Integer (11)	Merupakan <i>primary key</i> dari tabel tersebut.
Email	Varchar (25)	Merupakan <i>email</i> dari <i>user</i> yang di daftarkan.

<i>Token</i>	<i>Varchar (128)</i>	Merupakan <i>field</i> yang berfungsi untuk menampung token tersebut.
<i>Date_created</i>	<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
<i>Id_brg</i>	<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> .
Harga	<i>Integer (11)</i>	Merupakan <i>field</i> untuk menampung harga dari produk tersebut.
Stok	<i>Integer (4)</i>	Merupakan <i>field</i> untuk menampung jumlah stok pada produk tersebut.
Tlpn	<i>Varchar (15)</i>	Merupakan <i>field</i> untuk menampung nomor telpon <i>user</i> dengan level penjual.
Gambar	<i>Text</i>	Merupakan <i>field</i> untuk menampung gambar dari produk tersebut.

3.3 User Interface Sistem JURAGAN

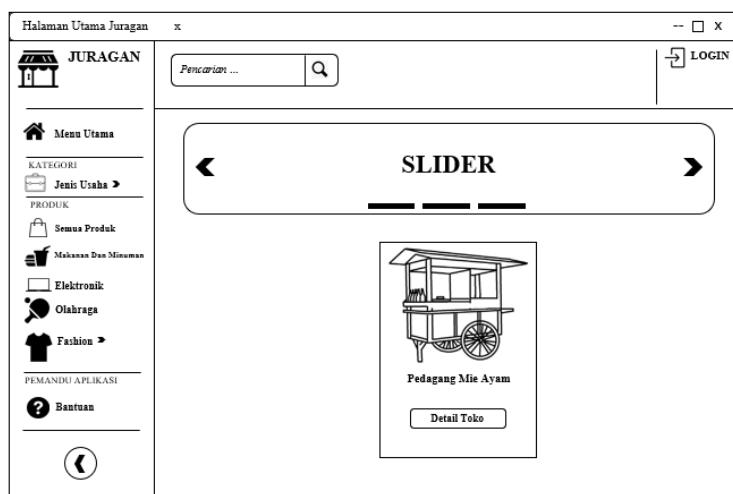
Pernahkah kalian melihat tampilan suatu sistem ? Template – template dari suatu sistem ? apa teman – teman tahu yang dimaksud dengan *user interface* ?. *User interface* atau biasanya disingkat dengan *UI* merupakan suatu bentuk visual, template, atau tampilan pada sebuah *software* atau *website* yang 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 antara *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 dikarenakan 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.

1. *User Interface* Halaman Utama



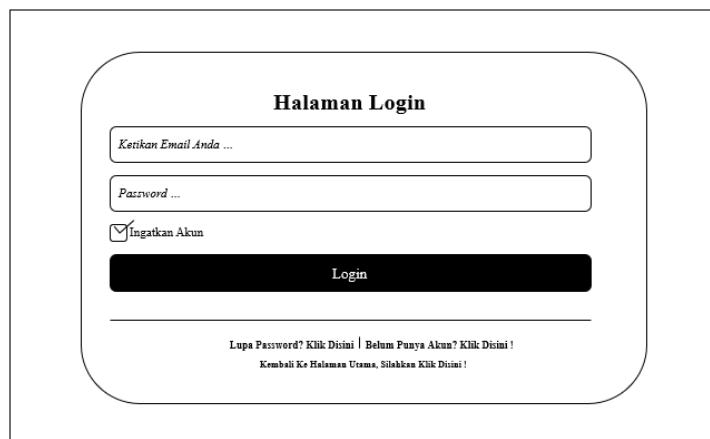
Gambar 3.8 Perancangan UI Halaman Utama

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.

2. *User Interface* Halaman Login

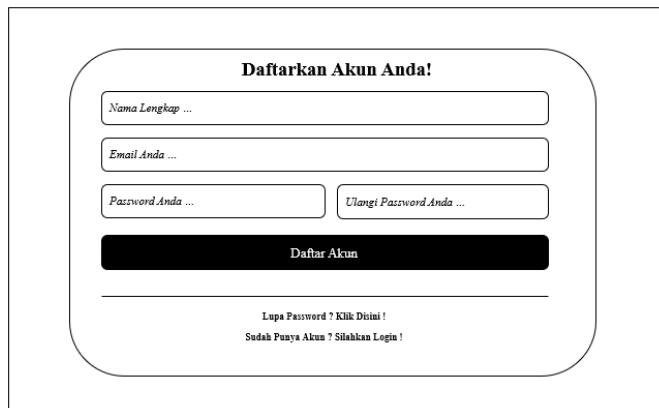


Gambar 3.9 Perancangan UI Halaman Login

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 *button* 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 halaman *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.

3. User Interface Halaman Daftar Akun



Daftarkan Akun Anda!

Nama Lengkap ...

Email Anda ...

Password Anda ... *Ulangi Password Anda ...*

Daftar Akun

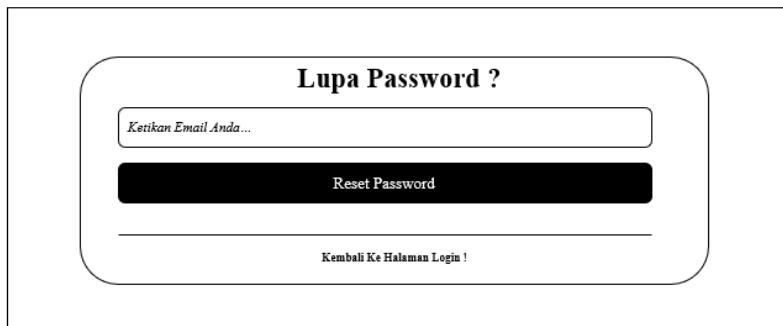
Lupa Password ? Klik Disini !
Sudah Punya Akun ? Silahkan Login !

A wireframe diagram of a user interface for account registration. It features a rounded rectangular frame with a title "Daftarkan Akun Anda!". Inside, there are four input fields: "Nama Lengkap ..." (Name), "Email Anda ..." (Email), "Password Anda ..." (Password), and "Ulangi Password Anda ..." (Repeat Password). Below these is a large, solid black rectangular button labeled "Daftar Akun". At the bottom of the form, there are two links: "Lupa Password ? Klik Disini !" (Forgot Password ? Click Here!) and "Sudah Punya Akun ? Silahkan Login !" (Already have an account ? Please 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.

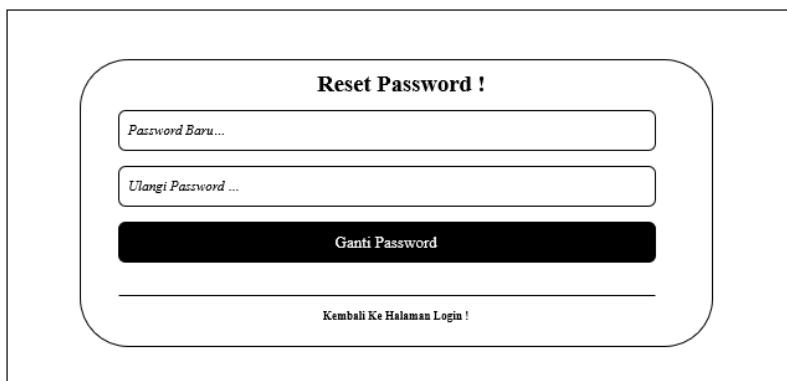
4. *User Interface* Halaman Lupa Password



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.

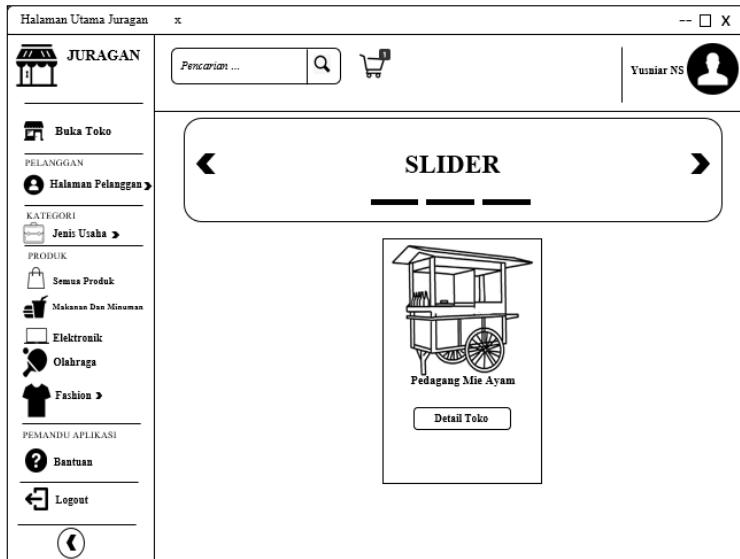
5. User Interface Halaman Reset Password



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*.

6. User Interface Halaman Utama Kondisi Login

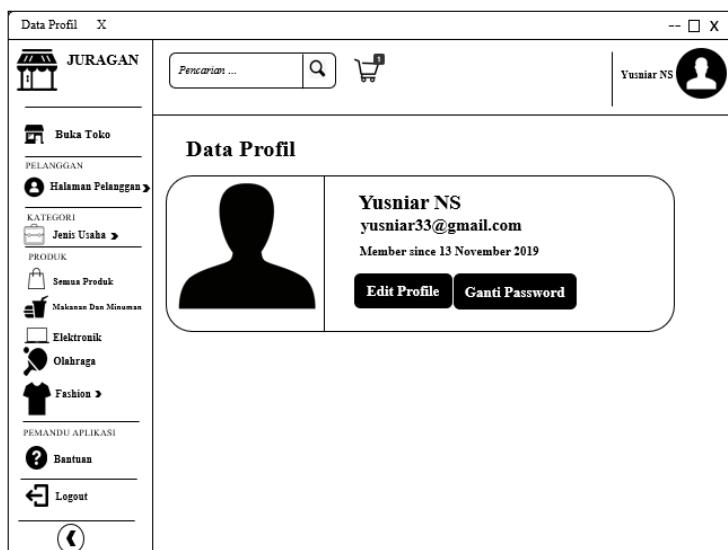


Gambar 3.13 Perancangan UI Halaman Utama Kondisi Login

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.

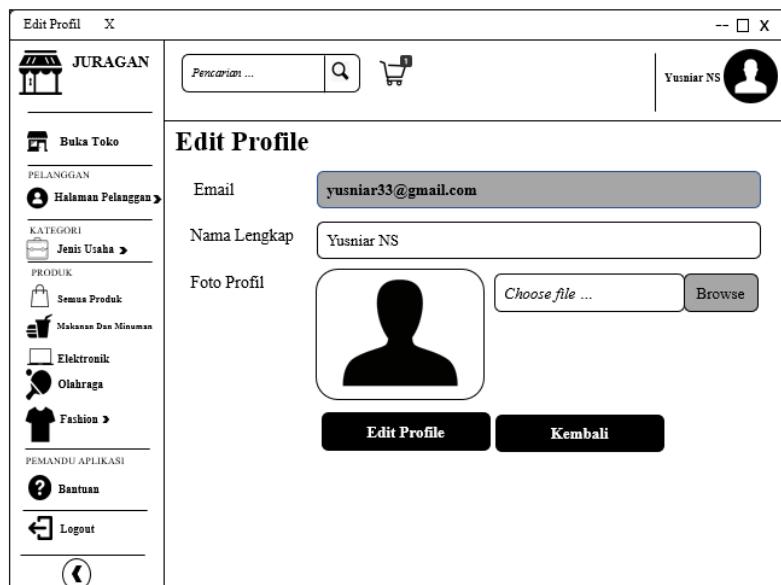
7. User Interface Halaman Data Profil



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*.

8. User Interface Halaman Edit Profil



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 bagian *main* halaman *edit* profil sistem JURAGAN terdapat tiga buah *textbox* yang dimana satu *textbox* bersifat *readonly*. Apa itu *readonly* ?, *readonly* merupakan fungsi *fzree* 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* dalam bentuk PNG, JPG, dan JPEG.

9. User Interface Halaman Ganti Password

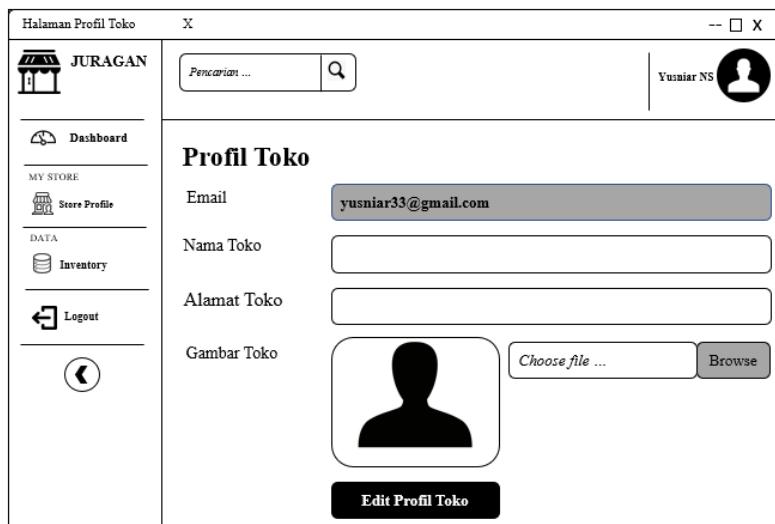
The screenshot shows the 'Ganti Password' (Change Password) page of the JURAGAN system. The page has a header with a search bar, a cart icon, and a user profile picture for 'Yusniar NS'. On the left is a sidebar with links for 'Buka Toko', 'PELANGGAN', 'Halaman Pelanggan', 'KATEGORI', 'Jenis Usaha', 'PRODUK', 'Semua Produk', 'Makanan Dan Minuman', 'Elektronik', 'Olahraga', 'Fashion', 'PEMANDU APLIKASI', 'Bantuan', 'Logout', and a circular icon. The main content area has a title 'Ganti Password' and three input fields for 'Password Lama', 'Password Baru', and 'Ulangi Password'. At the bottom are two buttons: 'Ganti Password' and 'Kembali'.

Gambar 3.16 Perancangan UI Halaman Ganti Password

Pada gambar 3.16 merupakan perancangan *user interface* halaman ganti password pada sistem JURAGAN. Halman 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.

10. User Interface Halaman Profil Toko

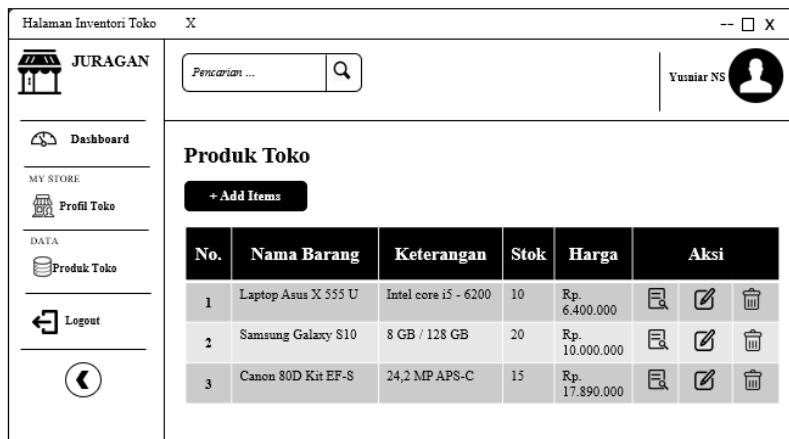


Gambar 3.17 Perancangan UI Halaman Profil Toko

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.

11. User Interface Halaman Produk Toko



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*.

12. User Interface Halaman Input Produk

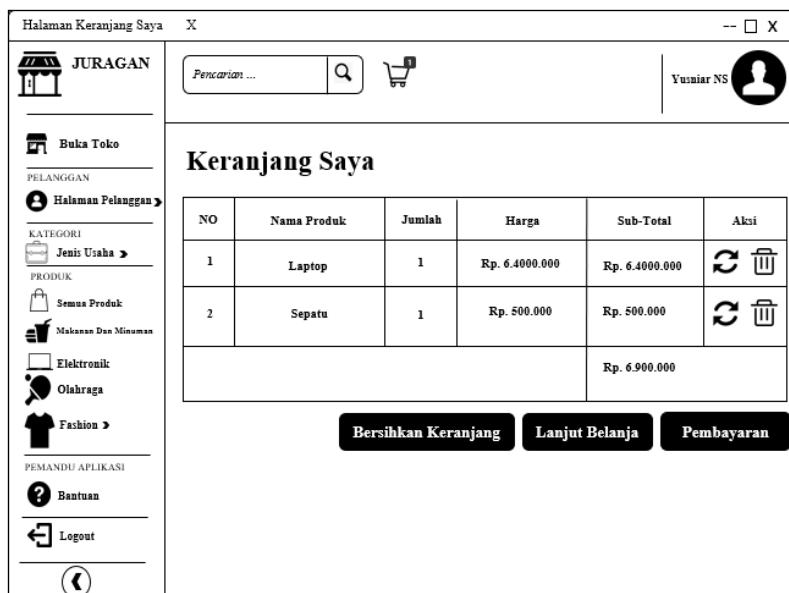
The form is titled "FORM INPUT PRODUK". It contains the following fields:

- Email:
- Nama Produk:
- Detail Produk:
- Kategori:
- Harga:
- Stok:
- Gambar Produk:
- Action Buttons: **Kembali** and **Simpan Produk**

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 *textbox*, dan *button* yang berfungsi sebagai pemberian aksi terhadap halaman tersebut.

13. User Interface Halaman



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 4

SOFTWARE PENDUKUNG

Sebelum teman – teman membuat suatu sistem dimana diperlukan *software – software* pendukung untuk membantu proses penggerjaan 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 memabntu kinerja pengembangan JURAGAN. Bagaimana cara melakukan *install software Xampp* tersebut ?, ikuti langkah – langkah nya sebagai berikut :

1. Silahkan teman – teman kunjungi *website* resmi *Xampp* pada *link* <https://www.apachefriends.org/index.html> dan lakukan proses *download*.



Gambar 4.1 Website Resmi Xampp

- 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

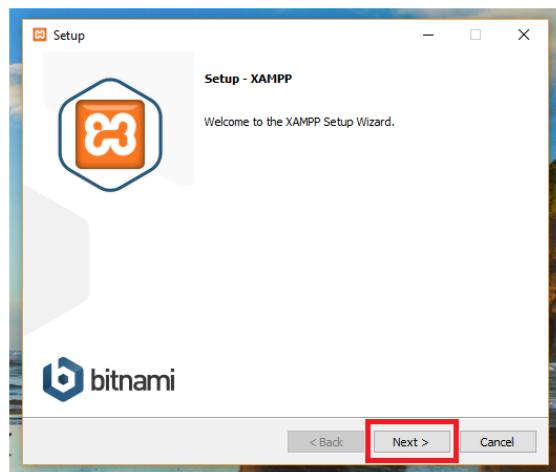
- Teman – teman akan di bawa ke halaman pemilihan versi yang akan di *download*. Sebelum melakukan *download* perlu teman – teman ketahui dimana dalam *website* resmi *Xampp* merupakan veris terbaru dan sudah menggunakan PHP 7. Apabila teman – teman tidak terbiasa dengan PHP 7 dapat menggunakan PHP 5 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*.

Version	Checksum	Size
7.2.26 / PHP 7.2.26	What's Included? md5 sha1	Download (64 bit) 145 Mb
7.3.13 / PHP 7.3.13	What's Included? md5 sha1	Download (64 bit) 145 Mb
7.4.1 / PHP 7.4.1	What's Included? md5 sha1	Download (64 bit) 145 Mb

Gambar 4.3 Halaman Versi Xampp

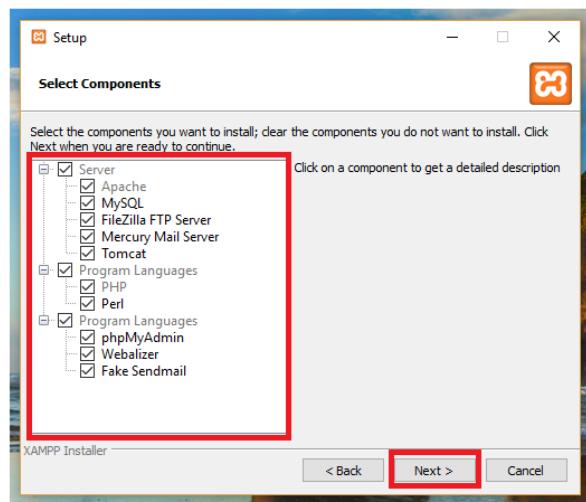
- Silahkan teman – teman *download software Xampp* tersebut. Tunggu beberapa menit hingga proses *download* selesai.
- Apabila proses *download* telah selesai, silahkan teman – teman buka file tersebut, jalankan dengan *run administrator*. Akan muncul sebuah

jendela baru yang menandakan proses *instalasi* akan dimulai, pilih *next*.



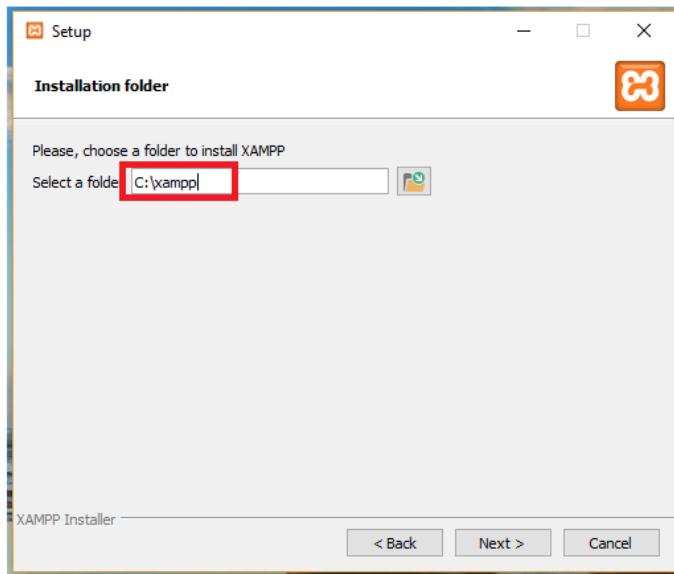
Gambar 4.4 Proses Awal Instalasi Xampp

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.



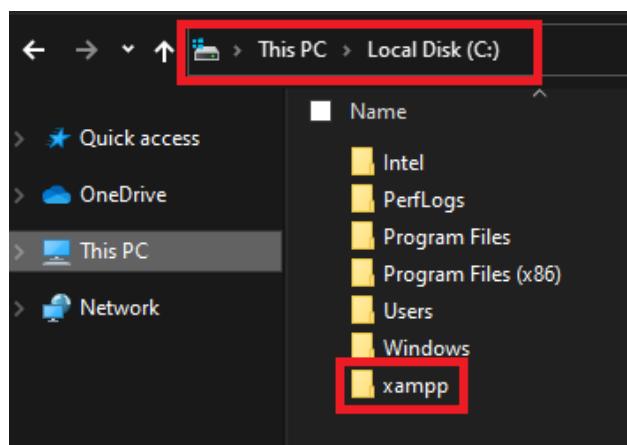
Gambar 4.5 Pemilihan Komponen - Komponen Pada Software Xampp

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.



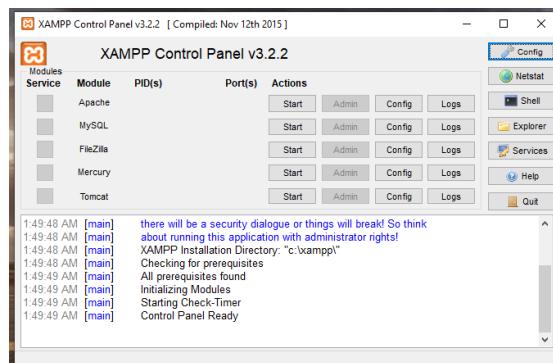
Gambar 4.7 Tempat Penyimpanan Xampp

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.



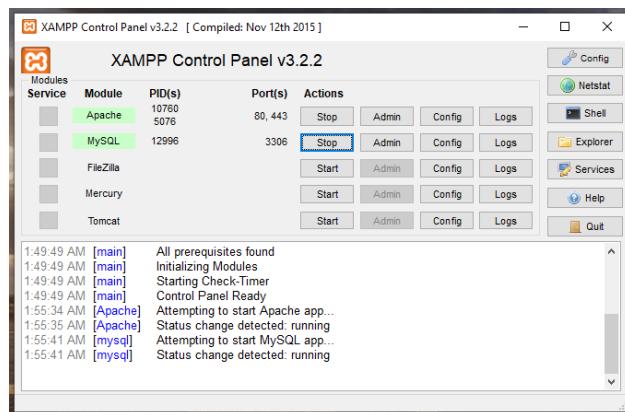
Gambar 4.8 Proses Installasi Xampp Sedang Berjalan

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



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.

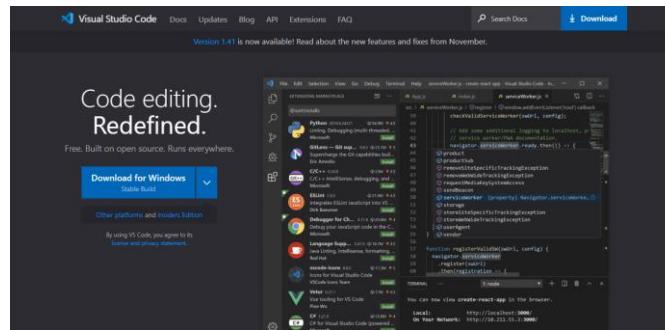
Sebelum melakukan penginstallan *software Vscode* dimana kita harus mengetahui spesifikasi atau *requirements* yang dibutuhkan. Spesifikasi yang dibutuhkan dapat dilihat pada tabel 4.1.

Tabel 4.1 Spesifikasi Software VS Code

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

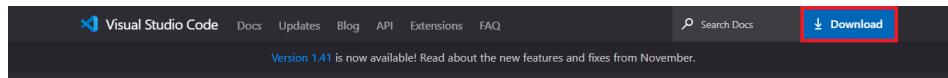
Apakah teman – teman tertarik ingin menggunakan *software visual studio code* ? Bagaimana sih cara melakukan *installasi* terhadap *software visual studio code* ?. Pada pembahasan kali ini dimana teman – teman akan belajar bagaimana cara melakukan *installasi* terhadap *software visual studio code* tersebut.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



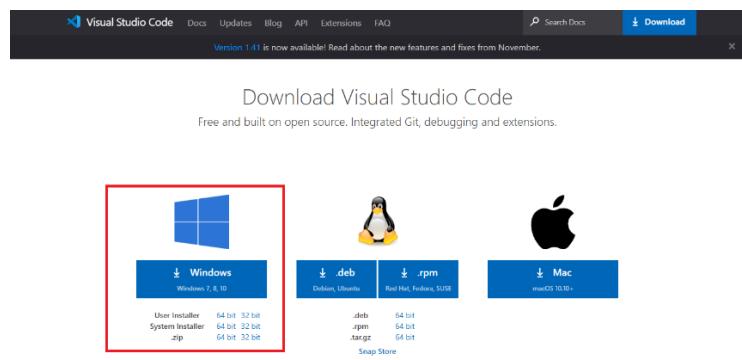
Gambar 4.12 Halaman Website Resmi VSCode

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



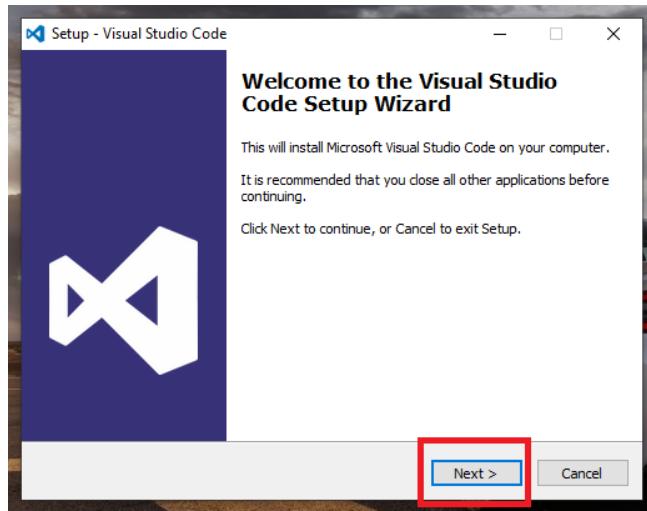
Gambar 4.13 Memilih Halaman Download VSCode

3. Dimana teman – teman akan dibawa ke halaman *download*, silahkan *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.



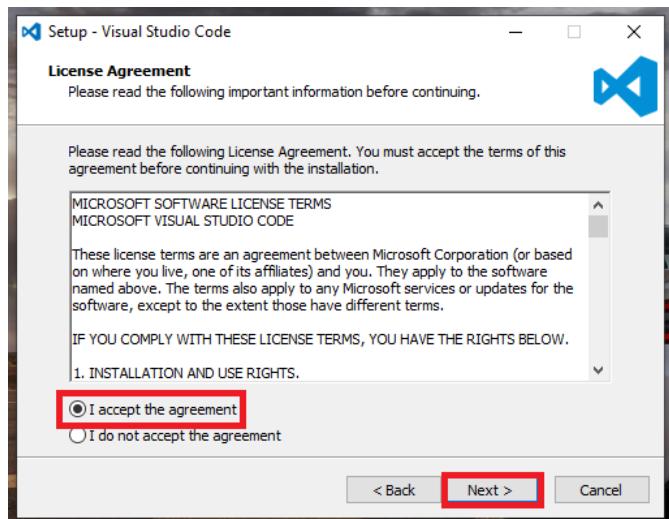
Gambar 4.14 Pemilihan OS Saat Download VSCode

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



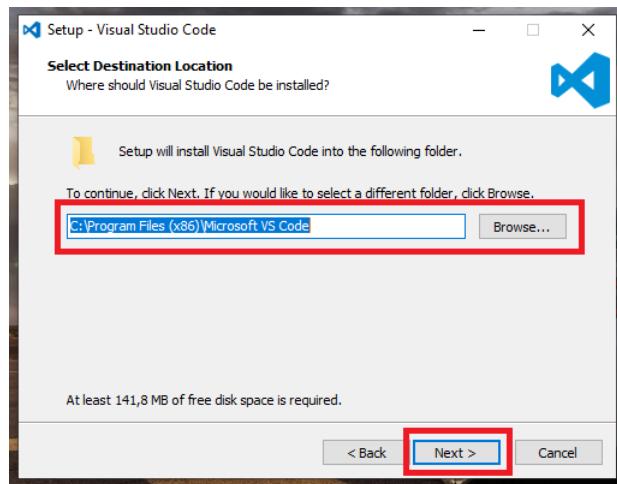
Gambar 4.15 Proses Awal Installasi VSCode

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



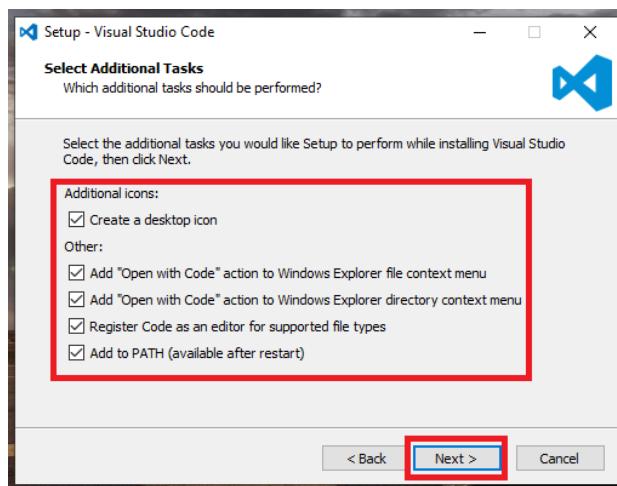
Gambar 4.16 Proses Installasi VSCode Tahap Dua

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.



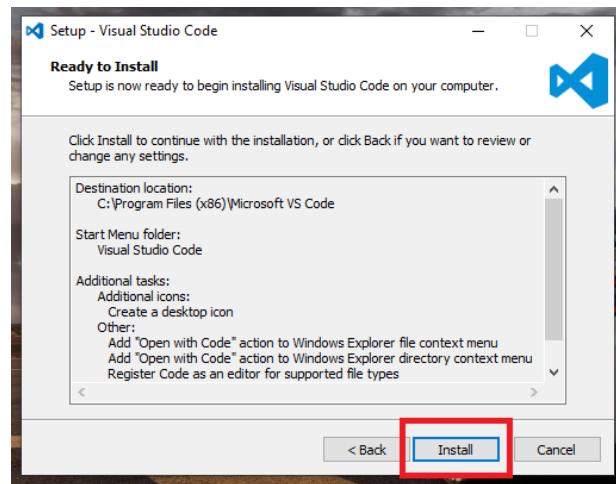
Gambar 4.17 Pemilihan Tempat Penyimpanan VSCode

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



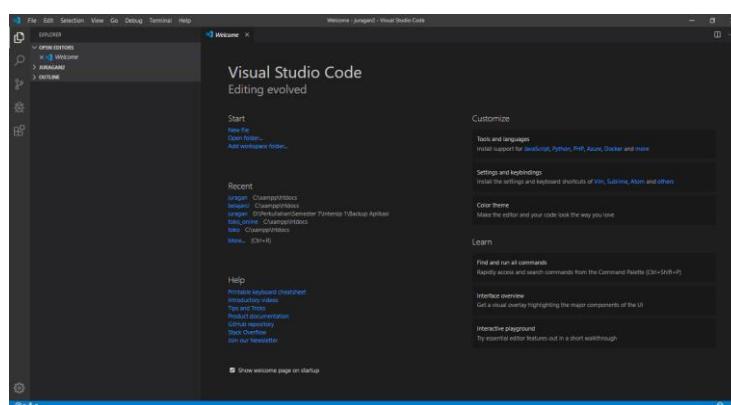
Gambar 4.18 Pemilihan Additional Tasks VSCode

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

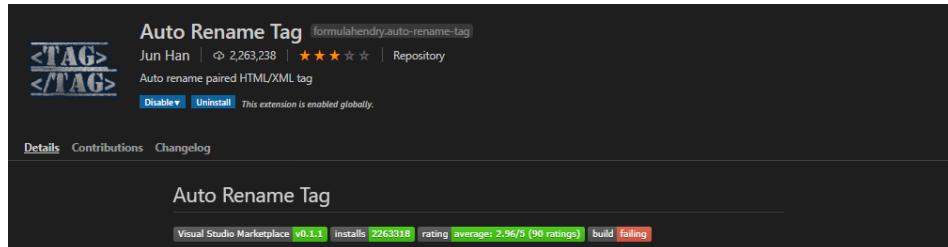
9. Apabila *progress bar* sudah terisi penuh menandakan proses *installasi software Visual Studio Code* teman – teman sudah selesai. Silahkan teman – teman buka *software Visual Studio Code* maka tampilan awalnya akan terlihat seperti pada gambar 4.20.



Gambar 4.20 Tampilan Visual Studio Code

4.2.1 Extensions Visual Studio Code

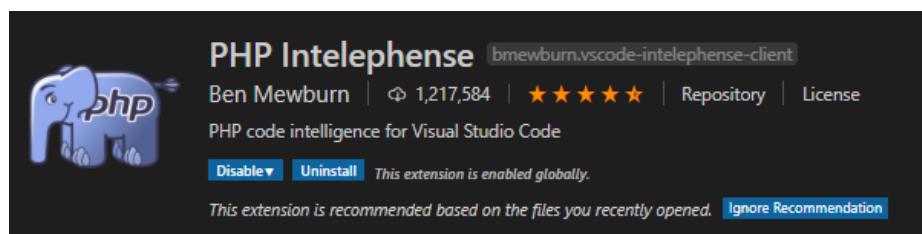
Apabila proses *install software Vscode* sudah selesai dimana teman - teman perlu melakukan *install* beberapa *extensions* untuk membantu proses pengkodean. Apa saja *extensions* tersebut ?, yang perlu teman - teman *install* adalah “*Auto Rename Tag*”.



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*”.

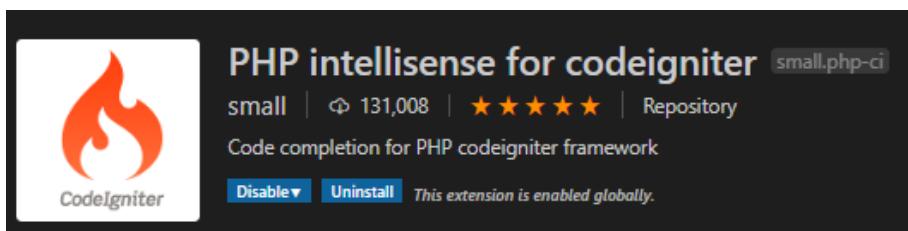


Gambar 4.22 Extension PHP Intelephense

Extensions tersebut dimana memiliki fungsi sebagai memberikan fitur lengkap terhadap *PHP* dengan saran yang sangat detail dan dukungan ”*Go*

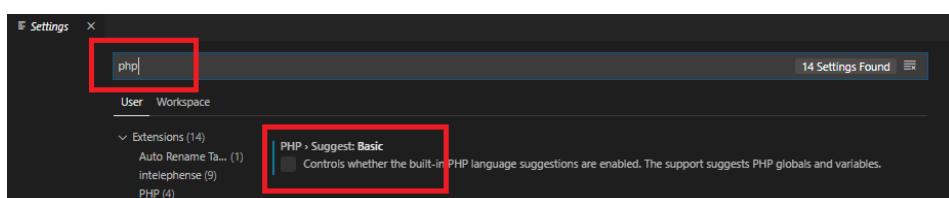
To” langsung kepada sumbernya. Dengan adanya *extensions* tersebut dimana teman - teman tidak perlu mengingat semua *sintaks* perintah yang akan teman - teman ketik, hal ini dikarenakan fitur *Intelephense* dapat bekerja tanpa harus melakukan konfigurasi.

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



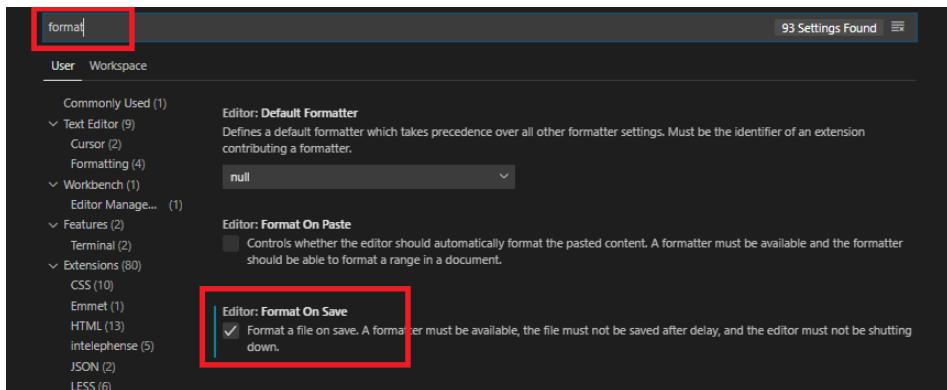
Gambar 4.23 Extension PHP Intellisese For CI

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

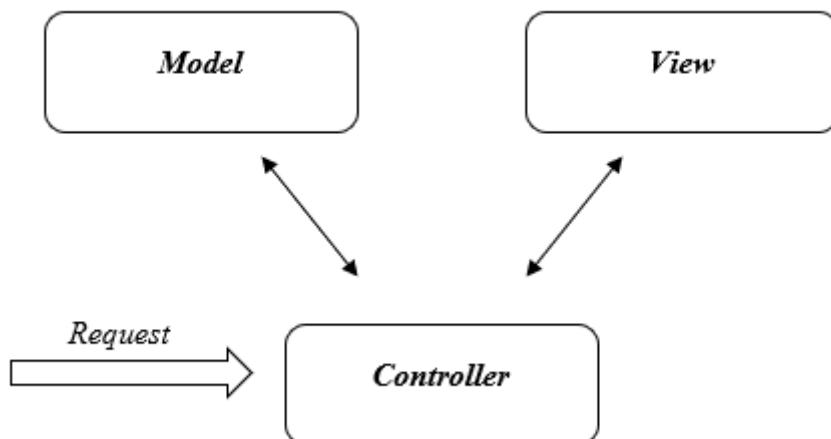
4.3 Codeigniter

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* *framework 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.



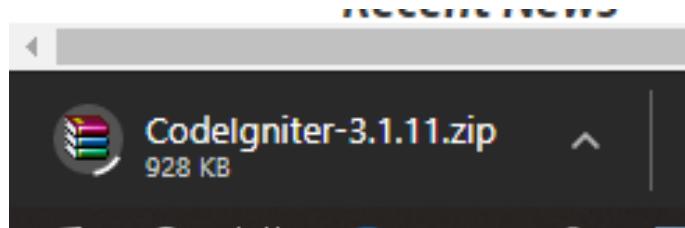
Gambar 4.27 Website Remi Codeigniter

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



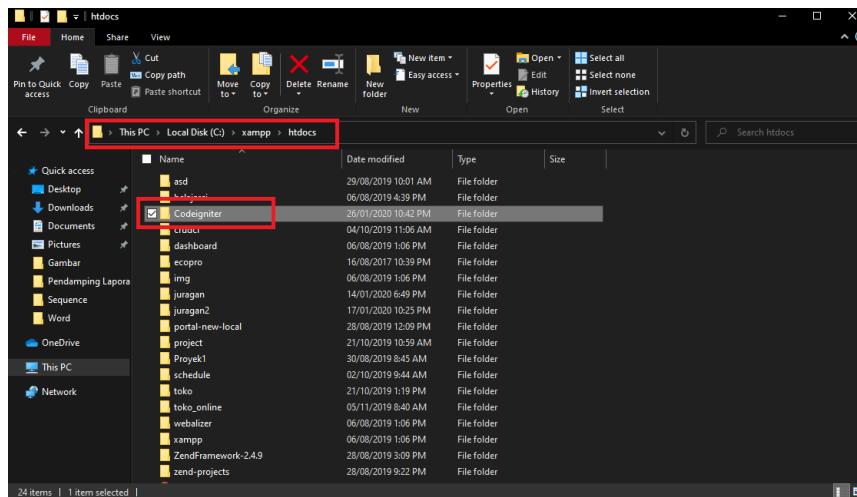
Gambar 4.28 Pilihan Download CI

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



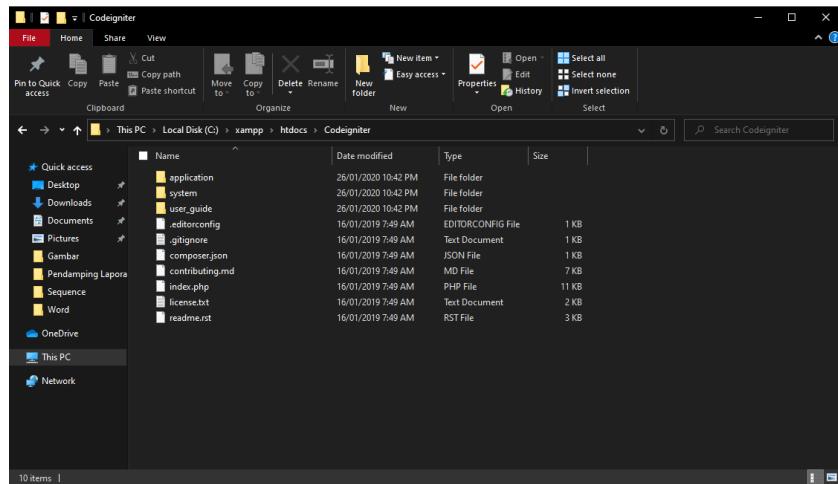
Gambar 4.29 Proses Download Codeigniter

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



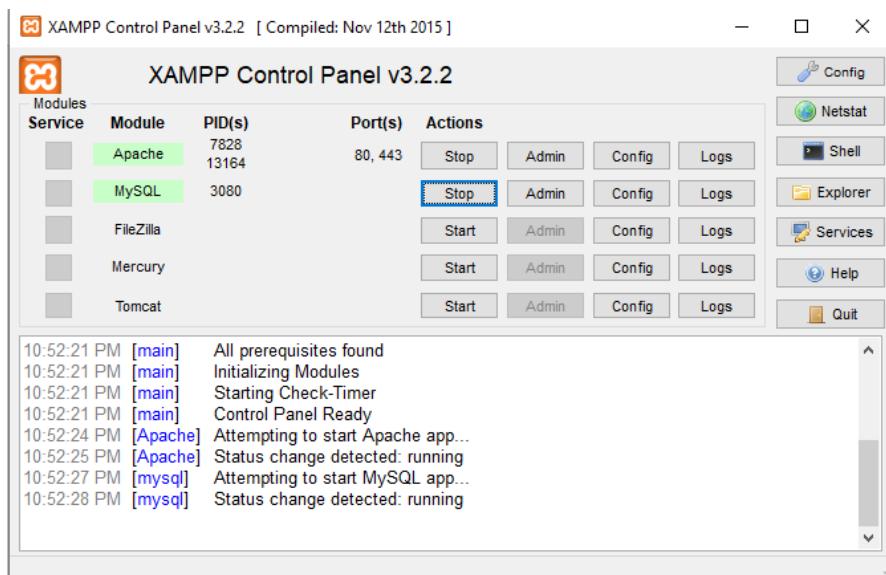
Gambar 4.30 Memindahkan File Codeigniter

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



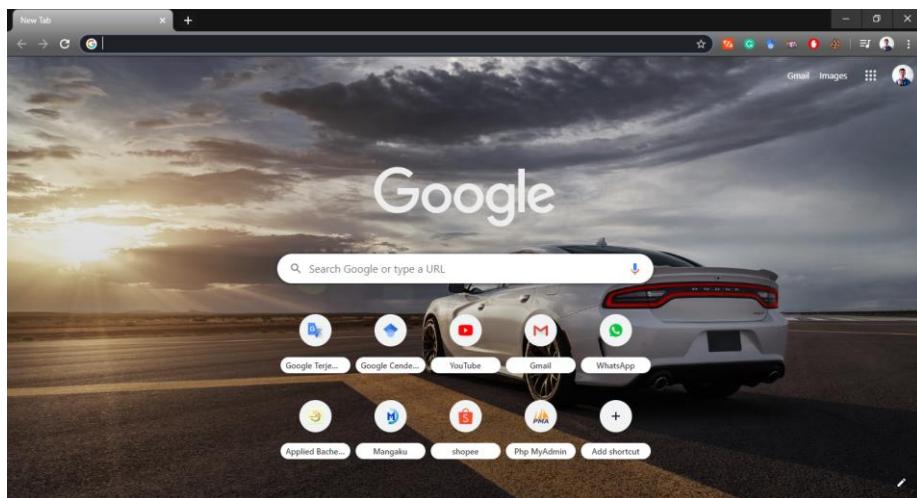
Gambar 4.31 Isi File Codeigniter

- 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.



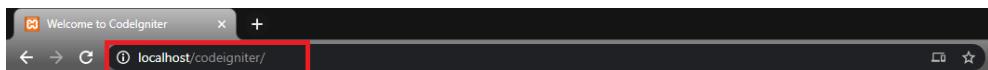
Gambar 4.32 Menjalankan Software Xampp

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



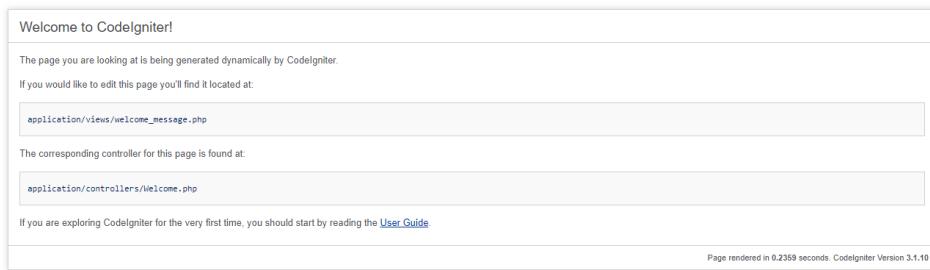
Gambar 4.33 Membuka Web Browser

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

11. Selamat teman – teman sudah berhasil memunculkan tampilan “Hello Word” pada *framework codeigniter* tersebut yang dimana menunjukkan bahwa *framework codeigniter* tersebut sudah berjalan dengan baik dan benar.

BAB 5

FRAMEWORK CODEIGNITER

5.1 Sejarah Codeigniter

Rick Ellis yang merupakan seorang musisi musik dengan genre rock yang dimana profesinya 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*.

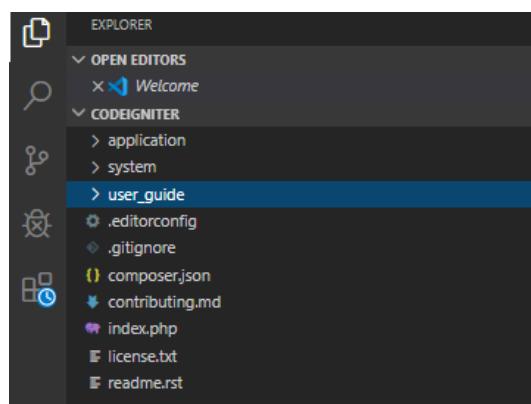


Gambar 5.1 Dokumentasi Codeigniter

- *Codeigniter* sangat cocok dan mudah dipelajari bagi pemula, hal ini dikarenakan *framework* tersebut tidak telalu bergantung pada *tool* tambahan sperti *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

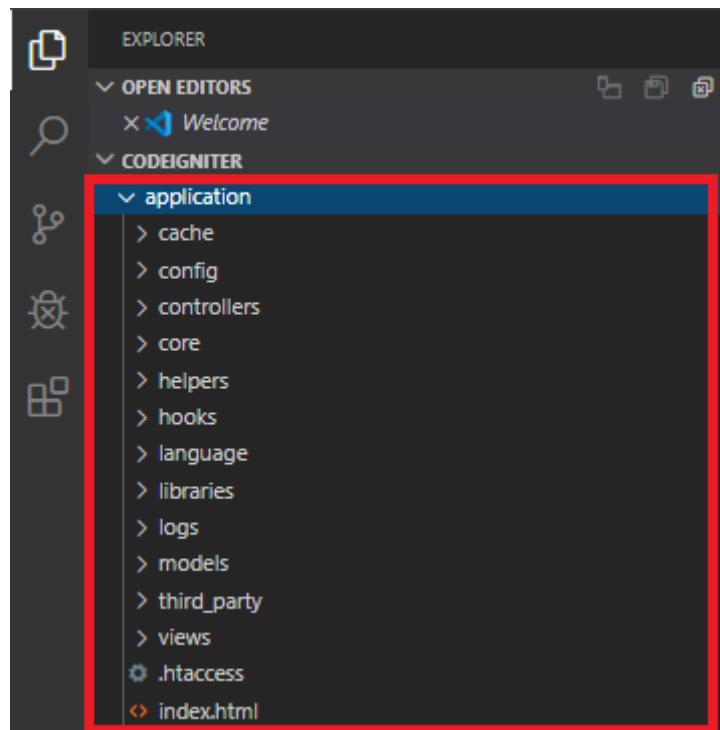
Sebelum teman – teman menggunakan *framewrok codeigniter* tersebut alangkah baiknya apabila teman – teman belajar dan memahami terlebih dahulu struktur direktori dari *framwork codeigniter* tersebut. Pada *framework Codeigniter* dimana struktur yang dimilikinya terlihat seperti pada gambar 5.2.



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.

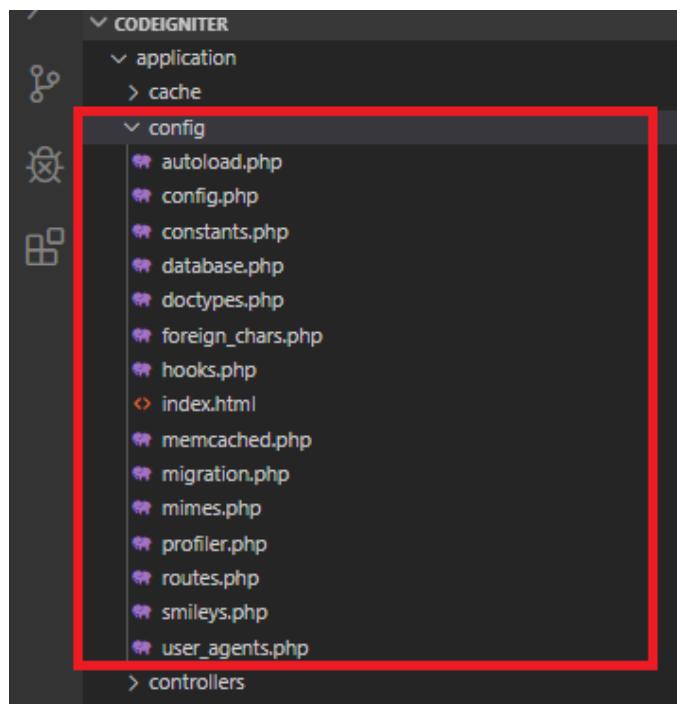
Direktori *application* merupakan direktori yang akan kita gunakan dalam pembuatan aplikasi kita nanti. Coba teman - teman klik pada direktori *application* tersebut dan lihat isinya, akan terlihat seperti pada gambar 5.3.



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.



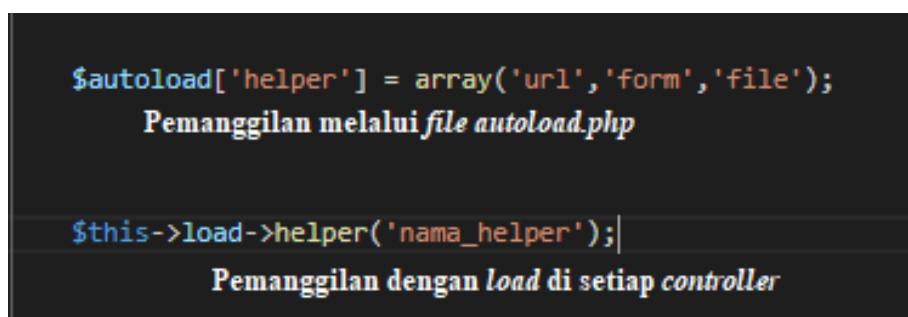
```

Welcome.php X
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 *custom 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

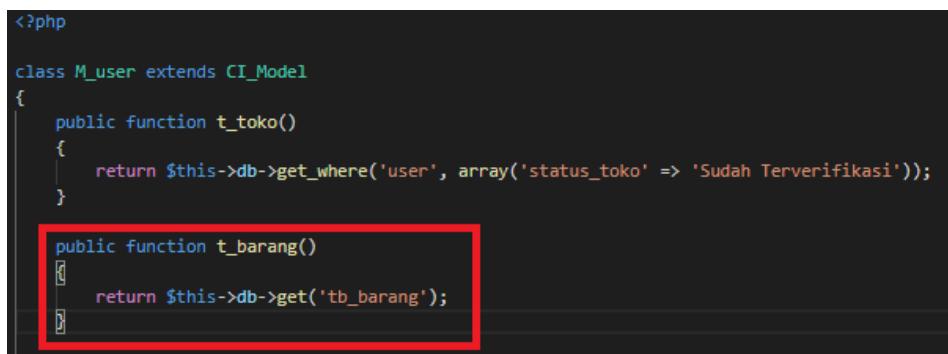
$this->load->helper('nama_helper');
Pemanggilan dengan load di setiap controller

```

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

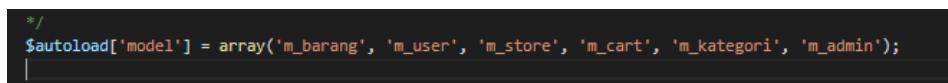
class M_user extends CI_Model
{
    public function t_toko()
    {
        return $this->db->get_where('user', array('status_toko' => 'Sudah Terverifikasi'));
    }

    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 Pemanggilan 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);           *m_user merupakan nama dari model yang kita buat.
    $this->load->view('user/templates/sidebar');                  *t_barang merupakan function dari m_user.
    $this->load->view('user/templates/topbar', $data);
    $this->load->view('user/halaman_produk', $data);
    $this->load->view('user/templates/footer');
}

```

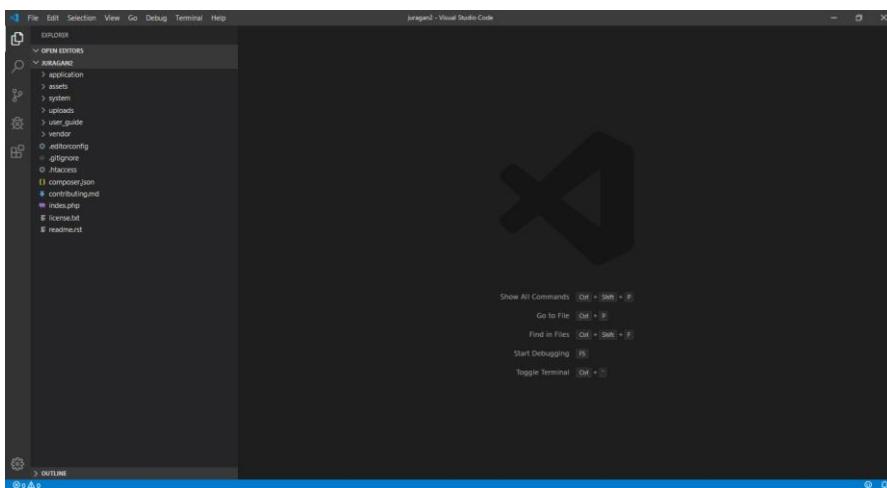
Gambar 5.9 Pemanggilan Fungsi Model Dari Controller

- *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 buka 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.

```

54 | $autoload['libraries'] = array('database', 'email', 'session');
55 |
56 | You can also supply an alternative library name to be assigned
57 | in the controller:
58 |
59 | $autoload['libraries'] = array('user_agent' => 'ua');
60 */
61 $autoload['libraries'] = array('database', 'email', 'session', 'form_validation', 'cart');
62
63 /*
64 | -----

```

Gambar 5.11 Konfigurasi Libraries Sistem JURAGAN

Langkah berikutnya masih pada file *autolod.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*.

```

86  | Auto-load Helper Files
87  | -----
88  | Prototype:
89  |
90  | $autoload['helper'] = array('url', 'file');
91  */
92  $autoload['helper'] = array('url', 'file', 'security');
93
94  /*
95  | -----
96  | Auto-load Config files
97  | -----

```

Gambar 5.12 Konfigurasi Fungsi Helpers Sistem JURAGAN

Langkah berikutnya masih tetap pada file *autolod.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*.

```

124 |     Auto-load Models
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['model'] = array('first_model' => 'first');
134 */
135 $autoload['model'] = array('m_barang', 'm_user', 'm_store', '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.

```

15 |
16 | If it is not set, then CodeIgniter will try guess the protocol and path
17 | your installation, but due to security concerns the hostname will be set
18 | to $_SERVER['SERVER_ADDR'] if available, or localhost otherwise.
19 | The auto-detection mechanism exists 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 | a PHP script and you can easily do that on your own.
24 |
25 */
26 $config['base_url'] = 'http://localhost/juragan2/';
27
28 /**
29 | -----
30 | Index File
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 so that it is blank.
36 |

```

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*.

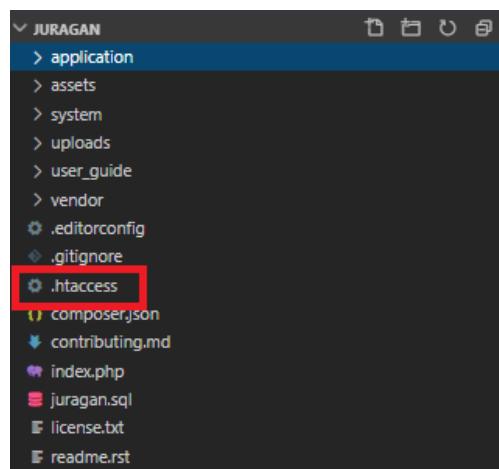
```

29 | -----
30 | Index File
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 so that it is blank.
36 |
37 */
38 $config['index_page'] = '';
39
40 /**
41 | -----

```

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 penyetinggan pada *mod_rewrite* nya. Silahkan teman – teman membuat *file* baru di luar direktori *application* dan berikan “*.htaccess*”.



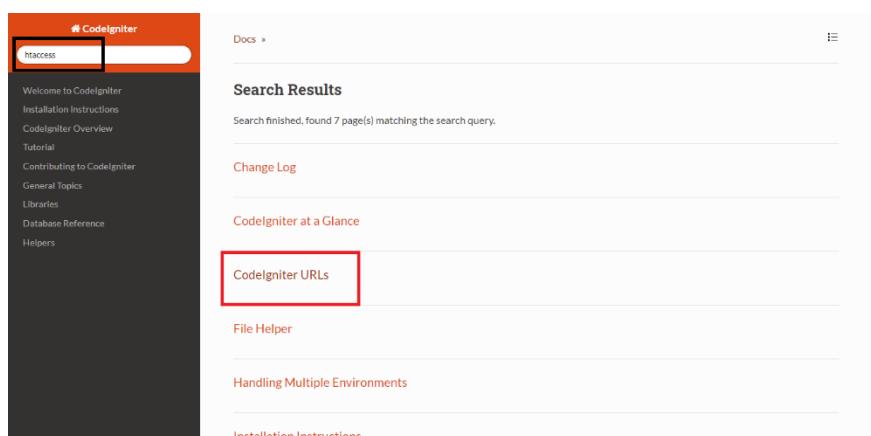
Gambar 5.16 Membuat File Baru *.htaccess*

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
```

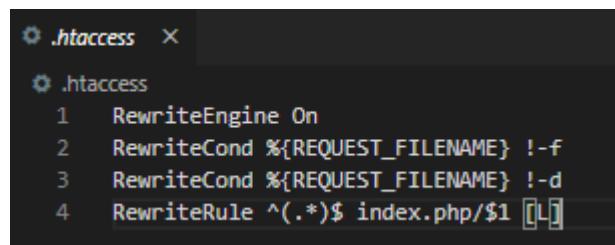
If your Apache server has `mod_rewrite` enabled, you can easily remove this file by using a `.htaccess` file with some simple rules. Here is an example of such a file, using the “negative” method in which everything is redirected except the specified items:

```
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.

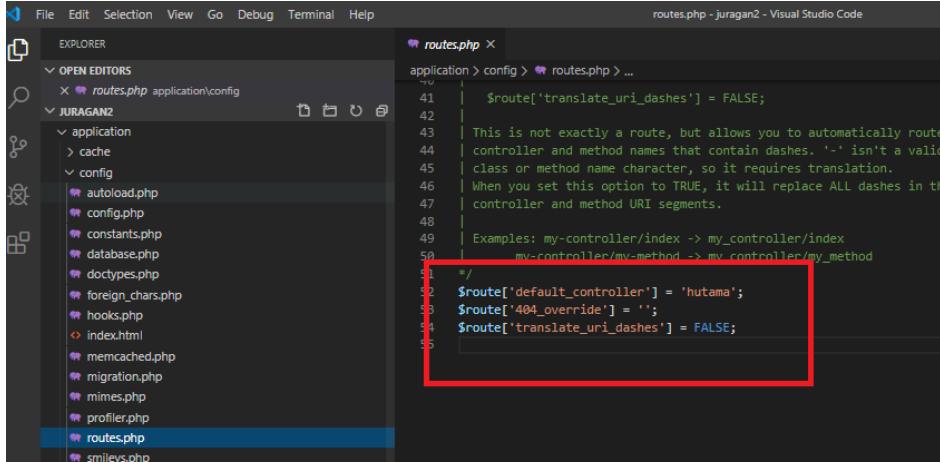


```
❶ RewriteEngine On
❷ RewriteCond %{REQUEST_FILENAME} !-f
❸ RewriteCond %{REQUEST_FILENAME} !-d
❹ RewriteRule ^(.*)$ index.php/$1 [L]
```

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 *welcome*. 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.



```

File Edit Selection View Go Debug Terminal Help
routes.php - juragan2 - Visual Studio Code
EXPLORER
OPEN EDITORS
routes.php application\config
JURAGAN2
application
cache
config
autoload.php
config.php
constants.php
database.php
doctypes.php
foreign_chars.php
hooks.php
index.html
memcached.php
migration.php
mimes.php
profiler.php
routes.php
smileys.php
routes.php X
application > config > routes.php > ...
41 | $route['translate_uri_dashes'] = FALSE;
42 |
43 | This is not exactly a route, but allows you to automatically route
44 | controller and method names that contain dashes. '-' isn't a valid
45 | class or method name character, so it requires translation.
46 | When you set this option to TRUE, it will replace ALL dashes in the
47 | controller and method URI segments.
48 |
49 | Examples: my-controller/index -> my_controller/index
50 | mv-controller/mv-method -> my_controller/my_method
51 */
52 $route['default_controller'] = 'hutama';
53 $route['404_override'] = '';
54 $route['translate_uri_dashes'] = FALSE;
55

```

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.

```

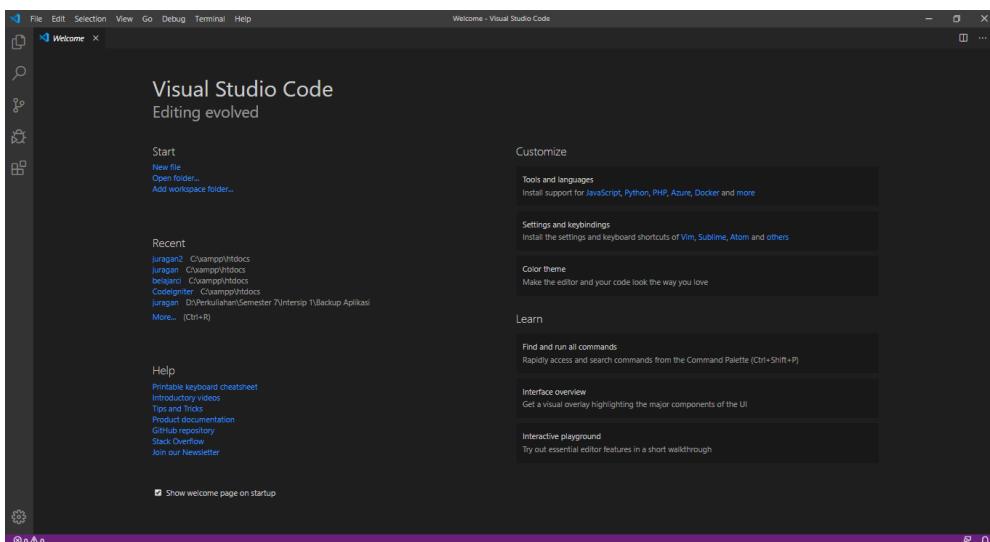
72
73 $active_group = 'default';
74 $query_builder = TRUE;
75
76 $db['default'] = array(
77     'dsn' => '',
78     'hostname' => 'localhost', Nama Server Xampp
79     'username' => 'root', Username Xampp
80     'password' => '', Password Xampp, Diisi Jika Ada
81     'database' => 'juragandua', Nama 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*, jangan lupa lakukan *start* pada *module Apache* dan *MySQL* yah teman – teman seperti pada gambar 4.10 dan buka *software visual studio code* teman – teman atau *software text editor* lainnya yang kalian sukai.



Gambar 5.23 Mempersiapkan Software VSCode

5.5.1 Membuat Database

Langkah awal dalam membuat aplikasi CRUD pada suatu sistem tentunya teman – teman perlu memiliki sebuah *database* untuk menampung data – data pada aplikasi teman – teman. Apakah teman – teman sudah bisa membuat *database* ?, jika sudah maka teman – teman bisa melewati tahap membuat *database* tersebut. Saya orang awam nih di dunia pemrograman, saya belum mengerti bagaimana cara membuat *database*, tenang teman – teman gak perlu bingung karena pada kesempatan kali ini penulis akan memberikan bagaimana cara membuat *database*.

Untuk mengekuti *tutorial* membuat *database* ini dimana pastikan teman – teman sudah mempunyai *software Xampp* pada laptop atau komputer teman – teman. *Xampp* ? apa itu *Xampp* ?, hayo teman – teman lupa yah coba buka kembali di halaman 48. Bagaimana apakah teman – teman sudah paham mengenai *software Xampp* ?, apabila sudah yuk kita

lanjut ke tahap berikutnya, pastikan *software* teman – teman sudah berjalan dengan baik.

Untuk mengakses halaman *Xampp* tersebut dimana teman – teman perlu mengunjungi alamat *http://localhost/dashboard/* pada aplikasi *web browser* teman – teman. Dimana teman – teman akan di bawa ke halaman *dashboard Xampp* seperti yang ditunjukkan pada gambar 5.24.



Welcome to XAMPP for Windows 5.6.40

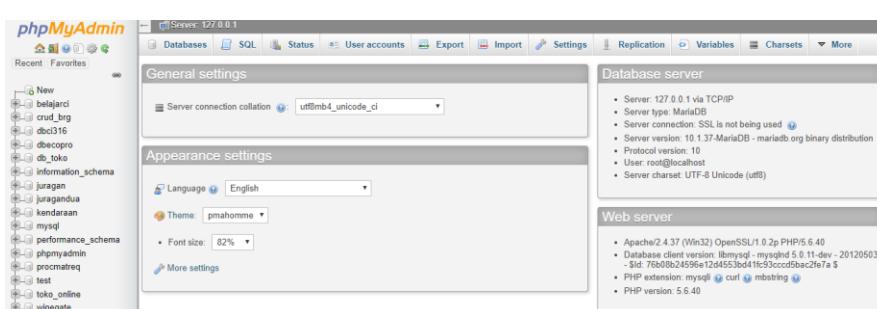
You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the FAQs to learn how to protect your site. Alternatively you can use WAMP, MAMP or LAMP which are similar packages which are more suitable for production.

Start the XAMPP Control Panel to check the server status.

Gambar 5.24 Dashboard Xampp

Perhatikan bagian *header* pada *dashboar Xampp* tersebut, dimana teman – teman dapat melihat menu *phpMyAdmin*, silahkan teman – teman klik menu tersebut dan teman – teman akan di bawa ke halaman *phpMyAdmin* tersebut seperti yang ditunjukkan pada gambar 5.25.



Gambar 5.25 Halaman phpMyAdmin

Halaman *phpMyAdmin* tersebut berfungsi untuk mengolah *database* yang akan teman – teman buat. Silahkan teman – teman pilih menu *database* pada bagian *header* di halaman *phpMyAdmin* tersebut.

The screenshot shows the phpMyAdmin interface with the following details:

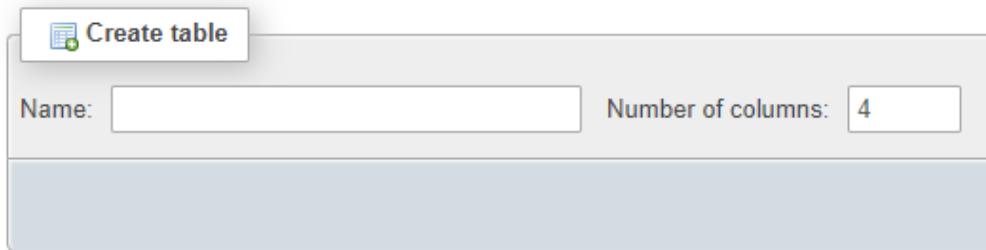
- Left Sidebar:** Shows a tree view of databases: New, belajarci, crud_brg, dbci316, dbecopro, db_toko, information_schema, juragan, juragandua, kendaraan, mysql, performance_schema, phpmyadmin, procmatreq, test, toko_online, winegate.
- Header:** Shows 'Server: 127.0.0.1' and tabs: Databases, SQL, Status, User accounts, Export, Import.
- Main Content:**
 - Databases Section:** Displays a 'Create database' form with 'Database name' set to 'latin1_swedish_ci' and a 'Create' button.
 - Filters Section:** Contains a search bar 'Containing the word:'.
 - Table Section:** Shows a table of existing databases:

Database	Collation	Action
belajarci	latin1_swedish_ci	
crud_brg	latin1_swedish_ci	
dbci316	latin1_swedish_ci	
dbecopro	latin1_swedish_ci	
db_toko	latin1_swedish_ci	

Gambar 5.26 Halaman Database Pada phpMyAdmin

Perhatikan yang diberikan tanda kotak merah pada gambar 5.26 tersebut, dimana terdapat sebuah *form* untuk membuat nama dari *database* kalian. Pada kolom *database* silahkan teman – teman isi nama dari *database* teman – teman contoh dimana penulis akan mengisikan dengan “juragandua” lalu pilih *button create* untuk membuat *database* tersebut dan selamat teman – teman sudah berhasil membuat *database*.

Apabila teman – teman sudah membuat *database* dimana teman – teman perlu membuat beberapa *table* yang dibutuhkan. Tabel tersebut yang akan berfungsi untuk menyimpan dan mengambil data pada aplikasi yang akan teman – teman buat. Bagaimana sih cara membuat tabel ?, tenang saja dikarenakan halaman *phpMyAdmin* tersebut dibuat dengan *user friendly* atau mudah untuk dimengerti bagi kalian yang masih awam dalam melakukan pemrograman.



Gambar 5.27 Membuat Tabel Pada Database

Perhatikan gambar 5.27 dimana teman – teman akan menemukan *form create table* pada *file database* yang sudah teman – teman buat. Pada kolom *Name* silahkan isi nama tabel yang teman – teman inginkan misalnya saja “*tb_barang*” sedangkan pada kolom *Number of columns* silahkan teman – teman isi dengan jumlah kolom yang akan di berikan pada tabel yang akan dibuat lalu pilih *button Go* untuk menuju proses berikutnya. Silahkan teman – teman buat kolom apa saja yang dibutuhkan sebagai contoh pada sistem JURAGAN isi dari tabel “*tb_barang*” dapat dilihat pada tabel 3.13 lalu pilih *button save* untuk menyimpan tabel tersebut dan selamat teman – teman sudah membuat tabel pada *database* teman – teman.

5.5.2 Membuat Controllers

Langkah selanjutnya teman – teman akan membuat *file controllers*, tapi sebelum itu silahkan teman – teman siapkan dulu *file codeigniter* yang sudah berjalan dengan baik. Jika teman – teman belum memiliki *file codeigniter* silahkan *download* terlebih dahulu dengan cara mengikuti *tutorial* pada sub bab 4.3.1. Apa bila teman – teman sudah memiliki *file codeigniter* tersebut silahkan lakukan konfigurasi terlebih dahulu, bagaimana sih cara konfigurasinya ? silahkan teman – teman buka kembali ke bagian sub bab 5.4.

Bagaimana apakah teman – teman sudah menyiapkan semua *file* yang dibutuhkan ?, jika sudah yuk kita masuk ke tahap membuat *controllers* pada *framework codeigniter*. Silahkan teman – teman buka direktori *controllers* tersebut di *Application/controllers*. Pada direktori *controllers* tersebut silahkan teman – teman buat *file* baru dengan bentuk PHP, sebagai contoh pada sistem JURAGAN telah terbentuk *file Store.php*. Catatan dimana nama *file controllers* tersebut bebas sesuai apa yang teman – teman inginkan jika teman – teman ingin mengubahnya menjadi *Toko.php* tidak di permasalahkan.

Apabila teman – teman sudah memiliki satu *file* baru dengan bentuk PHP pada *controllers* teman – teman silahkan lakukan pengkodean terhadap *file* tersebut. Yuk kita belajar mengedit *file controllers* tersebut.

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

class Store extends CI_Controller
{
```

Source Code 5.1 Tahap Awal Mengedit File Controllers

Perhatikan *script code 5.1* tersebut dimana kita akan membuat sebuah *class* dengan nama *Stroe*. *Class* tersebut merupakan sebuah nama *file* PHP yang telah kita buat sebelumnya. Perlu teman – teman ketahui terlebih dahulu bahwa *controllers* dibuat sebagai pengontrol dari aplikasi yang akan kita buat, maka dari itu kita akan buat beberapa *function* yang berguna untuk mengontrol fungsi dari aplikasi tersebut.

Silahkan teman – teman buat *function inventory* sebagai *function* pertama pada *file controllers* teman – teman, perhatikan *source code 5.2* berikut.

```

public function inventory()
{
    $data['title'] = 'Halaman Toko';
    $data['barang'] = $this->m_barang->
        inventory()->result();

    $this->load->view('store/templates/header',
        $data);
    $this->load->view('store/templates/sidebar');
    $this->load->view('store/templates/topbar',
        $data);
    $this->load->view('store/inventory');
    $this->load->view('store/templates/footer');

}

```

Source Code 5.2 Function Inventory Pada Controllers

Pada *script code* 5.2 tersebut merupakan *function inventory* yang berfungsi untuk menampilkan data – data pada tabel yang berada pada *database* yang sudah kita buat sebelumnya. Mari kita jelaskan *code* tersebut, dimana pada baris pertama teman – teman akan membuat sebuah ***function*** dengan nama ***inventory*** dan dibuat secara ***public*** agar dapat diakses secara menyeluruh. Pada baris selanjutnya dimana teman – teman akan membuat sebuah ***variabel data***, dalam *framework codeigniter* dimana *variabel* diawali dengan *symbol \$*. Pada *variabel* pertama diisi dengan *fungsi tittle* yang dimana untuk menampung judul dari halaman tersebut. Pada *variabel* berikutnya dimana diisikan dengan *fungsi model* yang berguna untuk memanggil *model* yang akan kita buat. Pada baris selanjutnya dimana teman – teman akan melakukan *load file* pada direktori *view* dengan cara memasukkan *code* ***\$this->load->view('file yang akan dipanggil');***. Contoh pada baris ke-4 dimana kita akan memanggil

file header.php yang berada pada *folder store/templates* pada direktori *view*.

Selanjutnya dimana kita akan membuat *function input*, silahkan teman – teman perhatikan *source code* berikut.

```
public function inputbarang()
{
    $email      = $this->input->post('email');
    $id         = $this->input->post('id');
    $nama_brg   = $this->input->post(
        'nama_brg');
    $keterangan = $this->input->post(
        'keterangan');
    $detail     = $this->input->post(
        'detail');
    $kategori   = $this->input->post(
        'kategori');
    $harga       = $this->input->post('harga');
    $stok        = $this->input->post('stok');
    $tlp_toko   = $this->input->post('tlpn');
    $alamat     = $this->input->post(
        'alamat');
    $kecamatan  = $this->input->post(
        'kecamatan');
    $kelurahan  = $this->input->post(
        'kelurahan');
    $kota        = $this->input->post(
        'kota');
    $k_pos       = $this->input->post(
        'kode_pos');

    $gambar      = $_FILES['gambar']['name'];
    if ($gambar == '') {
    } else {
        $config['upload_path'] = './uploads';
        $config['allowed_types'] = 'jpg|jpeg|png';

    $this->load->library('upload', $config);
}
```

```

        if (!$this->upload->do_upload('gambar')) {
            echo "Gagal dalam meng-Upload Gambar!";
        } else {
            $gambar = $this->upload->data('file_name');
        }
    }

    $data = array(
        'email'          => $email,
        'id'             => $id,
        'nama_brg'       => $nama_brg,
        'keterangan'    => $keterangan,
        'detail'         => $detail,
        'kategori'       => $kategori,
        'harga'          => $harga,
        'stok'           => $stok,
        'tlpn'           => $tlp_toko,
        'alamat'         => $alamat,
        'kecamatan'      => $kecamatan,
        'kelurahan'     => $kelurahan,
        'kota'           => $kota,
        'kode_pos'        => $k_pos,
        'gambar'          => $gambar
    );
}

$this->m_barang->inputdata($data, 'tb_barang');
redirect('store/inventory');
}

```

Source Code 5.3 Function Input Pada Controllers

Perhatikan *script code* 5.3 tersebut yang dimana merupakan *controllers inputbarang* yang berfungsi sebagai mengontrol fungsi *input* pada sistem yang akan teman – teman buat. Mari kita jelaskan dimana teman – teman akan membuat sebuah *function* baru yang bersifat *public* dengan nama *inputbarang*. Baris bari selanjutnya dimana teman – teman perlu membuat fungsi *post* dengan menerapkan ke dalam beberapa *variabel* dengan menggunakan *code* **\$this->input->post('nama kolom');**. Agar

mudah di ingat buat saja *variabel* tersebut dengan nama kolom pada tabel *database* yang sudah kita buat sebelumnya. Contoh, `$email = $this->input->post('email');`, yang dimana kita akan membuat *variabel email* yang akan menerima *inputan* dari *array* dan di *post* ke dalam tabel yang ada pada *database* ke bagian kolom *email*, begitu juga dengan *variabel – variabel* yang lain. Pada bagian gambar dimana terlihat berbeda, hal ini dikarenakan *codeigniter* memuliki cara khusus yang sudah dijelaskan pada *userguide codeigniter*, silahkan teman – teman buka saja pada *website resmi codeigniter* untuk penjelasannya. Selanjutnya teman – teman akan membuat data *array* yang dimana data *array* tersebut akan di panggil ke dalam *variabel* yang sudah kita buat di awal. Pada baris berikutnya dimana teman – teman akan memanggil fungsi *model* yang akan kita buat pada sub bab 5.5.3. Pada baris berikutnya dimana apabila kita sudah selesai melakukan proses *input* maka sistem akan melakukan *redirect* ke halaman *inventory*.

Dari kedua *function* yang sudah kita apakah teman – teman sudah paham ?, jika sudah yuk kita lanjut ke tahap berikutnya. Apabila ada *input* yang pasti ada *edit* dan *delete* dong, pada pembelajaran kali ini masih pada tahap membuat *controllers*, dimana kita akan mempelajari mengenai *controllers edit* dan *delete*, perhatikan *source code* berikut.

```
public function edit_brg($id)
{
    $where = array('id_brg' => $id);

    $data['barang'] = $this->m_barang->
        edit_barang($where, 'tb_barang')->result();
    $data['title'] = 'Edit Produk';

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

```

$this->load->view('store/templates/sidebar');
$this->load->view(
    'store/templates/topbar', $data);
$this->load->view('store/edit_brg', $data);
$this->load->view('store/templates/footer');
}

```

Source Code 5.4 Function Edit Pada Controllers

Perhatikan *source code* 5.4 tersebut dimana merupakan *function edit* ada *file controllers* kita. *Controllers edit* tersebut hanya berfungsi sebagai memanggil atau melakukan *load* untuk halaman *edit* dan belum bisa melakukan fungsi *edit*, akan tetapi coba teman – teman perhatikan *code* yang diberi warna merah tersebut. Dimana kita akan membuat *function edit_brg* yang bersifat *public* namun diisikan dengan *variabel id*. Hal ini dikarenakan pada halaman *edit* tersebut dimana kita hanya akan merubah satu data saja maka dari itu kita memerlukan sebuah *primary key* untuk memanggil data tersebut. Coba teman – teman perhatikan *code* bagian **\$where = array('id_brg' => \$id);**, dimana *code* tersebut berfungsi untuk mengirimkan *array* berdasarkan *id* yang dikirim dari kolom *id_brg*.

Pada pembelajaran sebelumnya dimana kita sudah membuat pemanggilan atau melakukan *load* pada *view edit_brg*, namun belum bisa melakukan fungsi *edit*. Bagaimana cara membuat fungsi *edit* tersebut ?, silahkan teman – teman perhatikan *source code* berikut.

```

public function update()
{
    $id          = $this->input->post('id_brg');
    $nama_brg    = $this->input->post(
                    'nama_brg');
    $keterangan = $this->input->post(
                    'keterangan');
}

```

```

$detail      = $this->input->post('detail');
$kategori    = $this->input->post(
                    'kategori');
$harga       = $this->input->post('harga');
$stok        = $this->input->post('stok');
$tlp_toko    = $this->input->post('tlpn');
$alamat      = $this->input->post('alamat');
$kecamatan   = $this->input->post(
                    'kecamatan');
$kelurahan   = $this->input->post(
                    'kelurahan');
$kota         = $this->input->post('kota');
$k_pos        = $this->input->post(
                    'kode_pos');

$data = array(
    'nama_brg'      => $nama_brg,
    'keterangan'    => $keterangan,
    'detail'        => $detail,
    'kategori'      => $kategori,
    'harga'          => $harga,
    'stok'           => $stok,
    'tlpn'           => $tlp_toko,
    'alamat'         => $alamat,
    'kecamatan'     => $kecamatan,
    'kelurahan'     => $kelurahan,
    'kota'           => $kota,
    'kode_pos'       => $k_pos
);

$where = array(
    'id_brg' => $id
);

$this->m_barang->update_data($where, $data,
                                'tb_barang');
redirect('store/inventory');
}

```

Source Code 5.5 Function Update Pada Controllers

Pada *source code* 5.5 tersebut merupakan *function update* pada *controllers* yang telah kita buat. *Function* tersebut akan melakukan kontrol terhadap fungsi *update/edit* pada sistem yang akan kita buat. Pada *function* sebenarnya sama saja seperti *function inputbarang* hanya saja yang membedakan terdapat pada *code* yang diberikan tanda merah tersebut. Perhatikan *code* **`$where = array('id_brg' => $id);`**, yang dimana akan membuat *variabel where* untuk memanggil *array* yang dikirim oleh *variabel data* pada tabel di dalam *database* kita dari kolom *id_brg* yang deskripsikan ke dalam *variabel id*. Pada baris berikutnya dimana kita akan membuat fungsi untuk memanggil *model* yang akan kita buat pada sub bab 5.5.3. Seperti pada *controllers inputbarang* dimana kita akan menerapkan fungsi *redirect* yang dituju ke halaman *inventory*.

Pada pembelajaran berikutnya masih pada tahap membuat *controllers* yang dimana merupakan pembuatan *function* terakhir dari *controllers* kita. *Function* tersebut merupakan *function delete* yang berfungsi untuk mengontrol proses *delete* pada sistem yang akan kita buat. Perhatikan *source code* berikut.

```
public function delete($id)
{
    $where = array('id_brg' => $id);
    $this->m_barang->hapus_data(
        $where, 'tb_barang');
    redirect('store/inventory');
}
```

Source Code 5.6 Function Delete Pada Controllers

Pada *source code* 5.6 tersebut dimana merupakan *function delete* pada *file controllers* kita. Untuk membuat *function delete* tersebut cukuplah sederhana,

seperti halnya pada *function edit_brg* dimana kita hanya akan melakukan hapus terhadap satu data saja maka dari itu kita akan membuat *function* yang bersifat *public* dengan nama *delete* dan di isi dengan *variabel id*. Pada selanjutnya dimana kita akan membuat *variabel where* yang berfungsi untuk membaca data berdasarkan *id* yang diberikan oleh tabel pada *database* kita dalam kolom *id_brg*. Untuk mengontrol fungsi *delete* tersebut dimana teman – teman cukup menjalankan fungsi *models* yang akan kita buat pada sub bab 5.5.3. Oh iyaa ketinggal nih menjelaskan bagaimana cara memanggil *models*, silahkan teman – teman perhatikan *code* berikut, **\$this->m_barang->hapus_data(\$where, 'tb_barang');**, dimana dapat dijelaskan teman – teman akan memanggil *models m_barang* dengan *function hapus_data* yang terdapat pada *models* tersebut dan mendeskripsikan *variabel where* terhadap **tb_barang**. Bagaimana apakah teman – teman sudah mengerti cara memanggil *models* pada *file controllers* ?, langkah selanjutnya dimana teman – teman tinggal jalankan fungsi *redirect* ke halaman *inventory*. Selamat *file controllers* kita sudah selesai sehingga tampilan full *source code controllers* dapat dilihat pada *source code 5.7* berikut.

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

class Store extends CI_Controller {

    public function inventory()
    {
        $data['title'] = 'Halaman Toko';
        $data['barang'] = $this->m_barang->
            inventory()->result();

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

```
$this->load->view('store/templates/sidebar');
$this->load->view('store/templates/topbar',
    $data);
$this->load->view('store/inventory');
$this->load->view('store/templates/footer');

}

public function inputbarang()
{
    $email      = $this->input->post('email');
    $id         = $this->input->post('id');
    $nama_brg   = $this->input->post(
        'nama_brg');
    $keterangan = $this->input->post(
        'keterangan');
    $detail     = $this->input->post(
        'detail');
    $kategori   = $this->input->post(
        'kategori');
    $harga       = $this->input->post('harga');
    $stok        = $this->input->post('stok');
    $tlp_toko   = $this->input->post('tlpn');
    $alamat     = $this->input->post(
        'alamat');
    $kecamatan  = $this->input->post(
        'kecamatan');
    $kelurahan  = $this->input->post(
        'kelurahan');
    $kota        = $this->input->post(
        'kota');
    $k_pos       = $this->input->post(
        'kode_pos');

    $gambar      = $_FILES['gambar']['name'];
    if ($gambar == '') {
    } else {
        $config['upload_path'] = './uploads';
        $config['allowed_types'] = 'jpg|jpeg|png';
    }
}
```

```

$this->load->library('upload', $config);
if (!$this->upload->do_upload('gambar')) {
echo "Gagal dalam meng-Upload Gambar!";
} else {
$gambar = $this->upload->data('file_name');
}
}

$data = array(
'email'          => $email,
'id'             => $id,
'nama_brg'       => $nama_brg,
'keterangan'    => $keterangan,
'detail'         => $detail,
'kategori'      => $kategori,
'harga'          => $harga,
'stok'            => $stok,
'tlpn'           => $tlp_toko,
'alamat'         => $alamat,
'kecamatan'     => $kecamatan,
'kelurahan'    => $kelurahan,
'kota'           => $kota,
'kode_pos'       => $k_pos,
'gambar'         => $gambar
);

$this->m_barang->inputdata(
    $data, 'tb_barang');
redirect('store/inventory');
}

public function edit_brg($id)
{
    $where = array('id_brg' => $id);

    $data['barang'] = $this->m_barang->
        edit_barang($where, 'tb_barang')->result();
    $data['title'] = 'Edit Produk';

    $this->load->view(
        'store/templates/header', $data);
}

```

```

$this->load->view('store/templates/sidebar');
$this->load->view(
    'store/templates/topbar', $data);
$this->load->view('store/edit_brg', $data);
$this->load->view('store/templates/footer');
}

public function update()
{
    $id          = $this->input->post('id_brg');
    $nama_brg    = $this->input->post(
                    'nama_brg');
    $keterangan = $this->input->post(
                    'keterangan');
    $detail      = $this->input->post('detail');
    $kategori   = $this->input->post(
                    'kategori');
    $harga       = $this->input->post('harga');
    $stok        = $this->input->post('stok');
    $tlp_toko   = $this->input->post('tlpn');
    $alamat     = $this->input->post('alamat');
    $kecamatan  = $this->input->post(
                    'kecamatan');
    $kelurahan  = $this->input->post(
                    'kelurahan');
    $kota        = $this->input->post('kota');
    $k_pos       = $this->input->post(
                    'kode_pos');

    $data = array(
        'nama_brg'      => $nama_brg,
        'keterangan'    => $keterangan,
        'detail'         => $detail,
        'kategori'      => $kategori,
        'harga'          => $harga,
        'stok'           => $stok,
        'tlpn'           => $tlp_toko,
        'alamat'         => $alamat,
        'kecamatan'      => $kecamatan,
        'kelurahan'      => $kelurahan,
    );
}

```

```

        'kota'          => $kota,
        'kode_pos'      => $k_pos
    );

$where = array(
    'id_brg' => $id
);

$this->m_barang->update_data($where, $data,
                                'tb_barang');
redirect('store/inventory');
}

public function delete($id)
{
    $where = array('id_brg' => $id);
    $this->m_barang->hapus_data(
        $where, 'tb_barang');
    redirect('store/inventory');
}
}

```

Source Code 5.7 Code Full Controllers

5.5.3 Membuat *Models*

Pada sub bab sebelumnya dimana kita sudah membahas bagaimana cara membuat *controllers* pada *framework codeigniter*, langkah selanjutnya dimana teman – teman akan belajar mengenai *models* pada *framework codeigniter*. Untuk membuat *models* pastikan teman – teman sudah melakukan konfigurasi terhadap *file autolod.php* yang terdapat pada direktori *application/config*.

Silahkan teman – teman buka direktori *application/models* dan *file* baru dengan bentuk PHP. Perlu diingat dimana *file* tersebut disesuaikan dengan yang sudah teman – teman konfigurasi pada *file autoload.php*.

Contoh pada saat saya melakukan konfigurasi pada *file autoload.php* terdapat *m_barang* dimana *file* yang akan saya buat pada direktori *models* tersebut adalah *m_barang.php*. Untuk konfiguras dimana teman – teman dapat melihatnya pada sub bab 5.4.

Sekarang saya sudah memiliki *file* baru pada direktori *models*, langkah pertama dimana kita perlu membuat sebuah *class* agar *models* tersebut dapat kita panggil pada *file controllers* kita. Perhatikan *source code* berikut.

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

Source Code 5.8 Membuat Class Pada File Models

Pada *source code* 5.8 tersebut dimana merupakan *code* awal dalam membuat *file models* kita, dimana dapat dijelaskan bahwa kita akan membuat sebuah *class* terhadap *file models* kita, *class* tersebut berfungsi untuk menampung fungsi – fungsi yang akan kita berikan pada *file models*. Pada umumnya *file models* berisi *code – code* yang terkait ke dalam *database*.

Sama halnya seperti *file controllers* dimana *models* juga memerlukan *function – function* untuk mengelola *code – code* tersebut. Perhatikan kembali *source code* 5.7 pada *function inventory*, dimana terdapat pemanggilan *models* dengan *class m_barang* dan *function inventory*. Mari kita buat *function* tersebut pada *file models* kita, perhatikan *source code* berikut.

```
public function inventory()
{
    return $this->db->get('tb_barang');
```

Source Code 5.9 Membuat Function Inventory Pada File Models

Pada *source code* 5.9 tersebut dimana merupakan *function inventory* yang diberikan fungsi pemanggilan isi dari *database*. Mari kita bahas dari tiap – tiap barisnya, dimana pada baris pertama kita akan membuat sebuah *function* yang bersifat *public* dengan *inventory*. Lalu pada baris berikutnya dimana kita akan membuat fungsi pemanggilan data tersebut dengan memasukkan *code* `$this->db->get('tb_barang');`. Untuk memanggil data dari sebuah *database* dimana *codeigniter* memiliki fungsi `$this->db`. Lalu terdapat *get* yang dimana merupakan sebuah fungsi untuk mengambil suatu data. Data mana yang akan di ambil ?, berikan *value* nama tabel yang sudah teman – teman buat pada *database*. Dengan menerapkan fungsi *get* tersebut dimana sistem akan menampilkan semua data yang berada pada tabel *tb_barang* tersebut.Jangan lupa tambahkan *return* pada awal kalimat, hal tersebut berguna untuk menghentikan proses fungsi yang sedang berjalan dan menyatakan bahwa hasil akhir dari fungsi tersebut akan menampilkan isi data pada tabel *tb_barang*.

Apabila fungsi tersebut memanggil semua data yang berada pada tabel *tb_barang*, lalu bagaimana jika saya ingin menampilkan isi data sesuai *session* yang diberikan ?. Hal tersebut biasanya berguna saat teman – teman menerapkan fungsi *session* pada sistem yang akan dibuat. Cara nya sangat mudah dimana teman – teman tinggal mengganti fungsi *get* tersebut dengan *get_where* dan masukkan *code* tersebut disertai dengan *session*, perhatikan *source code* 5.10 berikut.

```
return $this->db->get_where('tb_barang', array(
    'email' => $this->session->
        userdata('email')));
```

Source Code 5.10 Membuat Fungsi Get_Where Dengan Session

Bagaimana apabila sistem saya tidak menerapkan fungsi *session* tapi ingin meneapkan fungsi *get_where* ?. Hal tersebut pun dapat diterapkan, coba teman – teman perhatikan *source code* 5.11 berikut.

```
return $this->db->get_where('user', array(
    'status_toko' =>
        'Sudah Terverifikasi'));
```

Source Code 5.11 Membuat Fungsi Get_Where Tanpa Session

Pada *source code* tersebut dimana menjelaskan bahwa teman – teman akan memanggil tabel *user* yang ada pada *database* dan hanya menampilkan data apa yang memiliki *value* “**Sudah Terverifikasi**” pada kolom “**status_toko**”.

Langkah selanjutnya coba teman – teman perhatikan *function inputbarang* pada *file controllers* yang sudah kita buat pada sub bab sebelumnya. Pasti teman – teman menemukan sebuah *code* seperti berikut, **\$this->m_barang->inputdata(\$data, 'tb_barang');**. Pada *code* tersebut dimana teman – teman akan memanggil *function inputdata* yang terdapat pada *file models m_barang.php*. Karena *function* tersebut belum terbuat, yuk kita buat dulu *function inputdata* tersebut pada *file models* kita. Perhatikan *source code* berikut.

```
public function inputdata($data, $table)
{
    $this->db->insert($table, $data);
}
```

Source Code 5.12 Membuat Function Inputdata Pada File Models

Pada *source code* 5.12 tersebut merupakan *function inputdata* yang dimana berfungsi sebagai fungsi *input* ke dalam *database*. Dimana pada baris pertama seperti biasa kita akan membuat *function* baru yang bersifat *public* dengan nama *inputdata*. Pada *function* tersebut dimana kita akan memberikan *value* *\$data* dan *\$table*. Pada *varibel* data dimana dibuat untuk mengambil *array* yang telah kita buat pada *file controllers* kita sedangkan *variabel* *table* dibuat untuk membaca *table* yang kita panggil pada *file controllers* kita. Pada baris selanjutnya dimana kita akan memanggil fungsi *database* dengan *code* *\$this->db->* dan memberikan fungsi *insert* kedalamnya. *Insert* tersebut merupakan salah satu fungsi *database* yang digunakan untuk melakukan proses *input* atau *create* terhadap suatu *table* yang dituju, dimana fungsi *insert* tersebut memerlukan *value* untuk membaca data, masukkan *\$table* dan *\$data* yang sudah kita deskripsikan pada *function*.

Langkah selanjutnya dimana teman – teman perlu membuat *function* *edit_barang* yang dimana berfungsi untuk mengambil *data* berdasarkan *id* yang dikirim. Bagaimana cara membuatnya ?, silahkan teman – teman perhatikan *source code* berikut.

```
public function edit_barang($where, $table)
{
    return $this->db->get_where($table, $where);
}
```

Source Code 5.13 Membuat Function edit_barang Pada File Models

Pada *source code* 5.13 tersebut dimana merupakan *function edit_barang* pada *file models* kita. Pada baris pertama dimana kita akan membuat *function* baru yang bersifat *public* dengan nama *edit_barang* dan akan mendeskripsikan *variabel where* dan *variabel table*. Ingat *variabel* dalam *framework codeigniter* tersebut selalu di awali dengan *symbol* “\$”. Pada baris selanjutnya dimana kita gunakan fungsi *get_where* yang sudah kita bahas sebelumnya. Fungsi *get_where* tersebut akan membaca salah satu data sesuai *id* yang dikirim. Jangan lupa panggil kedua *variabel* tersebut kedalam fungsi *get_where* tersebut yaa teman – teman. Pada *variabel table* dimana berfungsi untuk memanggil *table* yang sudah kita deskripsikan pada *file controllers*, tentunya *table* tersebut sudah dibuat pada *database* kita yaa teman – teman. Sedangkan *variabel where* berfungsi untuk memanggil *id* yang terdapat pada *table* yang akan kita panggil. Jangan lupa tambahkan fungsi *return* di awal *code* tersebut.

Perlu diingat dimana pada *source code* 5.13 tersebut hanya berfungsi untuk pemanggilan data saja dan belum bisa melakukan proses *edit* tersebut. Gimana caranya agar data melakukan proses *edit* ?, coba teman – teman perhatikan kembali *file controllers* yang sudah kita buat sebelumnya, lihat *function update*, dimana isi dari *function* tersebut merupakan sebuah proses untuk melakukan *edit* data namun agar data tersebut dapat terhubung ke dalam *database* kita perlu menghubungkannya dengan fungsi *models* dengan cara membuat sebuah *function* baru pada *file models* kita. Perhatikan *source code* berikut.

```
public function update_data($where, $data,
                           $table)
{
    $this->db->where($where);
    $this->db->update($table, $data);
}
```

Source Code 5.14 Membuat Function *update_data* Pada File Models

Pada *source code* 5.14 tersebut dimana merupakan *function update_data* pada *file models* kita yang bertujuan untuk menghubungkan fungsi pada *file controllers* tersebut ke dalam *database*. Pada baris pertama dimana kita akan membuat *function* baru yang bersifat *public* dengan nama *update_data* yang di deskripsikan dengan variabel *where*, *data*, dan *table*. Pada baris selanjutnya dimana kita akan menerapkan fungsi *where* pada *database*. Apa sih bedanya fungsi *get_where* dan *where* ? dimana fungsi *get_where* digunakan untuk mengambil suatu data sedangkan fungsi *where* digunakan untuk merubah data. Karena kita akan melakukan *update* terhadap data kita yang dimana data tersebut akan berubah, maka gunakan fungsi *where*. Pada baris selanjutnya dimana teman – teman perlu menambahkan fungsi *update* sebagai membaca data mana yang mau kita *update* dan disimpan ke dalam *table* mana pada *database* kita.

Membuat *function* untuk menghubungkan proses *input* sudah, untuk menghubungkan *edit* dan *update* sudah, untuk mengbungkan fungsi *view* juga udah, satu lagi nih teman – teman yaitu *function* untuk menghubungkan proses *delete* jangan sampe ketinggalan yah. Bagaimana sih cara membuat *models* untuk memproses fungs *delete* ?, simpel kok teman – teman, coba perhatikan *source code* berikut ini.

```
public function hapus_data($where, $table)
{
    $this->db->where($where);
    $this->db->delete($table);
}
```

Source Code 5.15 Membuat Function Delete Pada File Models

Pada *source code* 5.15 tersebut dimana merupakan *function* yang berfungsi untuk menghubungkan proses *delete* ke dalam *database* pada *file models* kita. Seperti biasa yang sudah kita lakukan sebelumnya dimana kita akan membuat *function* baru yang bersifat *public* dengan nama *hapus_data* pada *file models* kita dan mendeskripsikan variabel *where* dan *table*. Pada baris selanjutnya dimana kita akan menghapus data tersebut berdasarkan *id* yang diberikan maka gunakan fungsi *where* dengan memanggil variabel *where* untuk mengambil *id* pada data tersebut. Pada baris selanjutnya dimana kita akan menggunakan fungsi *delete* pada *database* yang dimana berguna untuk menghapus data. Jangan lupa panggil variabel *table* pada fungsi tersebut yaa teman – teman untuk membaca pada *table* mana kita akan menghapus data tersebut. Bagaimana, simpel kan ?, selamat teman – teman sudah selesai membuat *file models* tersebut sehingga *source code full* dari *file models* kita akan terlihat seerti pada *source code* 5.16 berikut.

```
<?php

class M_barang extends CI_Model
{

public function inventory()
{
    return $this->db->get('tb_barang');
}
```

```

public function inputdata($data, $table)
{
    $this->db->insert($table, $data);
}

public function edit_barang($where, $table)
{
    return $this->db->get_where($table, $where);
}

public function update_data($where, $data,
                      $table)
{
    $this->db->where($where);
    $this->db->update($table, $data);
}

public function hapus_data($where, $table)
{
    $this->db->where($where);
    $this->db->delete($table);
}

}

```

Source Code 5.16 Code Full File Models

5.5.4 Membuat View

View merupakan salah satu bagian terpenting dalam suatu sistem. Pada pembuatan *view* kali ini dimana kita akan menggunakan fungsi *bootstrap* sebagai *css* dari sistem kita, untuk mempelajari bagaimana cara menghubungkan *framework codeigniter* silahkan teman – teman buka bagian BAB 6.

Untuk membuat *view* tersebut dimana teman – teman perlu membuat *folder* baru pada direktori *application/view*. Catatan dimana teman – teman sesuai nama *folder* tersebut seperti pada *file controllers*, sebagai contoh

dimana pada *file controllers* kita terdapat *code* untuk melakukan *load view* berikut, `$this->load->view('store/...');`, dimana kita akan membuat *folder* dengan nama *store*. Pada *folder store* tersebut dimana kita akan membuat beberapa *file PHP* baru.

Langkah awal dalam membuat *view* tersebut dimana teman – teman perlu membuat *view inventory* untuk menampilkan data – data yang ada pada *table* di dalam *database* teman – teman. Silahkan teman – teman berkreasi seperti apa *view* yang teman – teman inginkan. Dimana penulis akan membuat *file view inventory* yang sederhana saja. Buat *file PHP* baru dengan nama *inventory.php* pada *folder store*. Penulis akan menampung item – item pada *table* tersebut ke dalam suatu tabel dengan menggunakan *source code* berikut.

```
<!-- Begin Page Content -->
<div class="container-fluid">

    <!-- Page Heading -->
    <h1 class="h3 mb-4 text-gray-800">
        <?= $title; ?>
    </h1>

    <button class="btn btn-sm btn-primary mb-3"
        data-toggle="modal" data-target="#inputdata">
        <i class="fas fa-plus fa-sm"></i> Tambah Produk
    </button>

    <table class="table">

        <thead>
            <tr align="center">
                <th scope="col">#</th>
                <th scope="col">Nama Produk</th>
                <th scope="col">Informasi Singkat
                </th>
                <th scope="col">Jenis Produk</th>
```

```
<th scope="col">Harga</th>
<th scope="col">Stok</th>
<th scope="col" colspan="3">Aksi
</th>
</tr>
</thead>

<?php
$no = 1;
foreach ($barang as $brg) : ?>

<tbody>
<tr align="center">
<th scope="row"><?php echo $no ++
?></th>

<td><?php echo $brg->nama_brg
?></td>

<td><?php echo $brg->
keterangan ?></td>

<td><?php echo $brg->kategori
?></td>

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

<td><?php echo $brg->stok
?></td>

<td><?= anchor('store/
detail_barang/' . $brg->
id_brg, '<div class=
"btn btn-success btn-sm">
<i class="fas fa-search-plus">
</i></div>') ?>
</td>

<td><?= anchor('store/edit_brg
/' . $brg->id_brg, '<div class=
```

```
        "btn btn-primary btn-sm">
        <i class="fa fa-edit"></i>
        </div>') ?>
    </td>

    <td><?= anchor('store/delete/' .
        $brg->id_brg, ' <div class=
        "btn btn-danger btn-sm">
        <i class="fa fa-trash">
        </i></div>') ?>
    </td>
    </tr>
</tbody>

<?php endforeach; ?>

</table>

</div>
<!-- /.container-fluid -->

</div>
<!-- End of Main Content -->
```

Source Code 5.17 Membuat View Inventory

Pada *source code* 5.17 tersebut dimana merupakan isi dari *file inventory*. Silahkan teman – teman coba jalankan terlebih dahulu, ingat karena sistem kita belum melakukan *hosting* yang berarti teman – teman membutuhkan *software Xampp* untuk menjalankan sistem kita. Lakukan *start* pada *module apache* dan *MySQL* pada *software Xampp* tersebut. Buka *web browser* teman – teman dan panggil *controller* nya dengan cara *http://localhost/nama_folder_ci/nama_controllers/nama_function*. Hasil dari *file view* tersebut akan nampak seperti pada gambar 5.28 berikut.

#	Nama Produk	Informasi Singkat	Jenis Produk	Harga	Stok	Aksi
1	Batagor Kering	Menerima Pesanan Catering	Makanan Dan Minuman	12000	10	

Gambar 5.28 Hasil View Inventory

Dapat dilihat pada gambar 5.28 tabel yang kita buat telah menampung isi dari *database*, maka dari itu fungsi *read* pada CRUD sudah berhasil kita buat.

Selanjutnya kita akan membuat *form input* pada *view* kita. Dimana kita akan menambahkan fungsi *modals* untuk *form input* tersebut, maka dari itu teman – teman dapat menambahkan *source code* 5.18 pada *file inventory* kita. Masukkan di bagian paling bawah saja, perhatikan *source code* berikut.

```
<!-- Modal -->
<div class="modal fade" id="inputdata" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">
                    FORM INPUT PRODUK
                </h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
        </div>
        <div class="modal-body">
```

```
<form action="= base_url() . 'store/inputbarang'; ?>
?>" method="post" enctype="multipart/form-data">

<label>Email</label>
<div class="form-group row">
  <div class="col-sm-12">
    <input type="text" name="email" class="form-control" value="= $user['email']; ?" readonly>
  </div>
</div>

<label>Nama Produk</label>
<div class="form-group row">
  <div class="col-sm-12">
    <input type="text" name="nama_brg" class="form-control">
  </div>
</div>

<label>Informasi Singkat</label>
<div class="form-group row">
  <div class="col-sm-12">
    <input type="text" name="keterangan" class="form-control">
  </div>
</div>

<label>Alamat Toko</label>
<div class="form-group row">
  <div class="col-sm-12">
    <textarea name="alamat" id="alamat" class="form-control"></textarea>
  </div>
</div>

<div class='form-group row'>
  <div class="col-sm-6">
    <small id="emailHelp" class="form-text text-muted"><b>*Nama Kecamatan.</b></small>
  </div>
</div>
```

```
<input type="text" name="kecamatan"
       class="form-control">
</div>

<div class="col-sm-6">
  <small id="emailHelp" class="form-text
  text-muted"><b>*Nama Kelurahan.</b></small>
  <input type="text" name="kelurahan"
         class="form-control">
</div>
</div>

<div class='form-group row'>
<div class="col-sm-6">
  <small id="emailHelp" class="form-text
  text-muted"><b>*Nama Kota/Kabupaten.
  </b></small>
  <input type="text" name="kota"
         class="form-control">
</div>

<div class="col-sm-6">
  <small id="emailHelp" class="form-text
  text-muted"><b>*Kode Pos.</b></small>
  <input type="text" name="kode_pos"
         class="form-control">
</div>
</div>

<label>Detail Produk</label>
<div class="form-group row">
  <div class="col-sm-12">
    <textarea name="detail" class="form-control">
    </textarea>
  </div>
</div>

<label>Jenis Produk</label>
<div class="form-group row">
  <div class="col-sm-12">
    <select class="form-control" name="kategori">
```

```
<option>Makanan Dan Minuman</option>
<option>Elektronik</option>
<option>Sport</option>
<option>Fashion Pria</option>
<option>Fashion Wanita</option>
<option>Fashion Anak - Anak</option>
</select>
</div>
</div>

<label>Harga</label>
<div class="form-group row">
  <div class="col-sm-12">
    <input type="text" name="harga"
      class="form-control">
  </div>
</div>

<label>Stok</label>
<div class="form-group row">
  <div class="col-sm-12">
    <input type="text" name="stok"
      class="form-control">
  </div>
</div>

<label>No. Telpon</label>
<div class="form-group row">
  <div class="col-sm-12">
    <input type="text" name="tlpn"
      class="form-control" value="62">
  </div>
</div>

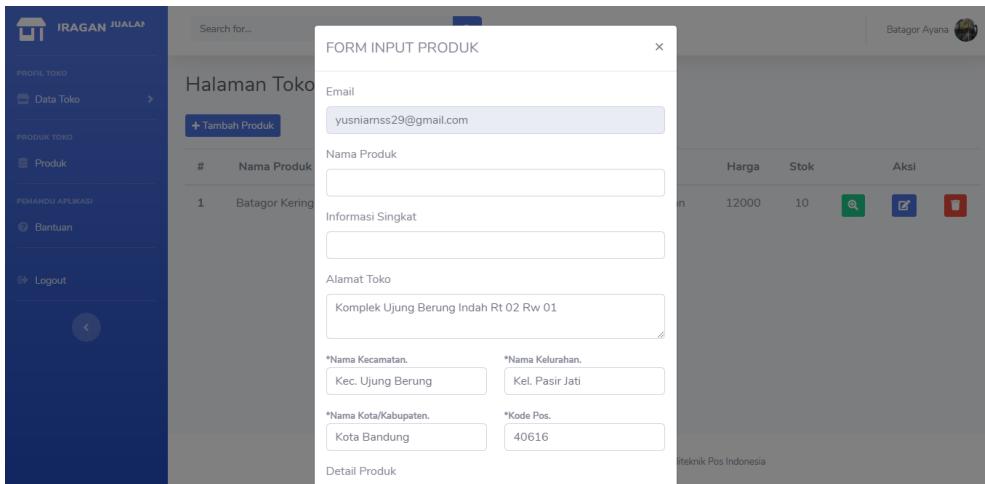
<label>Gambar Produk</label><br>
<div class="form-group row">
  <div class="col-sm-12">
    <input type="file" name="gambar"
      class="form-control">
  </div>
</div>
```

```
</div>
<div class="modal-footer">
    <button type="button" class="btn
    btn-secondary" data-dismiss="modal">Kembali
    </button>
    <button type="submit" class="btn
    btn-primary">Simpan Produk
    </button>
</div>

</form>
</div>
</div>
</div>
```

Source Code 5.18 Halaman Input Produk Dengan Fungsi Modals

Pada *source code* 5.18 tersebut merupakan tahap pembuatan *form input* barang dengan menggunakan fungsi *modals*. Dimana teman – teman akan menambahkan *source code* 5.18 tersebut di baris paling bawah pada *file inventory.php*. Coba sekarang teman – teman aplikasikan *button tambah produk* pada halaman *inventory* tersebut maka akan muncul sebuah *form* yang keluar seperti *pop up* pada sistem dan coba seperti yang ditunjukkan pada gambar 5.29 berikut.



Gambar 5.29 Halaman Input Barang

Langkah selanjutnya dimana teman – teman perlu membuat halaman *edit* yang berfungsi untuk melakukan proses *edit* tersebut. Untuk membuat halaman *edit* tersebut dimana teman – teman perlu membuat *file* baru pada *folder store*, silahkan teman – teman buat dan beri nama *edit_brg.php*, lalu masukkan *source code* berikut.

```
<div class="container-fluid">
    <h3><i class="fas fa-edit"></i>
        <?= $title; ?>
    </h3>

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

    <form method="post" action="<?= base_url() . 
        'store/update' ?>">

        <label>Nama Produk</label>
        <div class="form-group row">
            <div class="col-sm-12">
                <input type="text" name="nama_brg"
                    class="form-control" value="<?= $brg->
                    nama_brg ?>">
            </div>
```

```
</div>

<label>Informasi Singkat</label>
<div class="form-group row">
<div class="col-sm-12">
<input type="text" name="keterangan"
class="form-control" value="<?= $brg->
keterangan ?">
</div>
</div>

<label>Alamat Toko</label>
<div class="form-group row">
<div class="col-sm-12">
<textarea name="alamat" id="alamat"
class="form-control"><?= $brg->alamat ?>
</textarea>
</div>
</div>

<div class='form-group row'>
<div class="col-sm-6">
<small id="emailHelp" class="form-text
text-muted"><b>*Nama Kecamatan.</b></small>
<input type="text" name="kecamatan"
class="form-control" value="<?= $brg->
kecamatan ?">
</div>

<div class="col-sm-6">
<small id="emailHelp" class="form-text
text-muted"><b>*Nama Kelurahan.</b></small>
<input type="text" name="kelurahan"
class="form-control" value="<?= $brg->
kelurahan ?">
</div>
</div>

<div class='form-group row'>
<div class="col-sm-6">
<small id="emailHelp" class="form-text
```

```
text-muted"><b>*Nama Kota/Kabupaten.</b>
</small>
<input type="text" name="kota"
class="form-control" value="= $brg-&gt;kota
??"&gt;
&lt;/div&gt;
&lt;div class="col-sm-6"&gt;
&lt;small id="emailHelp" class="form-text
text-muted"&gt;&lt;b&gt;*Kode Pos.&lt;/b&gt;&lt;/small&gt;
&lt;input type="text" name="kode_pos"
class="form-control" value="<?= $brg-&gt;
kode_pos ??"&gt;
&lt;/div&gt;
&lt;/div&gt;

&lt;label&gt;Detail Produk&lt;/label&gt;
&lt;div class="form-group row"&gt;
&lt;div class="col-sm-12"&gt;
&lt;textarea name="detail" class="form-control"&gt;
<?= $brg-&gt;detail ??"&lt;/textarea&gt;
&lt;/div&gt;
&lt;/div&gt;

&lt;label&gt;Harga&lt;/label&gt;
&lt;div class="form-group row"&gt;
&lt;div class="col-sm-12"&gt;
&lt;input type="text" name="harga"
class="form-control" value="<?= $brg-&gt;harga
??"&gt;
&lt;/div&gt;
&lt;/div&gt;

&lt;label&gt;Stok&lt;/label&gt;
&lt;div class="form-group row"&gt;
&lt;div class="col-sm-12"&gt;
&lt;input type="text" name="stok"
class="form-control" value="<?= $brg-&gt;stok
??"&gt;
&lt;/div&gt;
&lt;/div&gt;</pre
```

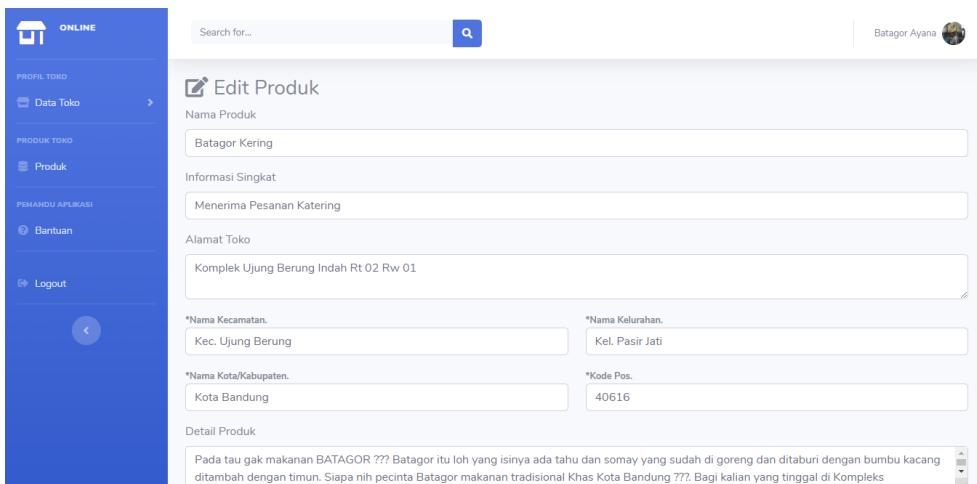
```
<label>No. Telpon</label>
<div class="form-group row">
    <div class="col-sm-12">
        <input type="text" name="tlpn"
            class="form-control" value=<?= $brg->tlpn
            ?>">
    </div>
</div>

<button type="submit" class="btn btn-primary
    btn-sm mt-3">Simpan Produk</button>
<?= anchor('store/inventory/', '<div
    class="btn btn-sm mt-3 btn-warning">
    Kembali</div>') ?>

</form>
<?php endforeach; ?>
</div>
```

Source Code 5.19 Membuat View Edit Pada Folder Store

Pada *source code* 5.19 tersebut merupakan *code full* dari *view edit* dimana sekilas sama dengan *code input* produk. Akan tetapi pada *view edit* tersebut akan memanggil isi pada *table* yang terdapat pada *database* kita dengan menggunakan fungsi *value* di setiap *inputan*. Silahkan teman – teman coba halaman *edit* tersebut dengan mengklik *button* dengan *icon edit* tersebut atau yang terletak pada bagian tengah kolom aksi, maka teman – teman akan akan di bawa ke dalam halaman *edit_brg* tersebut, seperti yang ditunjukkan pada gambar 5.30 berikut.



Gambar 5.30 Halaman Edit Barang

Silahkan teman – teman perhatikan fungsi *models* yang sudah kita buat sebelumnya, dimana kita telah merapkan fungsi *get_where* yang dimana untuk mengambil data berdasarkan *id*, hal tersebut dapat teman – teman terapkan pada bagian *detail* barang. Yuk kita buat satu *view* lagi pada *folder store* kita, silah teman – teman buat *file* baru dan simpan dengan nama *detail_barang.php* lalu isikan *source code* berikut.

```
<div class="container-fluid">
<div class="card">
    <div class="card-header">
        <?= $title; ?>
    </div>
<div class="card-body">

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

    <div class="row">
        <div class="col-md-4">
            
        </div>
        <div class="col-md-8">
```

```
<table class="table">
<tr>
    <td>Nama Produk</td>
    <td><strong><?= $brg->nama_brg ?>
    </strong>
    </td>
</tr>

<tr>
    <td>Detail Produk</td>
    <td><strong><?= $brg->detail ?></strong>
    </td>
</tr>

<tr>
    <td>Jenis Produk</td>
    <td><strong><?= $brg->kategori ?>
    </strong>
    </td>
</tr>

<tr>
    <td>Stok</td>
    <td><strong><?= $brg->stok ?>
    </strong>
    </td>
</tr>

<tr>
    <td>Kontak Penjual</td>
    <td><strong>+<?= $brg->tlpn ?>
        <a href="https://api.whatsapp.com/
            send?phone=<?= $brg->tlpn ?>
            &text=Selamat%20datang%20ada
            %20yang%20bisa%20kami%20bantu%20?,">
        <button type="submit" name="button"
            class="btn-sm btn-success">
            <i class="fab fa-fw fa-whatsapp"></i>
        </button></a></strong>
    </td>
</tr>
```

```

<tr>
    <td>Harga</td>
    <td><strong><div class="btn btn-sm
        btn-success">Rp. <?= number_format
        ($brg->harga, 0, ',', '.') ?></div>
    </strong></td>
</tr>

</table>
<?= anchor('store/edit_brg/' . $brg->
    id_brg, '<div class="btn btn-sm
    btn-primary">Edit Produk</div>') ?>
<?= anchor('store/inventory/', '<div class="'
    btn btn-sm btn-warning">
    Kembali</div>') ?>
</div>

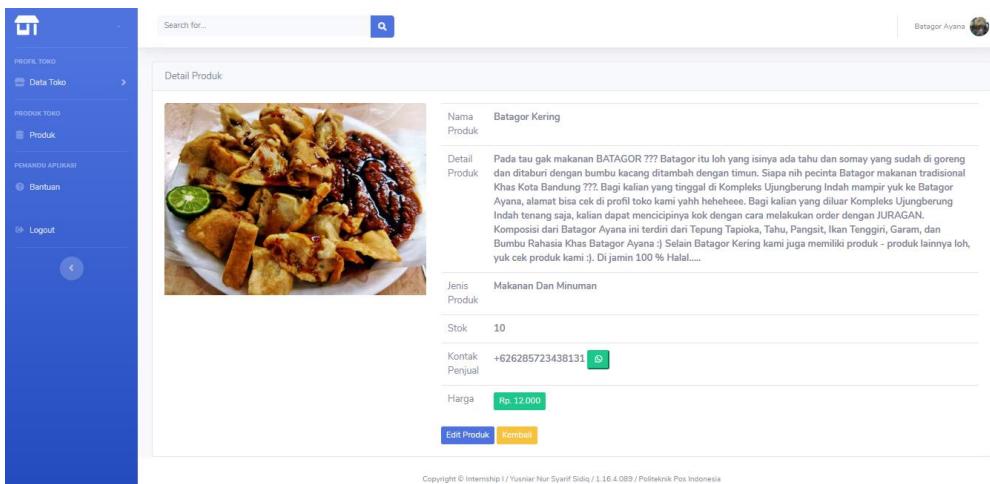
<?php endforeach; ?>
</div>
</div>

</div>
</div>

```

Source Code 5.20 View Detail Barang Pada Folder Store

Pada *source code* 5.20 tersebut dimana teman – teman akan membuat *view* halaman detail barang dengan memanggil *id* yang diberikan. Untuk mengakses halaman tersebut dimana teman – teman dapat mengklik *button* dengan bentuk *icon* detail pada kolom aksi. Sehingga hasil dari halaman detail barang tersebut akan terlihat seperti pada gambar 5.31 berikut.



Gambar 5.31 Halaman Detail Barang

Untuk *view delete* dimana teman – teman dapat melakukan klik pada *button* yang bernbentuk *icon delete* yang terdapat pada halaman *inventory*. Coba teman – teman klik apabila berhasil maka data tersebut akan terhapus dan sistem akan melakukan fungsi *redirect* ke halaman *inventory*. Dimana proses pembuatan sistem CRUD dengan *framework codeigniter* telah selesai.



BAB 6

E – COMMERCE DAN CODEIGNITER

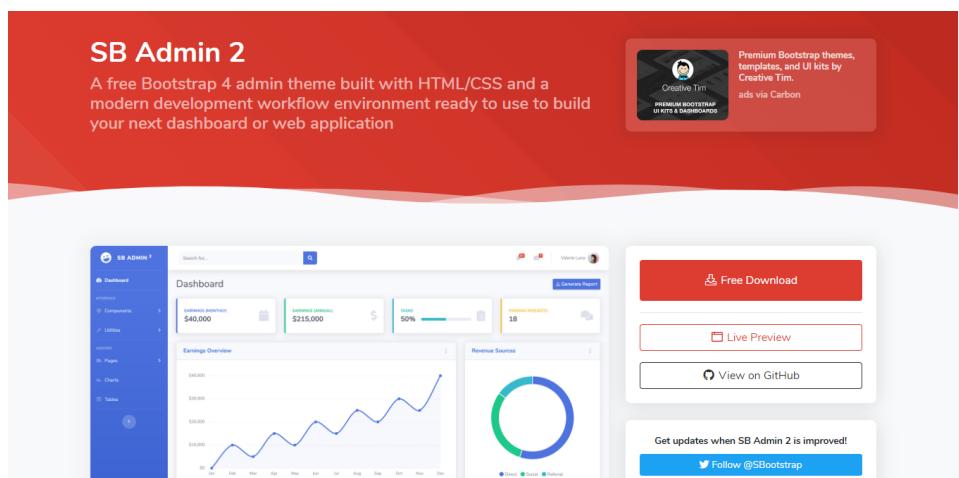
Seperti yang sudah kita bahas pada bab pendahuluan dimana *e-commerce* sangatlah penting di dunia RI 4.0 ini. Banyak pengusaha yang sudah menerapkan konsep *e-commerce* guna memperbesar daya saing mereka dalam dunia bisnis. *E-Commerce* sangatlah membantu baik bagi pelanggan maupun bagi penjual atau 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 Boostrap Dengan CI

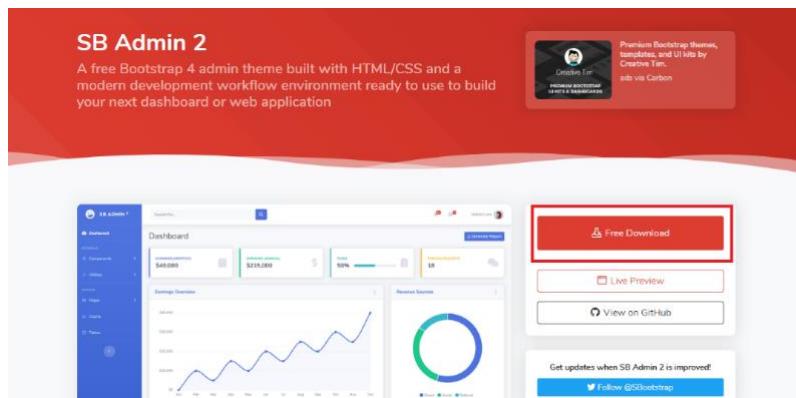
Bootstrap merupakan kerangka atau *library framework* CSS yang khusus dibuat untuk pengembangan *front end* pada sebuah *website*. CSS merupakan sebuah *template* pada sebuah *website*, yang dimana dengan adanya CSS dapat memperindah tampilan dari *website* yang akan kita bangun. Penampilan merupakan kunci utama dalam membangun sebuah sistem dengan konsep *e-commerce*. Apabila suatu sistem tersebut memiliki tampilan yang begitu menarik dan mudah digunakan, dimana *user* akan merasa nyaman dalam menggunakan sistem tersebut. *Bootstrap* merupakan solusi yang tepat bagi teman – teman yang tida mau ribet dalam mempercantik tampilan sistem kita, dimana *bootstrap* akan mempersingkat pekerjaan proses desain sistem kita.

Bagaimana cara menghubungkan *bootstrap* tersebut ke dalam *framework codeigniter* kita ?. Teman – teman tidak usah khawatir karena dalam sub bab kali kita akan belajar mengenai menghubungkan *templates bootstrap* kedalam *framework codeigniter*. Langkah awal dalam pembelajaran kali ini dimana teman – teman perlu memiliki satu *template bootstrap*. Pada sistem JURAGAN dimana telah menggunakan *templates bootstrap* SB Admin 2. Teman – teman dapat *mendownload template* yang lain namun sebagai contoh saya akan menggunakan *templates* SB Admin 2 tersebut. Untuk melakukan *download* pada *template* SB Admin 2 tersebut teman – teman dapat mengunjungi *link* berikut, <https://startbootstrap.com/themes/sb-admin-2/> yang dimana teman – teman akan di bawa ke suatu halaman yang ditunjukkan pada gambar 6.1 tersebut.



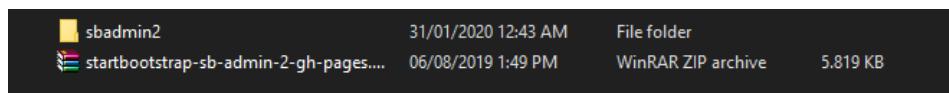
Gambar 6.1 Halaman Start Bootstrap SB ADMIN 2

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



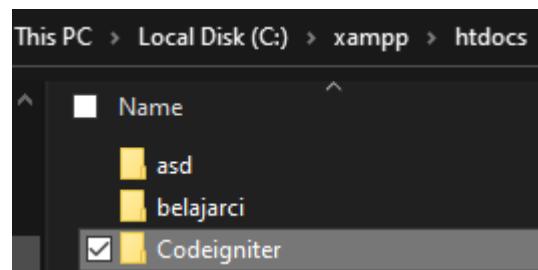
Gambar 6.2 Menu Download Template Bootstrap SB ADMIN 2

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



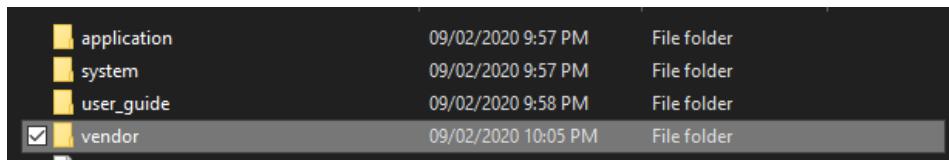
Gambar 6.3 Meng Extract File Zip Bootstrap

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 tersimpan pada direktori *xampp/htdocs*.



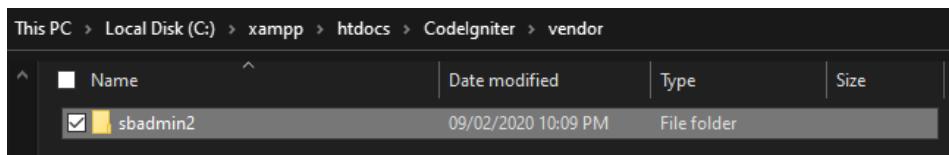
Gambar 6.4 File Codeigniter Pada Direktori Xampp/htdocs

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



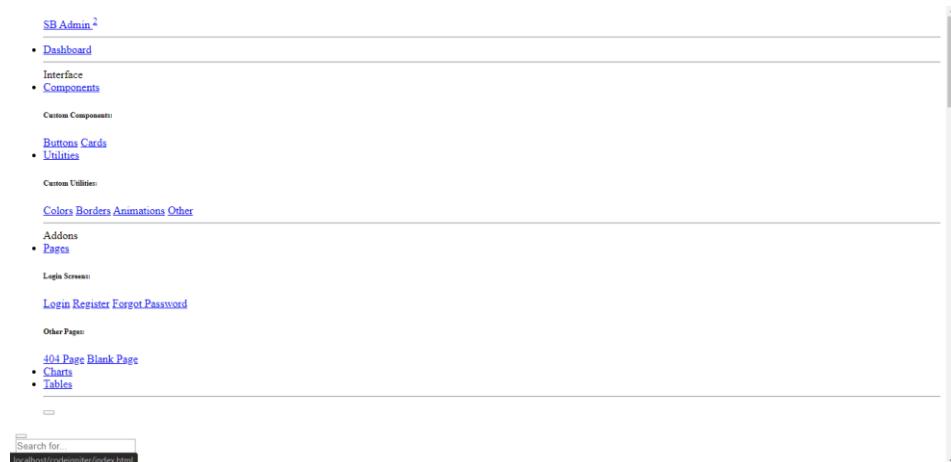
Gambar 6.5 Membuat Folder Vendor

Folder *vendor* tersebut berfungsi untuk menampung referensi *template bootstrap* SB ADMIN 2 tersebut, silahkan teman – teman pendahkan *file* SB ADMIN 2 tersebut kedalah *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.

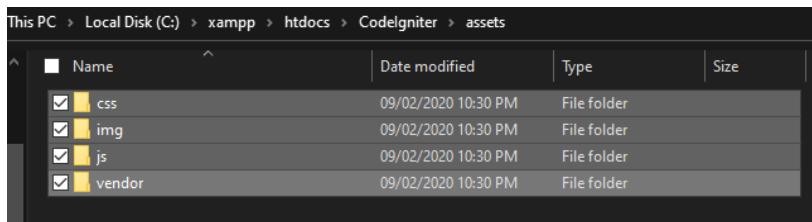
Langkah selanjutnya dimana teman – teman perlu membuat *folder* baru pada *file codeigniter* dan beri nama menjadi *assets*, seperti pada gambar 6.8 berikut.

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

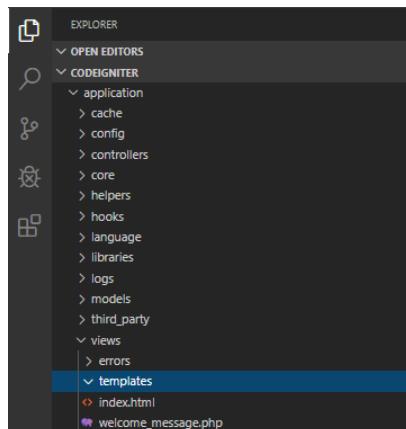
Folder *assets* berfungsi untuk menampung beberapa fungsi dari *templates bootstrap* SB ADMIN 2 tersebut, seperti css, js, dan jquery. Silahkan teman –

teman pindahkan *folder css, js, img* dan *vendor* yang terdapat pada *file SB ADMIN 2* tersebut kedalam *folder assets*, seperti pada gambar 6.9.



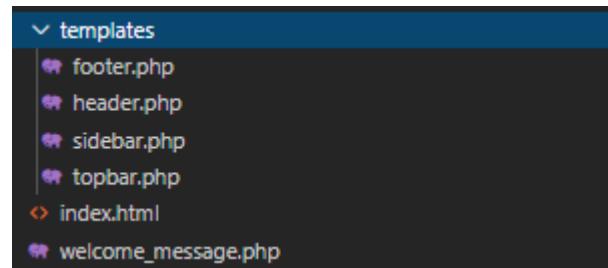
Gambar 6.9 Mengisi Folder Assets

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 *title* 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 bagai *header* kedalam *file header.php*.

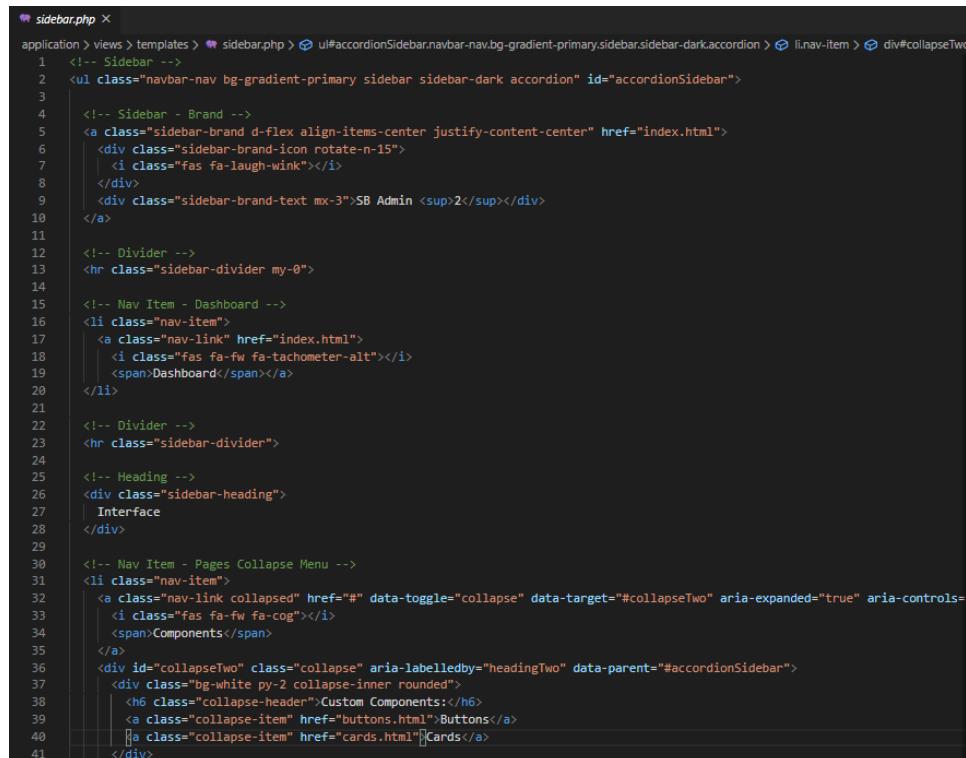
```

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 di isi 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.php 
application > views > templates > sidebar.php > ul#accordionSidebar.navbar.nav.bg-gradient-primary.sidebar.sidebar-dark.accordion > li.nav-item > div#collapseTwo
1  <!-- Sidebar -->
2  <ul class="navbar-nav bg-gradient-primary sidebar sidebar-dark accordion" id="accordionSidebar">
3
4  <!-- Sidebar - Brand -->
5  <a class="sidebar-brand d-flex align-items-center justify-content-center" href="index.html">
6    <div class="sidebar-brand-icon rotate-n-15">
7      <i class="fas fa-laugh-wink"></i>
8    </div>
9    <div class="sidebar-brand-text mx-3">SB Admin <sup>2</sup></div>
10   </a>
11
12  <!-- Divider -->
13  <hr class="sidebar-divider my-0">
14
15  <!-- Nav Item - Dashboard -->
16  <li class="nav-item">
17    <a class="nav-link" href="index.html">
18      <i class="fas fa-fw fa-tachometer-alt"></i>
19      <span>Dashboard</span>
20    </a>
21
22  <!-- Divider -->
23  <hr class="sidebar-divider">
24
25  <!-- Heading -->
26  <div class="sidebar-heading">
27    Interface
28  </div>
29
30  <!-- Nav Item - Pages Collapse Menu -->
31  <li class="nav-item">
32    <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseTwo" aria-expanded="true" aria-controls="collapseTwo">
33      <i class="fas fa-fw fa-cog"></i>
34      <span>Components</span>
35    </a>
36    <div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-parent="#accordionSidebar">
37      <div class="bg-white py-2 collapse-inner rounded">
38        <h6 class="collapse-header">Custom Components:</h6>
39        <a class="collapse-item" href="buttons.html">Buttons</a>
40        <a class="collapse-item" href="cards.html">Cards</a>
41      </div>

```

Gambar 6.13 Isi File Sidebar.php

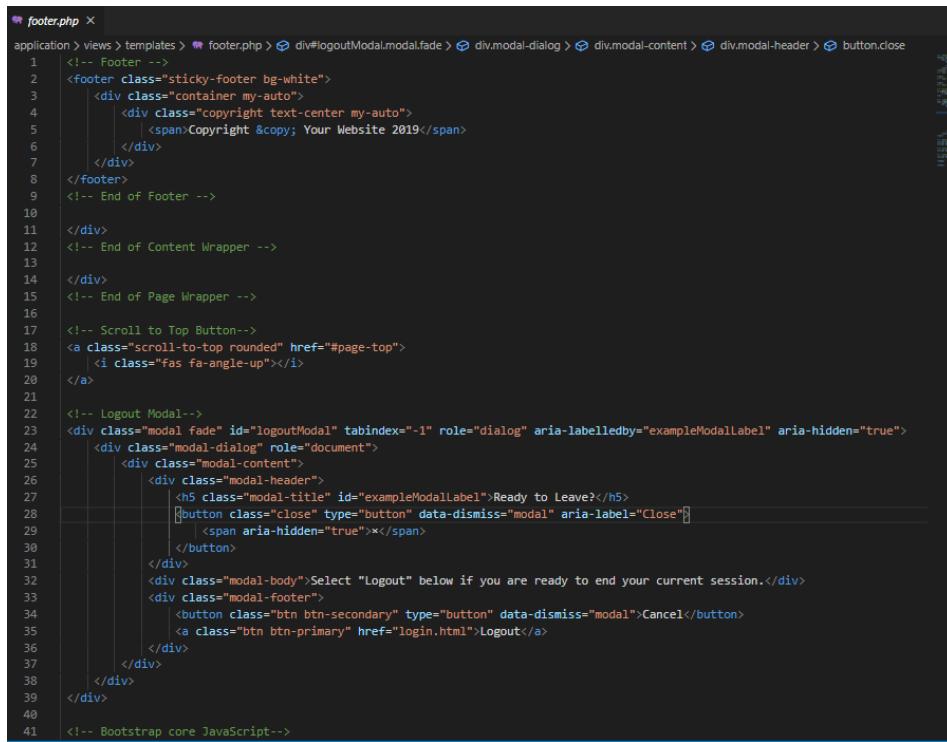
Topbar merupakan bagian ketiga pada *template* tersebut dimana biasanya terletak pada posisi paling atas dari sebuah sistem dan berikan fungsi – fungsi seperti *form* pecarian salah satu contohnya. Silahkan teman – teman *cut code* baris 142 – 329 pada file *welcome_message.php* yang merupakan bagian *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-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" aria-describedby="basic-addon2">
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 <!-- Scroll to Top Button-->
15 <a class="scroll-to-top rounded" href="#page-top">
16     | <i class="fas fa-angle-up"></i>
17 </a>
18 <!-- Logout Modal-->
19 <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModallLabel" aria-hidden="true">
20     <div class="modal-dialog" role="document">
21         <div class="modal-content">
22             <div class="modal-header">
23                 | <h5 class="modal-title" id="exampleModallLabel">Ready to Leave?</h5>
24                 | <button class="close" type="button" data-dismiss="modal" aria-label="Close">
25                     | <span aria-hidden="true">×</span>
26             </div>
27             <div class="modal-body">Select "Logout" below if you are ready to end your current session.</div>
28             <div class="modal-footer">
29                 | <button class="btn btn-secondary" type="button" data-dismiss="modal">Cancel</button>
30                 | <a class="btn btn-primary" href="login.html">Logout</a>
31             </div>
32         </div>
33     </div>
34 </div>
35 <!-- Bootstrap core JavaScript-->
36
37
38
39
40
41 <!-- 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">

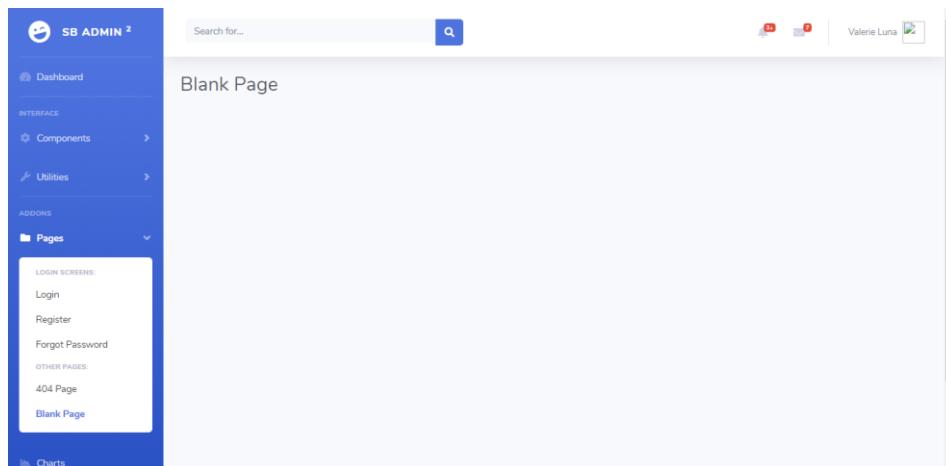
<!-- Custom styles for this template-->
<link href="<?= base_url(); ?>assets/css/
      sb-admin-2.min.css" rel="stylesheet">
</head>
```

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

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

Source Code 6.2 Merubah File header.php Pada Folder Templates

Pada *source code* 6.2 tersebut dimana teman – teman akan 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 kembali *file controllers welcome* pada *web browser* teman – teman dan lihat hasilnya, jika tidak terjadi masalah maka halaman yang ditampilkan akan terlihat seperti ada gamabar 6.16 berikut.



Gambar 6.16 Hasil Yang Diberikan Source Code 6.2

Bagaimana, sudah ada perubahan bukan ?, dengan menambahkan fungsi *helper* tersebut teman – teman telah memanggil *template css* dan fungsi vendor pada *folder assets* yang sudah terisi fungsi – fungsi pada *templates bootstrap* tersebut.

Langkah selanjutnya dimana teman – teman perlu menambahkan fungsi *helper* tersebut terhadap *java script* pada *bootstrap* tersebut supaya *java script* dapat digunakan. Untuk menemukan bagian *java script* tersebut silahkan teman – teman buka *file footer.php* pada *folder templates*. *Scroll* kebawah hingga menumakan *file java script/jquery*, silahkan rubah *file footer.php* tersebut menjadi seperti *source code* 6.3 berikut.

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

<!-- Core plugin JavaScript-->
<script src="<% base_url(); %>assets/vendor/
jquery-easing/jquery.easing.min.js"></script>

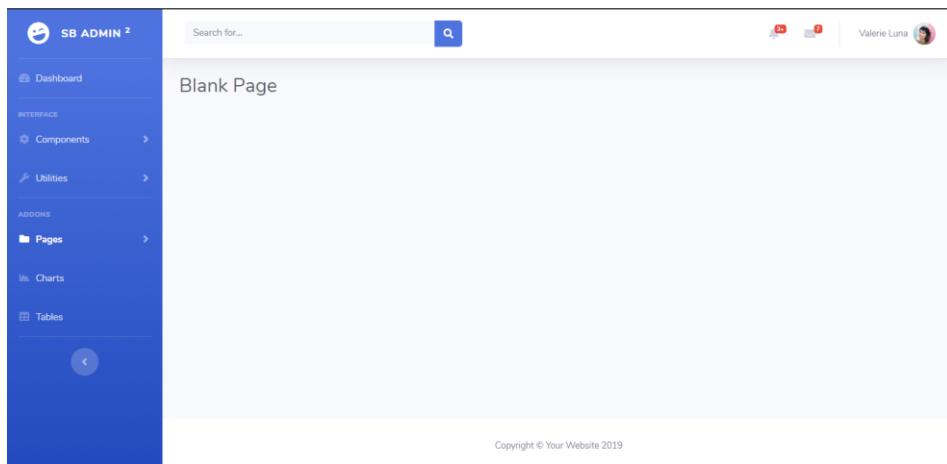
<!-- Custom scripts for all pages-->
<script src="<% base_url(); %>assets/js/
sb-admin-2.min.js"></script>

</body>

</html>
```

Source Code 6.3 Merubah File footer.php Pada Folder Templates

Silahkan teman – teman jalankan maka fungsi *java script* pada *templates bootstrap* teman – teman sudah aktif dan dapat digunkan. Perhatikan juga bagian bawah sistem dimana terdapat perubahan yang signifikan akan terlihat yaitu munculnya bagian *footer* seperti yang ditampilkan pada gambar 6.17 berikut.



Gambar 6.17 Hasil Dari Source Code 6.3

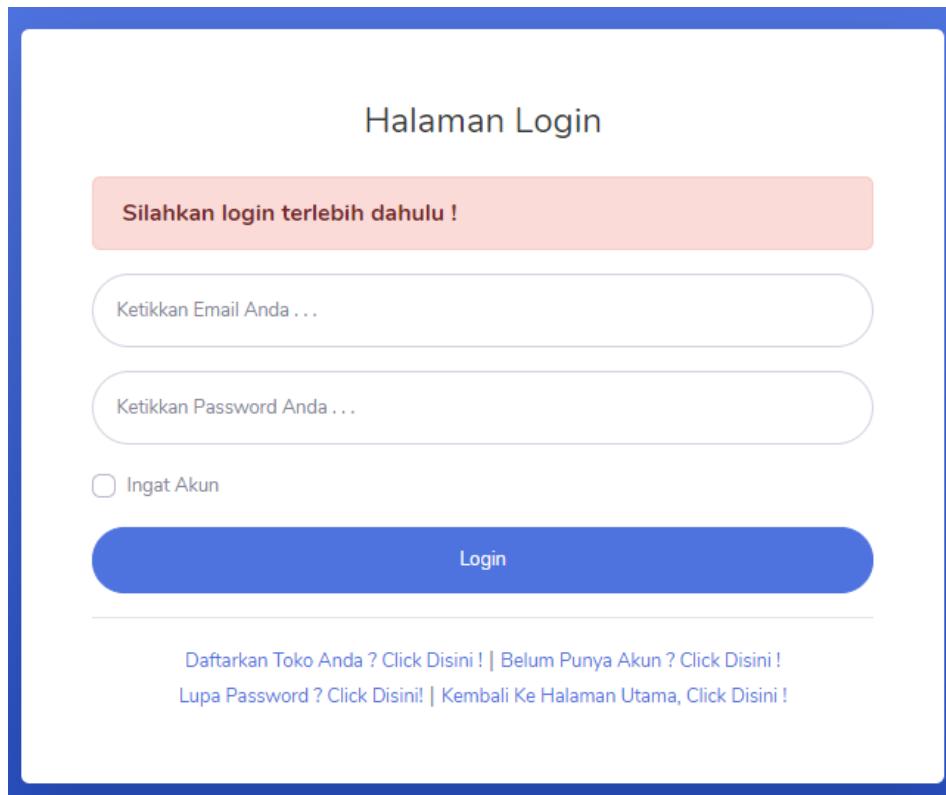
Dengan begini dimana *templates bootstrap* SB Admin 2 yang kita miliki sudah siap digunakan. Teman – teman dapat merubah bagian *sidebar* pada tersebut dengan cara melakukan pengeditan pada file *sidebar.php* pada folder *templates*.

6.2 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 di bangun. 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*.



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. Disni saya akan memberikan contoh 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

Perhatikan *source code* 6.4 tersebut, dimana pada sistem JURAGAN terdapat *file controllers* dengan nama *Auth*. *Controllers Auth* tersebut berfungsi sebagai pengontrol halaman *login* dan daftar akun pada sistem JURAGAN. Pada *controllers Auth* tersebut dimana penulis akan memanggil *library form_validation* tersebut ke dalam *controllers construct* dengan melakukan *load library* tersebut.

Dalam pembelajaran kali ini dimana teman – teman membutuhkan sebuah *form* daftar akun untuk mempraktekan fungsi *form validation* tersebut. Dalam sistem JURAGAN dimana sudah terdapat halaman daftar akun, penulis akan menggunakan *form* tersebut untuk mempraktekan fungsi *form validation*.

Perlu teman – teman ketahui pada umumnya dimana dalam membuat fungsi *form validation* pada halaman seperti daftar akun tersebut dimana kita akan membuat sebuah aturan atau *rules* terhadap halaman tersebut. Untuk membuat *rules* tersebut teman – teman dapat membuatnya pada *file controllers*. Perhatikan *source code* berikut.

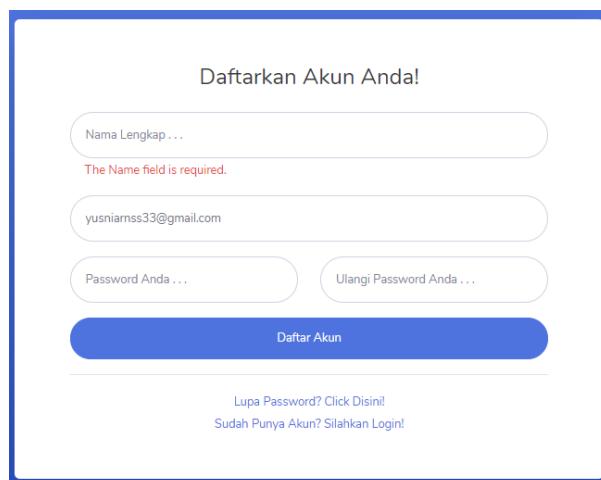
```
public function registrasi()
{
    $data['title'] = 'Halaman Registrasi';
    $this->form_validation->
        set_rules('name', 'Name', 'required|trim');
}
```

Source Code 6.5 Membuat Rules Pada Kolom Nama

Pada *source code* 6.5 tersebut dimana penulis akan membuat sebuah *rules* terhadap kolom *name* pada *function registrasi*. Untuk membuat *rules* dimana teman – teman dapat memanggil *library form_validation* lalu berikan fungsi *set_rules* seperti yang terlihat pada *source code* 6.5 tersebut. Pada kolom *name* dimana penulis telah memberikan *rules* berupa :

- **Required** dimana pada kolom *name* wajib di isi dan tidak boleh kosong.
- **Trim.**

Apabila *rules* tersebut diterapkan maka fungsi *form validation* tersebut akan terlihat seperti pada gambar 6.19 sebagai berikut.

*Gambar 6.19 Praktek Form Validation Pada Field Name*

Teman – teman dapat melihat pada gambar 6.9 tersebut merupakan sebuah *form* daftar akun pada sistem JURAGAN. Selain memiliki kolom nama dimana halaman daftar akun tersebut terdapat kolom *email*, penulis akan memberikan fungsi *form validation* terhadap kolom tersebut. Caranya sama saja dimana teman – teman perlu memanggil *library form validation* tersebut dan memberikan fungsi *set_rules*, yang membedakan disini adalah *rules* yang diberikan pada kolom *email* tersebut. Perhatikan *source code* berikut.

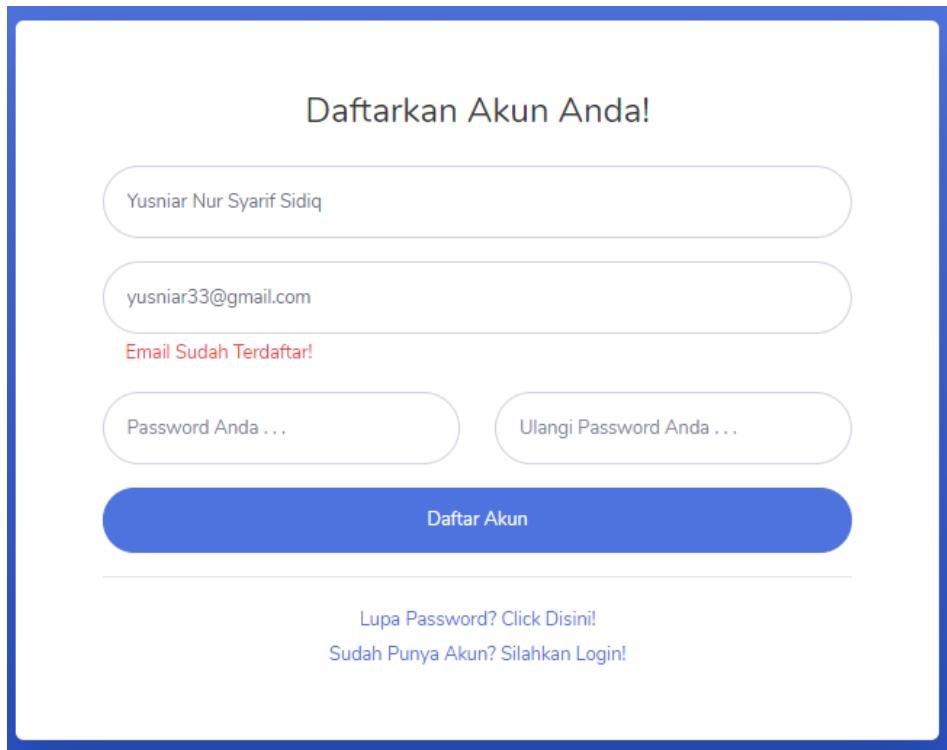
```
$this->form_validation->  
    set_rules('email', 'Email', 'required|trim|  
    valid_email|is_unique[user.email]',  
    [  
        'is_unique' => 'Email Sudah Terdaftar!'  
    ]);
```

Source Code 6.6 Menambahkan Rules Pada Kolom *email*

Dimana *rules* yang diberikan sama dengan kolom *name*, namun pada kolom *email* terdapat dua *rules* tambahan yaitu,

- *Valid_email* dimana *email* yang diberikan merupakan format yang valid atau format *email* yang sebenar – benarnya.
- *Is_unique* dimana *email* harus bersifat *unique*, artinya *email* yang sudah terdaftar tidak bisa dipergunakan kembali.

Apabila *rules* tersebut diterapkan maka *form validation* yang muncul akan terlihat seperti pada gambar 6.20 berikut.



Gambar 6.20 Praktek Form Validation Terhadap Field Email

Selain *name* dan *email* dimana pada halaman daftar akun sistem JURAGAN tersebut memiliki kolom *password*. Penulis akan menambahkan beberapa *rules* terhadap kolom *password* tersebut. Perhatikan *source code* berikut.

```
$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 Membuat Rules Pada Kolom Password

Pada *source code* 6.7 tersebut dimana penulis telah menambahkan *rules* pada kolom *password* tersebut. Dimana *rules* yang diberikan terlihat sama dengan yang ada pada kolom *name*, namun terdapat *rules* baru yaitu,

- **Min_length[6]** dimana panjang minimal dari *password* tersebut berupa 6 karakter.
- **Matches** dimana antara isi pada kolom *password* 1 dan 2 harus sama.

Apabila *rules* tersebut diterapkan maka *form validation* yang akan tampil adalah seperti pada gambar 6.21 berikut.

The screenshot shows a registration form titled "Daftarkan Akun Anda!". It has three input fields: a first name field containing "Yusniar Nur Syarif Sidiq", an email field containing "yusniar29@gmail.com", and two password fields. The left password field contains "Password Anda ..." and the right password field contains "Ulangi Password Anda ...". Below the fields, a red error message "Password Terlalu Pendek!" is displayed. At the bottom is a large blue "Daftar Akun" button. Below the button are links for password recovery ("Lupa Password? Click Disini!") and account login ("Sudah Punya Akun? Silahkan Login!").

Gambar 6.21 Praktek Form Validation Pada Field Password

Setelah teman – teman telah selesai menerapkan *rules – rules* di setiap kolom tersebut dimana teman – teman perlu menjalankan *library form validation* tersebut dengan cara memberikan fungsi *run* seperti berikut.

```

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 Code Full Function Registrasi JURAGAN

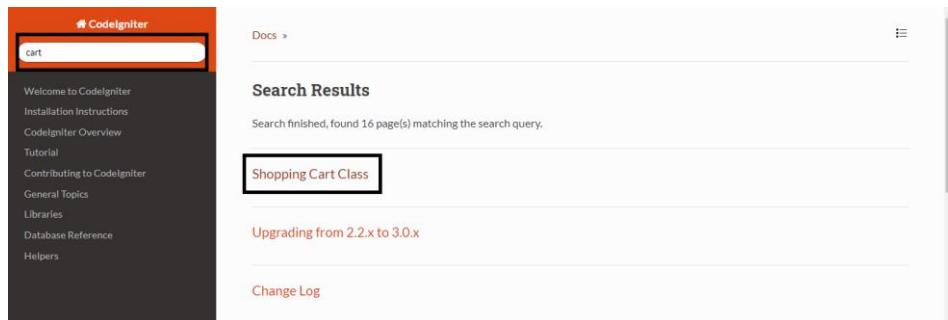
Pada *source code* 6.8 tersebut merupakan *code full* dari *function registrasi* pada sistem JURAGAN. Dimana pembahasan mengenai *form validation* sudah selesai. Yuk kita bahas fungsi – fungsi lainnya.

6.3 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);

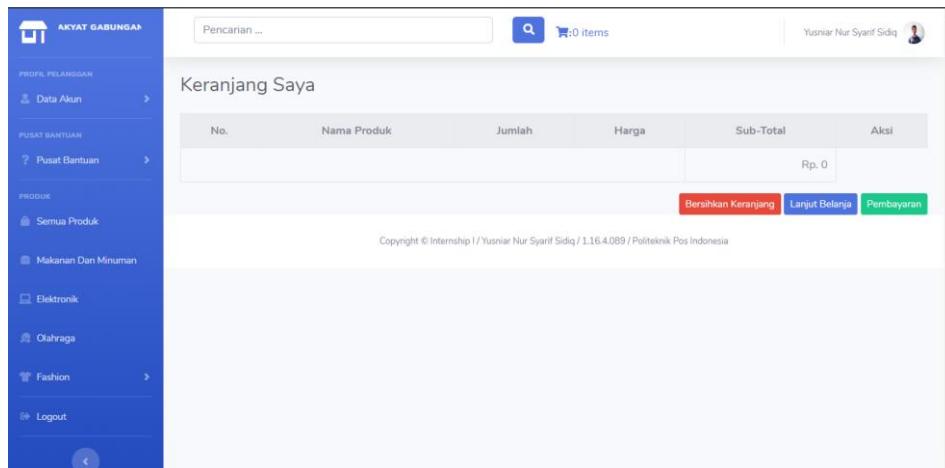
        return $ambilData->row();
    }
}
```

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.

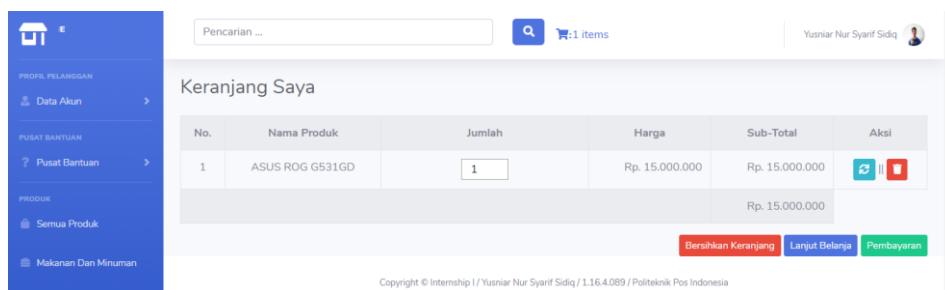
Selamat teman – teman sudah berhasil membuat fungsi *add* pada *library shopping cart*. Langkah selanjutnya dimana teman – teman membutuhkan *view* untuk melakukan uji coba terhadap fungsi – fungsi *library shopping cart* tersebut. Silahkan teman – teman buat *view* halaman keranjang belanja sesuai keinginan teman – teman. Namun untuk menampilkan *item* yang ditambahkan dimana teman – teman – 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 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>') ?>
```

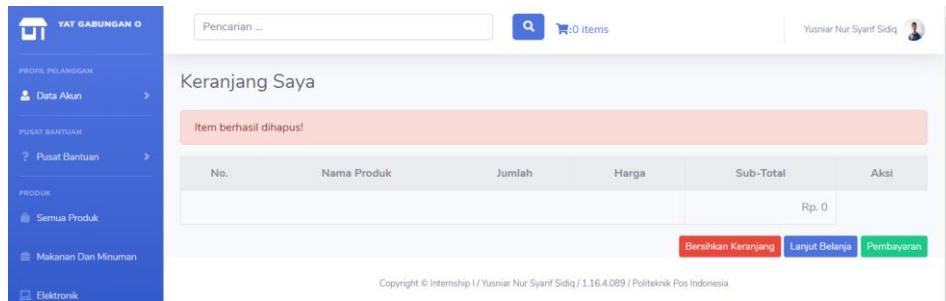
Source Code 6.14 Memanggil Function Delete Pada Button Delete

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.



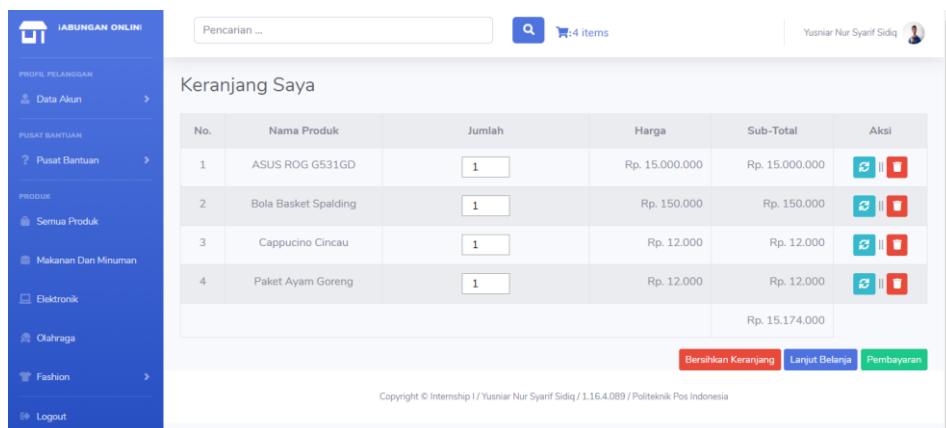
Gambar 6.25 Rowid Delete

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 ?.



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;</code
```

```
</div>
```

*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 – taman 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

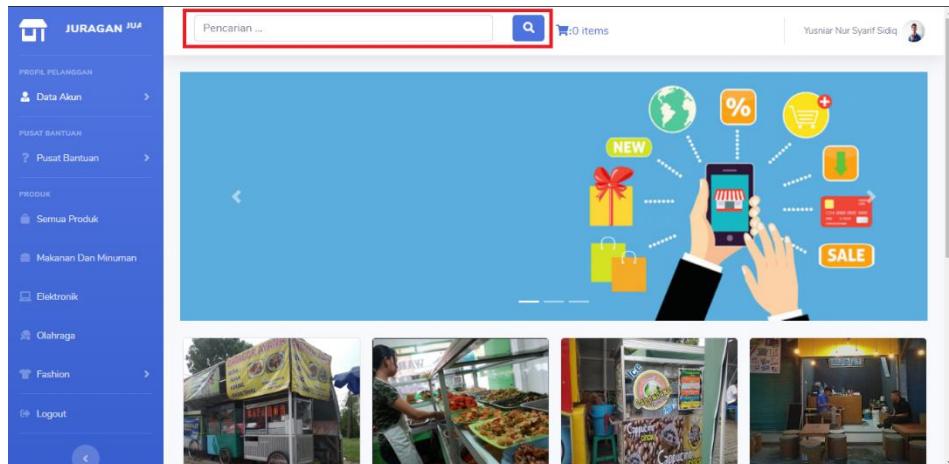
6.4 Membuat Fitur Pencarian

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 *error*, 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 akan tetapi ketika di praktikan *view* yang diberikan tidak akan berubah. Mengapa demikian ?, hal ini dikarenakan *form* pencarian tersebut membutuhkan sebuah *view* tersendiri untuk menampilkan data – data sesuai *keyword* yang diberikan. Silahkan teman – teman berkreasi dalam membuat halaman pencarian tersebut dengan menerapkan pembelajaran – pembelajaran sebelumnya.

Apabila teman – teman sudah membuat *view* halaman pencarian tersebut silahkan teman – teman jalankan kembali *fitur* pencarian tersebut. Contoh dimana penulis akan mencari makanan dengan *keyword* “Batagor”. Maka saat dilakukan pencarian dimana hanya *record* yang memiliki kemeridan dengan batagor yang akan ditampilkan, seperti pada gambar 6.29.



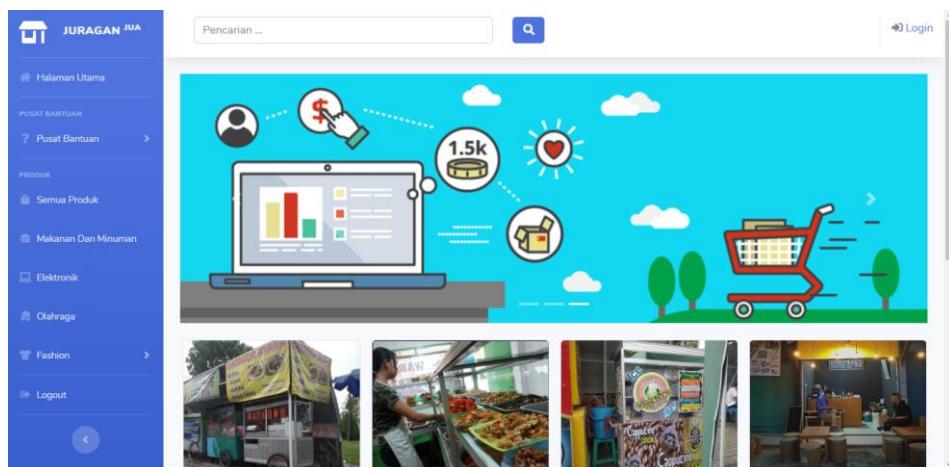
Gambar 6.29 Halaman Pencarian Pada Sistem JURAGAN

BAB 7

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

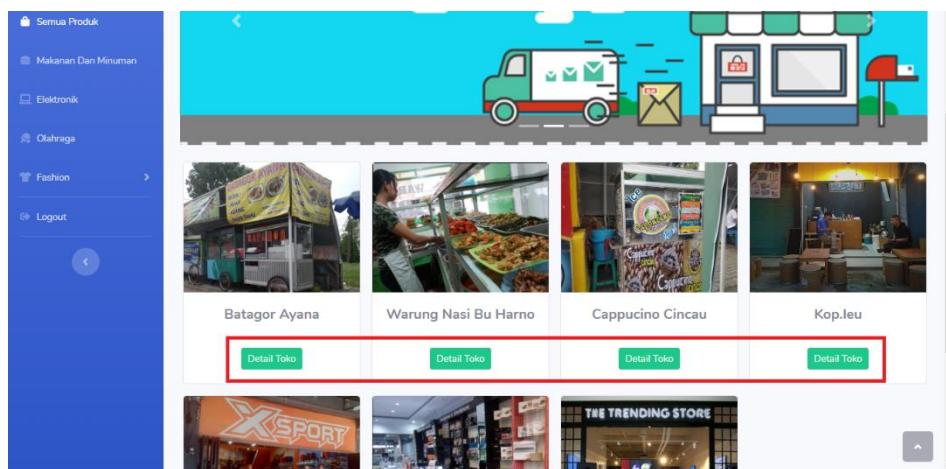


Gambar 7.1 Halaman Awal Sistem JURAGAN

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

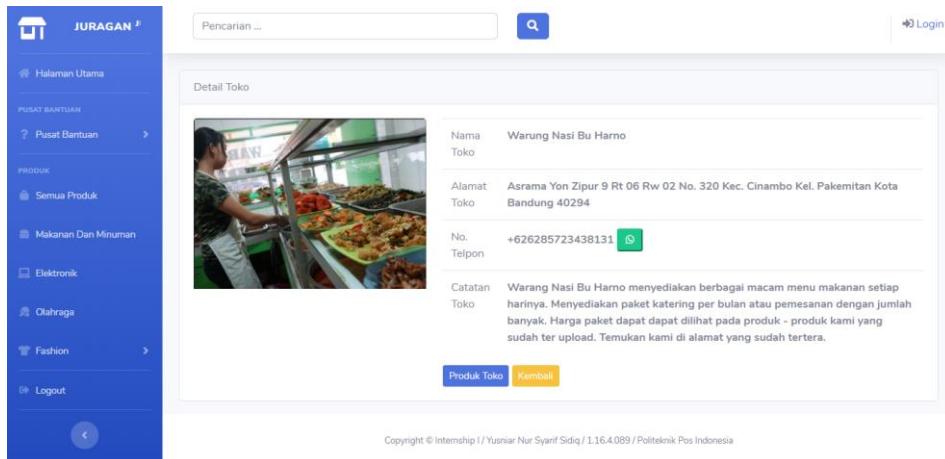
Pada halaman awal tersebut dimana kondisi *user* belum melakukan *login* sehingga *user* tidak dapat melakukan proses transaksi. Namun *user* dapat melihat produk – produk apa saja yang sedang ditawarkan pada sistem JURAGAN tersebut dan informasi – informasi yang diberikan baik informasi produk maupun 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.



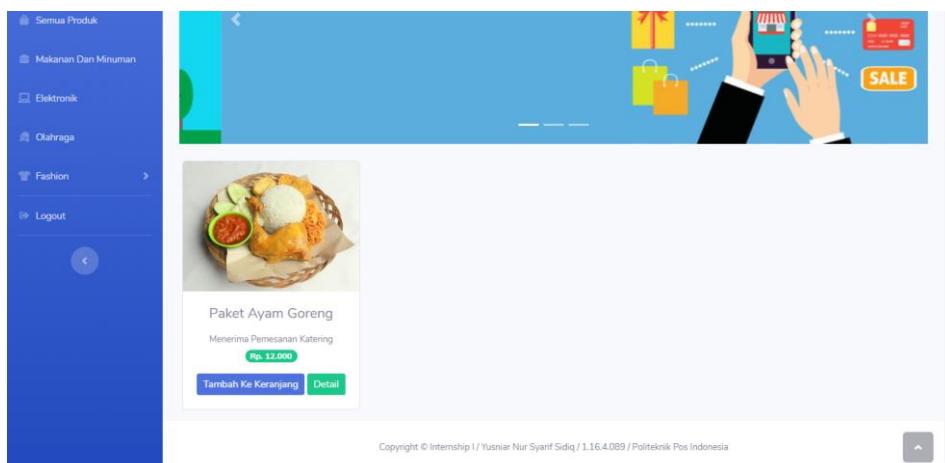
Gambar 7.2 Mengaplikasikan Button Detail Toko Pada Halaman Awal

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.



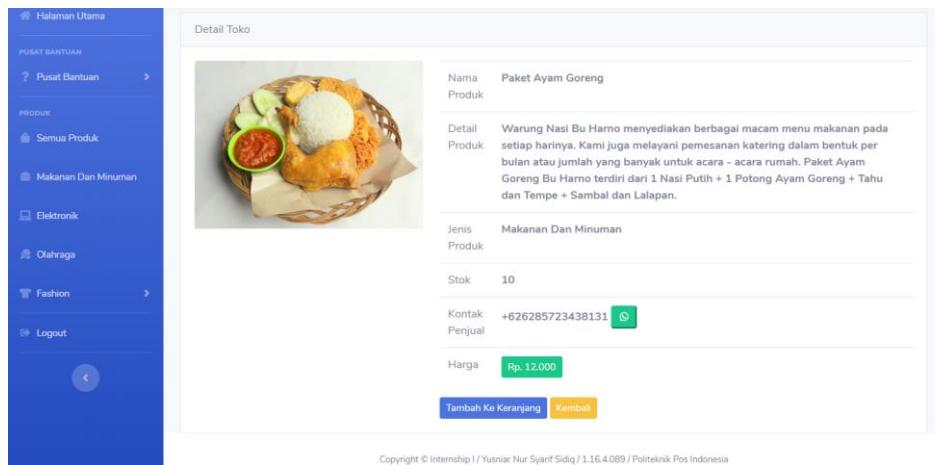
Gambar 7.3 Halaman Detail Toko Pada Sistem JURAGAN

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

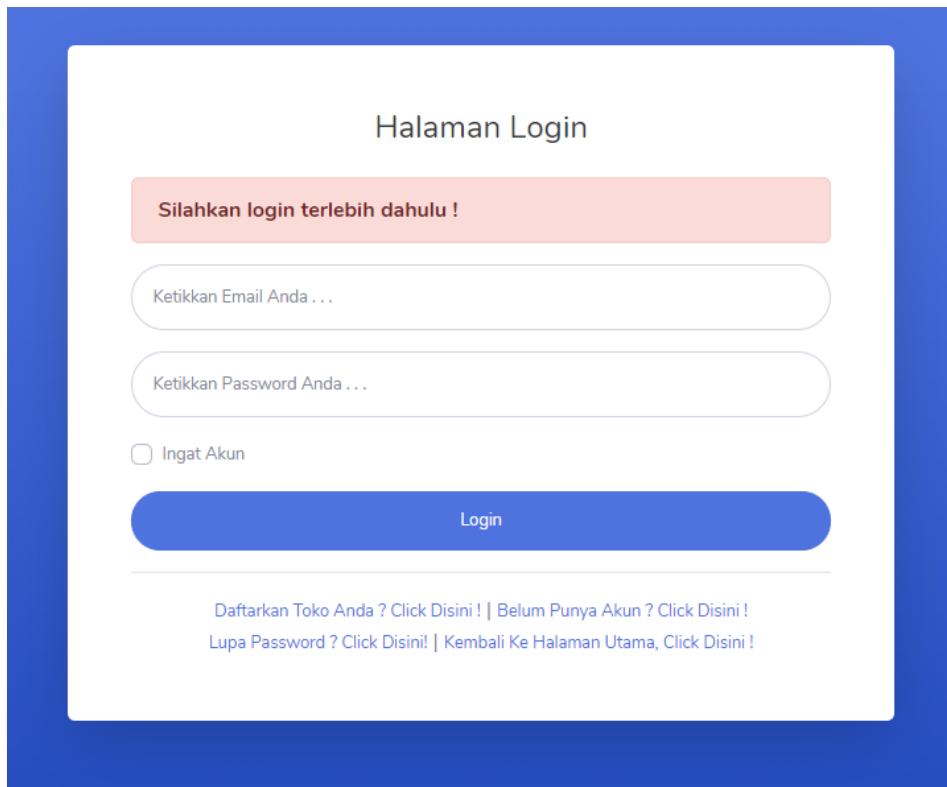
Silahkan teman – teman perhatikan pada halaman produk toko tersebut dimana saat di akses hanya menampilkan produk dari toko tersebut. Pada gambar 7.4 dimana kita telah memilih toko “**Warung Nasi Bu Harno**” sehingga yang di tampilkan adalah produk dari toko tersebut. Pada halaman tersebut dimana terdapat 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 mengembalikkan

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.



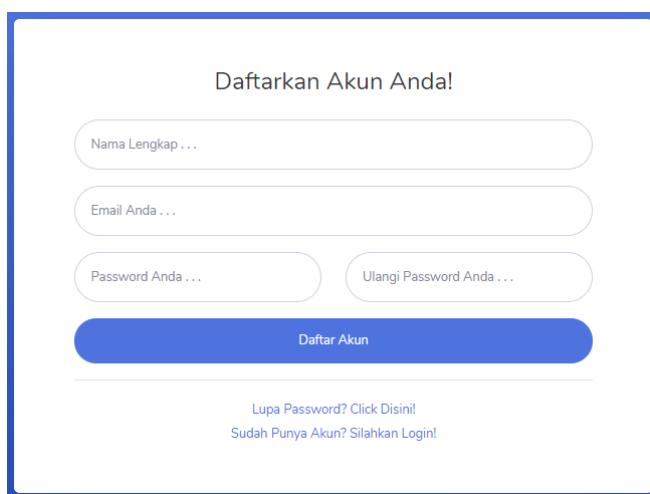
Gambar 7.6 Peringatan Wajib Login Terlebih Dahulu

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.



Gambar 7.7 Halaman Daftar Akun Pelanggan Pada Sistem JURAGAN

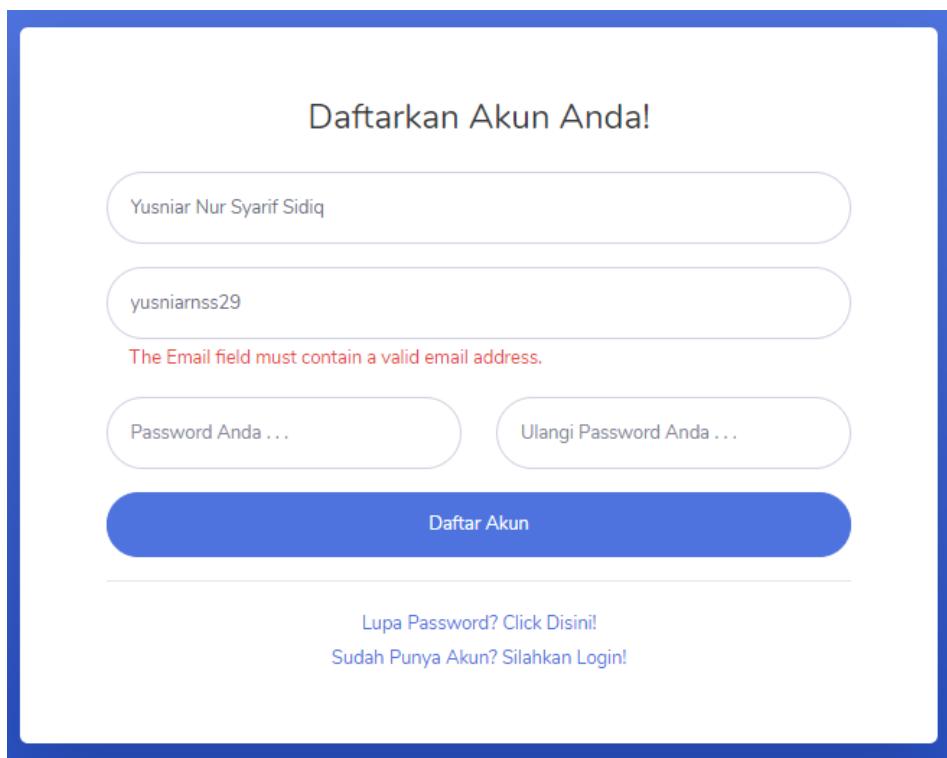
Perhatikan gambar 7.7 tersebut, dimana merupakan halaman pendaftaran akun bagi *user* dengan level pelanggan pada sistem JURAGAN. Untuk melakukan proses daftar akun tersebut dimana *user* perlu memberikan beberapa informasi data diri yang di butuhkan oleh sistem JURAGAN tersebut. Dimana *user* perlu 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 with the following fields and errors:

- Nama Lengkap ...**: Below the input field, the error message "The Name field is required." is displayed in red.
- Email Anda ...**: Below the input field, the error message "The Email field is required." is displayed in red.
- Password Anda ...** and **Ulangi Password Anda ...**: Both input fields have the error message "The Password field is required." displayed below them in red.
- Daftar Akun**: A large blue button at the bottom of the form.
- Lupa Password? Click Disini!** and **Sudah Punya Akun? Silahkan Login!**: Links at the bottom of the form.

Gambar 7.8 Peringatan Kolom Tidak Boleh Kosong

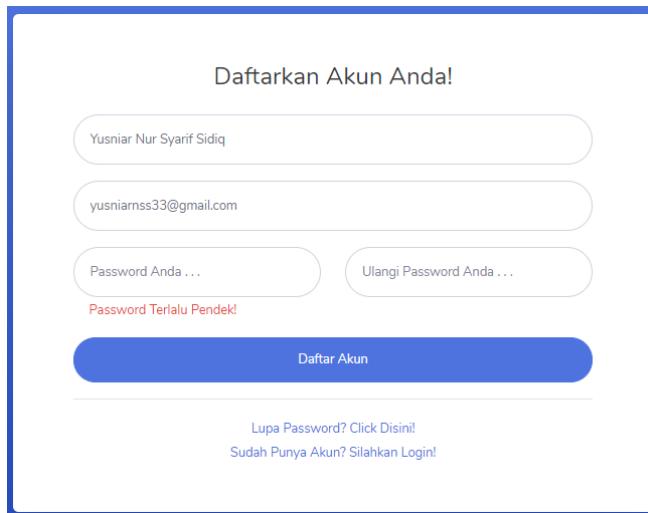
Peraturan selanjutnya dimana *format email* yang diberikan harus benar – benar sesuai dan tidak asal. Hal ini sangatlah penting, karena pada saat *user* selesai mendaftar akun maka akun tersebut belum aktif, dan untuk mengaktifkan akun tersebut dimana *user* perlu melakukan proses aktivasi melalui *email*.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



Gambar 7.9 Notifikasi Format Email Tidak Sesuai

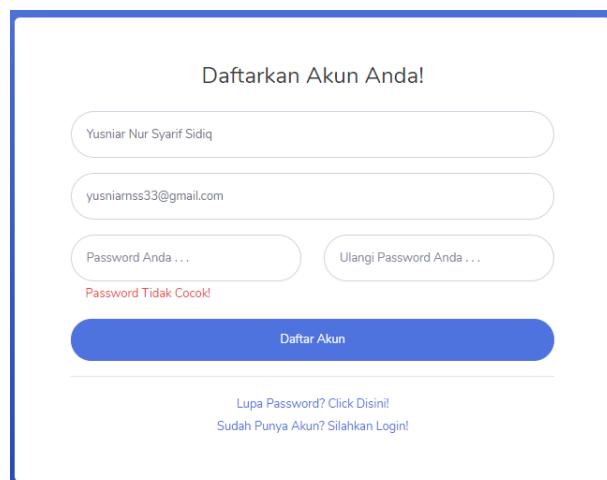
Peraturan berikutnya dimana terdapat pada bagian kolom *password*. Pada sistem JURAGAN *password* dapat dibuat dalam bentuk *text*, *angka*, dan campuran antara *text* dan angka, namun perlu di perhatikan bahwa *password* minimal berjumlah 6 huruf atau angka apabila kurang maka sistem akan memberikan notifikasi bahwa *password* terlalu pendek seperti pada gambar 7.10.

178 | BAB VII Panduang Menggunakan Sistem JURAGAN



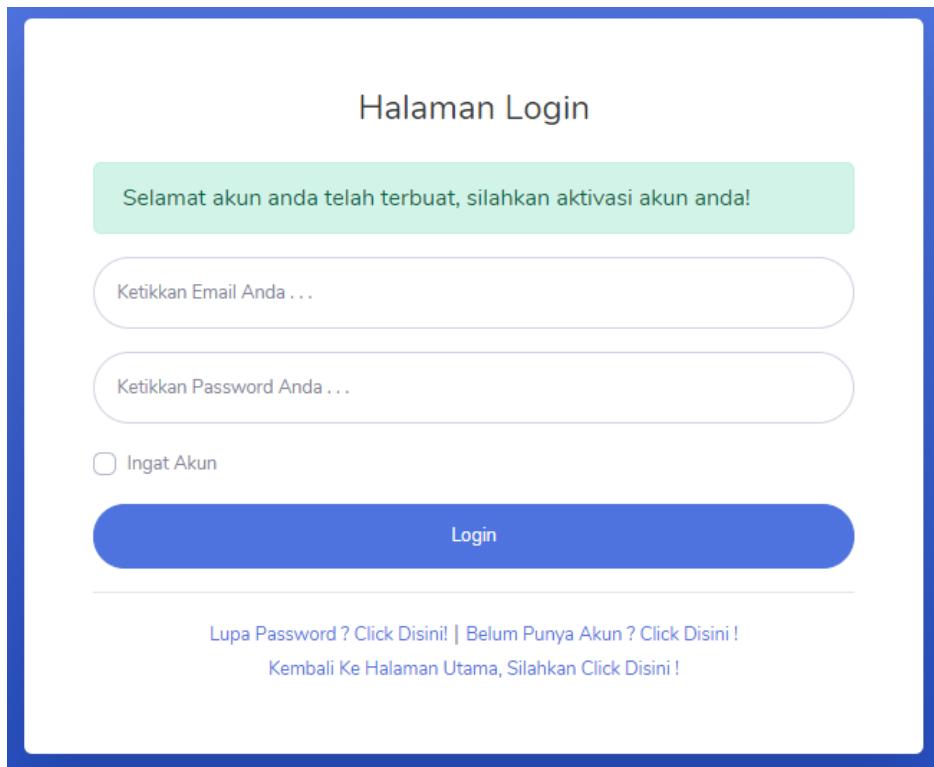
Gambar 7.10 Notifikasi Password Terlalu Pendek

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.



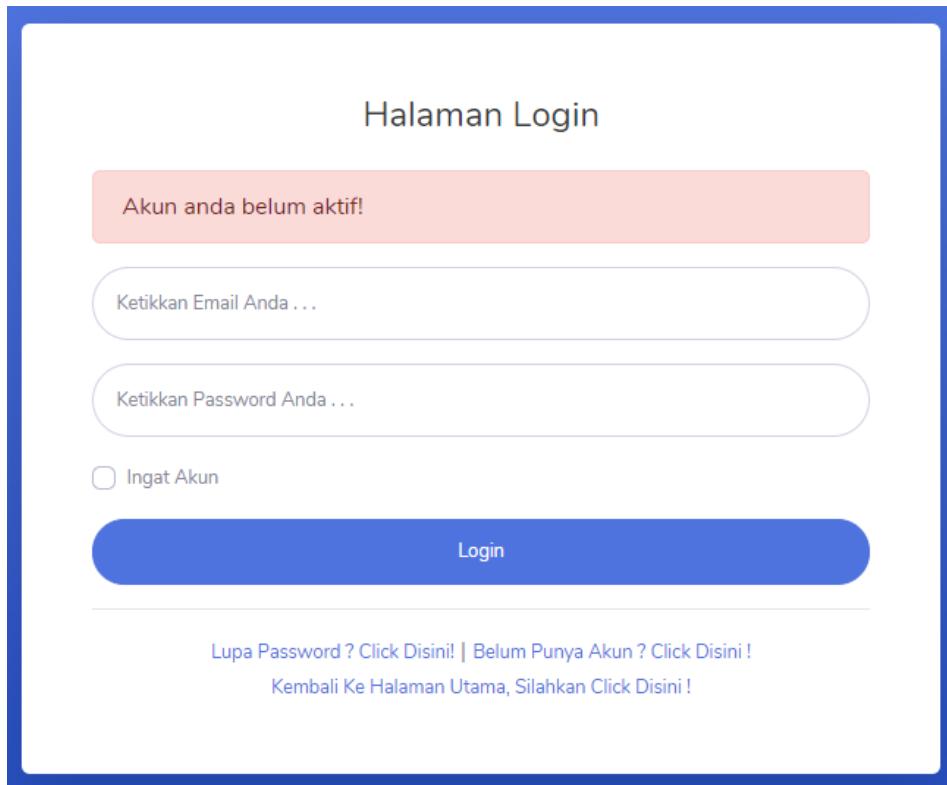
Gambar 7.11 Notifikasi Password Tidak Cocok

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.



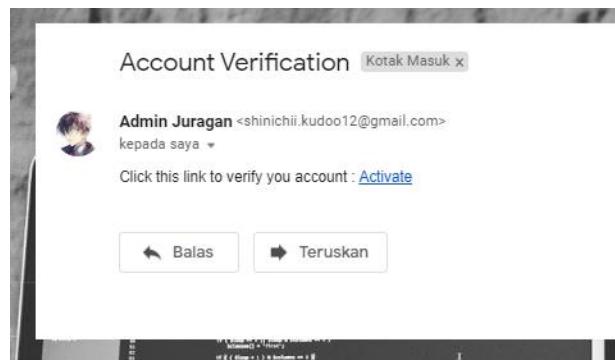
Gambar 7.12 Akun User Berhasil Terdaftar

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



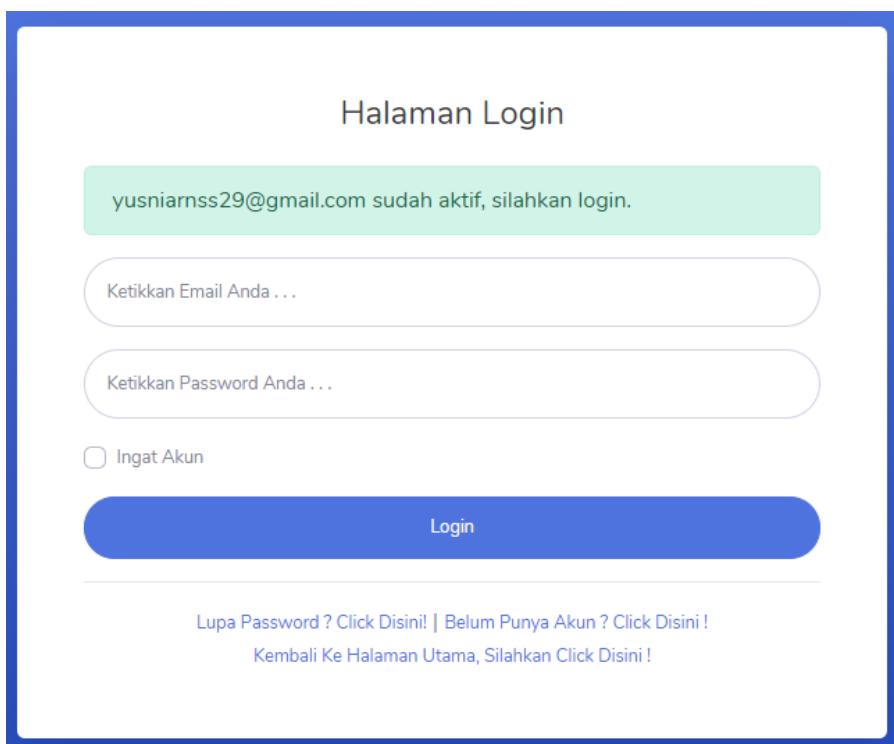
Gambar 7.13 Notifikasi Akun Belum Aktif

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.



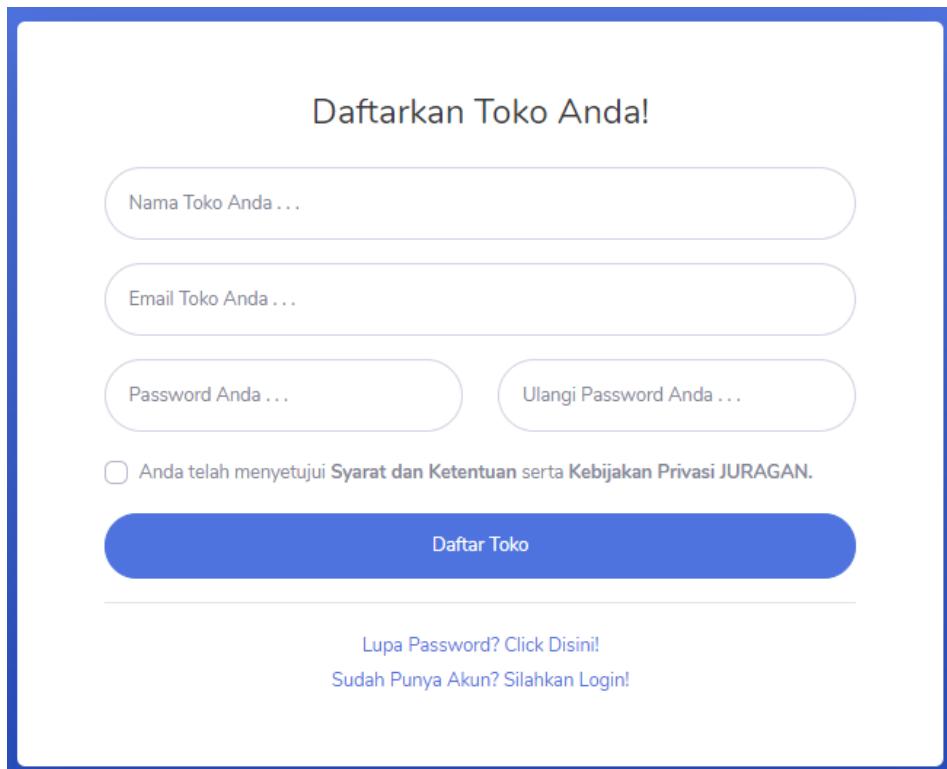
Gambar 7.14 Aktivasi Akun Melalui Email

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



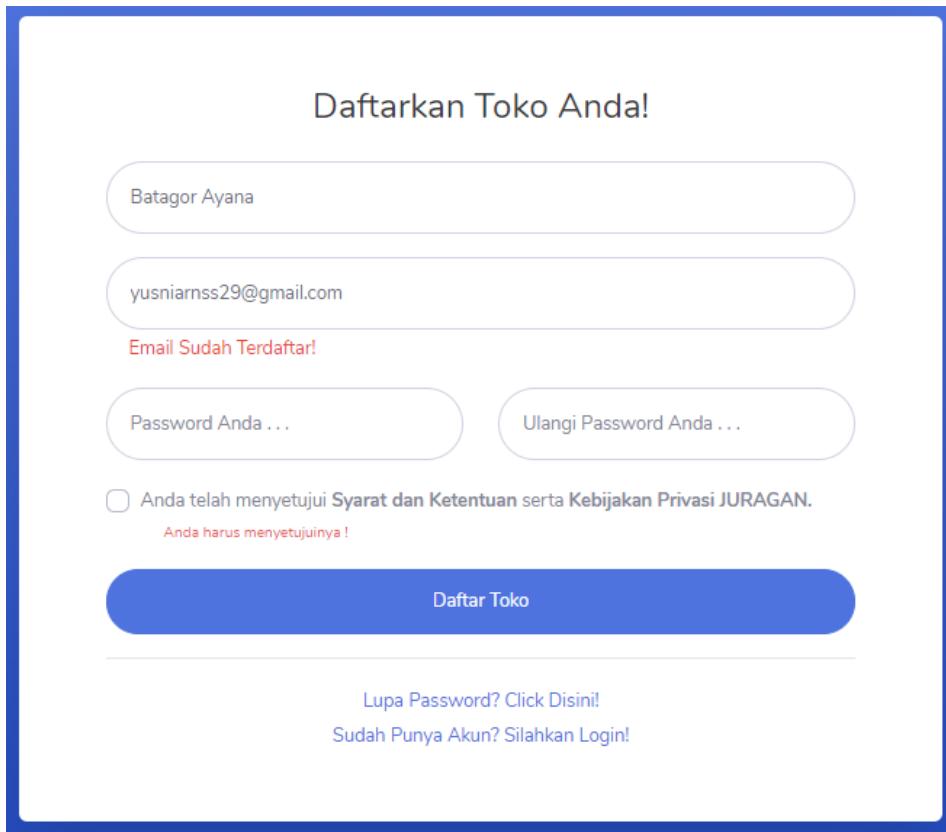
Gambar 7.15 Akun User Telah Aktif

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.



Gambar 7.16 Halaman Daftar Akun Toko

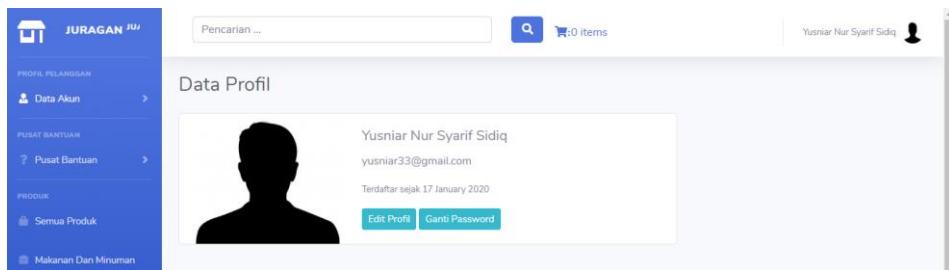
Mengenai prosedur pendaftaran akun toko tersebut terlihat sama seperti pendaftaran pada akun *user* pelanggan, hanya saja yang membedakan adalah dimana pada halaman daftar akun toko terdapat syarat dan ketentuan serta kebijakan privasi yang harus disetujui oleh *user* penjual. Dimana sistem JURAGAN hanya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



Gambar 7.17 Peringatan Pada Pendaftaran Akun Toko

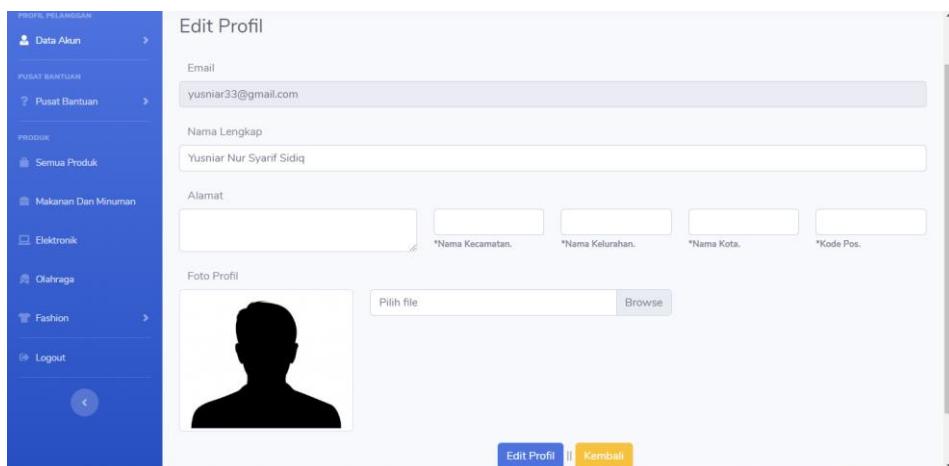
7.3 Mengelola Data Akun Pada Sistem JURAGAN

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.

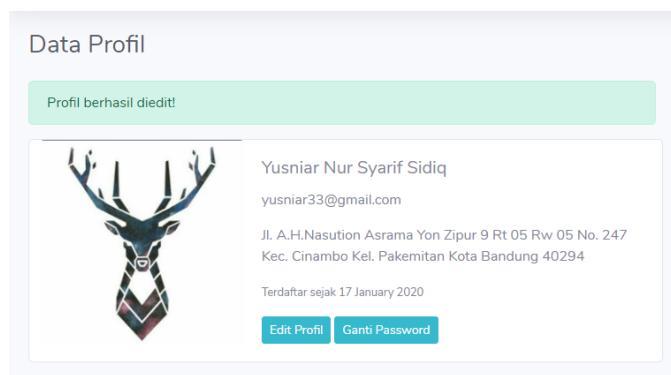
Foto Profil

Screenshot_2016-08-07-23-14-44-1.png Browse

[Edit Profil](#) || [Kembali](#)

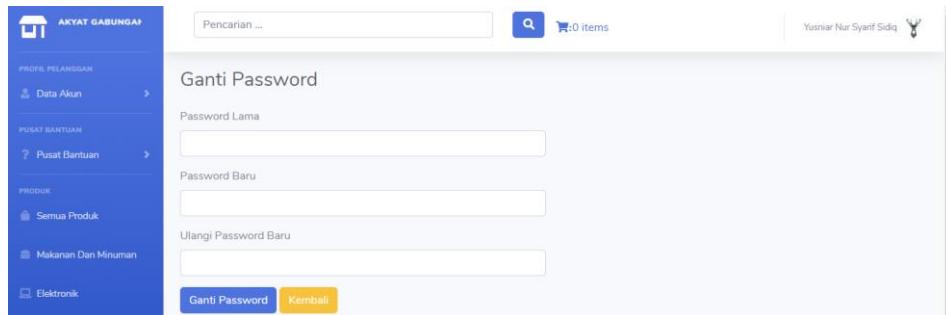
Gambar 7.20 Melakukan Pengeditan Data Profil

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.



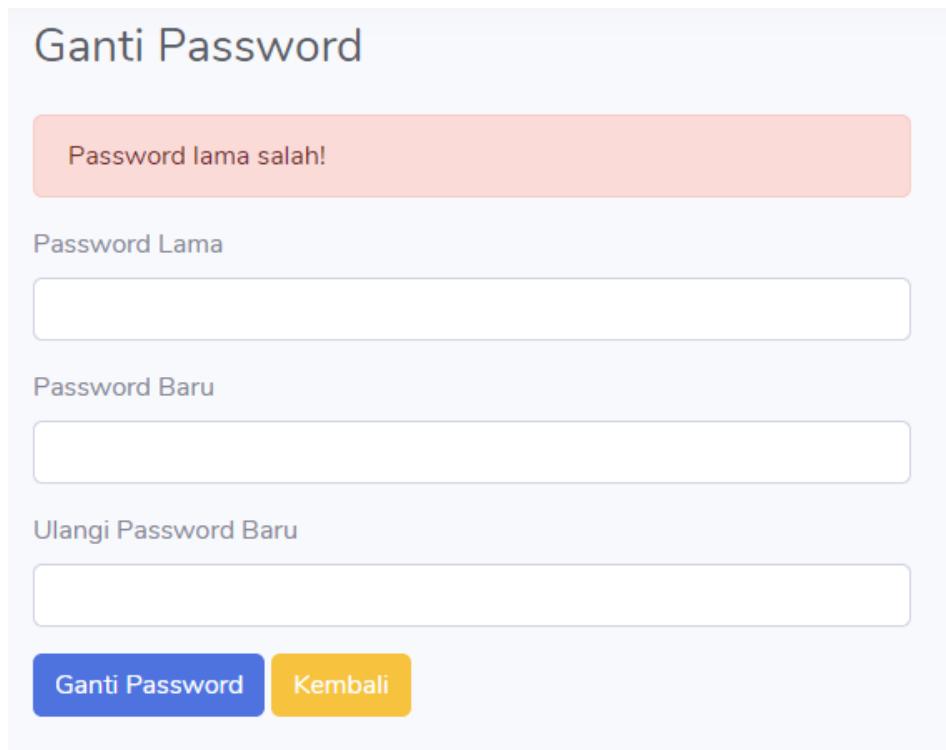
Gambar 7.21 Kelengkapan Data Profil User

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 melaluli sidebar “**Data Akun**”.



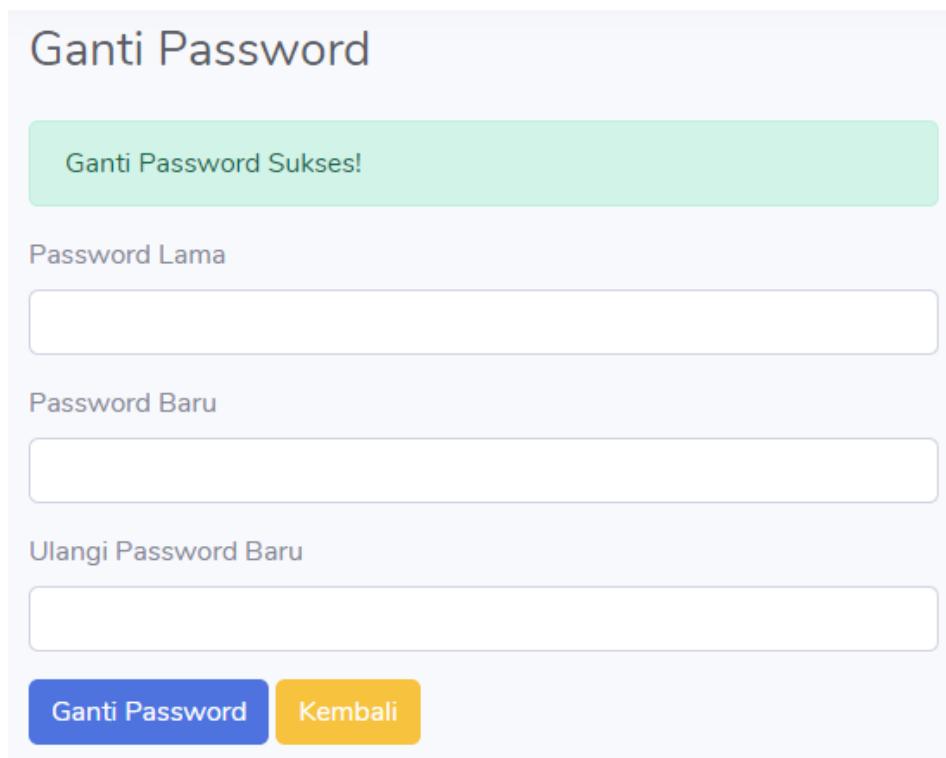
Gambar 7.22 Halaman Ganti Password Pada Sistem JURAGAN

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

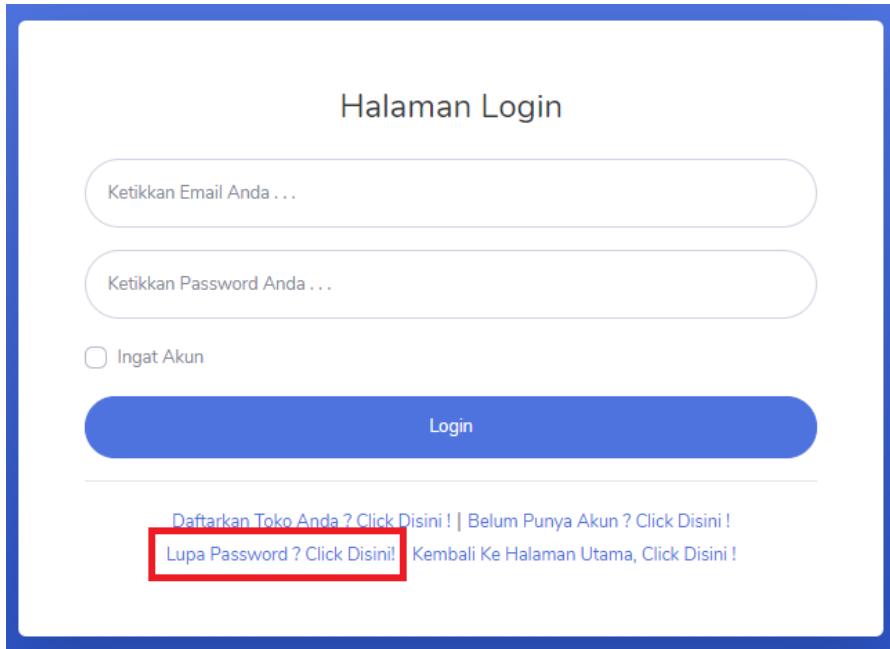
Untuk *rules* pada kolom *password* baru dan ulangi *password* baru sama dengan saat melakukan daftar akun yaitu dimana kedua *password* tidak boleh kosong dan bersifat *matches*. Apabila teman – teman berhasil melakukan proses ganti *password* dimana sistem akan memberikan respon tetep berada pada halaman 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

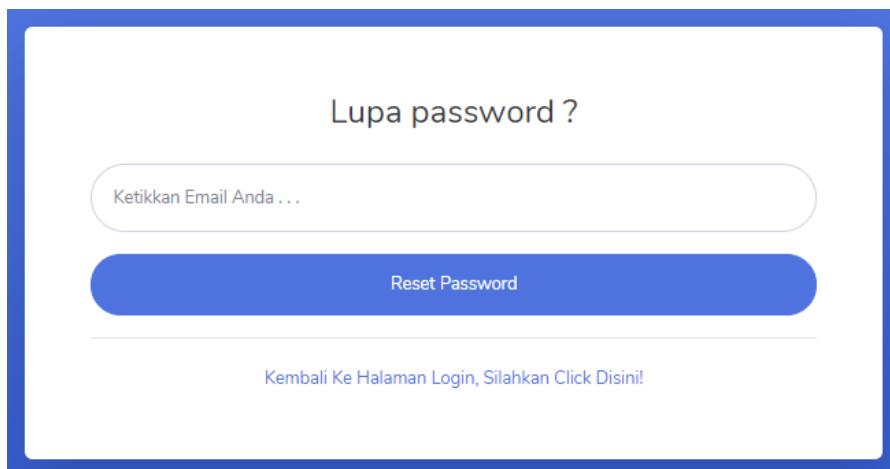
7.4 Proses Lupa Password Pada Sistem JURAGAN

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

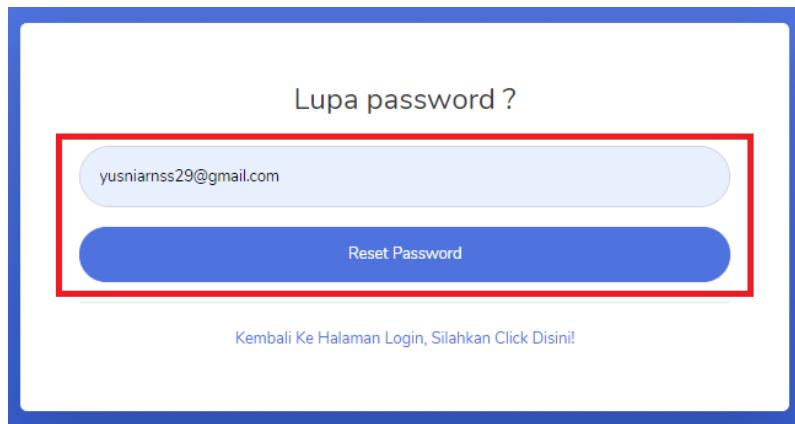
Perhatikan gambar 7.25 tersebut, apabila teman – teman mengklik *text* yang diberikan kotak merah tersebut, sistem akan membawa teman – teman kepada halaman lupa *password* yang terdapat pada sistem JURAGAN tersebut, perhatikan gambar 7.26 sebagai berikut.



Gambar 7.26 Halaman Lupa Password Pada Sistem JURAGAN

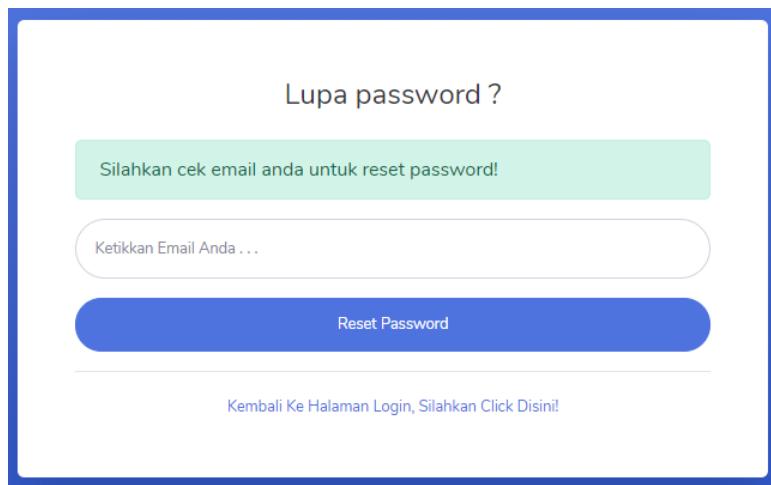
190 | BAB VII Panduang Menggunakan Sistem JURAGAN

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”.



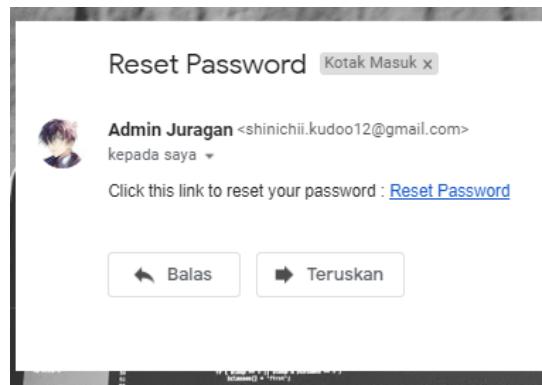
Gambar 7.27 Memberikan Email Yang Akan Di Reset Passwordnya

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.



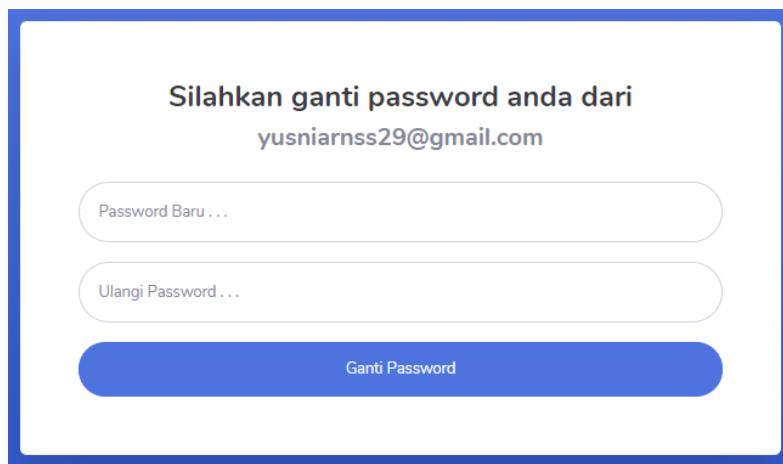
Gambar 7.28 Notifikasi Cek Email Untuk Melakukan Reset Password

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



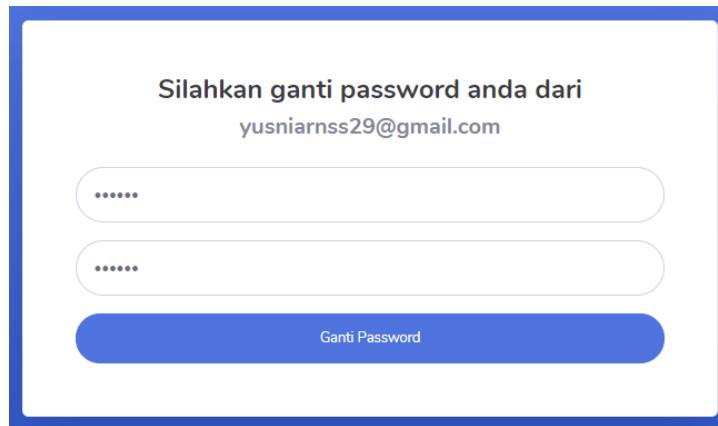
Gambar 7.29 Email Untuk Melakukan Reset Password

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.



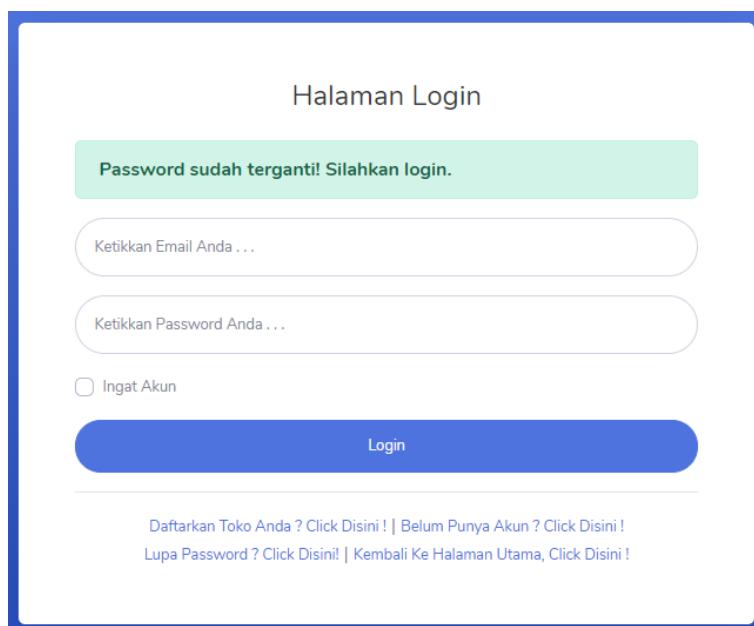
Gambar 7.30 Halaman Reset Password Sistem JURAGAN

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.

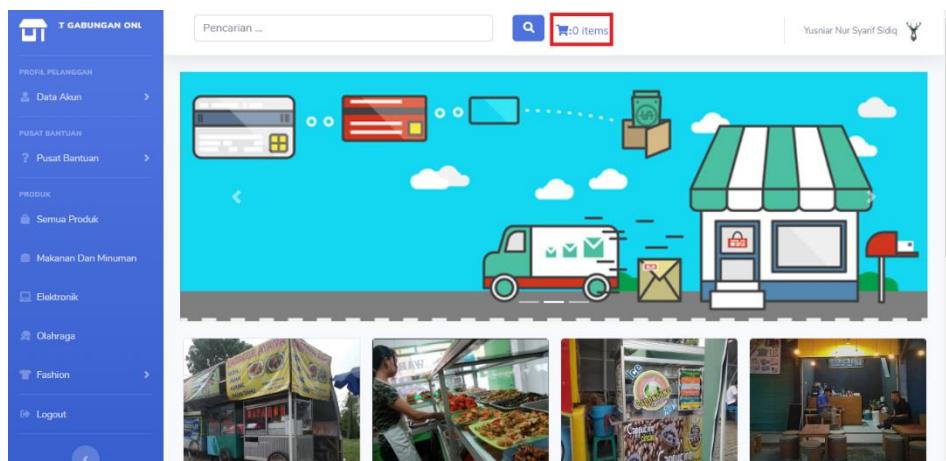


Gambar 7.32 Fitur Lupa Password Berhasil Dilakukan

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.

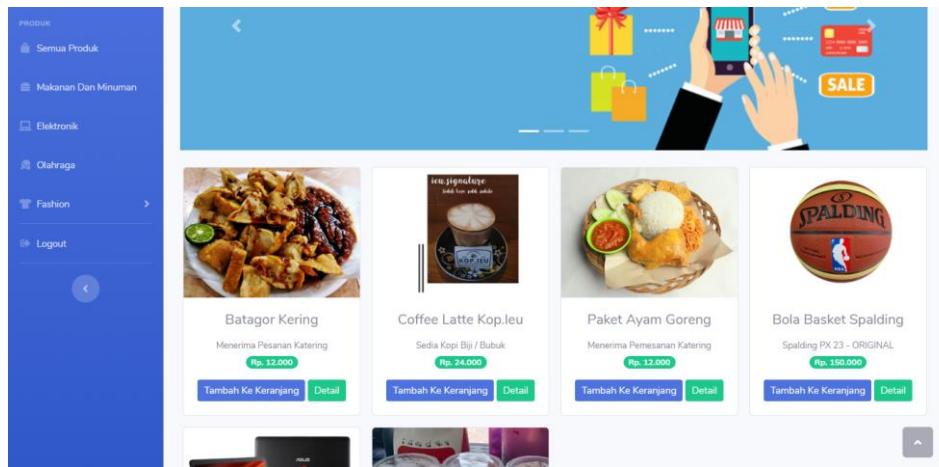
Langkah pertama dimana teman – teman perlu melakukan *login* terlebih dahulu terhadap sistem JURAGAN sehingga dapat masuk ke halaman utamanya, seperti pada gambar 7.33 berikut.



Gambar 7.33 Keranjang Belanja Belum Terisi

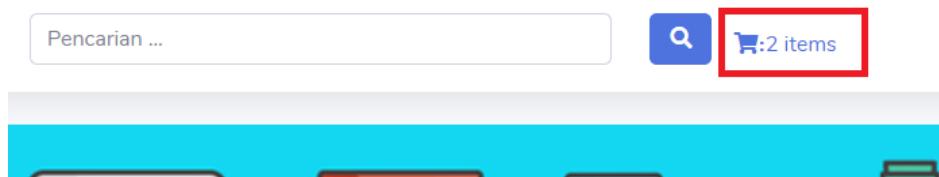
Perhatikan bagian *topbar* pada sistem JURAGAN tersebut, dimana terdapat *icon* keranjang dan masih belum terisi. Silahkan teman – teman tambahkan beberapa produk yang ditawarkan oleh sistem JURAGAN tersebut. Teman – teman dapat melihat semua produk yang terdapat pada sistem JURAGAN dengan cara mengakses 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

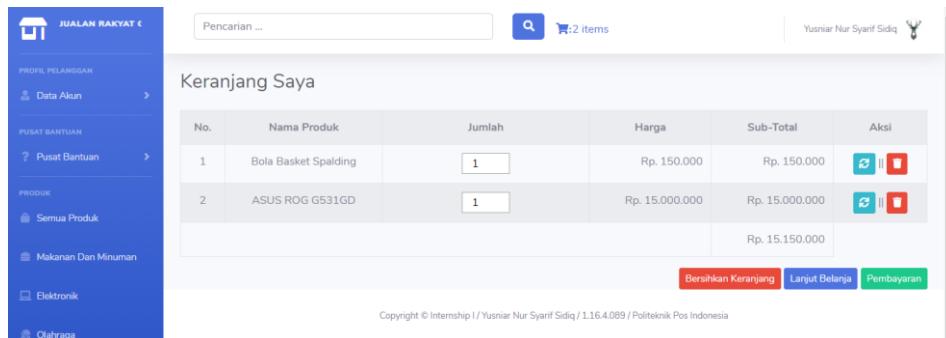
Perhatikan gambar 7.34 tersebut dimana merupakan tampilan dari halaman produk sistem JURAGAN. Pada halaman tersebut semua produk yang di tawarkan oleh para penjual akan ditampilkan. Terdapat dua *button* pada halaman tersebut yaitu *button* “**Tambah Ke Keranjang**” dan *button* “**Detail**” yang sebelumnya 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.



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*.

196 | BAB VII Panduang Menggunakan Sistem JURAGAN

No.	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Bola Basket Spalding	2	Rp. 150.000	Rp. 300.000	
2	ASUS ROG G531GD	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.

No.	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Bola Basket Spalding	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

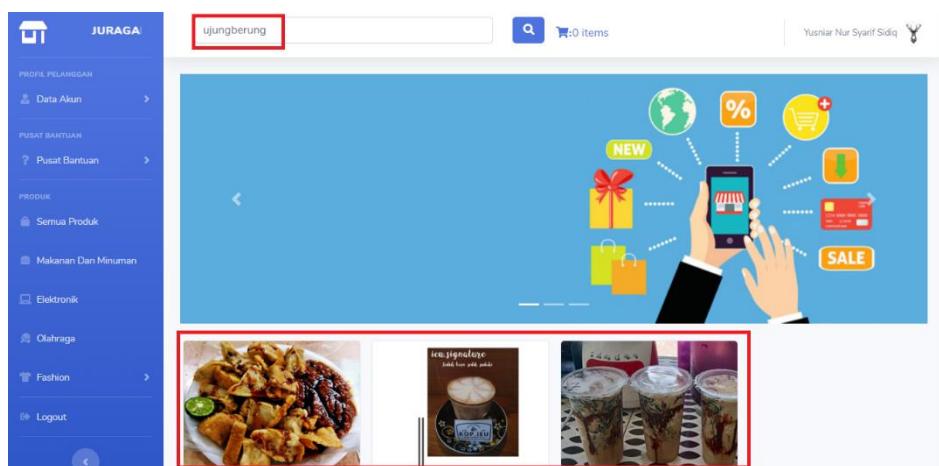
Gambar 7.38 Menghapus Salah Satu Items Pada Halaman Keranjang Belanja

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 berada 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.

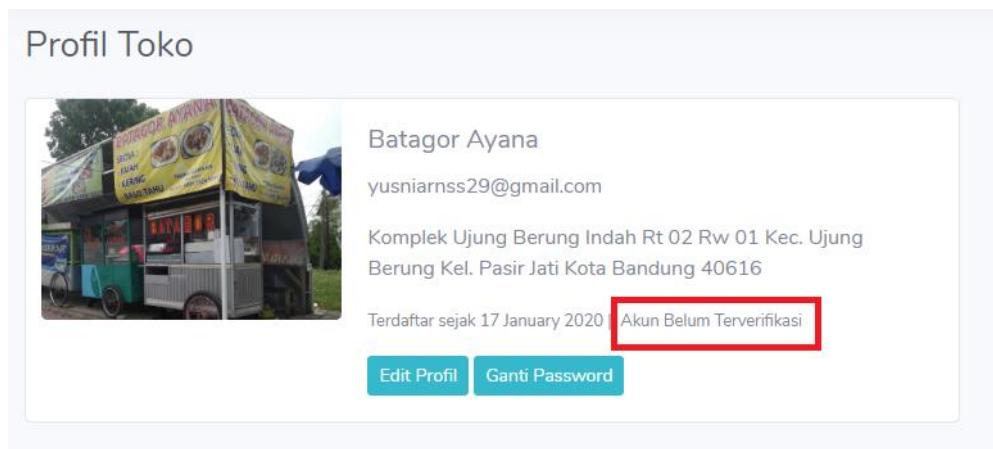


Gambar 7.39 Menampilkan Produk Berdasarkan Komunitas

7.7 Melakukan Verifikasi Akun Toko

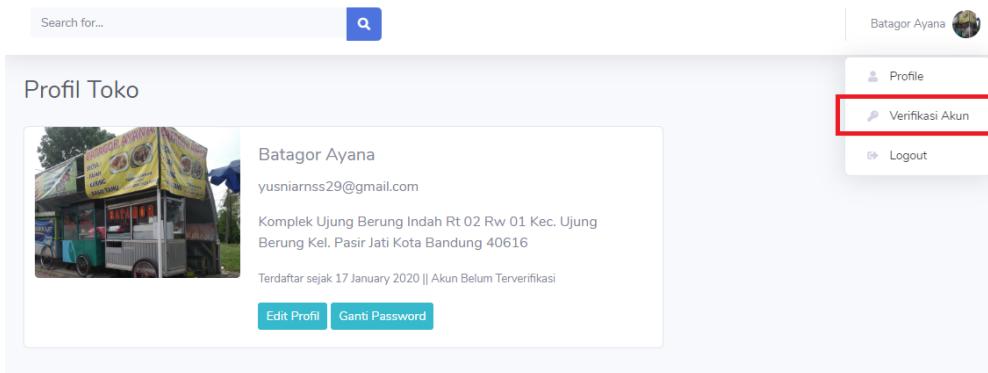
Pada sistem JURAGAN dimana *user* dengan status penjual dimana akan melakukan buka toko. Dimana agar toko mereka dapat di tampilkan pada halaman pelanggan, penjual perlu melakukan verifikasi terlebih dahulu terhadap akun toko tersebut. Dalam melakukan verifikasi tersebut dimana sistem JURAGAN membutuhkan beberapa data seperti no KTP, foto KTP, dan foto diri bersama KTP. Lalu bagaimana caranya melakukan verifikasi akun toko tersebut ?.

Hal pertama yang perlu penjual lakukan adalah tentunya mendaftarkan akun toko tersebut, untuk cara mendaftar akun silahkan teman – teman ikuti langkah – langkah yang ada pada sub bab 7.2. Lakukan *login* dan lihat profil toko anda, apa bila akun belum terverifikasi maka sistem akan menampilkan keterangan “**Akun Belum Terverifikasi**” seperti yang ditunjukkan pada gambar 7.40.



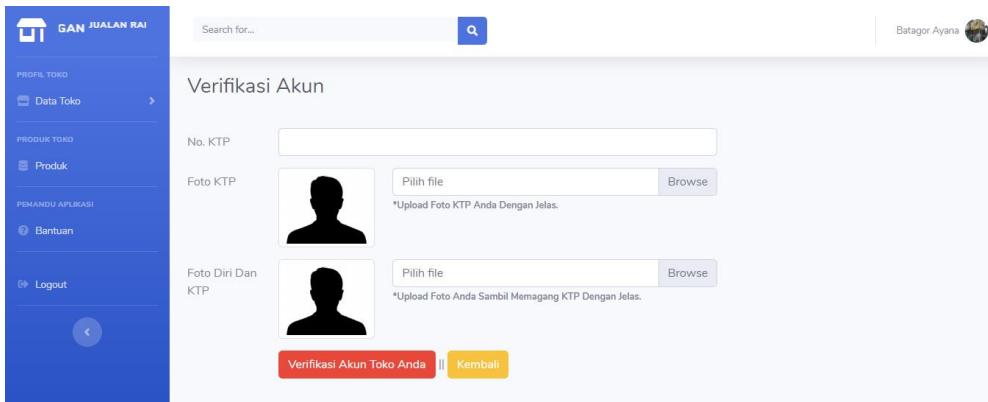
Gambar 7.40 Akun Toko Belum Terverifikasi

Untuk melakukan verifikasi silahkan teman – teman klik foto profil yang terdapat pada bagian *topbar*, maka akan muncul pilihan verifikasi akun, silahkan pilih menu tersebut.



Gambar 7.41 Menu Verifikasi Akun

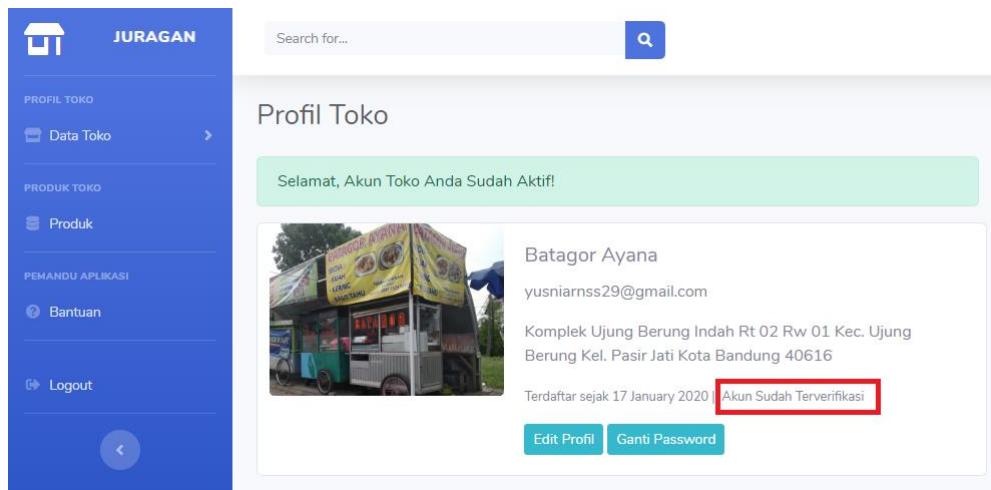
Halaman verifikasi akun akan terlihat seperti pada gambar 7.42, dimana teman – teman perlu melakukan *input* no KTP dan melakukan *upload* foto KTP serta foto diri bersama KTP, lalu pilih button “**Verifikasi Akun Toko Anda**”.



Gambar 7.42 Halaman Verifikasi Akun

Apabila teman – teman sudah mengisi data dengan benar dalam melakukan proses verifikasi tersebut maka status akun toko teman – teman akan berubah menjadi “**Akun Sudah Terverifikasi**” seperti yang ditampilkan pada gambar 7.43. Jika akun sudah terverifikasi dimana toko anda akan ditampilkan pada halaman utama pelanggan.

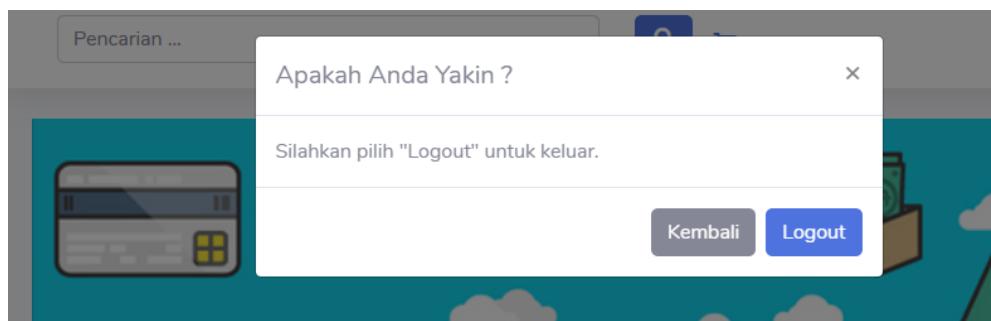
200 | BAB VII Panduang Menggunakan Sistem JURAGAN



Gambar 7.43 Akun Toko Sudah Terverifikasi

7.8 Melakukan Proses Logout

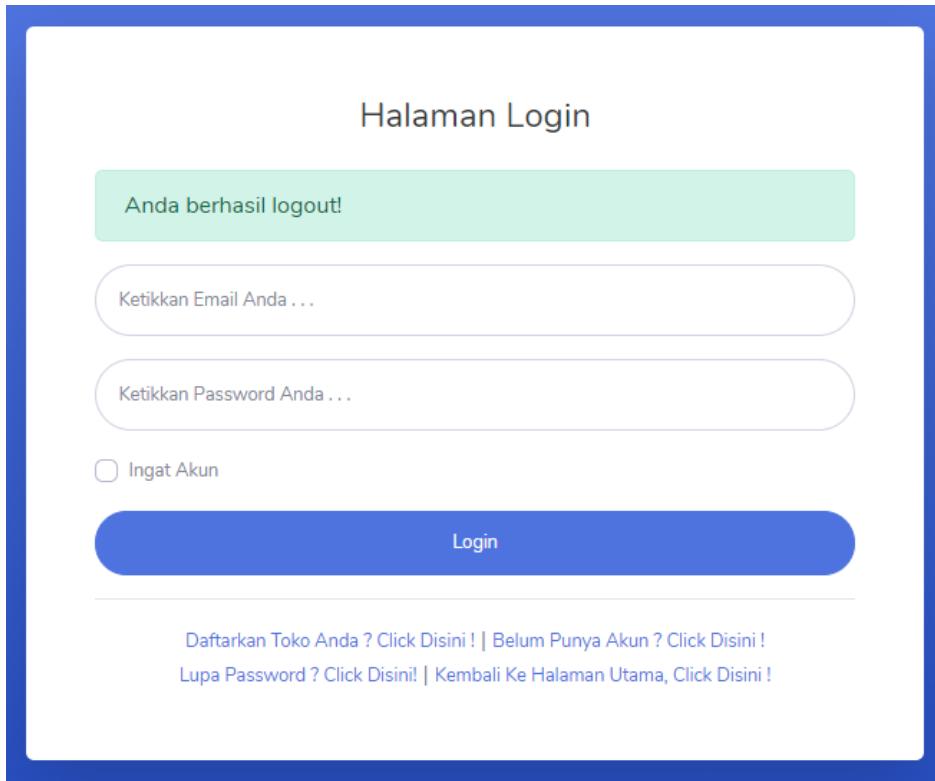
Proses *logout* merupakan kegiatan untuk mengakhiri pekerjaan akun yang sedang melakukan *login*. Dimana akses – akses seperti transaksi tidak dapat dilakukan apabila akun dalam kondisi *logout*. Untuk melakukan *logout* dimana teman – teman dapat melalui menu *logout* yang terdapat pada *sidebar*. Silahkan klik menu tersebut maka akan muncul fungsi *modals* pada sistem tersebut seperti yang ditunjukkan pada gambar 7.44.



Gambar 7.44 Fungsi Modals Pada Logout

Apabila *modals* sudah muncul akan terdapat dua *button* yaitu *button kembali* apabila teman – teman tidak jadi melakukan *logout* dan *button logout*

apabila teman – teman ingin melanjutkan proses *logiut* tersebut. Silahkan pilih *logout*, dimana teman – teman akan di kabawa ke halaman *login* seperti yang ditunjukkan pada gambar 7.45.



Gambar 7.45 Proses Logout Berhasil

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.