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

YUSNIAR NUR SYARIF SIDIQ

1.16.4.089



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

KATA PENGANTAR

DAFTAR ISI

KATA PENGANTAR	2
DAFTAR ISI	3
DAFTAR GAMBAR	5
DAFTAR TABEL	9
DAFTAR SOURCE CODE	10
BAB I	12
BAB II	15
2.1 Usaha Kecil Menengah (UKM).....	15
2.1.1 Usaha Mikro.....	15
2.1.2 Usaha Kecil.....	15
2.1.3 Usaha Menengah	16
2.2 Pedagang Kaki Lima (PKL).....	16
2.3 Promosi.....	16
2.4 Internet	17
2.5 E-Commerce.....	17
2.6 Komunitas	17
2.7 Aplikasi.....	18
2.8 Framework	18
2.9 CodeIgniter	18
2.10 Hypertext Preprocessor (PHP)	19
2.11 <i>Database</i> (Basis Data)	19
2.12 MariaDB	19
BAB III	21
3.1 <i>Use Case</i> JURAGAN	22
3.1.1 Definisi Aktor	26
3.1.2 Definisi <i>Use Case</i>	27
3.3.3 <i>Scenario Use Case</i> JURAGAN	28
3.2 Perancangan Basis Data (<i>Database</i>).....	36

3.2.1 Tabel <i>User</i>	37
3.2.2 Tabel <i>User_token</i>	39
3.2.3 Tabel tb_barang	40
3.3 <i>User Interface</i> Sistem JURAGAN.....	42
BAB IV	59
3.1 Xampp.....	59
4.1.1 Perbedaan PHP 7 Dan PHP 5	66
4.2 Visual Studio Code	67
4.3 <i>Codeigniter</i>	76
4.3.1 <i>Installasi Framework Codeigniter</i>	78
BAB V	84
5.1 Sejarah Codeigniter.....	84
5.2 Kelebihan Codeigniter.....	84
5.3 Struktur Direktori Codeigniter.....	85
5.4 Konfigurasi Framework Codeigniter	91
5.5 Membuat CRUD Dengan Codeigniter.....	99
BAB VI	144
6.1 Mengubungkan Boostrap Dengan CI.....	144
5.2 Fungsi Form Validation	158
DAFTAR PUSTAKA	201

DAFTAR GAMBAR

Gambar 3.1 Komponen Aktor Use Case.....	22
Gambar 3.2 Komponen UseCase.....	23
Gambar 3.3 Komponen Use Case Subject.....	23
Gambar 3.4 Relasi Association.....	24
Gambar 3.5 Relasi Generalization	24
Gambar 3.6 Relasi Despendency	25
Gambar 3.7 Use Case Sistem JURAGAN	25
Gambar 3.8 Perancangan UI Halaman Utama	43
Gambar 3.9 Perancangan UI Halaman Login	45
Gambar 3.10 Perancangan UI Halaman Daftar Akun	46
Gambar 3.11 Perancangan UI Halaman Lupa Password	47
Gambar 3.12 Perancangan UI Halaman Reset Password	48
Gambar 3.13 Perancangan UI Halaman Utama Kondisi Login.....	49
Gambar 3.14 Perancangan UI Halaman Data Profil.....	51
Gambar 3.15 Perancangan UI Halaman Edit Profil.....	52
Gambar 3.16 Perancangan UI Halaman Ganti Password	53
Gambar 3.17 Perancangan UI Halaman Profil Toko	54
Gambar 3.18 Perancangan UI Halaman Produk Toko.....	55
Gambar 3.19 Perancangan UI Halaman Input Produk.....	56
Gambar 3.20 Perancangan UI Halaman Keranjang Belanja.....	57
Gambar 4.1 Website Resmi Xampp.....	60
Gambar 4.2 Tahap Dua Dalam Mendownload Xampp	60
Gambar 4.3 Halaman Versi Xampp.....	61
Gambar 4.4 Proses Awal Installasi Xampp	61
Gambar 4.5 Pemilihan Komponen - Komponen Pada Software Xampp.	62
Gambar 4.6 Pemilihan Tempat Penyimpanan Software Xampp	63

Gambar 4.7 Tempat Penyimpanan Xampp	63
Gambar 4.8 Proses Installasi Xampp Sedang Berjalan.....	64
Gambar 4.9 Tampilan Sofware Xampp	65
Gambar 4.10 Menjalankan Module Apache Dan MySQL	65
Gambar 4.11 Halaman Localhost	66
Gambar 5.1 Dokumentasi Codeigniter	85
Gambar 5.2 Struktur Direktori CI.....	86
Gambar 5.3 Isi Dari Direktori Application	87
Gambar 5.4 File Dalam Folder Config	88
Gambar 5.5 Contoh Script Code Controller	88
Gambar 5.6 Pemanggilan Fungsi Helper	89
Gambar 5.7 Contoh Script Code Model	90
Gambar 5.8 Pemanggilang Fungsi Model Pada Autoload.....	90
Gambar 5.9 Pemanggilan Fungsi Model Dari Controller	90
Gambar 5.10 Folder CI JURAGAN.....	91
Gambar 5.11 Konfigurasi Libraries Sistem JURAGAN.....	92
Gambar 5.12 Konfigurasi Fungsi Helpers Sistem JURAGAN.....	93
Gambar 5.13 Konfigurasi Models Pada Sistem JURAGAN	94
Gambar 5.14 Konfigurasi File Config Sistem JURAGAN	94
Gambar 5.15 Konfigurasi index_page Sistem JURAGAN.....	95
Gambar 5.16 Membuat File Baru .htaccess	95
Gambar 5.17 Membuka Dokumentasi CI	96
Gambar 5.18 Mencari Dokumentasi htaccess.....	96
Gambar 5.19 Script Code htaccess	97
Gambar 5.20 Mengisikan File .htaccess	97
Gambar 5.21 Konfigurasi Default Controller Sistem JURAGAN	98
Gambar 5.22 Konfigurasi Database Sistem JURAGAN	99

Gambar 5.23 Mempersiapkan Software VSCode	100
Gambar 5.24 Open Folder CI Pada VSCode	100
Gambar 5.25 Halaman PHP MyAdmin	101
Gambar 5.26 Membuat Nama Database	101
Gambar 5.27 Membuat Tabel tb_barang	102
Gambar 5.28 Membuat Struktur Tabel tb_barang	102
Gambar 5.29 Struktur Tabel tb_barang	103
Gambar 5.30 Membuat Controllers Barang.php	104
Gambar 5.31 Membuat File m_barang Pada Models	112
Gambar 5.32 Membuat Folder Barang Pada Views	121
Gambar 5.33 Membuat File PHP Pada Folder Barang	122
Gambar 6.1 Halaman Start Bootstrap SB ADMIN 2	145
Gambar 6.2 Menu Download Template Bootstrap SB ADMIN 2	146
Gambar 6.3 Meng Extract File Zip Bootstrap	146
Gambar 6.4 File Codeigniter Pada Direktori Xampp/htdocs	147
Gambar 6.5 Membuat Folder Vendor	147
Gambar 6.6 Memindahkan File SB ADMIN 2 Ke Dalam Vendor.....	147
Gambar 6.7 Tampilan View Welcome_message.php	148
Gambar 6.8 Membuat Folder Assets	148
Gambar 6.9 Mengisi Folder Assets.....	149
Gambar 6.10 Membuat Folder templates Pada Views.....	149
Gambar 6.11 Membuat 4 File PHP Pada Folder Templates	150
Gambar 6.12 Isi File Header.php	151
Gambar 6.13 Isi File Sidebar.php	152
Gambar 6.14 Isi File Topbar.php	153
Gambar 6.15 Isi File Footer.php	154
Gambar 6.16 Perubahan Sidebar Dan Topbar	156

Gambar 6.17 Menerapkan Templates Boostrap Terhadap CI Berhasil .158

Gambar 6.18 Contoh Form Validarion Pada Sistem JURAGAN159

DAFTAR TABEL

Tabel 3. 1 Definisi Aktor Pada Use Case JURAGAN.....	26
Tabel 3. 2 Definisi Komponen Use Case Pada Use Case JURAGAN	27
Tabel 3.3 Scenario Use Case Daftar Akun	28
Tabel 3.4 Scenario Use Case Login.....	29
Tabel 3.5 Scenario Use Case Kelola Profil.....	30
Tabel 3.6 Scenario Use Case Transaksi	31
Tabel 3.7 Scenario Use Case Verifikasi Akun.....	32
Tabel 3.8 Scenario Use Case Kelola Produk (Penjual).....	33
<i>Tabel 3.9 Scenario Use Case Kelola Produk (Admin)</i>	35
Tabel 3.10 Scenario Use Case Kelola User	35
Tabel 3.11 Struktur Tabel User.....	37
Tabel 3.12 Struktur Tabel User_token.....	40
Tabel 3.13 Struktur Tabel tb_barang	40
Tabel 4. 1 Spesifikasi Software VS Code	68

DAFTAR SOURCE CODE

Source Code 5.1 SQL Create Table	104
Source Code 5.2 Code Pertama Controllers	105
Source Code 5.3 Pemanggilan Fungsi Models Dan Form_validation ...	106
Source Code 5.4 Membuat Function Index	106
Source Code 5.5 Membuat Function Tambah Pada Controllers.....	107
Source Code 5.6 Membuat Function Edit Pada Controllers	108
Source Code 5.7 Membuat Function Delete Pada Controllers	109
Source Code 5.8 Code Full Controllers Barang.php.....	111
Source Code 5.9 Code Awal Pada m_barang.php	112
Source Code 5.10 Mendeskripsikan Field Pada Models	113
Source Code 5.11 Function Fungsi Read.....	114
Source Code 5.12 Fungsi Read Berdasarkan Id.....	114
Source Code 5.13 Membuat Function Simpan Pada Models	115
Source Code 5.14 Membuat Fungsi Update Pada Models	116
Source Code 5.15 Membuat Fungsi Delete Pada Models.....	117
Source Code 5.16 Code Full Models File m_barang.php.....	120
Source Code 5.17 Memanggil Header Pada View v_barang.....	123
Source Code 5.18 Memanggil Template Topbar Dan Sidebar Pada View v_barang.....	123
Source Code 5.19 Membuat Tabel Pada View v_barang	125
Source Code 5.20 Membuat Table Body Pada View v_barang.....	126
Source Code 5.21 Membuat Footer Pada View v_barang	127
Source Code 5.22 Code Full File v_barang	130
Source Code 5.23 Memanggil Flashdata Pada View inputbarang.....	131
Source Code 5.24 Membuat Button Kembali Pada Form inputbarang .	132
Source Code 5.25 Membuat Textbox Pada View inputbarang.....	134

Source Code 5.26 Code Full View File inputbarang.php	137
Source Code 5.27 Membuat Flashdata Pada View Edit	138
Source Code 5.28 Membuat Button Kembali Pada View Edit.....	138
Source Code 5.29 Membuat Form Input Dengan Value Pada Edit	140
Source Code 6.1 Merubah Controllers Welcome	154
Source Code 6.2 Merubah File Header.php Pada Controllers Welcome	156
Source Code 6.3 Merubah Code Footer.php.....	157

BAB I

PENDAHULUAN

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

tangan atau keterampilan tangan sehingga menciptakan sebuah seni atau karya.

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

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

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

BAB II

LANDASAN TEORI

2.1 Usaha Kecil Menengah (UKM)

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

2.1.1 Usaha Mikro

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

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

2.1.2 Usaha Kecil

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

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

2.1.3 Usaha Menengah

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

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

2.2 Pedagang Kaki Lima (PKL)

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

2.3 Promosi

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

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

2.4 Internet

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

2.5 E-Commerce

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

2.6 Komunitas

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

2.7 Aplikasi

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

2.8 Framework

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

2.9 CodeIgniter

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

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

2.10 Hypertext Preprocessor (PHP)

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

2.11 Database (Basis Data)

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

2.12 MariaDB

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

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

BAB III **PERANCANGAN**

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

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

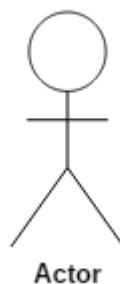
Untuk membangun aplikasi dengan konsep *e-commerce* seperti JURAGAN (Jualan Rakyat Gabungan *Online*) dimana perlu dilakukannya tahap analisis dan perancangan terlebih dahulu. Dalam buku ini dimana penulis telah menyediakan perancangan dalam membuat sistem JURAGAN tersebut yang terdiri dari perancangan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

3.1 Use Case JURAGAN

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

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

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



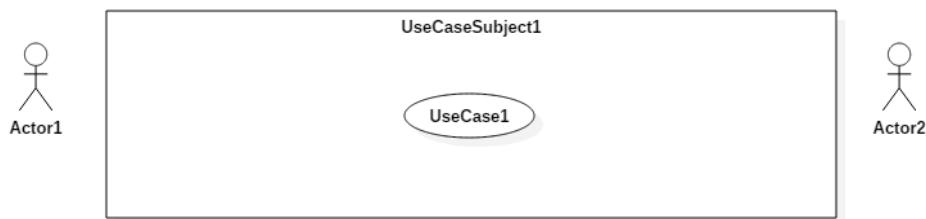
Gambar 3.1 Komponen Aktor Use Case

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



Gambar 3.2 Komponen UseCase

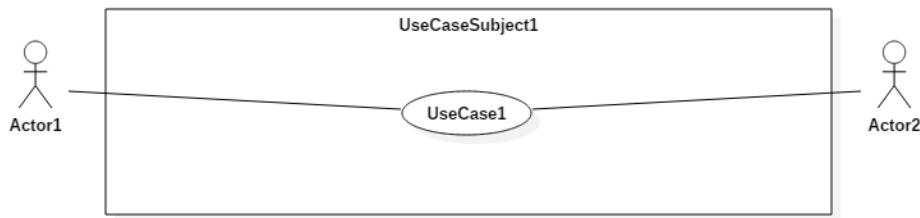
UseCase merupakan komponen yang berfungsi sebagai memberikan gambaran fungsional sistem yang akan dibangun yang dimana dapat membuat pengguna lebih mengerti dalam mengaplikasikan sistem tersebut.



Gambar 3.3 Komponen Use Case Subject

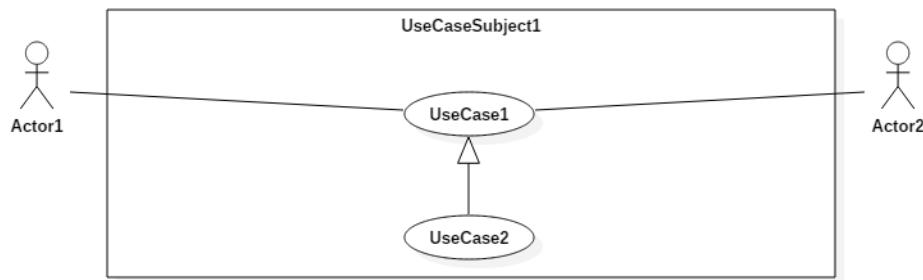
Use case subject merupakan komponen yang berfungsi sebagai tempat menampungnya *subject – subject use case* atau sebagai sebuah *frame* dari *user case* tersebut.

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



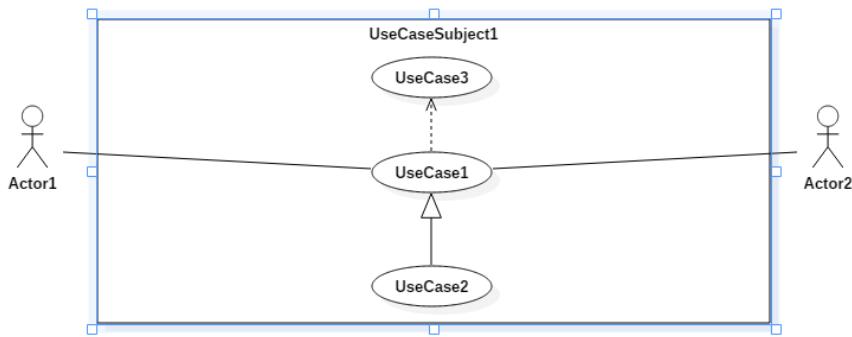
Gambar 3.4 Relasi Association

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



Gambar 3.5 Relasi Generalization

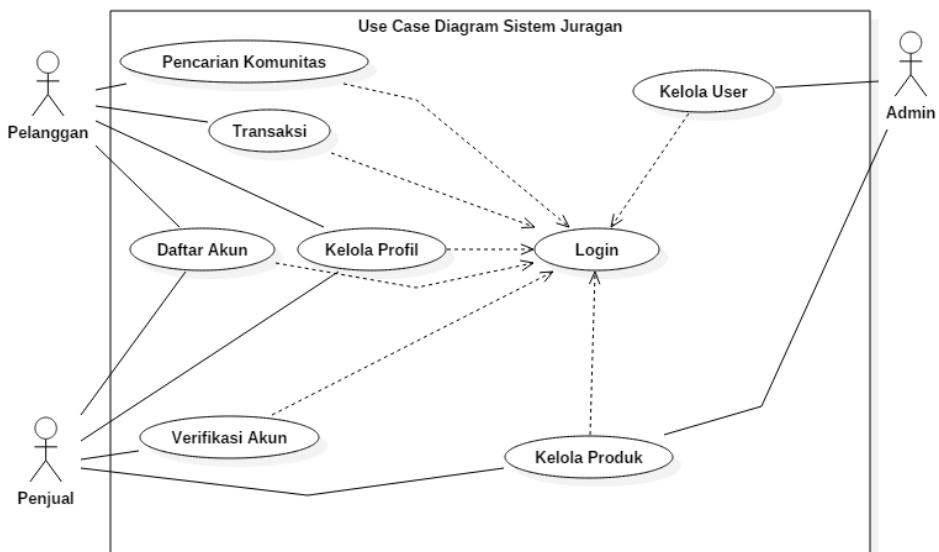
Relasi *generalization* berfungsi sebagai penghubung sebuah elemen yang menjadi spesialisasi terhadap elemen yang lain. *Generalization* biasa berbentuk garis lurus panjang disertai anak panah pada ujungnya.



Gambar 3.6 Relasi Despendency

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

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



Gambar 3.7 Use Case Sistem JURAGAN

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

3.1.1 Definisi Aktor

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

Tabel 3. 1 Definisi Aktor Pada Use Case JURAGAN

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

3.1.2 Definisi Use Case

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

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

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

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

3.3.3 Scenario Use Case JURAGAN

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

Tabel 3.3 Scenario Use Case Daftar Akun

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

Skenario	
Kondisi Awal	Halaman registrasi
Aksi Aktor	Reaksi Sistem
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	Reaksi Sistem
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.
2. -	b. Melakukan validasi
3. -	c. Menambahkan produk ke dalam keranjang.

Tabel 3.7 Scenario Use Case Verifikasi Akun

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

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

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

Tabel 3.9 Scenario Use Case Kelola Produk (Admin)

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

3.2 Perancangan Basis Data (*Database*)

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

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

3.2.1 Tabel *User*

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

Tabel 3.11 Struktur Tabel User

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

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

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

3.2.2 Tabel *User_token*

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

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

Tabel 3.12 Struktur Tabel User_token

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

3.2.3 Tabel tb_barang

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

Tabel 3.13 Struktur Tabel tb_barang

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

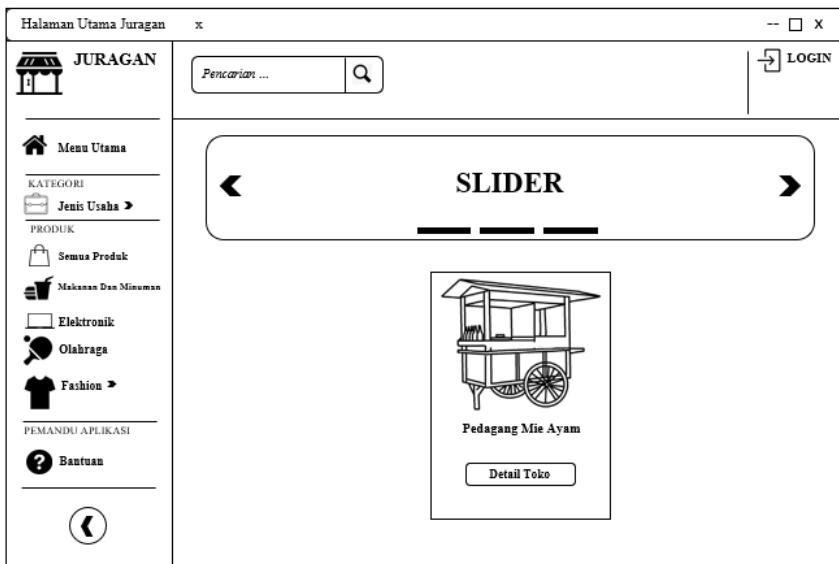
<i>Id</i>	<i>Integer (11)</i>	Merupakan <i>id</i> dari <i>user</i> dengan level penjual.
<i>Email</i>	<i>Varchar (15)</i>	Merupakan email dari <i>user</i> dengan level <i>penjual</i> .
<i>Nama_brg</i>	<i>Varchar (25)</i>	Merupakan nama dari produk yang akan di <i>input</i> .
<i>Keterangan</i>	<i>Varchar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

3.3 User Interface Sistem JURAGAN

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

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

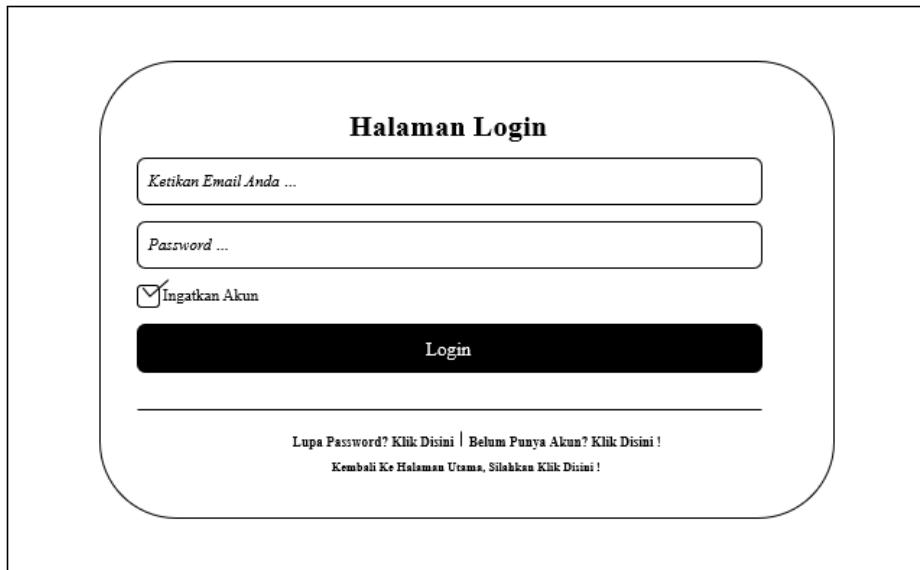
Dalam sistem JURAGAN ini dimana *user interface* akan di desain dengan cukup nyaman dan bersifat *user friendly*. Adapun perancangan yang dilakukan untuk membangun sistem JURAGAN dapat teman – teman lihat pada gambar 3.8 – 3.20.



Gambar 3.8 Perancangan UI Halaman Utama

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

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



Gambar 3.9 Perancangan UI Halaman Login

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

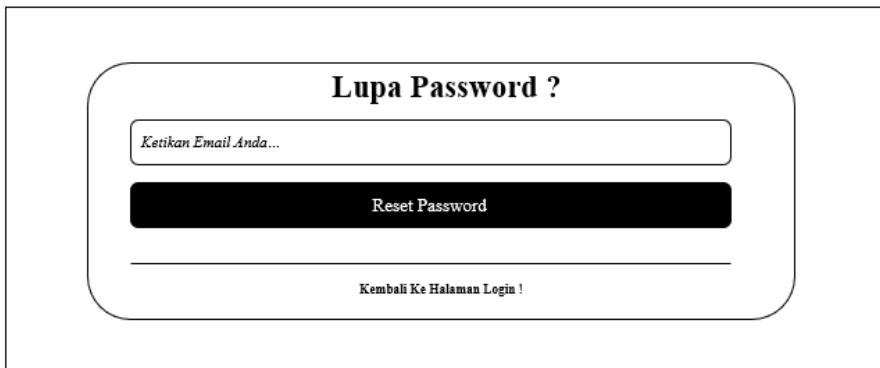
nama *login*, hal seperti ini dapat kita contohkan sebagai desain interaksi. *Button* berfungsi sebagai memberikan aksi terhadap sistem tersebut, sebagai contoh apabila *button login* tersebut di klik maka aksi *login* akan dijalankan. Di bagian bawah *buttoh* dimana terdapat fungsi “hr” yang merupakan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

The diagram illustrates a user interface design for account registration. At the top center, the title "Daftarkan Akun Anda!" is displayed. Below it are four input fields: "Nama Lengkap ...", "Email Anda ...", "Password Anda ...", and "Ulangi Password Anda ...". A large, dark rectangular button labeled "Daftar Akun" is positioned below these fields. At the bottom of the form, there is a horizontal line with two links: "Lupa Password ? Klik Disini !" and "Sudah Punya Akun ? Silahkan Login !".

Gambar 3.10 Perancangan UI Halaman Daftar Akun

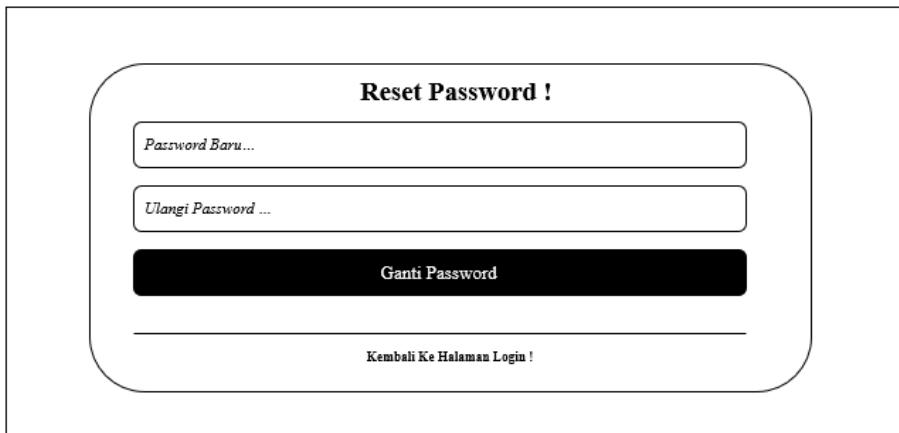
Pada gambar 3.10 merupakan perancangan *user interface* pada halaman daftar akun pada sistem JURAGAN. Halaman Daftar Akun ini berfungsi sebagai *form* untuk melakukan registrasi data diri *user*. Pada halaman daftar akun tersebut dimana komponen – komponen *template* yang

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



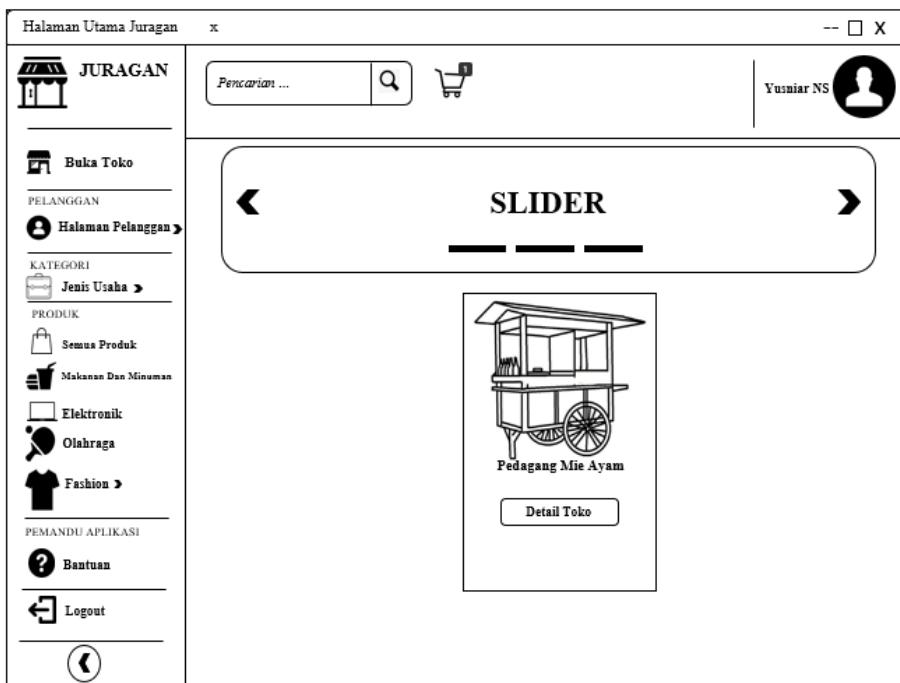
Gambar 3.11 Perancangan UI Halaman Lupa Password

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



Gambar 3.12 Perancangan UI Halaman Reset Password

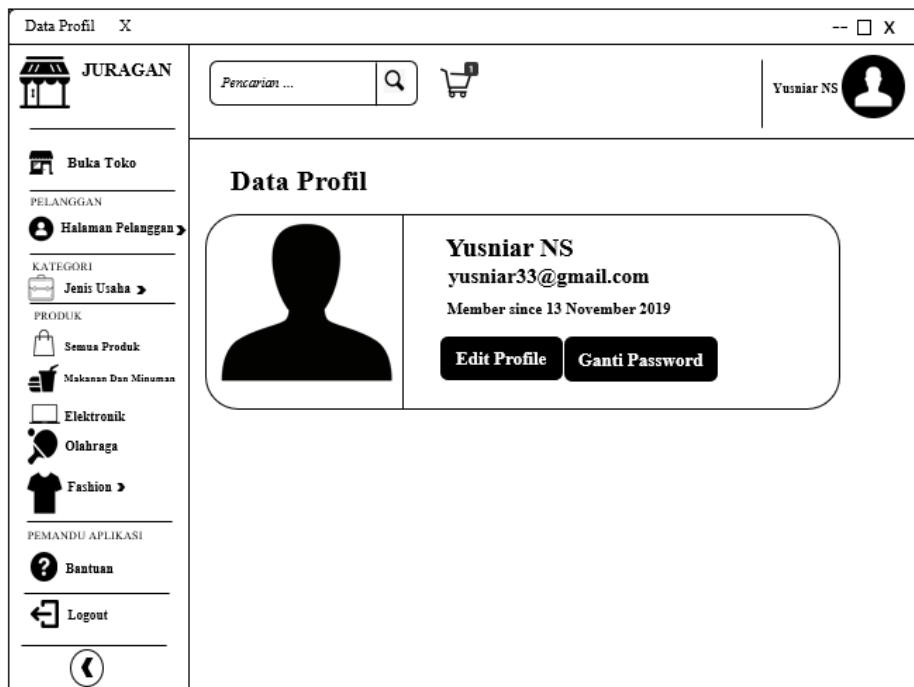
Pada gambar 3.12 merupakan perancangan *user interface* halaman reset *password* pada sistem JURAGAN. Pada halaman reset *password* tersebut dimana lanjutan dari halaman lupa *password*. Halaman reset *password* tersebut berfungsi untuk melakukan pembuatan *password* baru terhadap *user* tersebut. Pada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



Gambar 3.13 Perancangan UI Halaman Utama Kondisi Login

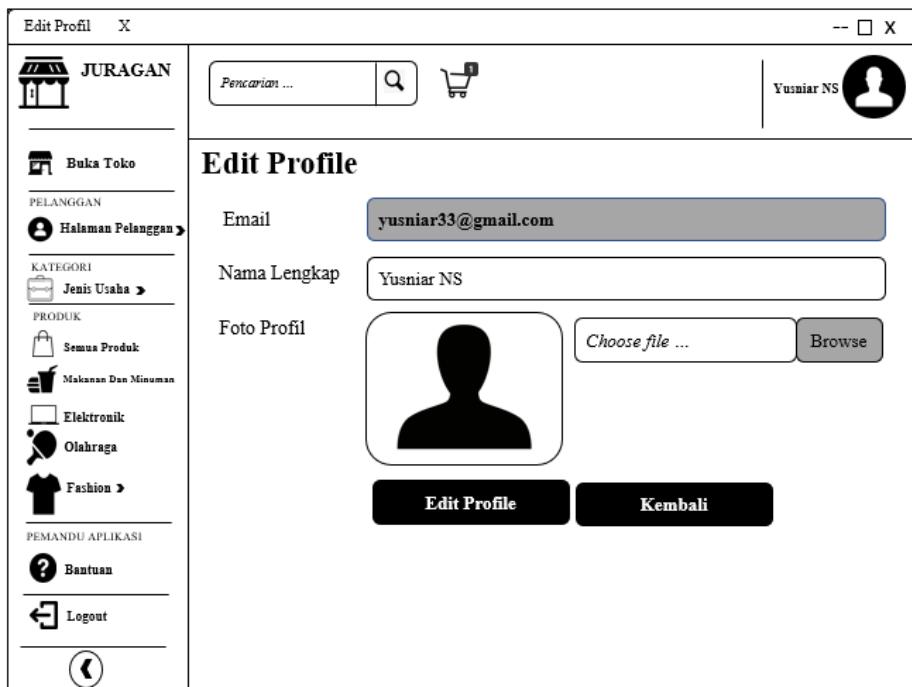
Pada gambar 3.13 merupakan perancangan *user interface* halaman utama pada sistem JURAGAN. Pada sistem JURAGAN ini dimana terdapat dua halaman utama, yang membedakan dari kedua halaman tersebut adalah kondisi dari tiap – tiap *user*. Pada gambar 3.8 dimana merupakan halaman utama dengan kondisi *user*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

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



Gambar 3.14 Perancangan UI Halaman Data Profil

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

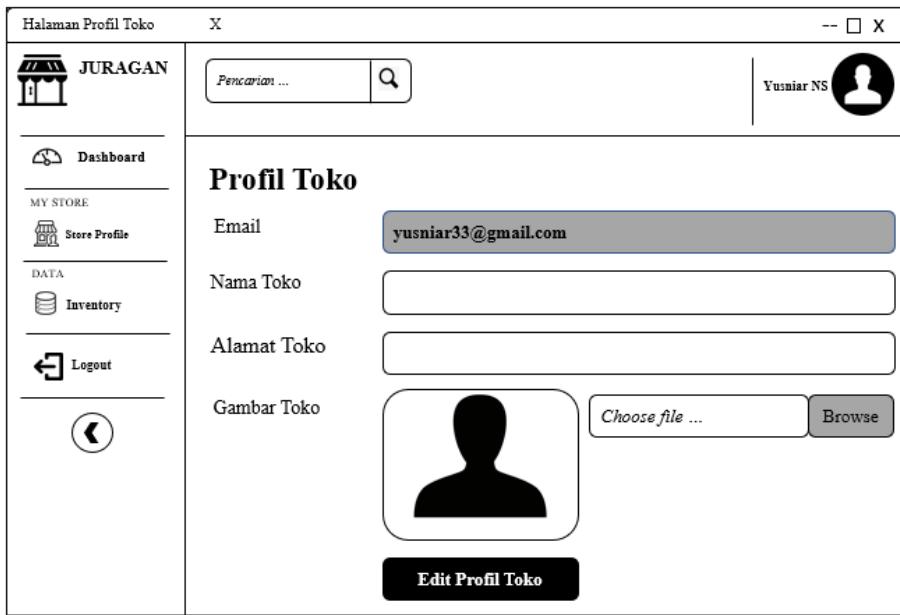


Gambar 3.15 Perancangan UI Halaman Edit Profil

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

Gambar 3.16 Perancangan UI Halaman Ganti Password

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



Gambar 3.17 Perancangan UI Halaman Profil Toko

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

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

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

Gambar 3.18 Perancangan UI Halaman Produk Toko

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

FORM INPUT PRODUK	
Email	<input type="text" value="yusniar33@gmail.com"/>
Nama Produk	<input type="text"/>
Detail Produk	<input type="text"/>
Kategori	<input type="text"/>
Harga	<input type="text"/>
Stok	<input type="text"/>
Gambar Produk	<input type="text"/>
Kembali 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 *texbox*, dan *button* yang berfungsi sebagai pemberian aksi terhadap halaman tersebut.

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

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

Gambar 3.20 Perancangan UI Halaman Keranjang Belanja

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

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

BAB IV

SOFTWARE PENDUKUNG

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

3.1 Xampp

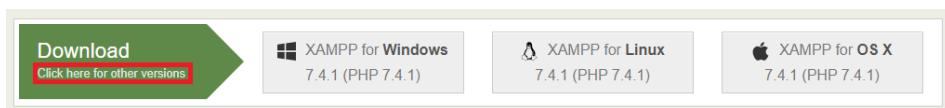
Xampp merupakan sebuah *software* yang memiliki fungsi sebagai server lokal dimana berguna sebagai membuat sebuah *website* yang sifatnya masih dikembangkan. *Xampp* bekerja tanpa menggunakan koneksi internet atau secara *offline* yang dimana layaknya *web hosting* namun tidak dapat diakses oleh banyak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

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



Gambar 4.1 Website Resmi Xampp

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



Gambar 4.2 Tahap Dua Dalam Mendownload Xampp

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

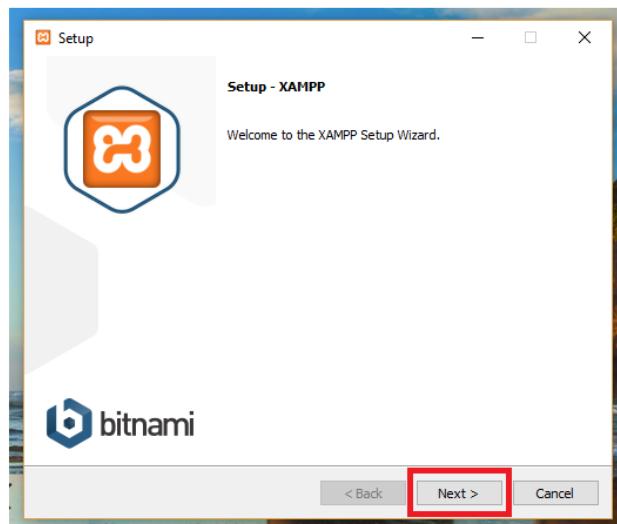
Download

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

The screenshot shows the XAMPP download page. It features a table with three rows, each representing a different version of XAMPP. The columns are 'Version', 'Checksum' (with md5 and sha1 options), and 'Size' (145 Mb). Each row has a 'Download (64 bit)' button. Below the table, there are links for 'Requirements', 'Add-ons', and 'More Downloads'. A note states that Windows XP or 2003 are not supported and provides a link to a compatible version. To the right, there are sections for 'Documentation/FAQs' (with a note about no real manual) and 'Add-ons and Themes' (listing Bitnami).

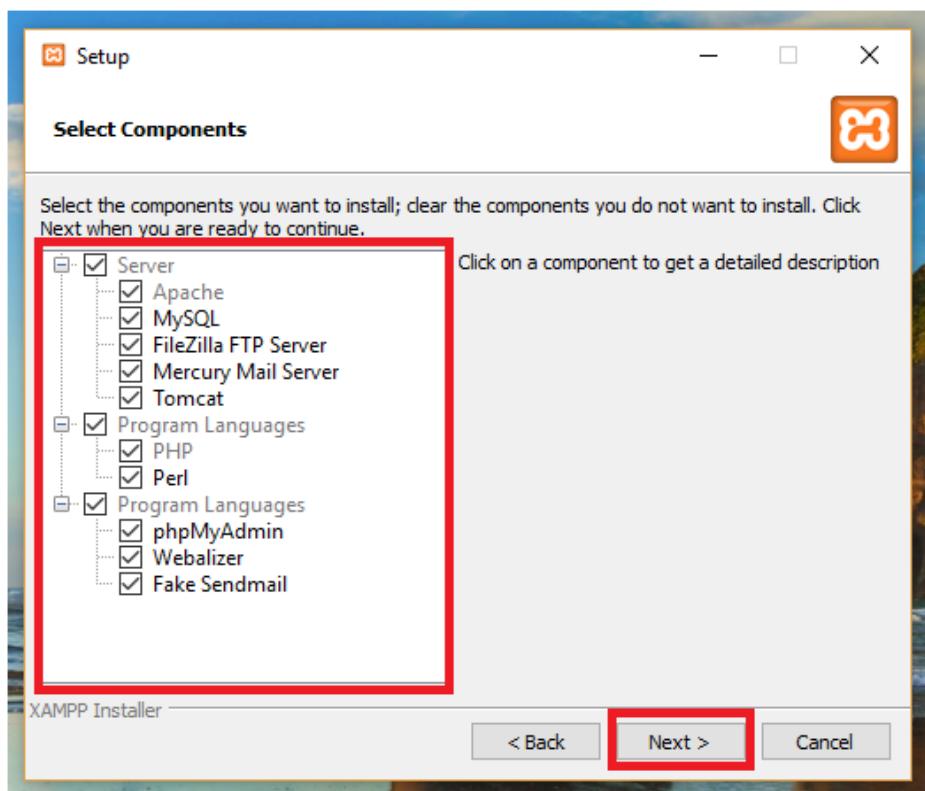
Gambar 4.3 Halaman Versi Xampp

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



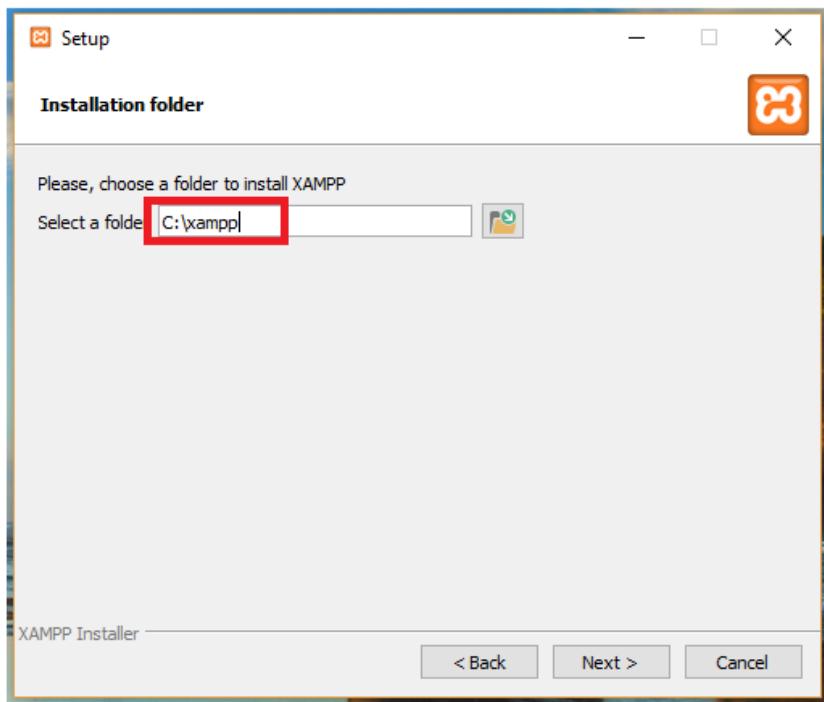
Gambar 4.4 Proses Awal Installasi Xampp

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



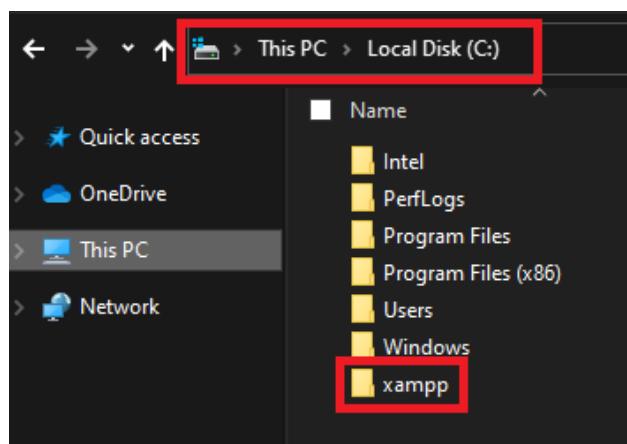
Gambar 4.5 Pemilihan Komponen - Komponen Pada Software Xampp

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



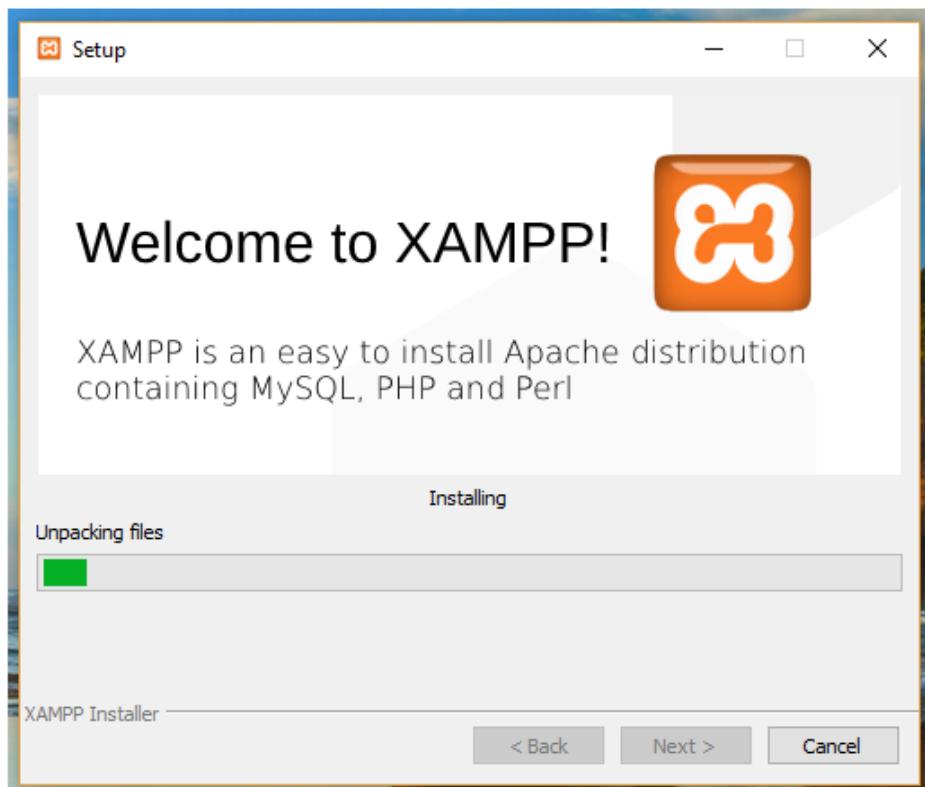
Gambar 4.6 Pemilihan Tempat Penyimpanan Software Xampp

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



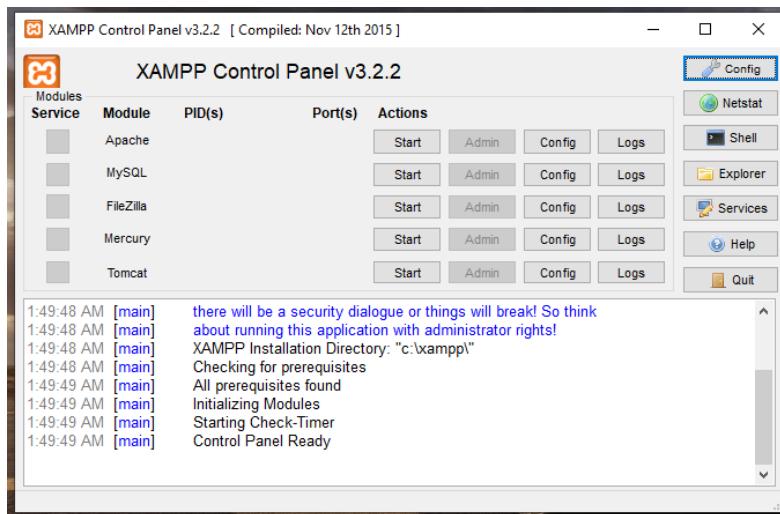
Gambar 4.7 Tempat Penyimpanan Xampp

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



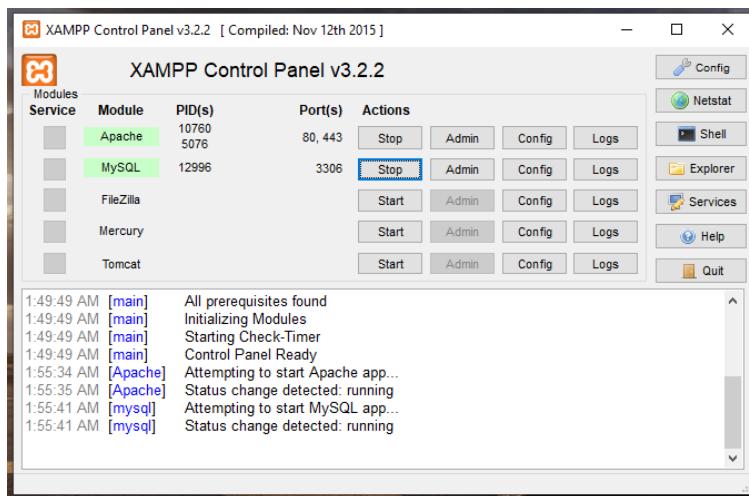
Gambar 4.8 Proses Installasi Xampp Sedang Berjalan

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



Gambar 4.9 Tampilan Software Xampp

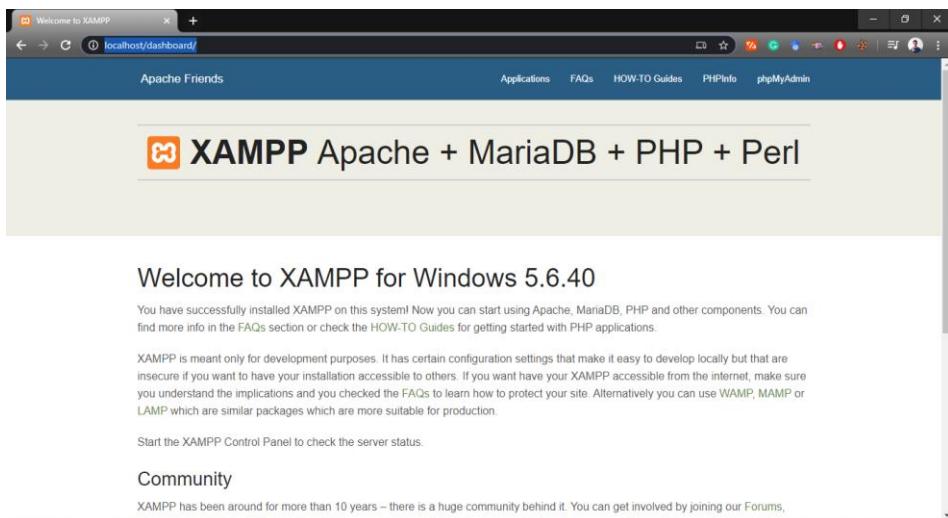
12. Ada beberapa hal yang perlu teman – teman perhatikan. Silahkan teman – teman jalankan *module Apache* dan *MySQL* dengan cara memberikan *actions start*. Apabila *module Apache* dan *MySQL* berubah warna menjadi hijau menandakan *software xampp* teman – teman sudah siap digunakan.



Gambar 4.10 Menjalankan Module Apache Dan MySQL

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

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



Gambar 4.11 Halaman Localhost

4.1.1 Perbedaan PHP 7 Dan PHP 5

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

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

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

4.2 Visual Studio Code

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

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

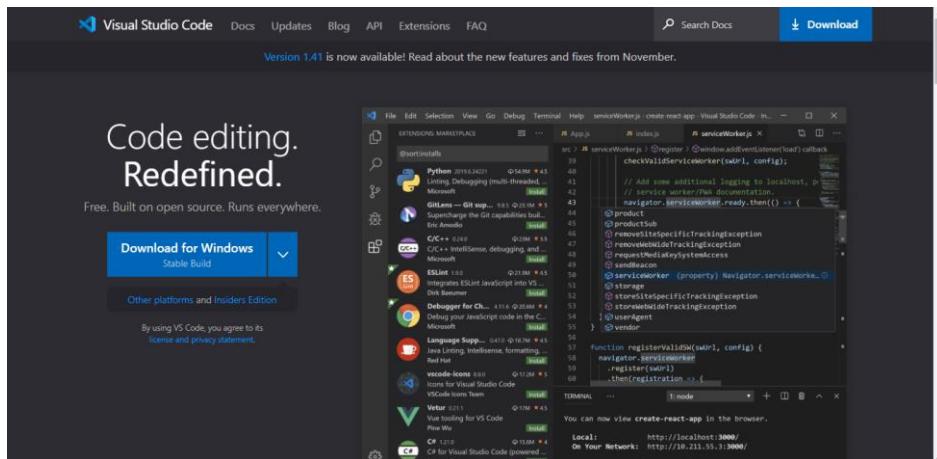
Sebelum melakukan penginstallan *software Vscode* dimana kita harus mengetahui spesifikasi atau *requirements* yang dibutuhkan. Spesifikasi yang dibutuhkan dapat dilihat pada tabel 4.1.

Tabel 4.1 Spesifikasi Software VS Code

<i>Hardware</i>	<i>Operation System</i>	<i>Sarat</i>
<i>Processor 1.6 GHz 1 GB RAM</i>	<i>Windows 7, 8.0, 8.1, 10</i>	<i>32/64 Bit</i>
	<i>Linux Debian: Ubuntu 14.04, Debian 7</i>	<ul style="list-style-type: none"> • 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>

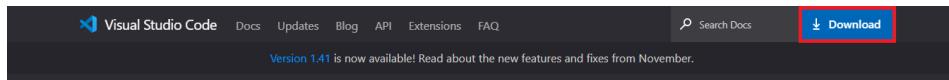
Apakah teman – teman tertarik ingin menggunakan *software visual studio code* ? Bagaimana sih cara melakukan *installasi* terhadap *software visual studio code* ?. Pada pembahasan kali ini dimana teman – teman akan belajar bagaimana cara melakukan *installasi* terhadap *software visual studio code* tersebut.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



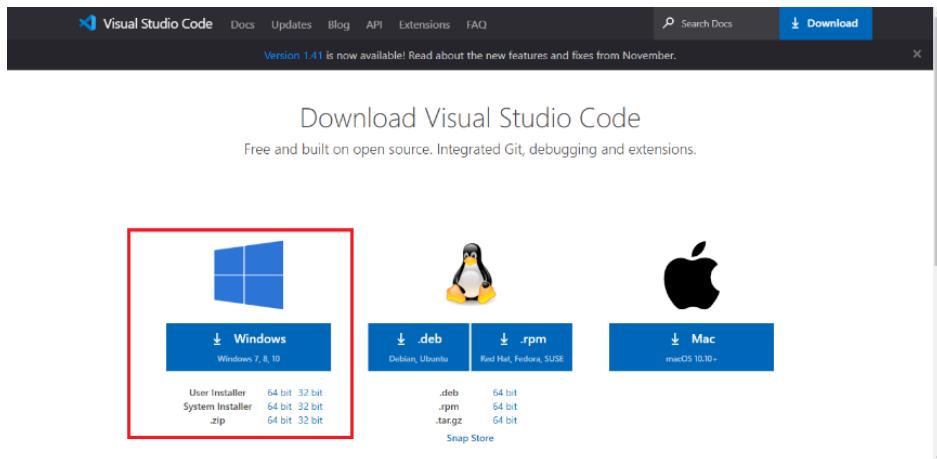
Gambar 4.12 Halaman Website Resmi VSCode

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



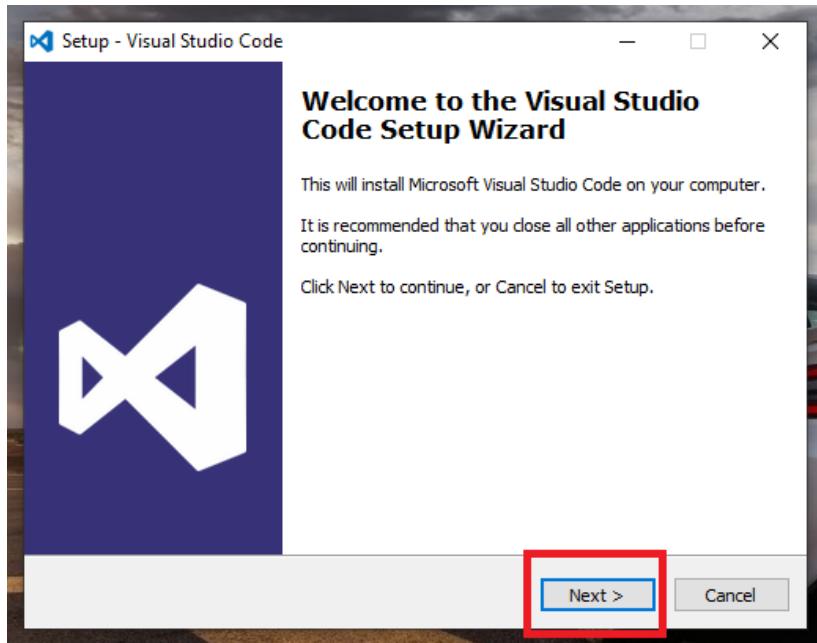
Gambar 4.13 Memilih Halaman Download VSCode

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



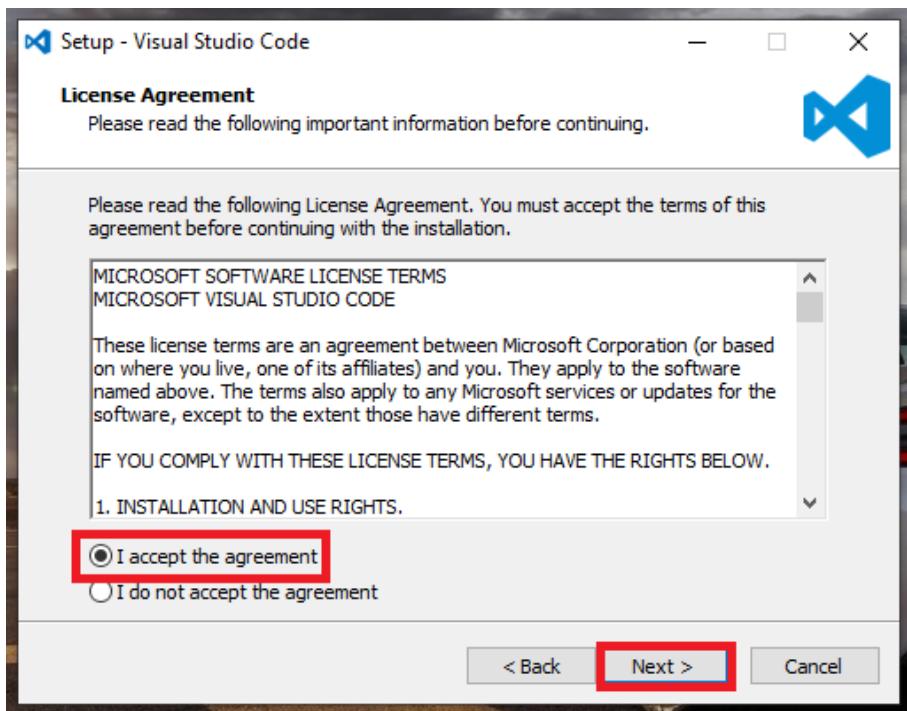
Gambar 4.14 Pemilihan OS Saat Download VSCode

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



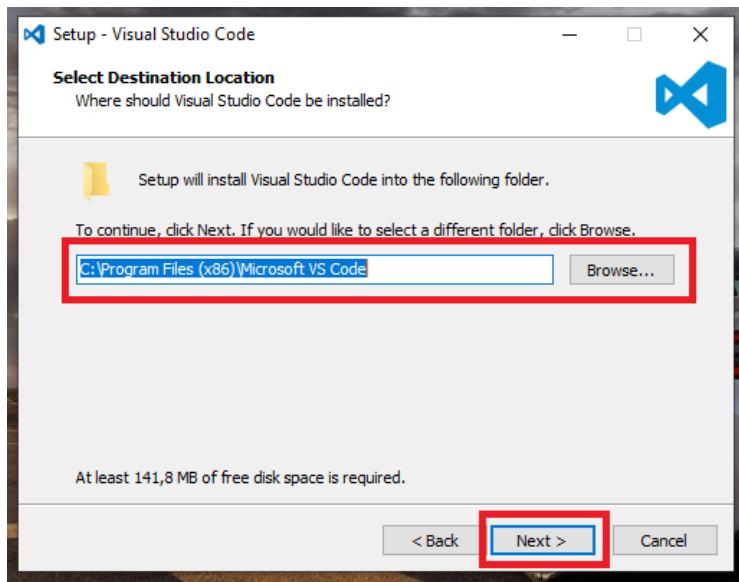
Gambar 4.15 Proses Awal Installasi VSCode

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



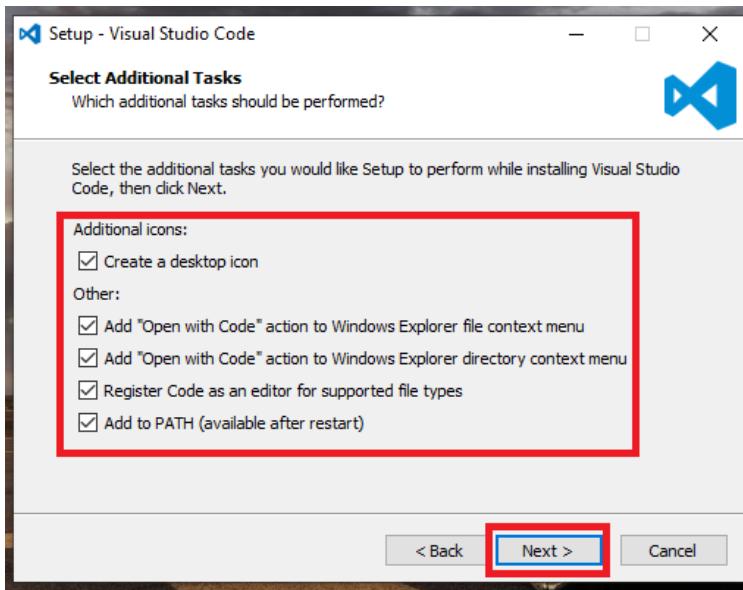
Gambar 4.16 Proses Installasi VSCode Tahap Dua

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



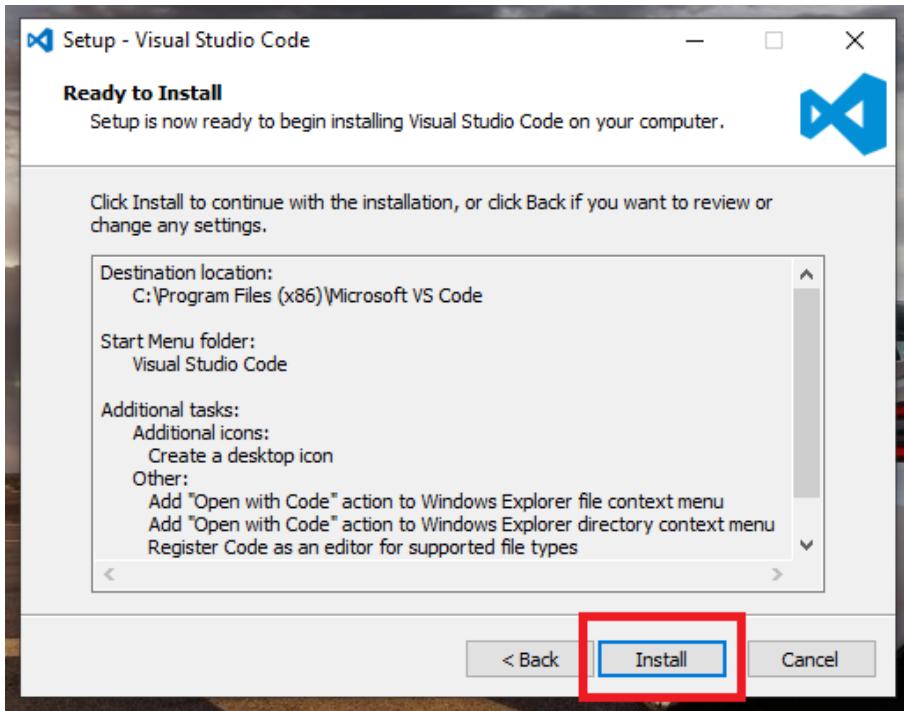
Gambar 4.17 Pemilihan Tempat Penyimpanan VSCode

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



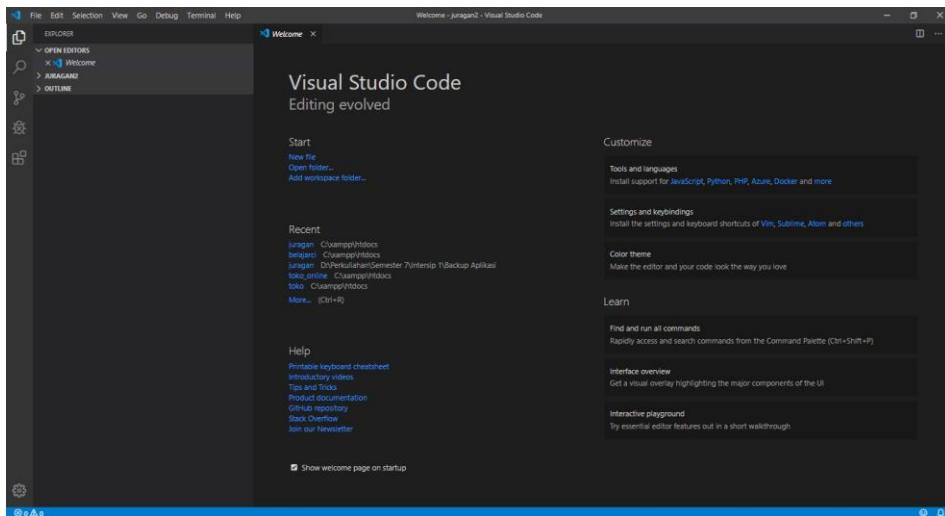
Gambar 4.18 Pemilihan Additional Tasks VSCode

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



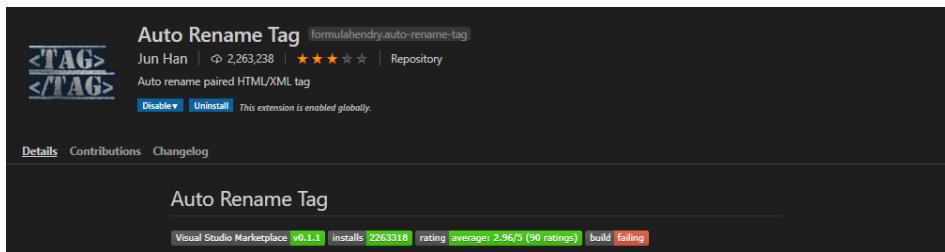
Gambar 4.19 Proses Installasi VSCode

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



Gambar 4.20 Tampilan Visual Studio Code

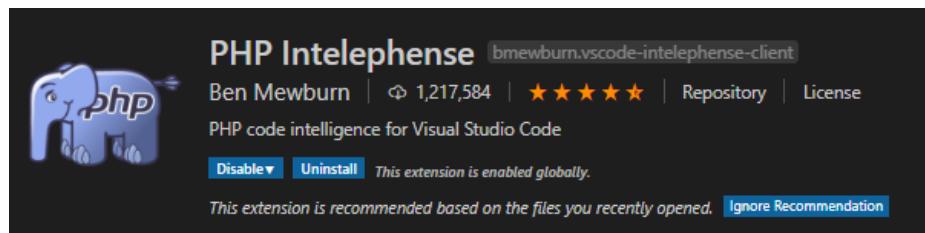
Apabila proses *install software Vscode* sudah selesai dimana teman - teman perlu melakukan *install* beberapa *extensions* untuk membantu proses pengkodean. Apa saja *extensions* tersebut ?, yang perlu teman - teman *install* adalah “*Auto Rename Tag*”.



Gambar 4.21 Extension Auto Rename Tag

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

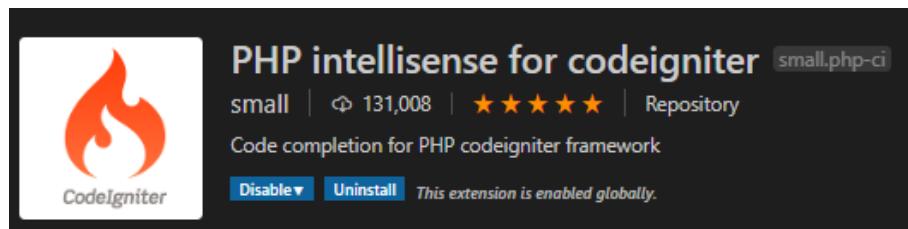
Extensions yang kedua dimana teman - teman perlu meng-*install* “*PHP Intelephense*”.



Gambar 4.22 Extension *PHP Intelephense*

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

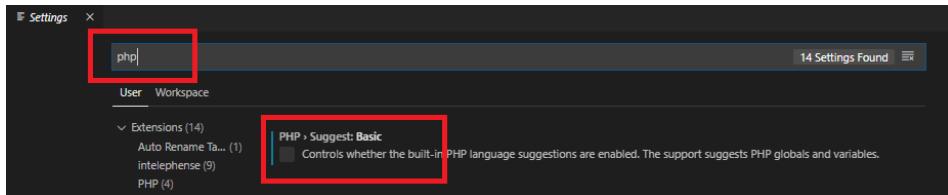
Ektensions yang terakhir dimana teman - teman perlu melakuakan *install* “*PHP Instellisense for codeigniter*”.



Gambar 4.23 Extension *PHP Intellisese For CI*

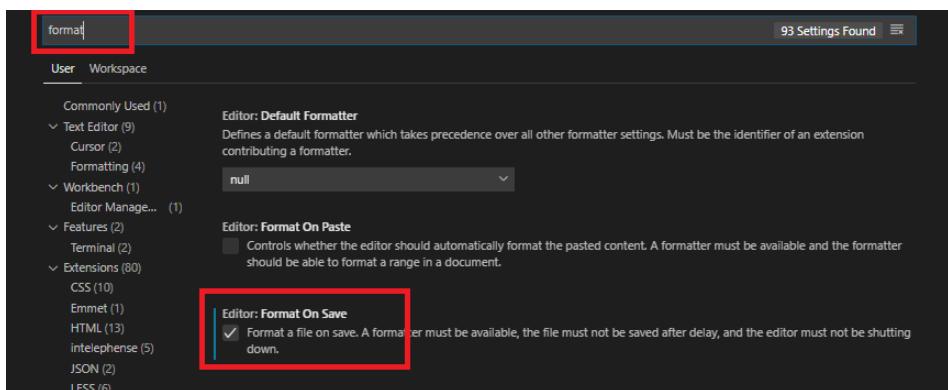
Extensions tersebut memiliki fungsi untuk memudahkan teman - teman dalam melakukan *script code* pemanggilan *PHP* yang berada pada *codeigniter*. Untuk menggunakan semua *extensions* yang telah teman - teman *install* dimana perlu dilakukan *setting* kembali pada *software*

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



Gambar 4.24 Setting Awal VSCode

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



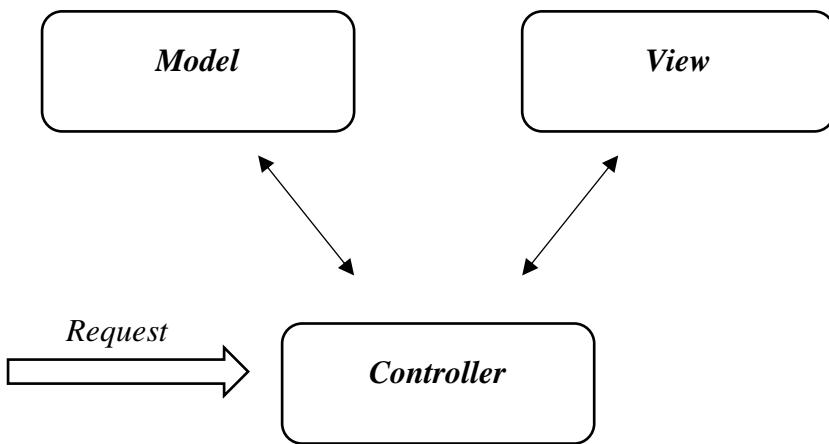
Gambar 4.25 Setting Kedua VSCode

4.3 Codeigniter

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

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

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



Gambar 4.26 Konsep MVC

4.3.1 Installasi Framework Codeigniter

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

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



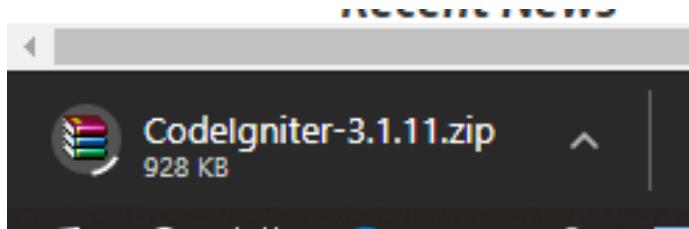
Gambar 4.27 Website Remi Codeigniter

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



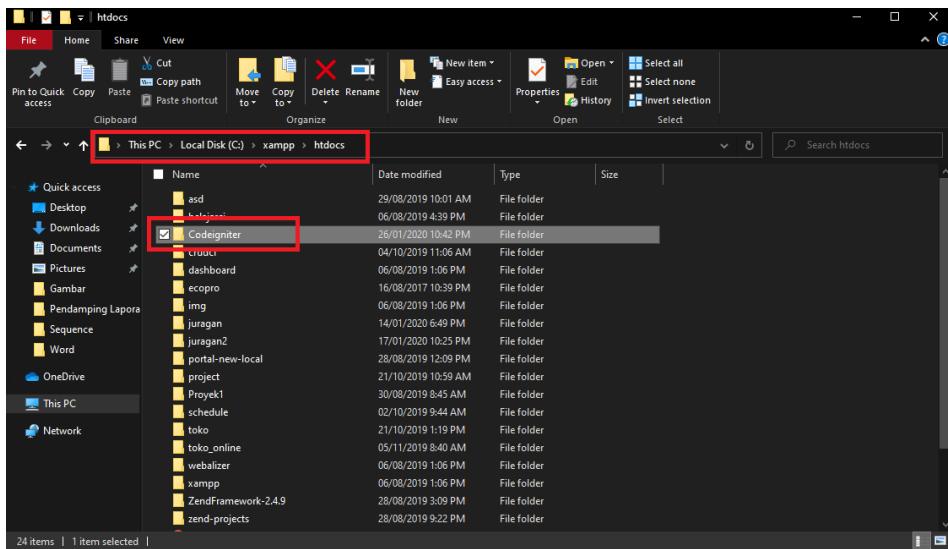
Gambar 4.28 Pilihan Download CI

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



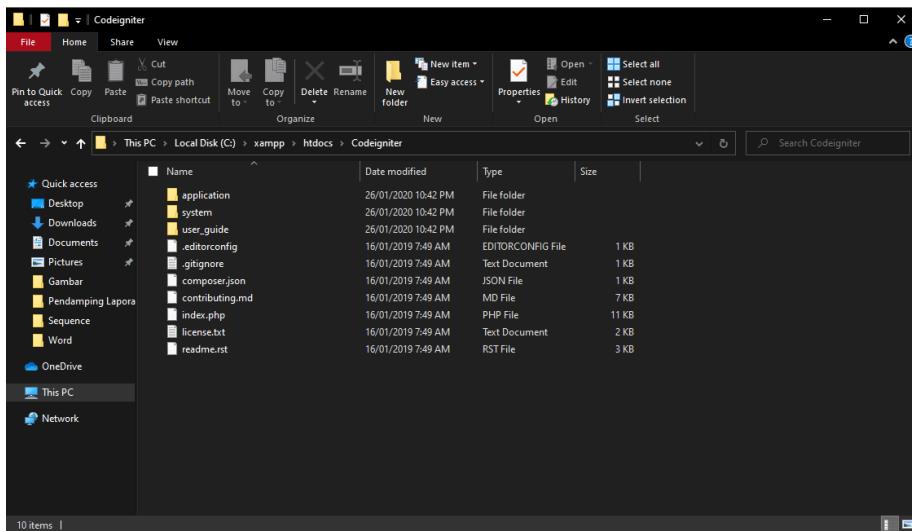
Gambar 4.29 Proses Download Codeigniter

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



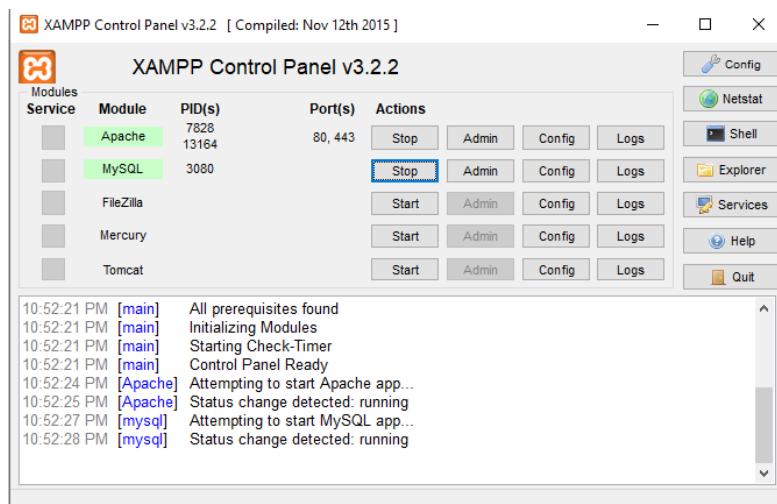
Gambar 4.30 Memindahkan File Codeigniter

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



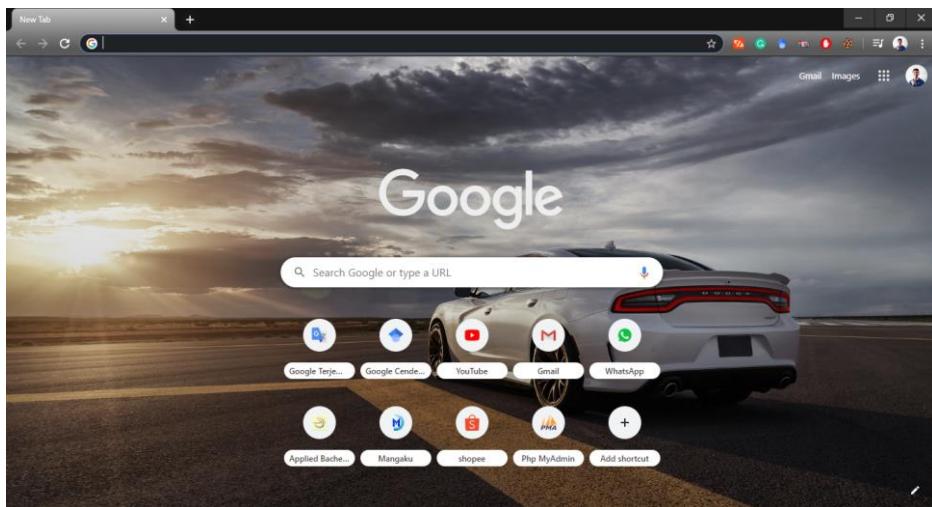
Gambar 4.31 Isi File Codeigniter

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



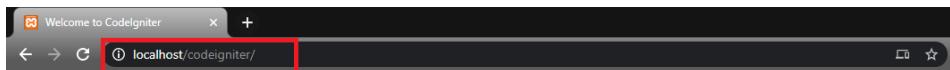
Gambar 4.32 Menjalankan Software Xampp

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



Gambar 4.33 Membuka Web Browser

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



Gambar 4.34 Menguji Framework Codeigniter

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



Gambar 4.35 Pengujian CI Berhasil

BAB V

FRAMEWORK CODEIGNITER

5.1 Sejarah Codeigniter

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

5.2 Kelebihan Codeigniter

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

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

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

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

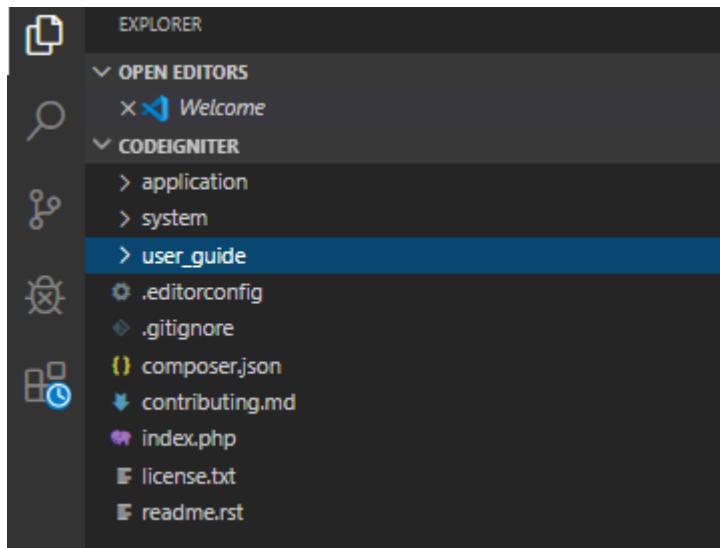


Gambar 5.1 Dokumentasi Codeigniter

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

5.3 Struktur Direktori Codeigniter

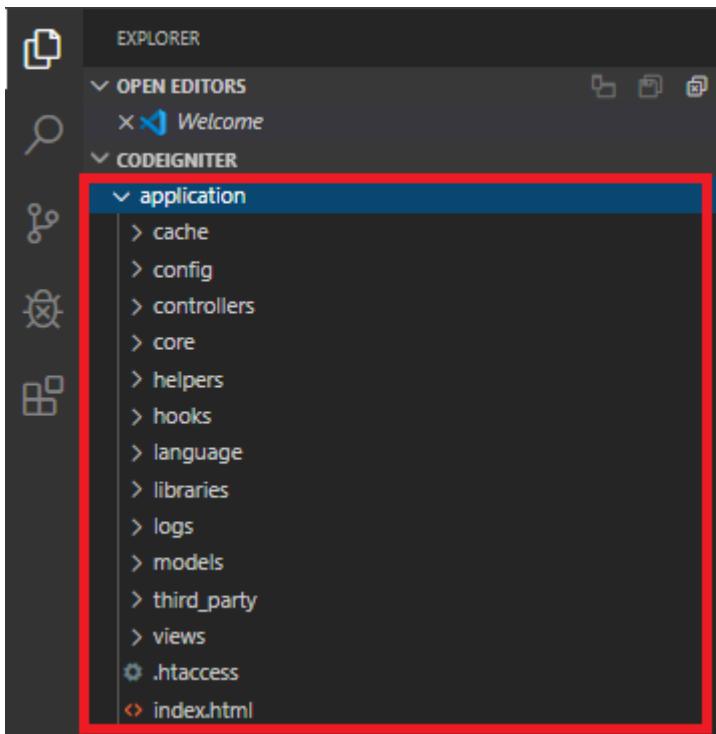
Sebelum teman – teman menggunakan *framewrok codeigniter* tersebut alangkah baiknya apabila teman – teman belajar dan memahami terlebih dahulu struktur direktori dari *framwork codeigniter* tersebut. Pada *framework Codeigniter* dimana struktur yang dimilikinya terlihat seperti pada gambar 5.2.



Gambar 5.2 Struktur Direktori CI

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

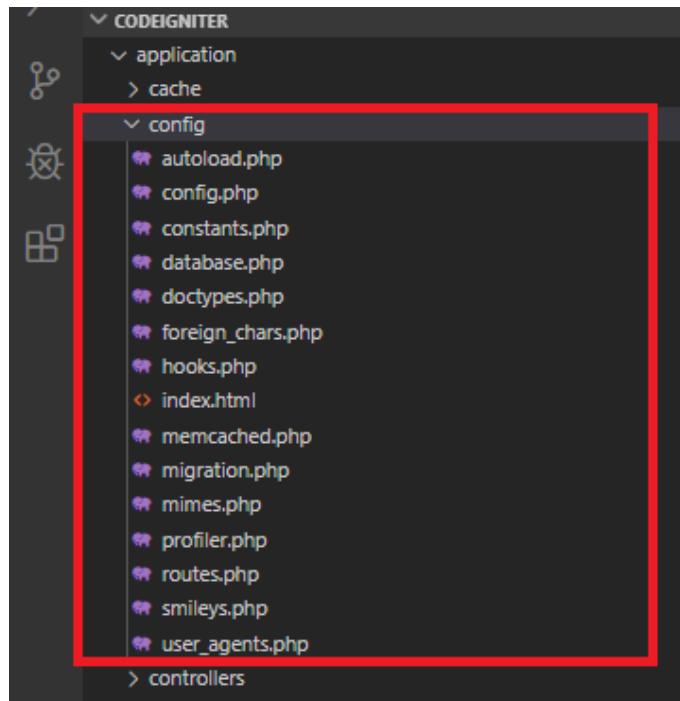
Direktori *application* merupakan direktori yang akan kita gunakan dalam pembuatan aplikasi kita nanti. Coba teman - teman klik pada direktori *application* tersebut dan lihat isinya, akan terlihat seperti pada gambar 5.3.



Gambar 5.3 Isi Dari Direktori Application

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

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



Gambar 5.4 File Dalam Folder Config

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

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

Gambar 5.5 Contoh Script Code Controller

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

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

```
$this->load->helper('nama_helper');
```

Pemanggilan dengan *load* di setiap *controller*

Gambar 5.6 Pemanggilan Fungsi Helper

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

```
<?php

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

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

Gambar 5.7 Contoh Script Code Model

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

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

Gambar 5.8 Pemanggilang Fungsi Model Pada Autoload

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

```
public function halaman_produk()
{
    $data['barang'] = $this->m_user->t_barang()->result();
    $data['user'] = $this->db->get_where('user', ['email' => $this->session->userdata('email')])->row_array();
    $data['title'] = 'Semua Produk';

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

**m_user* merupakan nama dari model yang kita buat.
**t_barang* merupakan function dari *m_user*.

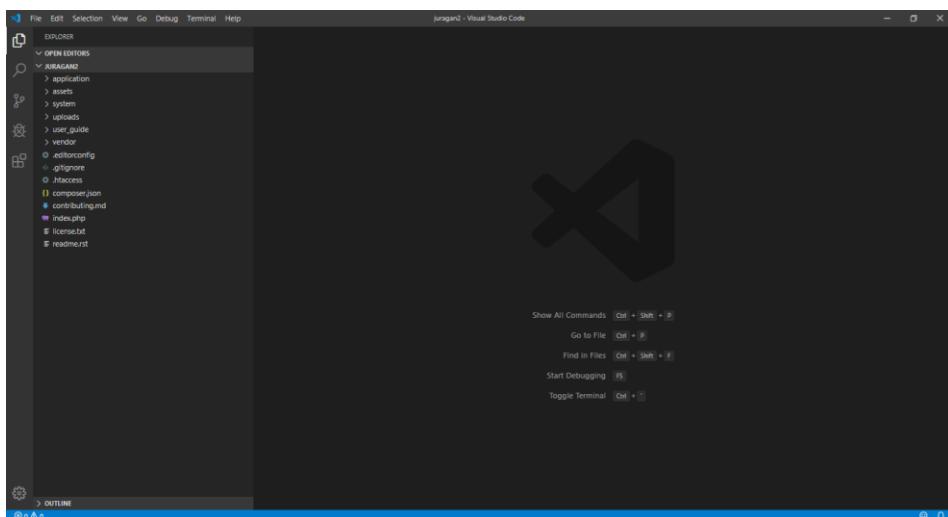
Gambar 5.9 Pemanggilan Fungsi Model Dari Controller

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

5.4 Konfigurasi Framework Codeigniter

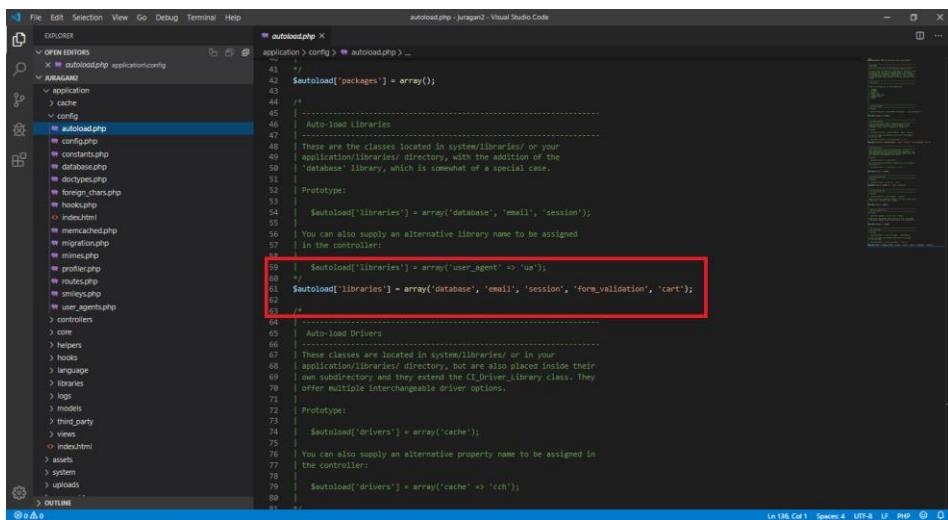
Sebelum teman – teman menggunakan *framework codeigniter* tersebut dimana teman – teman perlu melakukan konfigurasi terlebih dahulu. Bagaimana cara melakukan konfigurasi tersebut ?, berikut akan saya berikan konfigurasi apa saja yang dilakukan dalam membuat sistem JURAGAN.

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



Gambar 5.10 Folder CI JURAGAN

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



```

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

```

Gambar 5.11 Konfigurasi Libraries Sistem JURAGAN

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

dimana berfungsi sebagai pemanggilan “`base_url()`”, file yang dimana berfungsi sebagai *upload* gambar, dan *security*.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `application/config/autoload.php`, `application/controllers/Controller.php`, and `application/models/Model.php`.
- Code Editor (Right):** Displays the `autoload.php` file content. The file defines several autoloading arrays for different components like drivers, helpers, security, config, and language files.

```
File Edit Selection View Go Debug Terminal Help
autoloader.php - juniper2 - Visual Studio Code

EXPLORER
  OPEN EDITORS
    < application> config > autoload.php ...
    > application
      > cache
      > config
        > autoload.php
        > config.php
        > constants.php
        > database.php
        > doctrine.php
        > doctrine_orm.php
        > foreign_chars.php
        > helper.php
        > index.html
        > memcached.php
        > migration.php
        > mimes.php
        > profiler.php
        > routes.php
        > smiley.php
        > user_agents.php
        > controllers
          > core
          > helpers
          > hooks
          > language
          > libraries
          > logs
          > models
          > think_party
          > views
          > vendor
        > index.html
    > assets
    > system
    > uploads
  > OUTLINE

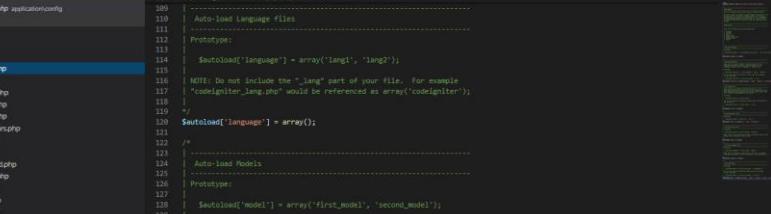
autoloader.php X
autoloader.php - juniper2 - Visual Studio Code

application > config > autoload.php ...
80
81
82 Autoload['drivers'] = array();
83
84 /**
85 * Auto-load Helper Files
86 */
87 /**
88 * Prototype:
89 */
90
91 Autoload['helper'] = array('url', 'file');
92 /**
93 * Prototype:
94 */
95 Autoload['helper'] = array('url1', 'file', 'security');
96 /**
97 * Prototype:
98 */
99 /**
100 * Prototype:
101 */
102 Autoload['config'] = array('config1', 'config2');
103 /**
104 * NOTE: This item is intended for use ONLY if you have created custom
105 * config files. Otherwise, leave it blank.
106 */
107 /**
108 * Prototype:
109 */
110 Autoload['language'] = array('lang1', 'lang2');
111 /**
112 * Prototype:
113 */
114 Autoload['language'] = array('codeigniter_lang.php' => 'codeigniter_lang');
115 /**
116 * NOTE: Do not include the "lang" part of your file. For example
117 * "codeigniter_lang.php" would be referenced as array('codeigniter_lang');
118 */
119 /**
120 * Prototype:
121 */
122 Autoload['language'] = array();

In the code editor, the line Autoload['language'] = array('codeigniter_lang.php' => 'codeigniter_lang'); is highlighted with a red box, indicating it is the target of the current refactoring operation.
```

Gambar 5.12 Konfigurasi Fungsi Helpers Sistem JURAGAN

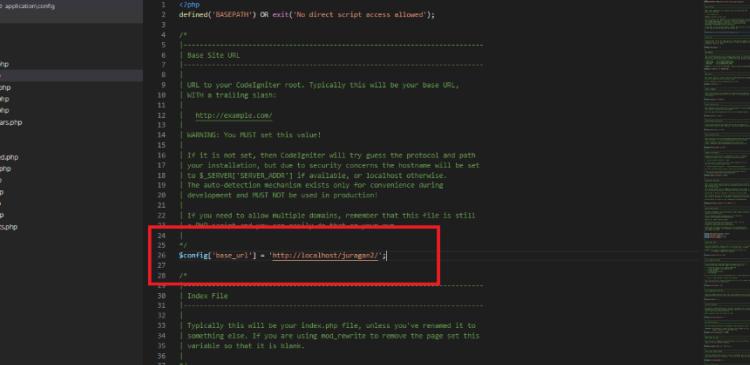
Langkah berikutnya masih tetap pada file *autolod.php*. Apabila teman – teman ingin menggunakan fungsi *model* dimana teman – teman perlu melakukan konfigurasi terlebih dahulu pada file *autoload.php*. Silahkan teman – teman lakukan scroll hingga baris 135. Konfigurasi fungsi *model* pada sistem JURAGAN 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*.



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

Gambar 5.13 Konfigurasi Models Pada Sistem JURAGAN

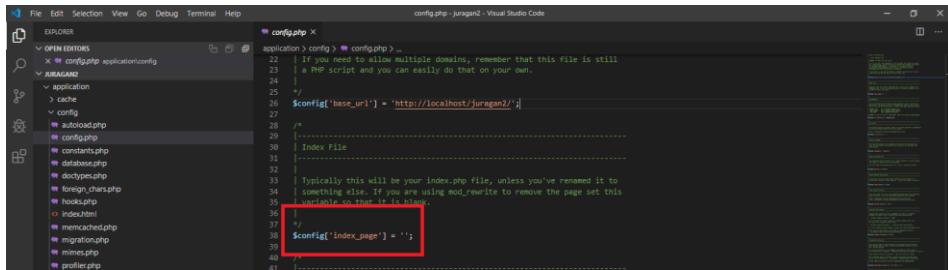
Langkah berikutnya teman – teman masih tetap berada pada direktori yang sama, buka file config.php, dan jangan lupa lakukan save pada file autolod.php. Pada file config.php dimana teman – teman perlu melakukan konfigurasi pada bagian base_url nya. Isikan dengan link website teman – teman. Pada sistem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



```
config.php >
application <-- config.php > ...
1 | <?php
2 | defined('BASEPATH') OR exit('No direct script access allowed');
3 |
4 |
5 |
6 | Base Site URL
7 |
8 |
9 | URL to your CodeIgniter root. Typically this will be your base URL,
10 | WITH a trailing slash!
11 |
12 | http://example.com/
13 |
14 | WARNING: You MUST set this value!
15 |
16 | If it is not set, then Codeigniter will try guess the protocol and path
17 | to your installation, but due to security concerns the hostname will be set
18 | to $_SERVER['SERVER_NAME'] if available, or localhost otherwise.
19 |
20 | The auto-detection mechanism exists only for convenience during
21 | development and MUST NOT be used in production!
22 |
23 | If you need to allow multiple domains, remember that this file is still
24 |
25 |/*
26 |$config['base_url'] = "http://localhost/juniperage/?";
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 |
37 |
38 | $config['index_page'] = '';
39 |
40 */
41 |
```

Gambar 5.14 Konfigurasi File Config Sistem JURAGAN

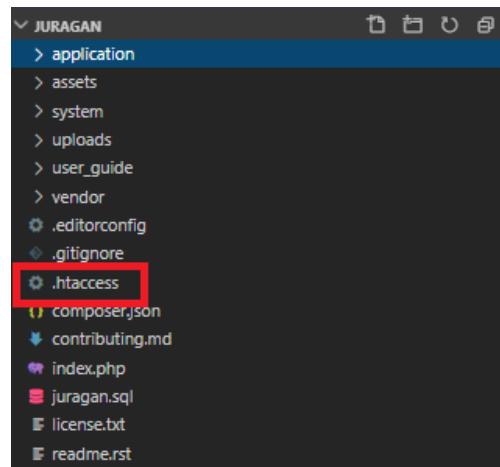
Langkah berikutnya masih dalam *file* yang sama yaitu *config.php* dimana teman – teman perlu menghapus isi pada bagian *index_page* pada baris 38. Hal ini berfungsi agar teman – teman tidak perlu mengetikkan kembali *index.php*.



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



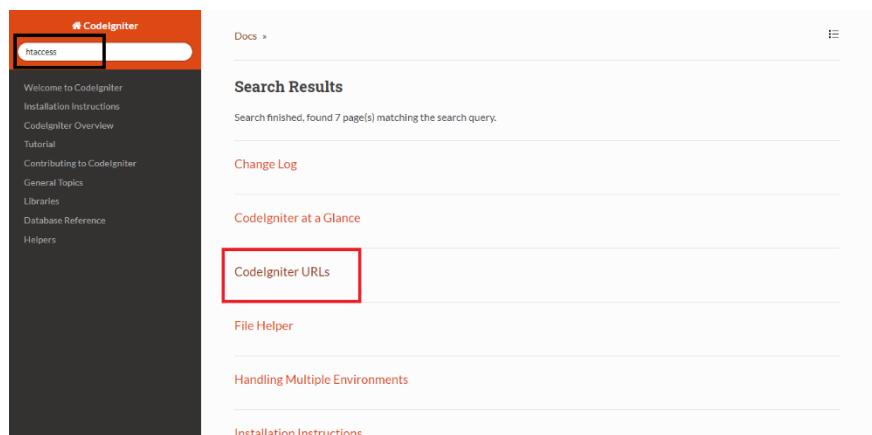
Gambar 5.16 Membuat File Baru *.htaccess*

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



Gambar 5.17 Membuka Dokumentasi CI

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



Gambar 5.18 Mencari Dokumentasi htaccess

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

Removing the index.php file

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

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

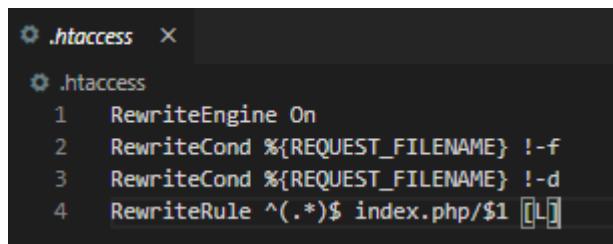
If your Apache server has `mod_rewrite` enabled, you can easily remove this file by using a `.htaccess` file with some simple rules. Here is an example of such a file, using the “negative” method in which everything is redirected except the specified items:

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

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

Gambar 5.19 Script Code htaccess

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

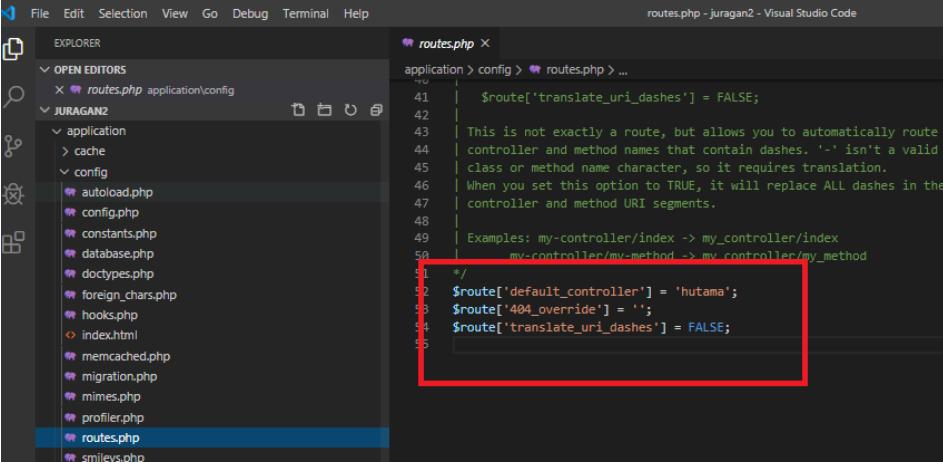


```
❶ .htaccess ✘
❷ .htaccess
❸ 1 RewriteEngine On
❹ 2 RewriteCond %{REQUEST_FILENAME} !-f
❺ 3 RewriteCond %{REQUEST_FILENAME} !-d
❻ 4 RewriteRule ^(.*)$ index.php/$1 [L]
```

Gambar 5.20 Mengisikan File .htaccess

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

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



The screenshot shows the Visual Studio Code interface with the file "routes.php" open. The code editor displays the following PHP code:

```
application > config > routes.php ...  
40 | $route['translate_uri_dashes'] = FALSE;  
41 |  
42 | This is not exactly a route, but allows you to automatically route  
43 | controller and method names that contain dashes. '-' isn't a valid  
44 | class or method name character, so it requires translation.  
45 | When you set this option to TRUE, it will replace ALL dashes in the  
46 | controller and method URI segments.  
47 |  
48 | Examples: my-controller/index -> my_controller/index  
49 |           mv-controller/mv-method -> my_controller/my_method  
50 |  
51 */  
52 $route['default_controller'] = 'hutama';  
53 $route['404_override'] = '';  
54 $route['translate_uri_dashes'] = FALSE;  
55
```

A red rectangular box highlights the line of code: `$route['default_controller'] = 'hutama';`.

Gambar 5.21 Konfigurasi Default Controller Sistem JURAGAN

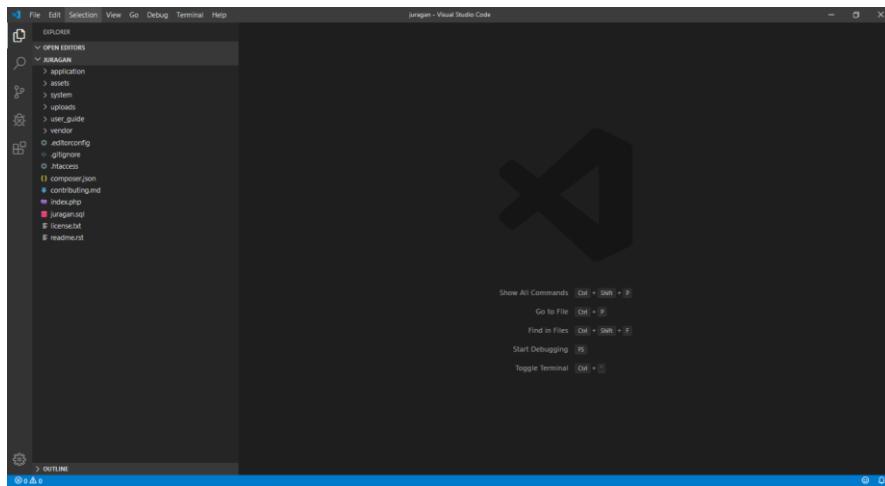
Langkah selanjutnya dimana teman – teman perlu melakukan *setting* terhadap *database* yang akan digunakan. Masih pada direktori yang sama, silahkan teman – teman buka file *database.php*. Konfigurasi *database* yang digunakan oleh sistem JURAGAN adalah sebagai berikut.

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

Gambar 5.22 Konfigurasi Database Sistem JURAGAN

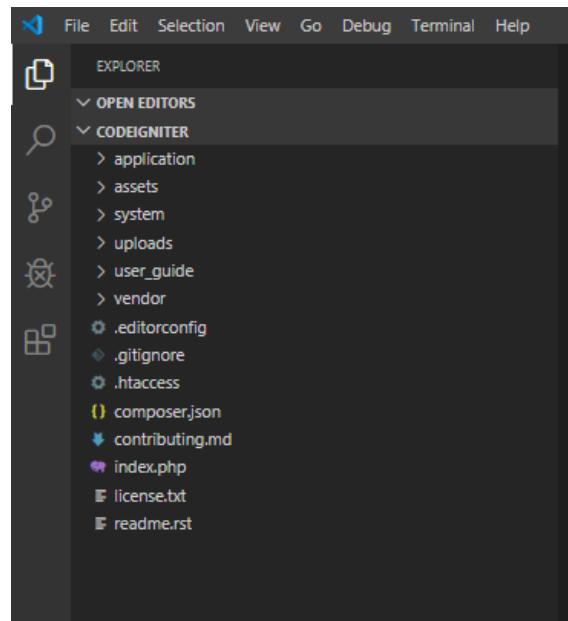
5.5 Membuat CRUD Dengan Codeigniter

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



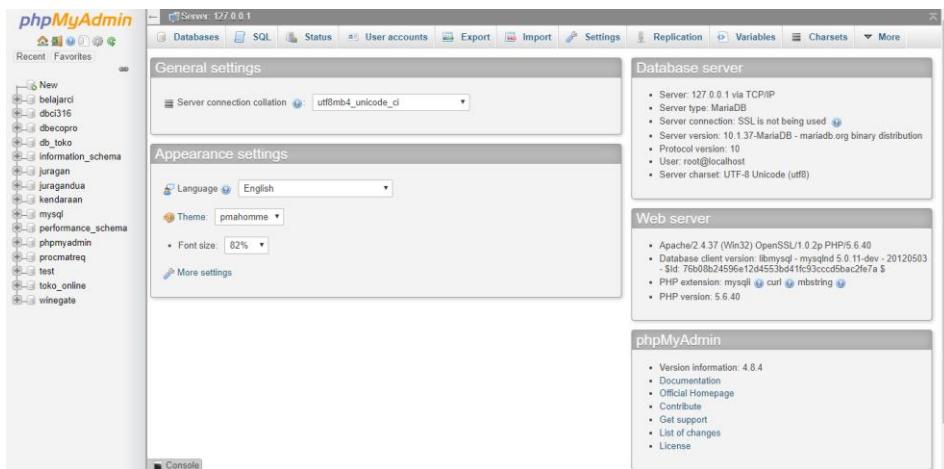
Gambar 5.23 Mempersiapkan Software VSCode

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



Gambar 5.24 Open Folder CI Pada VSCode

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



Gambar 5.25 Halaman PHP MyAdmin

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

A screenshot of a 'Create database' dialog box. It has a 'Create database' button with a plus icon. Below it is a text input field containing 'crud_brg'. To the right of the input field is a dropdown menu showing 'latin1_swedish_ci'. Next to the dropdown is a 'Create' button with a checkmark icon.

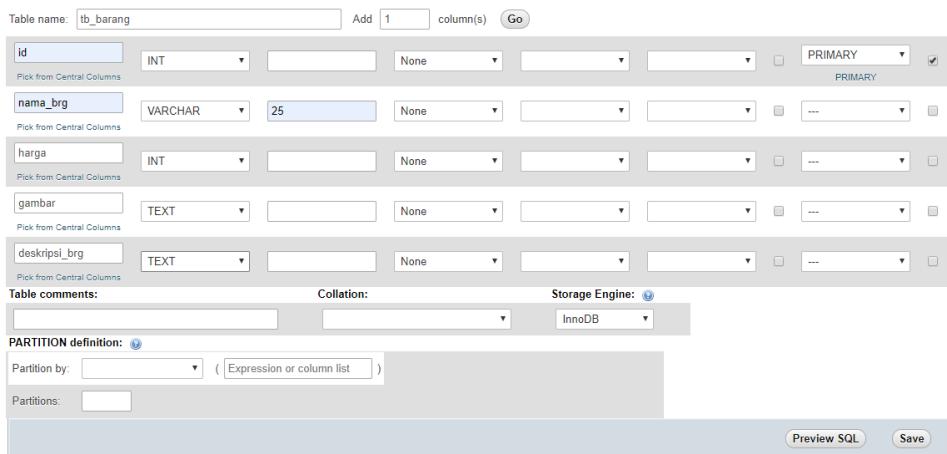
Gambar 5.26 Membuat Nama Database

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



Gambar 5.27 Membuat Tabel tb_barang

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



Gambar 5.28 Membuat Struktur Tabel tb_barang

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

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

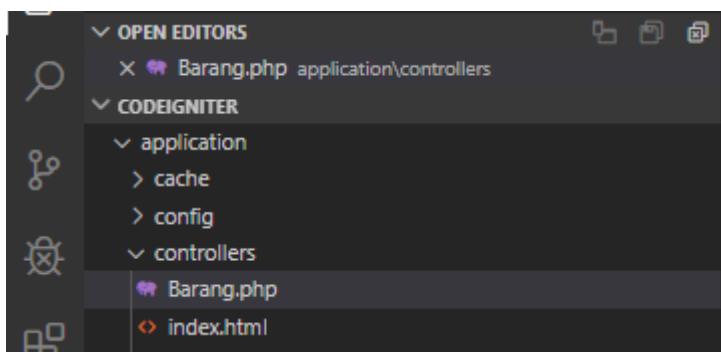
Gambar 5.29 Struktur Tabel tb_barang

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

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

Source Code 5.1 SQL Create Table

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



Gambar 5.30 Membuat Controllers Barang.php

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

```
<?php

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

class Barang extends CI_Controller
{



}
```

Source Code 5.2 Code Pertama Controllers

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

```
<?php

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

class Barang extends CI_Controller
{

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

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

}
```

Source Code 5.3 Pemanggilan Fungsi Models Dan Form_validation

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

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

Source Code 5.4 Membuat Function Index

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

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

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

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

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

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

Source Code 5.5 Membuat Function Tambah Pada Controllers

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

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

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

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

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

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

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

Source Code 5.6 Membuat Function Edit Pada Controllers

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

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

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

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

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

Source Code 5.7 Membuat Function Delete Pada Controllers

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

```
<?php

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

class Barang extends CI_Controller
{
```

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

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

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

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

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

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

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

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

```

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

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

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

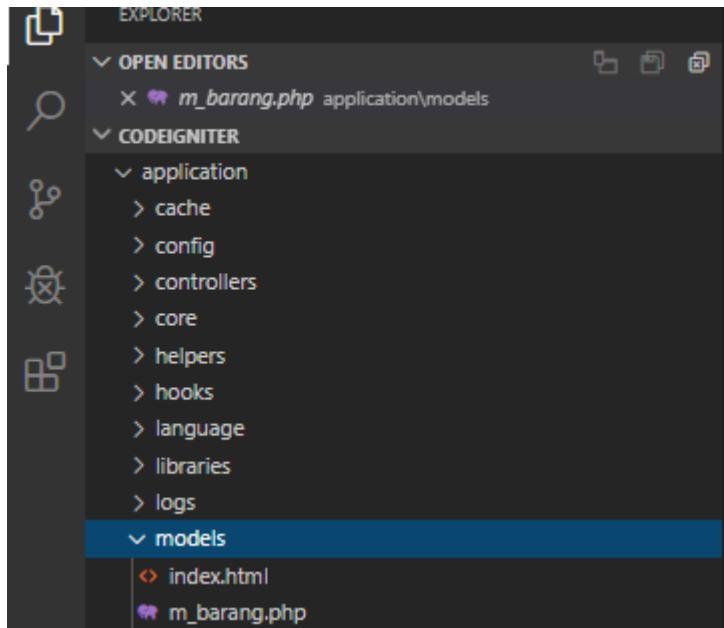
public function delete($id)
{
    if (!isset($id)) show_404();

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

```

Source Code 5.8 Code Full Controllers Barang.php

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



Gambar 5.31 Membuat File m_barang Pada Models

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

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

Source Code 5.9 Code Awal Pada *m_barang.php*

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

```

private $_table = "tb_barang";

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

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

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

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

```

Source Code 5.10 Mendeskripsikan Field Pada Models

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

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

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

Source Code 5.11 Function Fungsi Read

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

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

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

Source Code 5.12 Fungsi Read Berdasarkan Id

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

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

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

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

Source Code 5.13 Membuat Function Simpan Pada Models

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

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

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

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

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

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

Source Code 5.14 Membuat Fungsi Update Pada Models

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

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

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

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

Source Code 5.15 Membuat Fungsi Delete Pada Models

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

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

```
<?php

class M_barang extends CI_Model
{

    private $_table = "tb_barang";

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

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

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

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

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

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

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

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

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

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

```

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

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

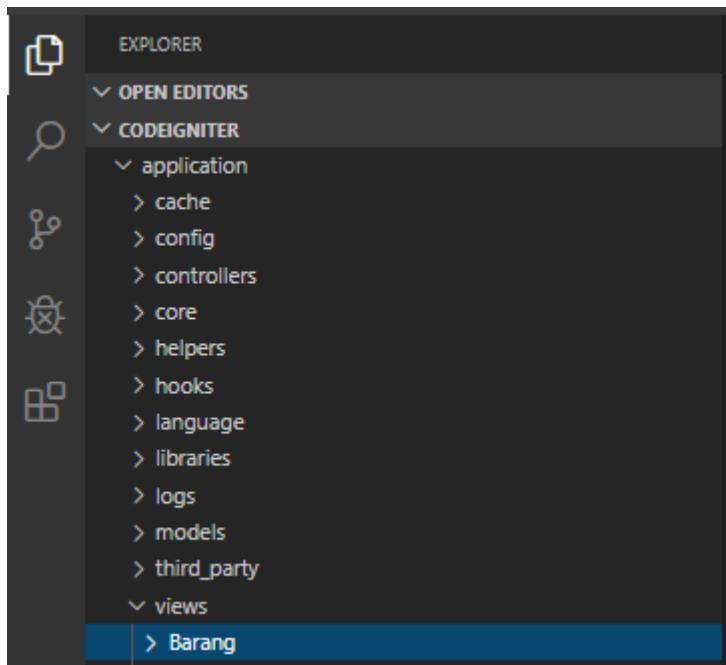
public function delete($id)
{
    return $this->db->delete($this->_table,
        array("id" => $id));
}

}

```

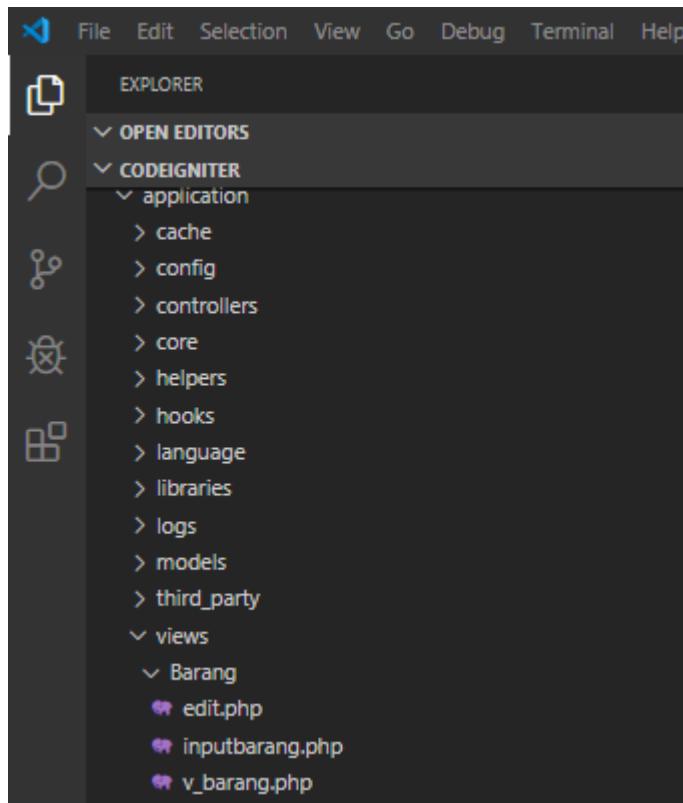
Source Code 5.16 Code Full Models File m_barang.php

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



Gambar 5.32 Membuat Folder Barang Pada Views

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



Gambar 5.33 Membuat File PHP Pada Folder Barang

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

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

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

Source Code 5.17 Memanggil Header Pada View v_barang

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

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

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

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

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

Source Code 5.18 Memanggil Template Topbar Dan Sidebar Pada View

v_barang

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

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

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

    <div class="container-fluid">

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

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

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

Source Code 5.19 Membuat Tabel Pada View v_barang

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

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

```

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

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

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

            <td>
                
            </td>

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

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

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

```

Source Code 5.20 Membuat Table Body Pada View v_barang

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

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

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

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

</body>

</html>
```

Source Code 5.21 Membuat Footer Pada View v_barang

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

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

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

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

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

    <div class="container-fluid">

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

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

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

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

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

<td>

</td>

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

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

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

```

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

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

</body>

</html>

```

Source Code 5.22 Code Full File v_barang

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

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

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

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

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

Source Code 5.23 Memanggil Flashdata Pada View inputbarang

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

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

<div class="card-header">

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

</div>
```

Source Code 5.24 Membuat Button Kembali Pada Form inputbarang

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

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

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

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

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

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

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

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

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

Source Code 5.25 Membuat Textbox Pada View inputbarang

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

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

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

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

```
<div id="wrapper">

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

    <div class="container-fluid">

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

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

</div>
<?php endif; ?>

<div class="card mb-3">

    <div class="card-header">

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

    </div>

    <div class="card-body">

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

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

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

```

</div>

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

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

</body>

</html>

```

Source Code 5.26 Code Full View File inputbarang.php

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

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

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

Source Code 5.27 Membuat Flashdata Pada View Edit

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

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

Source Code 5.28 Membuat Button Kembali Pada View Edit

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

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

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

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

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

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

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

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

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

```

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

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

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

<?php endforeach; ?>

</form>
</div>

```

Source Code 5.29 Membuat Form Input Dengan Value Pada Edit

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

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

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

    <body id="page-top">

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

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

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

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

                <div class="card-body">

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

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

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

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

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

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

```

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

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

    <?php endforeach; ?>

    </form>
    </div>

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

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

    </body>

    </html>

```

Source Code 5.30 Code Full File Edit.php

BAB VI

E-COMMERCE DAN CODEIGNITER

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

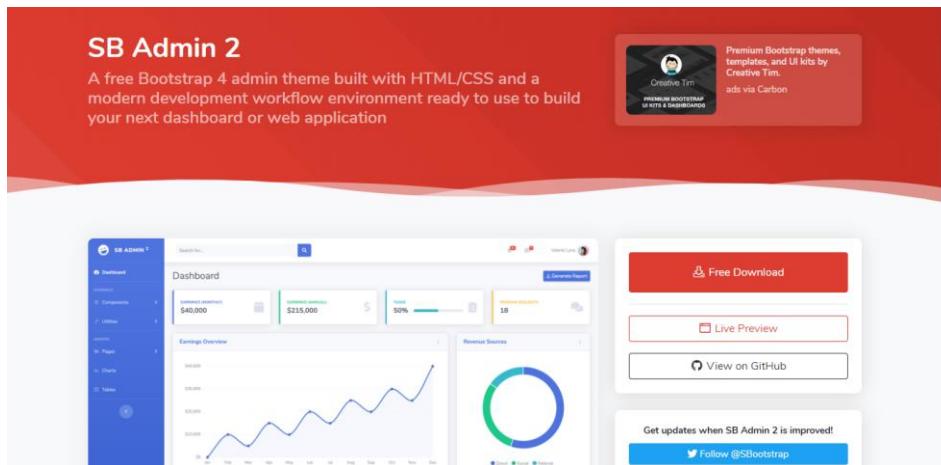
Apakah teman – teman tertarik belajar membangun sebuah *website* dengan konsep *e-commerce* ?, banyak sekali *website* – *website* yang bertebaran dan dibuat dengan *framework codeigniter*. Sistem JURAGAN adalah salah satunya yang menerapkan konsep *e-commerce* dan dibangun dengan menggunakan *framework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

6.1 Mengubungkan Bootstrap Dengan CI

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

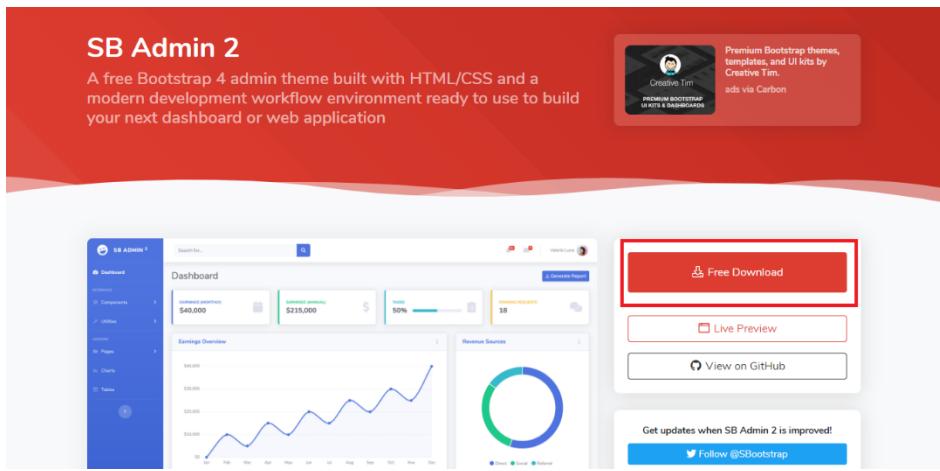
website dengan konsep *e-commerce*. Dengan tampilan yang menarik dan mudah digunakan dimana *user* akan terlihat lebih nyaman dan betah untuk mengakses dan menggunakan *website* tersebut. Dengan adanya *bootstrap* tersebut dimana dapat mempermudah dan mempercepat para developers dalam melakukan pengembangan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

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



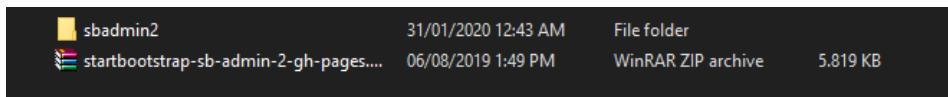
Gambar 6.1 Halaman Start Bootstrap SB ADMIN 2

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



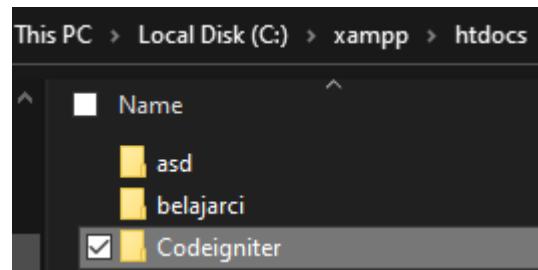
Gambar 6.2 Menu Download Template Bootstrap SB ADMIN 2

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



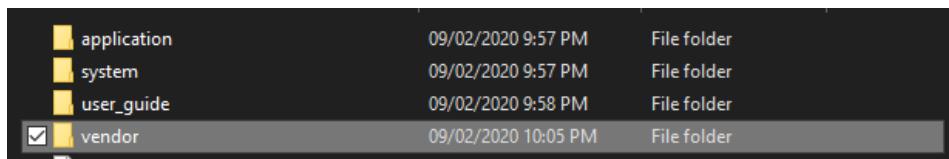
Gambar 6.3 Meng Extract File Zip Boostrap

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



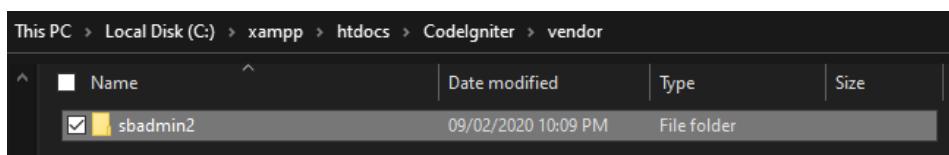
Gambar 6.4 File Codeigniter Pada Direktori Xampp/htdocs

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



Gambar 6.5 Membuat Folder Vendor

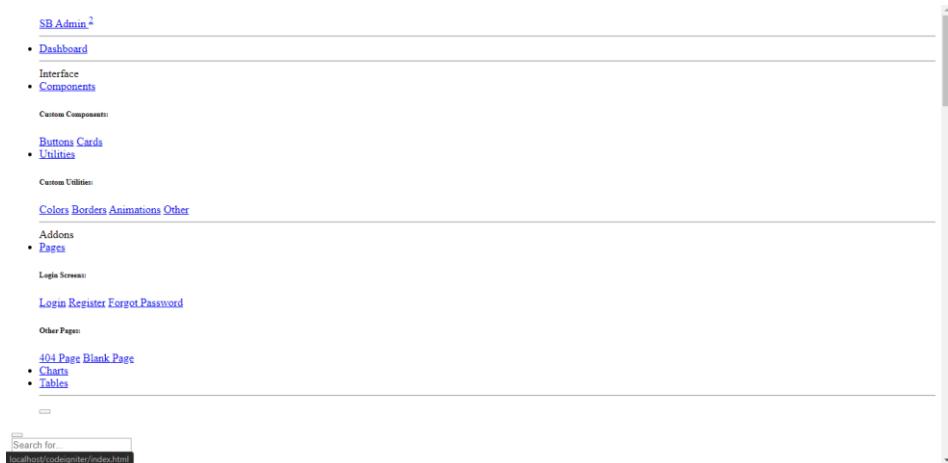
Folder *vendor* tersebut berfungsi untuk menampung referensi *template bootstrap* SB ADMIN 2 tersebut, silahkan teman – teman pindahkan *file* SB ADMIN 2 tersebut kedalam *folder vendor* seperti pada gambar 6.6.



Gambar 6.6 Memindahkan File SB ADMIN 2 Ke Dalam Vendor

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

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



Gambar 6.7 Tampilan View Welcome_message.php

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

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

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

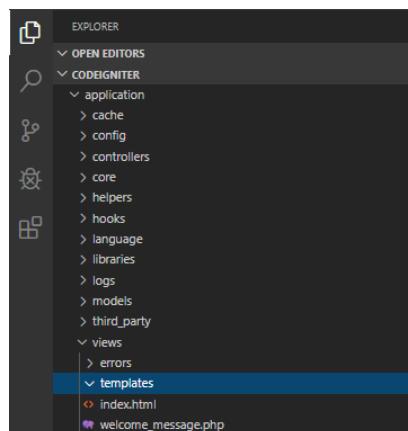
Gambar 6.8 Membuat Folder Assets

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

This PC > Local Disk (C:) > xampp > htdocs > CodeIgniter > assets				
	Name	Date modified	Type	Size
<input checked="" type="checkbox"/>	css	09/02/2020 10:30 PM	File folder	
<input checked="" type="checkbox"/>	img	09/02/2020 10:30 PM	File folder	
<input checked="" type="checkbox"/>	js	09/02/2020 10:30 PM	File folder	
<input checked="" type="checkbox"/>	vendor	09/02/2020 10:30 PM	File folder	

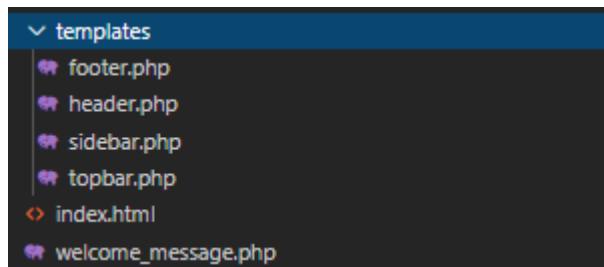
Gambar 6.9 Mengisi Folder Assets

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



Gambar 6.10 Membuat Folder templates Pada Views

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



Gambar 6.11 Membuat 4 File PHP Pada Folder Templates

Setelah teman – teman sudah selesai mempersiapkan *folder – folder* untuk menampung *templates bootstrap* SB ADMIN 2 tersebut langkah selanjutnya kita akan belajar bagaimana caranya menghubungkan *css* tersebut dengan *framework codeigniter*. *Template* yang pertama adalah bagian *header*, yang dimana merupakan bagian awal pada suatu sistem dan biasanya berisi dengan judul atau *tittle* dari sistem tersebut. Silahkan teman – teman buka kembali *file welcome_message.php* yang sudah diberikan *script code blank.html*, mari kita pisahkan terlebih dahulu bagian – bagian *template* tersebut. Silahkan teman – teman *cut* baris 1 – 26 pada *file welcome_message.php* tersebut yang merupakan 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 diisi oleh beberapa kolom yang dimana mengandung fungsi *link navigasi*. Silahkan teman – teman *cut code* baris 28 - 140 pada file *welcome_message.php* yang merupakan bagian *sidebar* kedalam file *sidebar.php*.

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

Gambar 6.13 Isi File Sidebar.php

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

```

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

```

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

```

Gambar 6.15 Isi File Footer.php

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

```

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

```

Source Code 6.1 Merubah Controllers Welcome

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

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

<head>

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

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

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

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

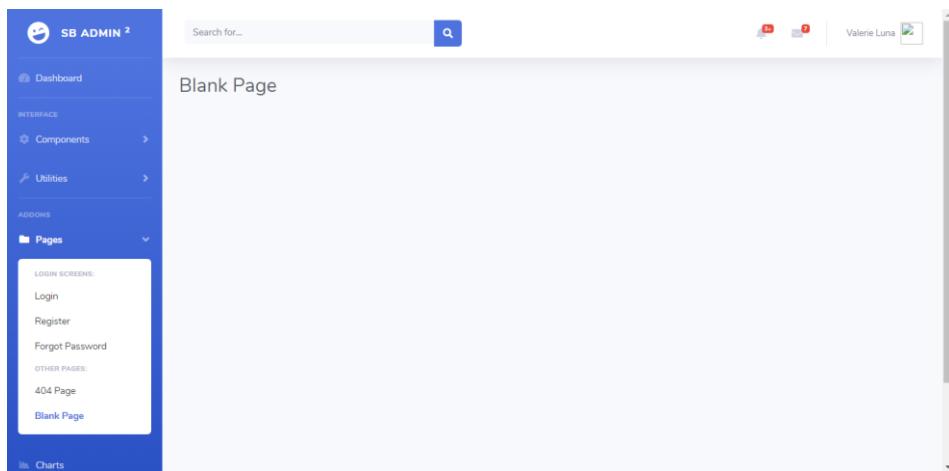
```
</head>

<body id="page-top">

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

Source Code 6.2 Merubah File Header.php Pada Controllers Welcome

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



Gambar 6.16 Perubahan Sidebar Dan Topbar

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

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

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

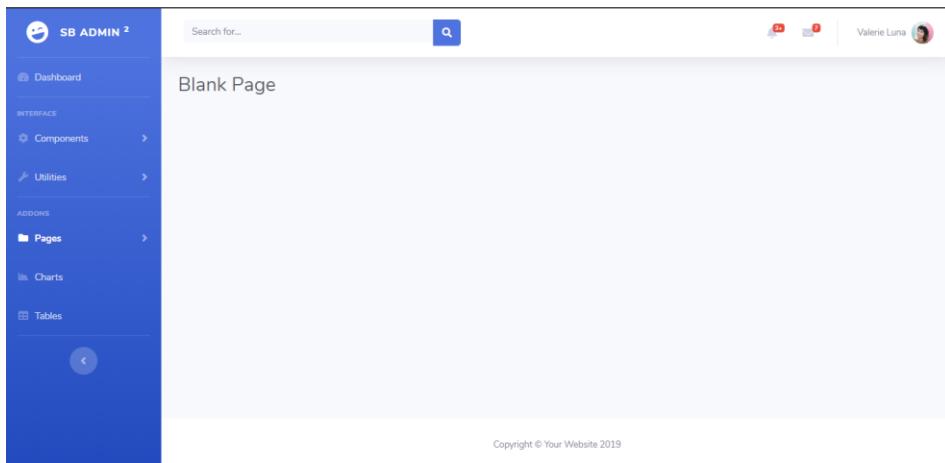
&lt;!-- Custom scripts for all pages--&gt;
&lt;script src="<?= base_url(); ?&gt;assets/js/sb-admin-2.min.js"&gt;&lt;/script&gt;

&lt;/body&gt;

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

Source Code 6.3 Merubah Code Footer.php

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



Gambar 6.17 Menerapkan Templates Bootstrap Terhadap CI Berhasil

5.2 Fungsi Form Validation

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

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

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

The screenshot shows a login page with a blue header and footer. The main content area has a white background. At the top center, it says "Halaman Login". Below that is a red rectangular box containing the text "Silaikan login terlebih dahulu !". Below the message are two input fields: one for email with placeholder text "Ketikkan Email Anda ..." and one for password with placeholder text "Ketikkan Password Anda ...". There is also a checkbox labeled "Ingat Akun". A large blue button at the bottom is labeled "Login". At the very bottom of the page, there are two links in blue text: "Daftarkan Toko Anda ? Click Disini !" and "Belum Punya Akun ? Click Disini !", followed by "Lupa Password ? Click Disini! | Kembali Ke Halaman Utama, Click Disini !".

Gambar 6.18 Contoh Form Validation Pada Sistem JURAGAN

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

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

Source Code 6.4 Memanggil Library Form Validation

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

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

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

Source Code 6.5 Penerapan Form Validation Pada Field Name

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

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

Gambar 6.19 Praktek Form Validation Pada Field Name

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

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

Source Code 6.6 Penerapan Form Validation Pada Field Email

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

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

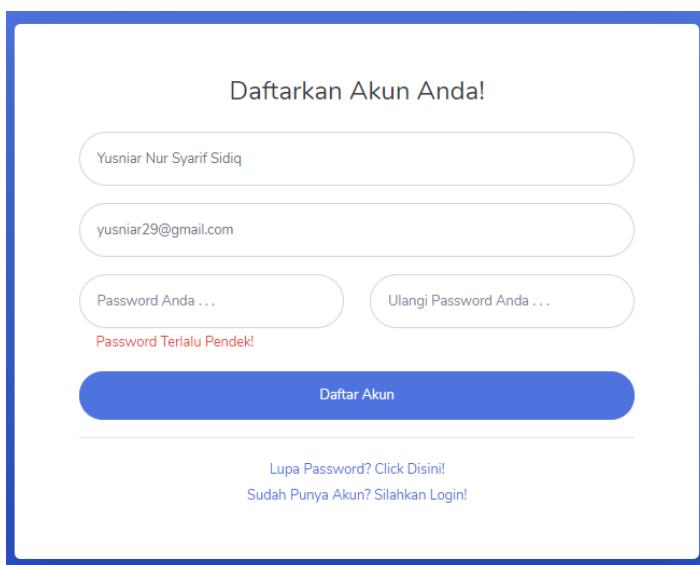
Gambar 6.20 Praktek Form Validation Terhadap Field Email

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

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

Source Code 6.7 Penerapan Form Validation Pada Field Password

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



Gambar 6.21 Praktek Form Validation Pada Field Password

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

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

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

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

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

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

    } else {

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

```
$this->input->post('name', true)),  
'email' => htmlspecialchars($email),  
'password' => password_hash(  
    $this->input->post('password1'),  
    PASSWORD_DEFAULT)  
];  
  
$this->db->insert('user', $data);  
  
$this->session->set_flashdata('message',  
    '<div class="alert alert-success" role="alert">  
        Selamat akun anda telah terbuat!</div>');  
  
redirect('auth');  
}  
  
}
```

Source Code 6.8 Contoh Pemanggilan Fungsi Form Validation

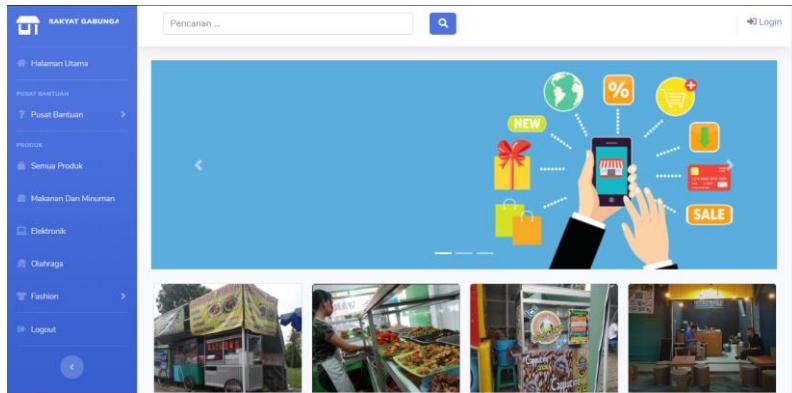
5.3 Membuat Fitur Keranjang Belanja

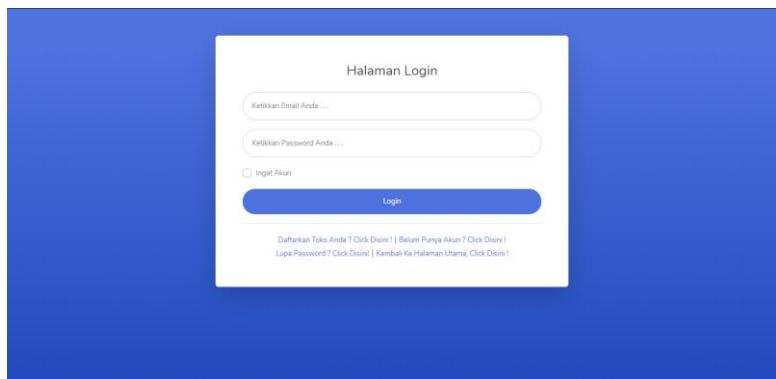
Shopping cart atau bisa juga disebut dengan keranjang belanja yang dimana biasanya sering ditemukan pada *website – website e-commerce*. Keranjang belanja sangatlah berperan penting pada sistem *e-commerce* yang dimana memiliki *fungsi* untuk menampung *item – item* atau produk – produk yang akan *user*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

Dalam *framework codeigniter* dimana dokumentasi yang diberikan sangatlah lengkap.

BAB VII

PANDUAN PENGGUNAAN APLIKASI JURAGAN





Daftarkan Akun Anda!

Nama Lengkap ...

Email Anda ...

Password Anda ...

Ulangi Password Anda ...

Daftar Akun

Lupa Password? [Click Disini!](#)
Sudah Punya Akun? [Silahkan Login!](#)

Daftarkan Toko Anda!

Nama Toko Anda ...

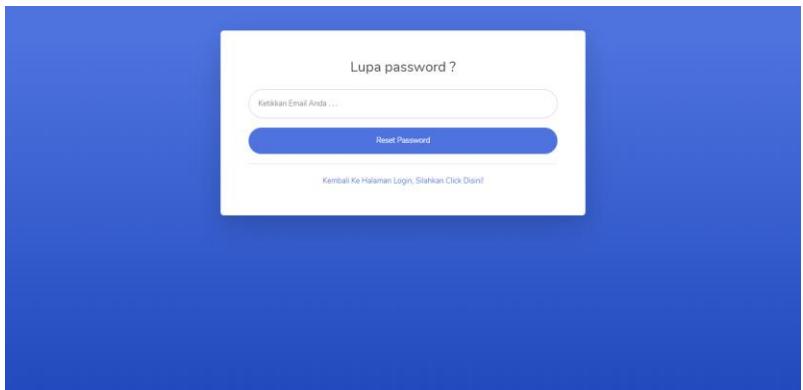
Email Toko Anda ...

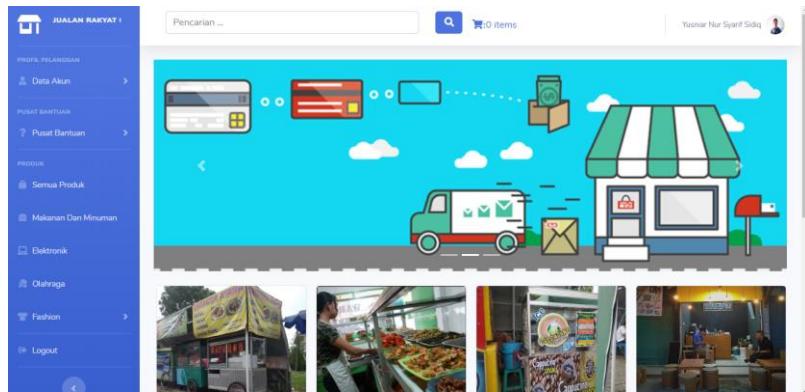
Password Anda ... Ulangi Password Anda ...

Anda telah menyetujui [Syarat dan Ketentuan](#) serta [Kebijakan Privasi JURAGAN](#).

[Daftar Toko](#)

[Lupa Password? Click Disini!](#)
[Sudah Punya Akun? Silahkan Login!](#)





JALAN RAKYAT GA

Pencarian ... 0 items

Yusniar Nur Syarif Sidiq

Data Profil

 Yusniar Nur Syarif Sidiq
yusniar33@gmail.com
Jl.A.H.Nasution Asrama Yon Zipur 9 Rt 05 Rw 02 No. 247
Kec. Cinambo Kel. Pakemitan Kota Bandung 40294
Terdaftar sejak 17 January 2020

[Edit Profil](#) [Ganti Password](#)

Copyright © Internship | Yusniar Nur Syarif Sidiq | 1.16.4.089 | Poitsenik Pos Indonesia

Logout

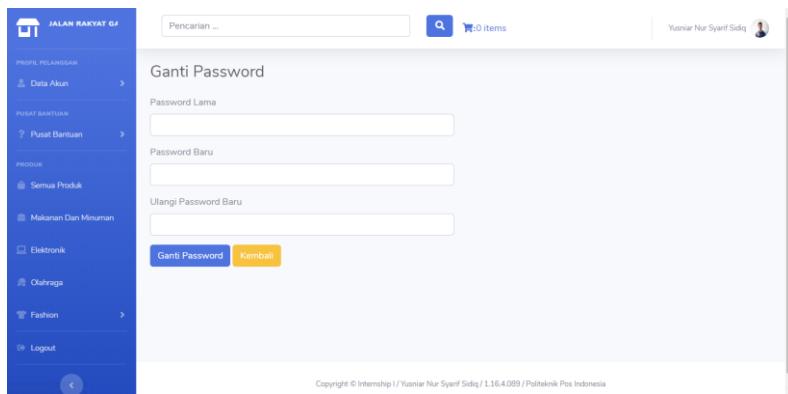
Screenshot of a user profile editing page titled "Edit Profil".

The left sidebar shows a navigation menu with categories like PRODUSEN, PEMBELIANSAR, Pusat Bantuan, Produk, Makanan Dan Minuman, Elektronik, Olahraga, Fashion, and Logout.

The main form fields include:

- Email: yusniar33@gmail.com
- Nama Lengkap: Yusniar Nur Syaef Sidiq
- Alamat: JLA.H.Nasution Asrama Yon Zipur 9 Rt 05 Rw 02 No. 247
Kec. Cinambo *Nama Kecamatan.
Kel. Pakemitan *Nama Kelurahan.
Kota Bandung *Nama Kota.
40294 *Kode Pos.
- Foto Profil: A small thumbnail image of a person smiling, with a file input field labeled "Pilih file" and a "Browse" button.

A search bar at the top right contains "Pencarian ...". The top right corner shows the user's name "Yusniar Nur Syaef Sidiq" and a profile picture.



KYAT SABUNGAN

PENGELUARAN
Data Akun

PROSES
Pusat Bantuan

PRODUK
Semua Produk

Makanan Dan Minuman

Eletronik

Olahraga

Fashion

Logout

Pencarian ...

0 items

Yusnir Nur Syarif Sidiq

Bantuan Aplikasi JURAGAN

* Panduan Mendaftar Toko
Anda ingin bergabung bersama kami dengan menjadi mitra JURAGAN ? ikutiilah tata caranya sebagai berikut:

Halaman Login

Kontak Email Anda:

Kontak Password Anda:

Ingat Akun

Log In

Akses sistem JURAGAN dan daftarkan akun toko anda dengan memilih menu daftar toko seperti pada gambar diatas.

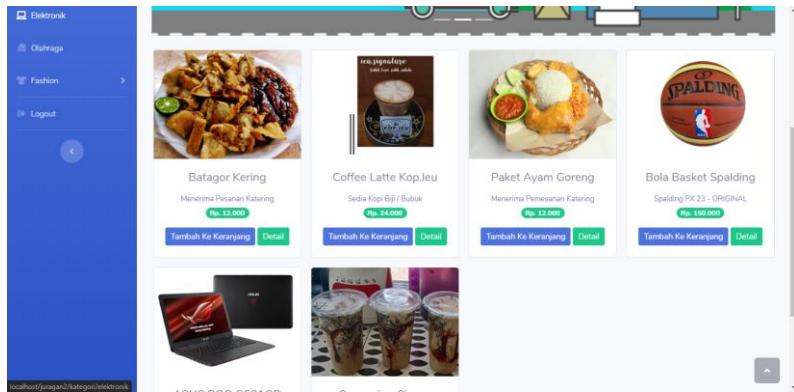
The screenshot shows the JURAGAN application interface. On the left is a sidebar with a logo and navigation links: PROSES PELAKUAN (Data Akun), BANTUAN (Pusat Bantuan), PRODUK (Semua Produk, Makanan Dan Minuman, Elektronik, Olahraga, Fashion), and LogOut. At the top right are a search bar, a cart icon (0 items), and a user profile for Yuniar Nur Syarif Sidiq.

Bantuan Aplikasi JURAGAN

* Panduan Mendaftar Akun
Bagaimana Cara Mendaftarkan Akun Saya Pada Sistem JURAGAN ? Silahkan ikuti Langkah Berikut :

Akses sistem JURAGAN dan daftarkan akun anda dengan memilih menu daftar akun seperti pada gambar diatas.

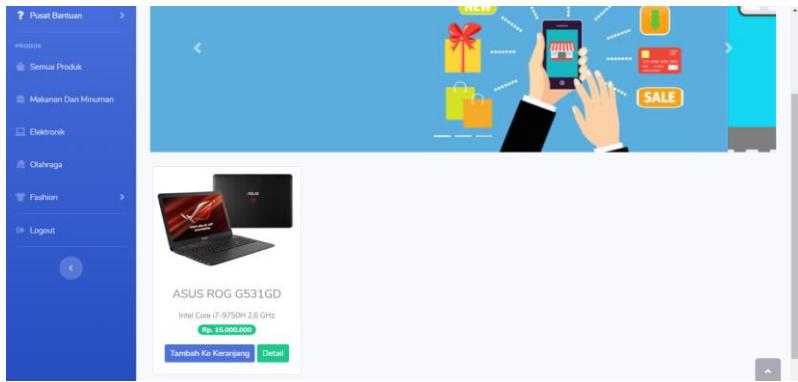
The central part of the screen displays a modal window titled "Halaman Login". It contains fields for "Email" and "Password", a "Forgot Account" link, and a "Login" button. Below the button is a link "Daftar Akun Anda" (highlighted with a red box) and "Lupa Password?" (also highlighted with a red box).

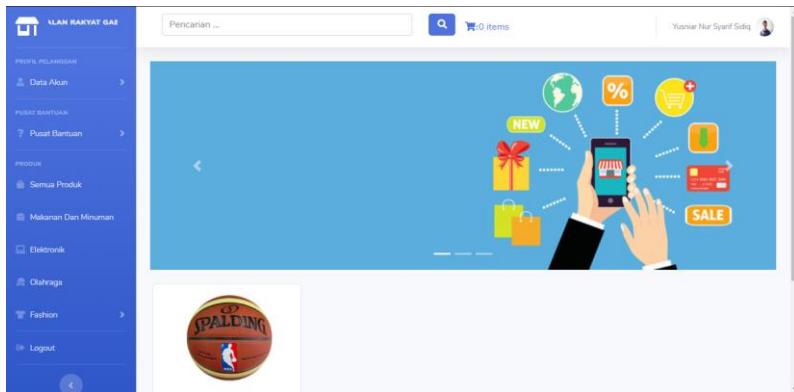


The screenshot displays a mobile application interface for an online store. On the left, a vertical sidebar menu is visible with the following categories: 'Semua Produk' (All Products), 'Makanan Dan Minuman' (Food and Beverage), 'Elektronik' (Electronics), 'Olahraga' (Sports), 'Fashion' (Fashion), and 'Logout'. Below the sidebar, there are four product items listed:

- Batagor Kering**
Menerima Pesanan Catering
Rp. 12.000
[Tambah Ke Keranjang](#) [Detail](#)
- Coffee Latte Kop.Ieu**
Sedia Kopi Biji / Bubuk
Rp. 24.000
[Tambah Ke Keranjang](#) [Detail](#)
- Paket Ayam Goreng**
Menerima Pemesanan Catering
Rp. 12.000
[Tambah Ke Keranjang](#) [Detail](#)
- Cappuccino Cincau**
Harap Cek Catatan Toko Kami
Rp. 12.000
[Tambah Ke Keranjang](#) [Detail](#)

At the top of the main content area, there is a decorative banner featuring a hand holding a smartphone, surrounded by shopping bags and a 'SALE' sign. The footer of the screen contains the copyright notice: 'Copyright © Internship I / Yuniar Nur Syarif Sisq | 1.16.4.089 / Politeknik Pos Indonesia'.





The screenshot shows a user interface for an e-commerce platform. On the left is a sidebar with navigation links: PROFIL PELANGGAN (Data Akun), PUSAT BANTUAN (Pusat Bantuan), PRODUK (Semua Produk, Makanan Dan Minuman, Elektronik, Olahraga, Fashion), and Logut. At the top right are search fields for 'Pencarian ...' and a magnifying glass icon, followed by a shopping cart icon showing '1 items'. A user profile for 'Yusnir Nur Syarif Sidiq' is also present.

Keranjang Saya

No.	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Bola Basket Spalding	1	Rp. 150.000	Rp. 150.000	

RP. 150.000

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

Copyright © Internship I / Yusnir Nur Syarif Sidiq | 1.16.4.009 | Politeknik Pos Indonesia

Halaman Pembayaran

Total Belanja Anda: Rp. 150.000

Nama Lengkap
Yusniar Nur Syarif Sidiq

Alamat Lengkap
JL.A.H.Nasution Asrama Yon Zipur 9 Rt 05 Rw 02 No. 247

*Nama Kecamatan.
Kec. Cinambo

*Nama Kelurahan.
Kel. Pakemitan

*Nama Kota/Kabupaten.
Kota Bandung

*Kode Pos.
40294

No. Telpoin
No. Telpoin Anda

Nama Pesanan Jumlah Pesanan
Bola Basket Spalding 1



UNGAN ONLINE

Search for...

Batagor Ayana

The screenshot shows the 'Edit Profil Toko' (Edit Shop Profile) page of the GAN ONLINE platform. On the left, there is a sidebar with navigation links: PROFILE & TOKO (Data Toko selected), PRODUK TOKO (Produk), PEMAHAMU APLIKASI (Bantuan), and LOGOUT. The main content area has a search bar at the top right. The profile information for 'Batagor Ayana' is displayed, including the email (yutniams29@gmail.com) and name (Batagor Ayana). Below this, the address is listed as 'Komplek Ujung Berung Indah Rt 02 Rw 01' with dropdown fields for 'Kec. Ujung Berung', 'Kel. Pasir Jati', 'Kota Bandung', and '40616'. There is also a field for 'No. WhatsApp' containing the number 6285723438131. At the bottom, a note in bold capital letters reads: '*MOHO DI BACA DENGAN SEKSAMA* Batagor Ayana merupakan usaha di bidang kuliner dimana produk yang dijual berupa batagor kering, batagor kuah, dan baso tahu. Adapun rules dan Batagor Ayana sebagai berikut : a). Di rekomendasikan menggunakan jasa pengiriman GRAB.

Search for...

Batagor Ayana

Edit Profil Toko

Email
yutniams29@gmail.com

Nama Lengkap
Batagor Ayana

Alamat
Komplek Ujung Berung Indah Rt 02 Rw 01
Kec. Ujung Berung
Nama Kecamatan.
Kel. Pasir Jati
Nama Kelurahan.
Kota Bandung
Nama Kota.
40616
Nama Pos.
*Kode Pos.

No. WhatsApp
6285723438131
*Gunakan format 62xxxxxxxxxx.

Catatan Toko
MOHO DI BACA DENGAN SEKSAMA Batagor Ayana merupakan usaha di bidang kuliner dimana produk yang dijual berupa batagor kering, batagor kuah, dan baso tahu. Adapun rules dan Batagor Ayana sebagai berikut : a). Di rekomendasikan menggunakan jasa pengiriman GRAB.

The screenshot displays a mobile application interface for a store. At the top, there is a header bar with a search field containing "Search for..." and a magnifying glass icon. To the right of the search field is a user profile section labeled "Batagor Ayana" with a small profile picture.

The main content area is titled "Halaman Toko" (Store Page). Below the title, there is a button labeled "+ Tambah Produk" (Add Product). A table lists a single product:

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

At the bottom of the screen, there is a footer note: "Copyright © Internship I / Yutnir Nur Syarif Sidiq / L16.A.089 / Politeknik Pos Indonesia".

TOKO SAKYAT GABUNGA

Search for..

Halaman Toko

+ Tambah Produk

Nama Produk

1 Batagor Kering

Informasi Singkat

Alamat Toko

Komplek Ujung Berung Indah Rt 02 Rw 01

*Nama Kecamatan.
Kec. Ujung Berung

*Nama Kelurahan.
Kel. Pasir Jati

*Nama Kota/Kabupaten.
Kota Bandung

*Kode Pos.
40616

Harga Stok Aksi

12000 10

Detail Produk

FORM INPUT PRODUK

Email
yusniarmss29@gmail.com

Nama Produk

Aksi

Batagor Ayana

PROFILE TOKO

Data Toko

PRODUK TOKO

Produk

PENGALAMAN APLIKASI

Bantuan

Logout

Detail Toko

Nama Produk Batagor Kering

Detail Produk Pada tau gak makanan BATAGOR ??? Batagor itu loh yang isinya ada tahu dan somay yang sudah di goreng dan ditaburi dengan bumbu kacang ditambah dengan timun. Siapa nih pecinta Batagor makanan tradisional Khas Kota Bandung ??? Bagi kalian yang tinggal di Kompleks Ujungberung Indah mampir yuk ke Batagor Ayana, alamat bisa cek di profil toko kami yahh hehehee. Bagi kalian yang diluar Kompleks Ujungberung Indah tenang saja, kalian dapat mencicipinya kok dengan cara melakukan order dengan JURAGAN. Komposisi dari Batagor Ayana ini terdiri dari Tepung Tapioka, Tahu, Pangsit, Ikan Tenggiri, Garam, dan Bumbu Rahasia Khas Batagor Ayana :) Selain Batagor Kering kami juga memiliki produk - produk lainnya loh, yuk cek produk kami :). Di jamin 100 % Halal....

Jenis Produk Makanan Dan Minuman

Stok 10

Kontak Penjual +626285723438131

Search for...

Batagor Ayana

Edit Produk

Nama Produk

Informasi Singkat

Alamat Toko

*Nama Kecamatan.
Kec. Ujung Berung

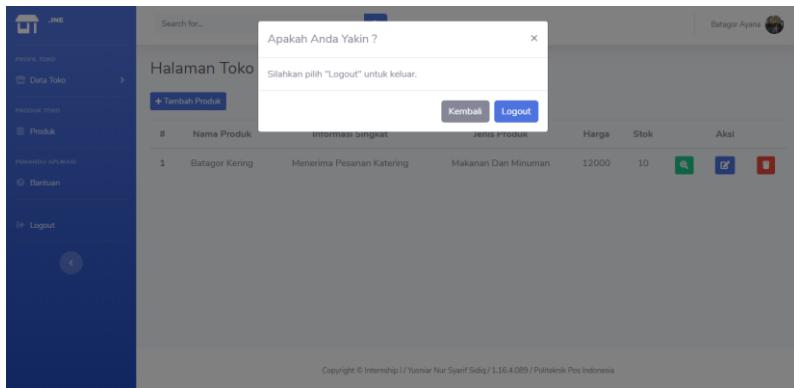
*Nama Kelurahan.
Kel. Pasir Jati

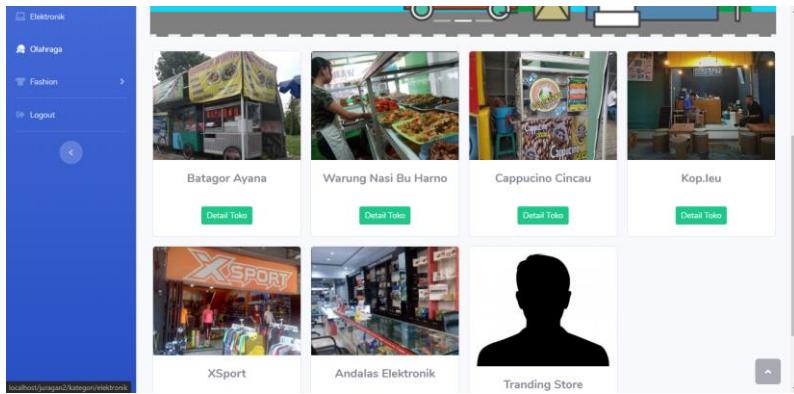
*Nama Kota/Kabupaten.
Kota Bandung

*Zipcode Pos.
40616

Detail Produk

Pada tau gak makanan BATAGOR ??? Batagor itu loh yang isinya ada tahu dan somay yang sudah di goreng dan ditaburi dengan bumbu kacang ditambah dengan timun. Siapa nih pecinta Batagor makanan tradisional Khas Kota Bandung ??? Bagi kalian yang tinggal di Kompleks





Detail Toko

Nama Toko: Batagor Ayana

Alamat Toko: Komplek Ujung Berung Indah Rt 02 Rw 01 Kec. Ujung Berung Kel. Pasir Jati Kota Bandung 40616

No. Telpon: +626285723438131

Catatan Toko: *MOHO DI BACA DENGAN SEKSAMA* Batagor Ayana merupakan usaha di bidang kuliner dimana produk yang dijual berupa batagor kering, batagor kuah, dan baso tahu. Adapun rules dari Batagor Ayana sebagai berikut : a). Di rekomendasi menggunakan jasa pengiriman GRAB INSTAN bagi pelanggan yang memesan dalam keadaan mistang. b). Pemesanan via GRAB hanya berlaku sampai berlaku dari pukul 08.00 - 14.00. c). Pelanggan juga dapat memesan dalam keadaan mentah untuk di goreng sendiri. d). Pelanggan dapat mengunjungi offline store sesuai alamat yang tertera. e). Jika pelanggan memesan lewat dari pukul 14.00 diharapkan menghubungi ADMIN Batagor Ayana terlebih dahulu agar tidak terjadi kesalahan orderan.

Produk Toko | Lihat Semua

6.3.2 Hasil Uji Lingkungan Aplikasi

Pada tabel 6.4 tersebut akan menampilkan mengenai hasil uji lingkungan aplikasi sebagai berikut :

Tabel 6.1 Hasil Uji Lingkungan Aplikasi

Identifikasi	Deskripsi	Prosedur Pengujian	Masukan	Keluaran Yang Diharapkan	Kriteria Evaluasi	Hasil Yang Didapat	Kesimpulan
A_01	Registration	User melakukan registrasi rationakan	User input data yang dibutuhkan oleh sistem	Data yang dikirim masuk ke dalam database	-	Datar masuk kedalam data base pada tabel user	Diterima

A_0 2	<i>Logi n</i>	<i>User melak ukan login</i>	<i>User melak ukan input email</i>	berha sil masu k ke halam	<i>User dashb oard</i>	-	Mas uk ke hala man das hbo ard	Diter ima
A_0 3	<i>View Prod uk Pelan ggan</i>	Pelan ggan melih at infor masi produ k	Mem berika n <i>action</i> terhad ap <i>button</i> detail	Mena mpilk an infor masi produ k	<i>Mena mpilk an infor masi produ k</i>	-	Mas uk ke hala man deta il prod uk	Diter ima
A_0 4	<i>View Prod</i>	Penju al	Mem berika	Mena mpilk	-	Mas uk	Diter ima	

	uk Penj ual	melih at infor masi produ k yang sudah di <i>input</i>	n <i>action</i> terhad ap <i>button</i> detail	an infor masi produ k		ke hala man deta il prod uk	
A_0 5	<i>Edit Profi le</i> Pelan ggan	Pelan ggan melak ukan penge ditan <i>profil</i> <i>edit</i> <i>profil</i> <i>baru</i>	Pelan ggan meng ubah data <i>profil</i> <i>edit</i> <i>profil</i> <i>baru</i>	Data <i>profil</i> <i>edit</i> berub ah	-	Dat a <i>prof</i> <i>ile</i> terg anti den gan yan g baru	Diter ima
A_0 6	<i>Edit Profi le</i>	Penju al melak	Penju al meng	Infor masi toko	-	Dat a info	Diter ima

		Penjual	ukan <i>edit</i> <i>profil</i> <i>e</i>	ubah infor masi <i>profil</i> <i>e</i> tokon ya pada halam an <i>Store</i> <i>Profil</i> <i>e</i>	berub ah		rma si toko terg anti den gan yan g baru	
A_0 7			<i>Chan</i> <i>ge</i> <i>Pass</i> <i>word</i>	<i>User</i> merub ah <i>passw</i> <i>ord</i> meng ganti <i>passw</i> <i>ord</i>	<i>User</i> denga n cara melak ukan <i>input</i> <i>passw</i> <i>ord</i> baru	<i>Pass</i> <i>word</i> <i>berub</i> <i>ah</i>	<i>Pas</i> <i>swo</i> <i>rd</i> terg anti den gan yan g baru	Diter ima

			pada halam an <i>Chan ge Passw ord</i>				
A_0 8	<i>Input</i> Prod uk	Penju al mena mbah kan <i>item</i> atau produ k baru	Pelan ggan melak ukan <i>input</i> produ k pada halam an <i>form</i> <i>input</i> produ k	Data produ k bertha mbah	-	Dat a prod uk bert amb ah	Diter ima
A_0 9	Tam bah Prod uk	Mena mbah kan produ	Pelan ggan mena mbah	Produ k bertha mbah	-	Hal ama n Cart	Diter ima

	Ke <i>Cart</i>	k kedala m <i>Cart</i>	kan produ k ke dalam <i>Cart</i> melal ui <i>button</i> <i>add to</i> <i>cart</i>	pada halam an Cart		bert amb ah prod uk	
--	-------------------	---------------------------------	--	-----------------------------	--	---------------------------------	--

6.3.3 Hasil Uji Antarmuka Pengguna

Pada tabel 6.5 tersebut dimana akan menampilkan mengenai hasil uji antarmuka pengguna sebagai berikut :

Tabel 6.2 Hasil Uji Antarmuka Pengguna

Identifikasi	Deskripsi	Prosedur Pengujian	Masukka n	Keluaran Yang Diharapkan	Kriteria Eva luasi	Hasil Yan g	Kesimpulan
B_01	Bahasa yang	-	-	Aplikasi	-	Aplikasi Ditul	Diterima

	digu naka n dala m aplik asi		Meng gu nakan Bahas a Indon esia Yang Baik Dan benar	is dala m Baha sa Ingg ris dan Indo nesia	
B_02	Peng ujian pew arna an	Meli hat Pewa rnaa n Yang Digu naka n layer	Warn a Yang Digu nakan Untu k setiap layar nya	War na Yan g Digu naka n Untu k	Diteri ma

		Dem i layar		konsi sten		Setia p Laya rnya kons isten	
B_03	Pesa n Kesa laha n	Men coba Mela kuka n Pros edur Yang salah	Dala m Pros es Vali das i Deng an User name Dan Pass word Yang tidak	Munc ul Saatu Pesan Kesal ahan	Mun cu Satu Pesa n kesa laha n	-	Diteri ma

			terda ftar					
B-04	Pena ta Leta kan men u	Meli hat Tatal etak Men u Laya r Dem i layar		Tata Letak -	Menu Yang sudah sesuai	- -	Tata Leta k Men u Yan g sesu ai kaid ah	Diteri ma

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.