

---

---

**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**

## **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, assesoris dan sesuatu yang dapat menunjang gaya hidup. Sektor terakhir yaitu kerajinan dimana merupakan sebuah usaha yang menghasilkan produk – produk kreatif dari buatan

tangan atau keterampilan tangan sehingga menciptakan sebuah seni atau karya.

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

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

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

## **BAB II**

## **LANDASAN TEORI**

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

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

#### **2.1.1 Usaha Mikro**

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

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

#### **2.1.2 Usaha Kecil**

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

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

### **2.1.3 Usaha Menengah**

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

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

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

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

## **2.3 Promosi**

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

memberikan sebuah pengenal produk sehingga membuat pelanggan yang asalnya tidak tahu menjadi tahu dan menarik minat untuk membeli produk tersebut.

## **2.4 Internet**

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

## **2.5 E-Commerce**

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

## **2.6 Komunitas**

Komunitas pada umumnya diartikan sebagai sebuah perkumpulan sosial dari beberapa organisme yang dimana saling berbagi lingkungan dan umumnya memiliki ketertarikan serta habitat yang sama [10]. Komunitas

juga dapat diartikan sebagai identifikasi serta interaksi sosial yang dibentuk dengan berbagai dimensi kebutuhan fungsional.

## **2.7 Aplikasi**

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

## **2.8 Framework**

*Framework* berfungsi dalam memfasilitasi pemograman web dan membuatnya menjadi lebih teratur [12]. Dimana *framework* akan meningkatkan produktivitas pemograman 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**

## **PERSIAPAN**

Sebelum melakukan pembuatan aplikasi *e-commerce* JURAGAN ada beberapa hal yang harus dipersiapkan terlebih dahulu. Ada beberapa hal yang perlu di *download*, apa saja yang perlu kalian siapkan ?, akan dibahas pada sub-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. *Download* terlebih dahulu *software Xampp*. Untuk melakukan *download*, dimana perlu mengunjungi *website* resminya adalah <https://www.apachefriends.org/index.html>. Maka akan terbuka seperti gambar 3.1.



Gambar 3.1 Halaman Utama Web Xampp

2. Klik pada bagian *text* “Click here for other versions”.



Gambar 3.2 Melihat Versions Xampp

3. Teman - teman akan dibawa ke halaman seperti pada gambar 3.3. *download Xampp* versi berapapun. Perlu kalian ketahui dimana dalam *website resmi Xampp* merupakan versi terbaru yang dimana sudah menggunakan PHP 7. Jika teman - teman belum terbiasa dengan PHP 7 silahkan teman - teman mencari *Xampp* veris 3 – 5 keatas yang dimana masih menggunakan PHP 5. Apasih perbedaan PHP 7 dan PHP 5 ?, akan kita bahas di akhir *tutorial instalasi software 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.

Version	Checksum	Size
7.2.26 / PHP 7.2.26	What's Included? md5 sha1	<a href="#">Download (64 bit)</a> 145 Mb
7.3.13 / PHP 7.3.13	What's Included? md5 sha1	<a href="#">Download (64 bit)</a> 146 Mb
7.4.1 / PHP 7.4.1	What's Included? md5 sha1	<a href="#">Download (64 bit)</a> 146 Mb

Requirements Add-ons More Downloads »

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms here.

**Documentation/FAQs**

There is no real manual or handbook for XAMP. We wrote the documentation in the form of FAQs. Have a burning question that's not answered here? Try the Forums or Stack Overflow.

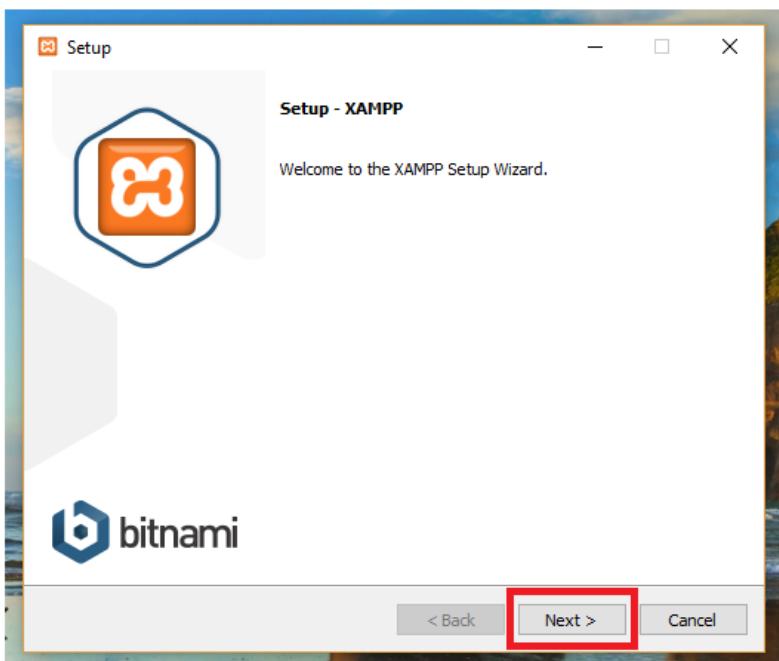
- Linux FAQs
- Windows FAQs
- OS X FAQs
- OS X XAMPP-VM FAQs

**Add-ons and Themes**

Bitnami provides a free all-in-one tool to install Drupal, Joomla!, WordPress

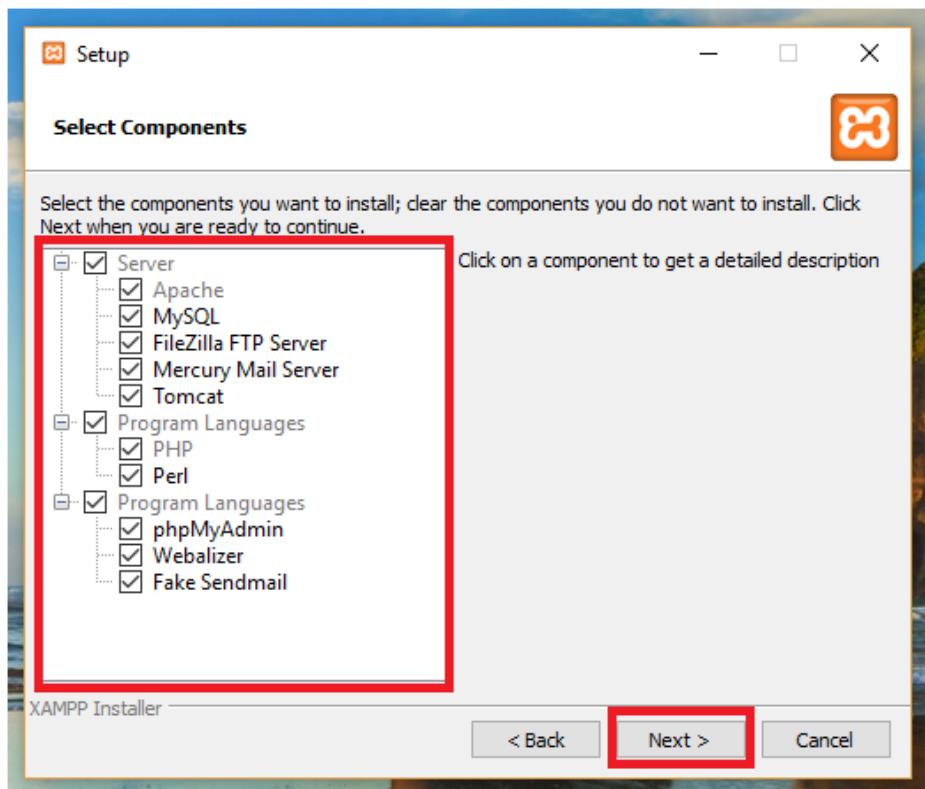
Gambar 3.3 Versi Xampp

4. Jalankan file yang sudah teman - teman download sebelumnya.
5. Dimana akan muncul sebuah jendela baru yang merupakan indikasi dimulainya proses instalasi. Pilih next.



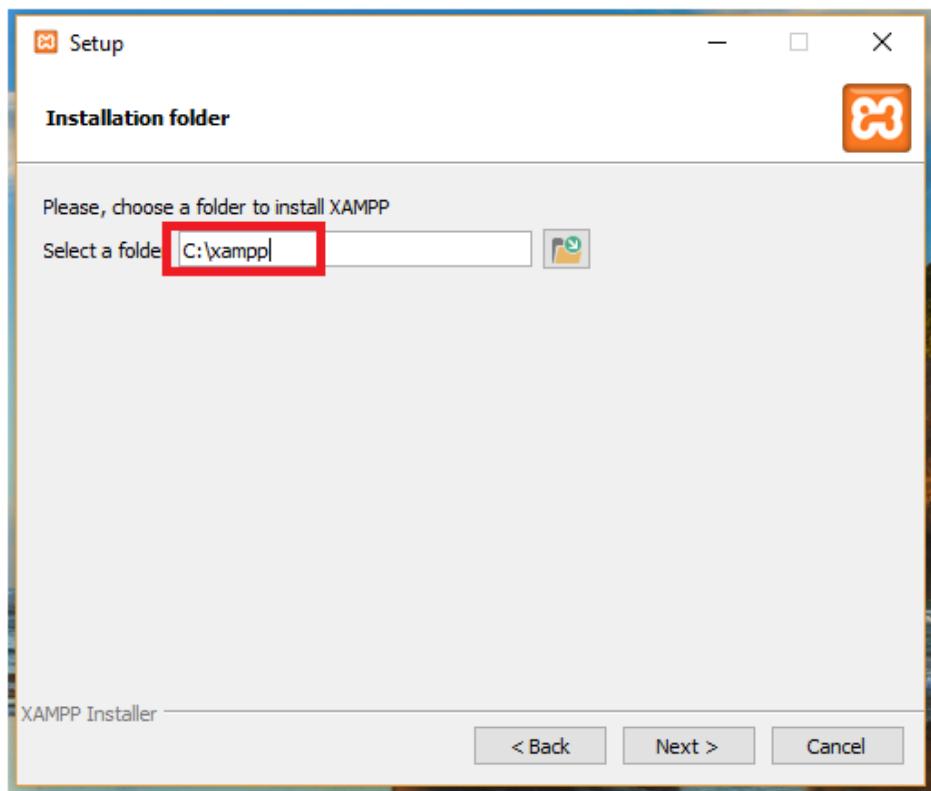
Gambar 3.4 Proses Instalasi Xampp Pertama

6. Pada halaman berikutnya dimana teman - teman akan diminta untuk memilih komponen – komponen yang akan digunakan. Perhatikan gambar 3.5. Klik *next* untuk menuju ke tahap berikutnya.



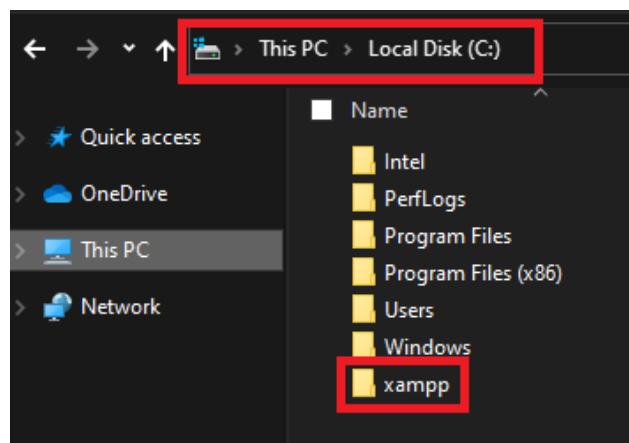
Gambar 3.5 Proses Instalasi Xampp Kedua

7. Pada halaman berikutnya dimana teman - teman diminta untuk memilih tempat *software Xampp* tersebut disimpan. Simpan saja di *local disk C:* lalu klik *next*.



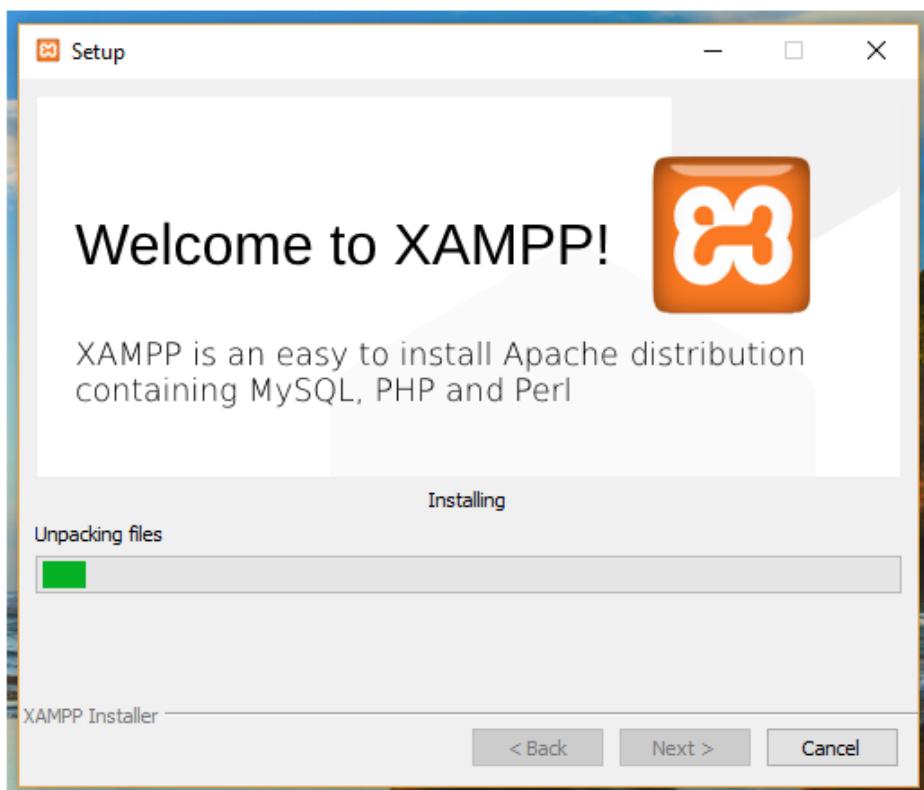
Gambar 3.6 Proses Instalasi Xampp Ketiga

8. Dimana *folder Xampp* kalian akan tersimpan.



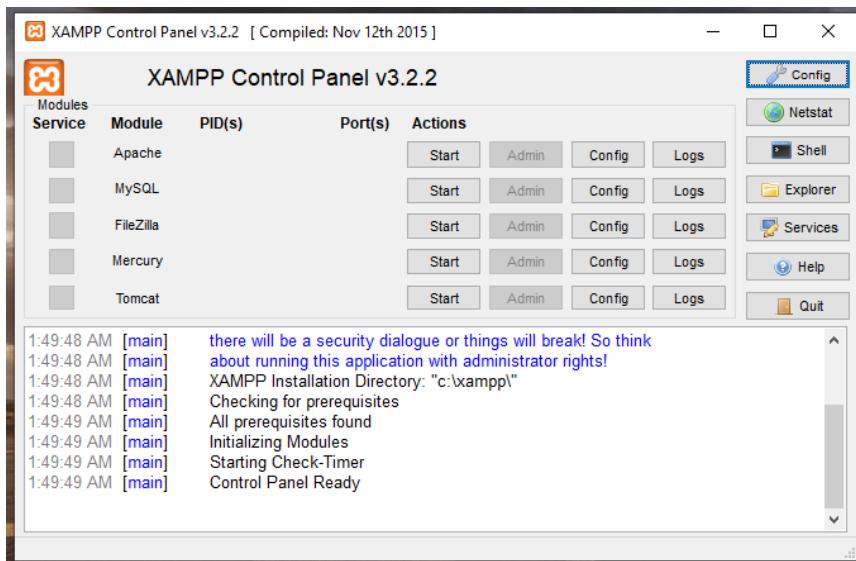
Gambar 3.7 Proses Instalasi Xampp Keempat

9. Tunggu beberapa menit hingga *bar progres* terisi penuh.



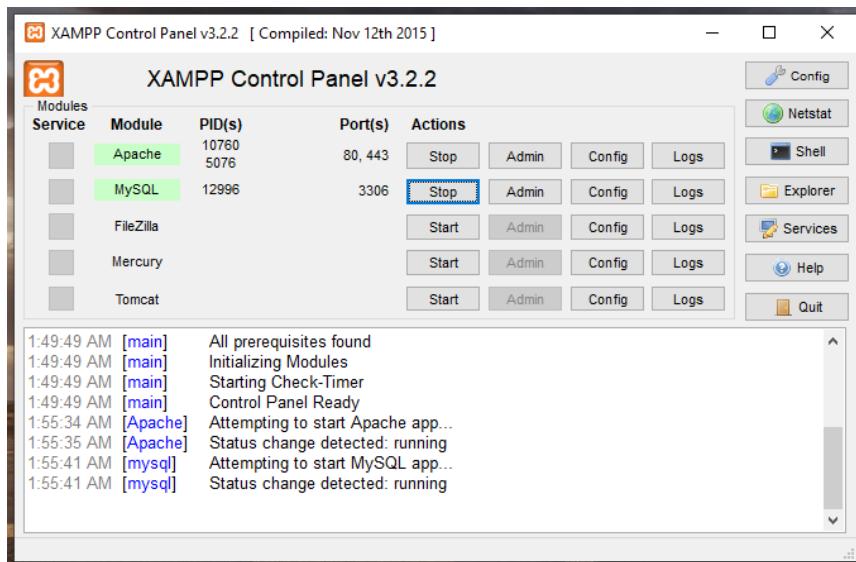
*Gambar 3.8 Proses Instalasi Xampp Kelima*

10. Jika sudah klik *next* dan coba jalankan *software Xampp* yang sudah terinstall pada laptop / PC kalian.



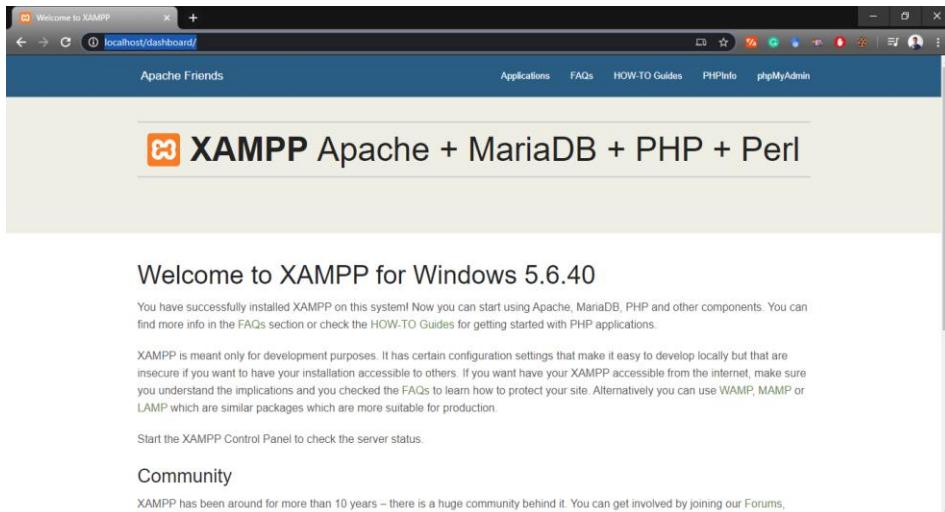
Gambar 3.9 Proses Instalasi Xampp Keenam

- Berikan *actions start* pada *module Apache* dan *MySQL*. Apabila berganti menjadi wana hijau *software Xampp* teman - teman sudah dapat digunakan.



Gambar 3.10 Proses Instalasi Xampp Ketujuh

12. Buka *web browser* kalian dan kunjungi <http://localhost/dashboard/> dan apabila nampak seperti pada gambar 3.11 maka proses *instalasi software teman - teman* berhasil dan siap untuk digunakan.



Gambar 3.11 Porses Instalasi Xampp Selesai

### 3.1.1 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.

### 3.3 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.

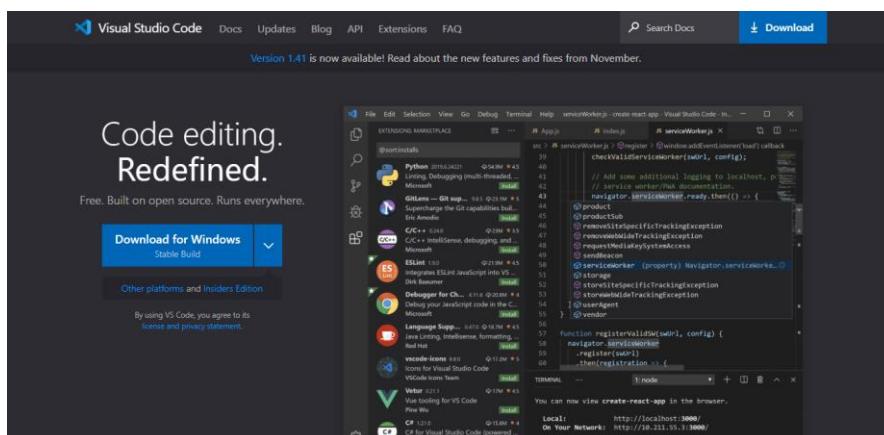
Sebelum melakukan penginstallan *software Vscode* dimana kita harus mengetahui spesifikasi atau *requirements* yang dibutuhkan. Spesifikasi yang dibutuhkan dapat dilihat pada tabel 3.1.

Tabel 3.1 Spesifikasi VScode

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

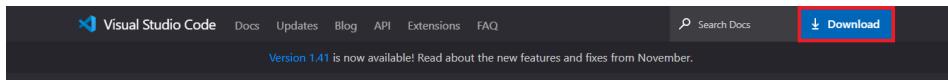
Pada kesempatan kali ini dimana penulis akan memberikan *tutorial* bagaimana cara melakukan *instalasi software Vscode* dengan menggunakan *sistem operation windows*. Ikuti langkah – langkah berikut ini :

1. Kunjungi *link* berikut <https://code.visualstudio.com/>



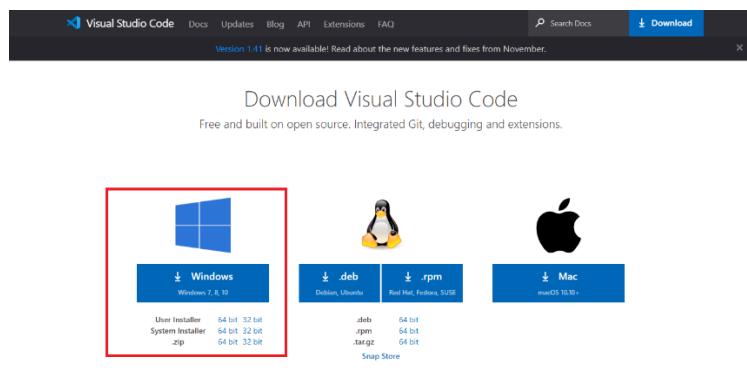
Gambar 3.12 Proses Instalasi Vscode Pertama

2. Pilih *download* pada bagian pojok kanan atas



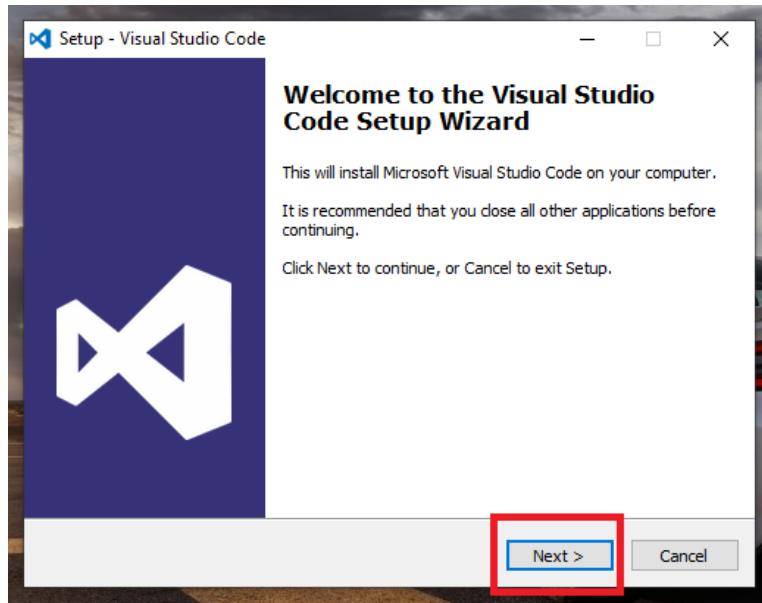
Gambar 3.13 Proses Instalasi Vscode Kedua

3. Karena kita disini akan menggunakan *operatios system windows* silahkan pilih *windows*. Maka sistem akan melakukan proses *download*, tunggu hingga selesai.



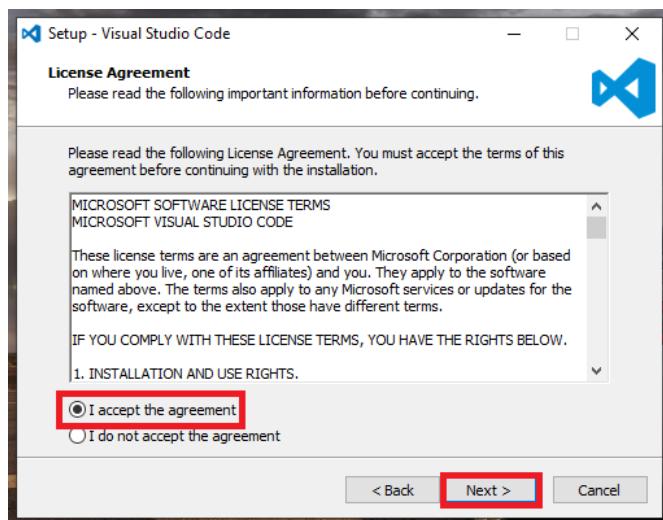
Gambar 3.14 Proses Instalasi Vscode Ketiga

4. Apabila sudah *download* sudah selesai silahkan jalankan *file* tersebut dengan cara “*Run Administrator*” lalu klik *next*.



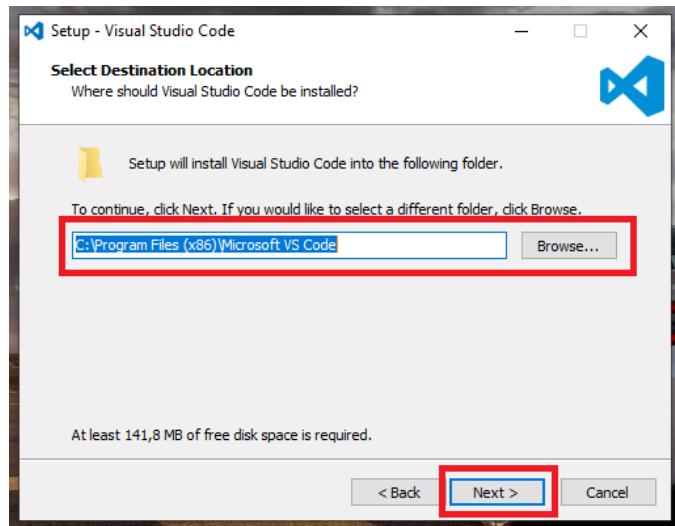
Gambar 3.15 Proses Instalasi Vscode Kempat

5. Pada tahap selanjutnya silahkan pilih “*I accept the agreement*” lalu klik *next*.



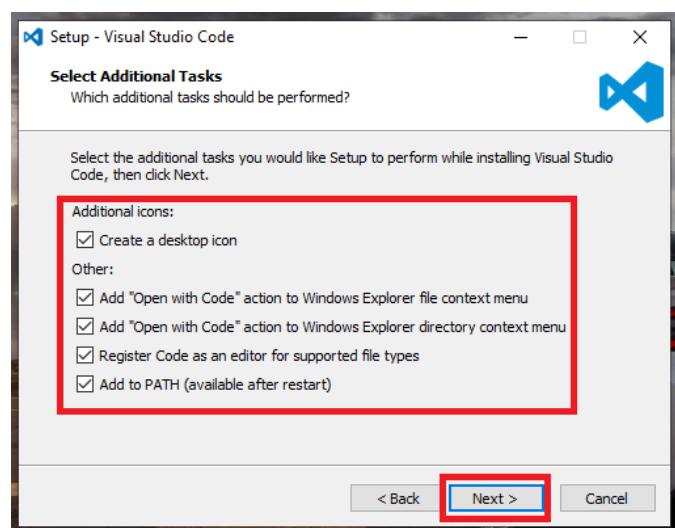
Gambar 3.16 Proses Instalasi Vscode Kelima

6. Silahkan pilih dimana teman - teman akan menyimpan file Vscode tersebut lalu klik *next*.



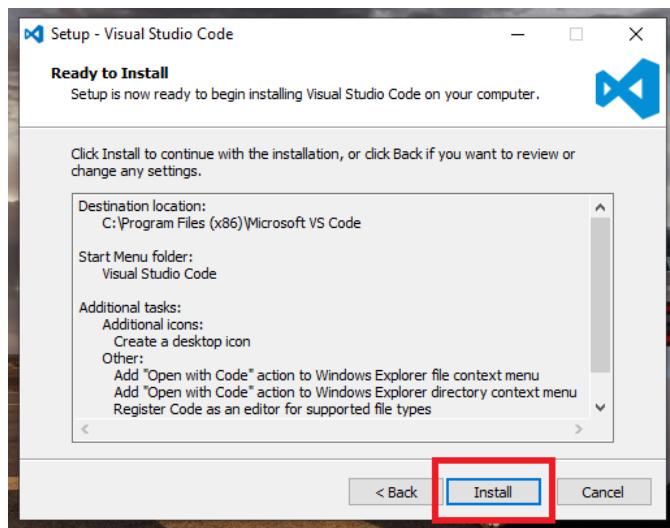
Gambar 3.17 Porses Instalasi Vscode Keenam

7. Pada bagian *select additional task* silahkan centang semua dan klik *next*.



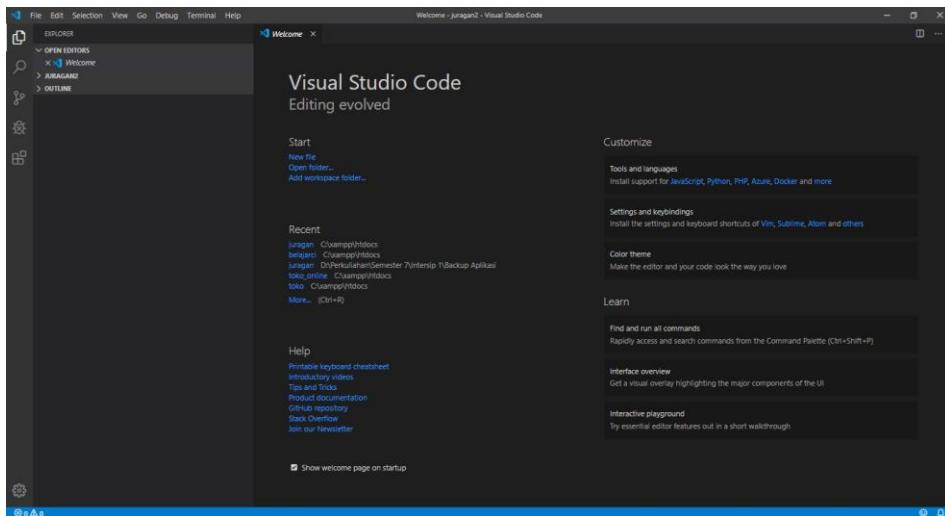
Gambar 3.18 Proses Instalasi Vscode Ketujuh

8. Klik *install* untuk memulai proses. Dimana akan muncul progres bar, silahkan tunggu hingga penuh.



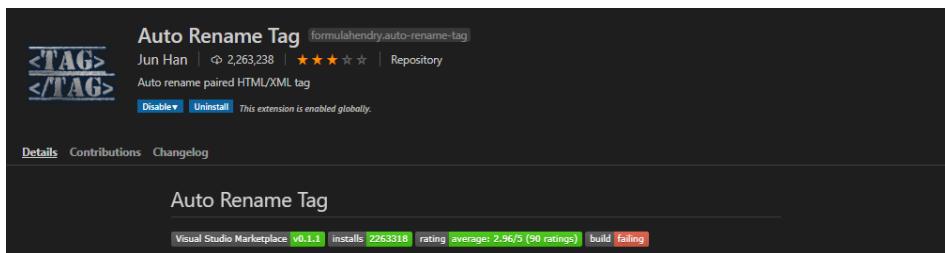
Gambar 3.19 Proses Instalasi Vscode Kedelapan

9. Apabila progres bar sudah penuh maka proses *install* telah selesai dan Vscode akan berada pada halaman *desktop* teman - teman. Buka Vscode yang sudah ter-*install*. Tampilan Vscode akan terlihat seperti pada gambar 3.20.



Gambar 3.20 Proses Instalasi Vscode Selesai

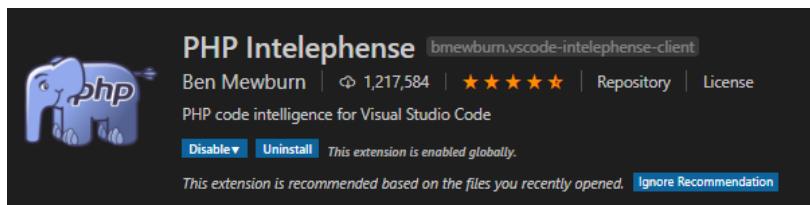
Apabila proses *install software Vscode* sudah selesai dimana teman - teman perlu melakukan *install* beberapa *extensions* untuk membantu proses pengkodean. Apa saja *extensions* tersebut ?, yang perlu teman - teman *install* adalah “*Auto Rename Tag*”.



Gambar 3.21 Extensions Auto Rename Tag

*Extensions* tersebut berfungsi untuk merapikan *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 dirapikan. Hal tersebut sangat bermanfaat karena *script code* yang teman - teman ketikan akan terlihat lebih rapih.

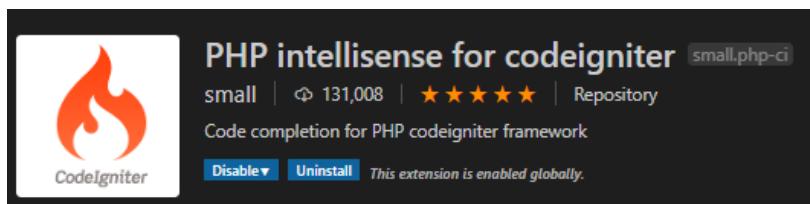
*Extensions* yang kedua dimana teman - teman perlu meng-*install* “*PHP Intelephense*”.



Gambar 3.22 Extensions *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.

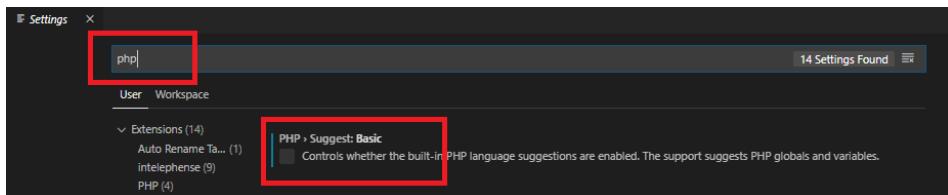
*Ektensions* yang terakhir dimana teman - teman perlu melakuakan *install* “*PHP Instellisense for codeigniter*”.



Gambar 3.23 Ekstensions *PHP Intellisense For CodeIgniter*

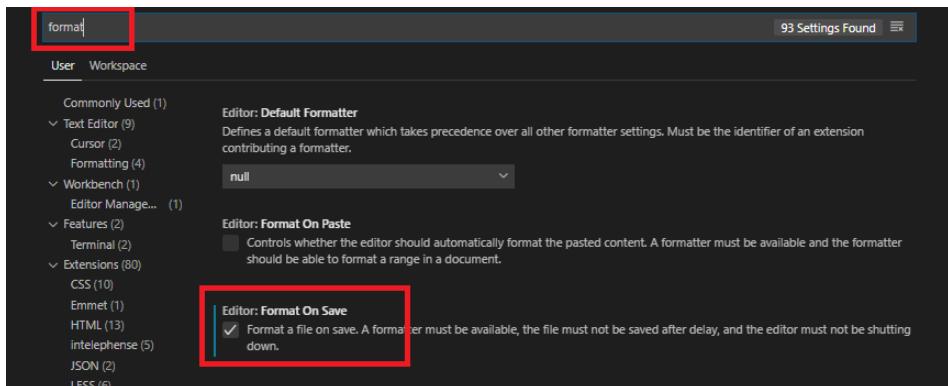
*Extensions* tersebut memiliki fungsi untuk memudahkan teman - teman dalam melakukan *script code* pemanggilan *PHP* yang berada pada *codeigniter*. Untuk menggunakan semua *extensions* yang telah teman - teman *install* dimana perlu dilakukan *setting* kembali pada *software* *Vscode*. Pilih *preferences* → *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 3.24 Setting Vscode 1

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.



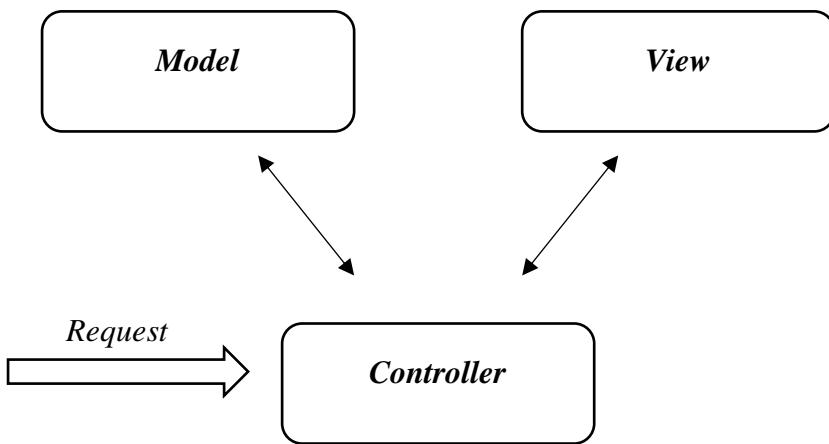
Gambar 3.25 Setting Vscode 2

### 3.4 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 3.26 Konsep MVC

### 3.4.1 Tutorial install Codeigniter

Bagaimana cara melakukan *instalasi framework codeigniter* ?, kali ini penulis akan memberikan *tutorial download framework codeigniter*. Ikuti langkah – langkah berikut :

1. Kunjungi website resmi *codeigniter* di <https://codeigniter.com/>.



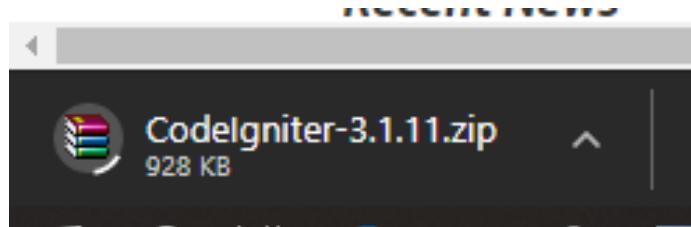
Gambar 3.27 CI Tahap 1

2. Klik pada bagian *download* seperti pada gambar 3.28.



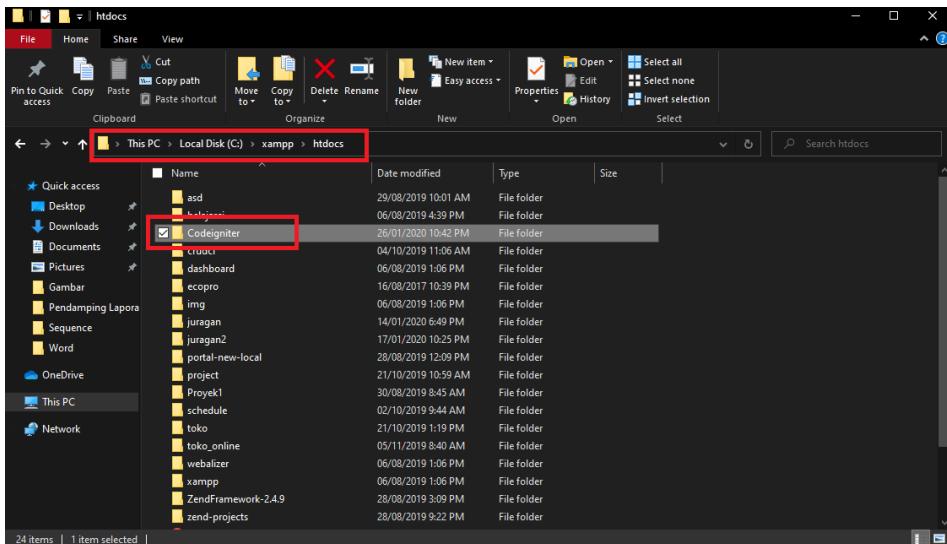
Gambar 3.28 CI Tahap 2

3. Tunggu beberapa menit hingga proses *download* selesai.



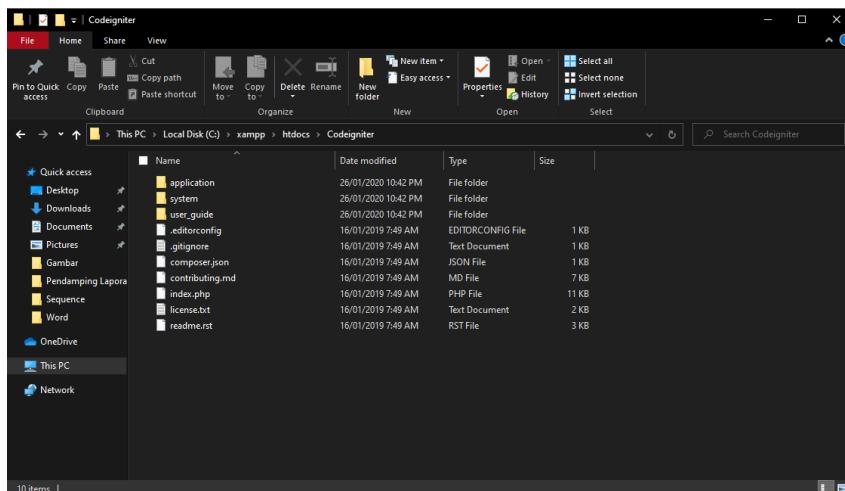
Gambar 3.29 CI Tahap 3

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



Gambar 3.30 CI Tahap 4

5. Buka folder *codeigniter* tersebut pastikan isinya sama seperti pada gambar 3.31.

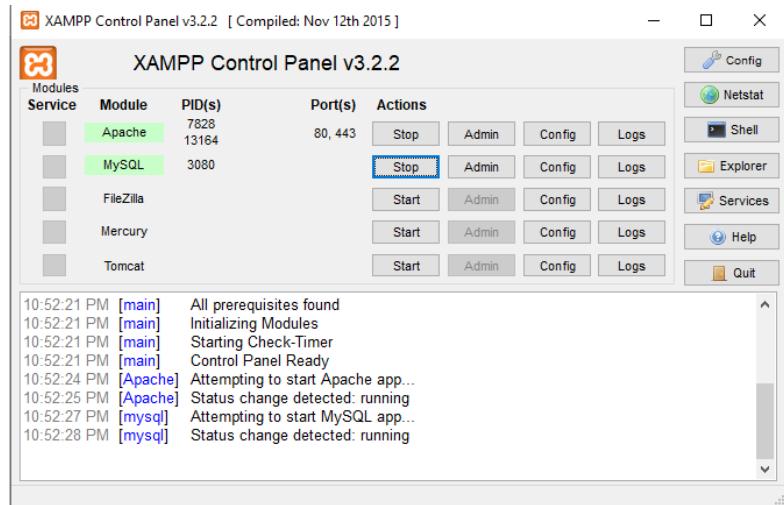


Gambar 3.31 CI Tahap 5

6. Untuk menguji *framework codeigniter* tersebut bisa digunakan atau tidak dimana teman - teman perlu melakukan pengujian

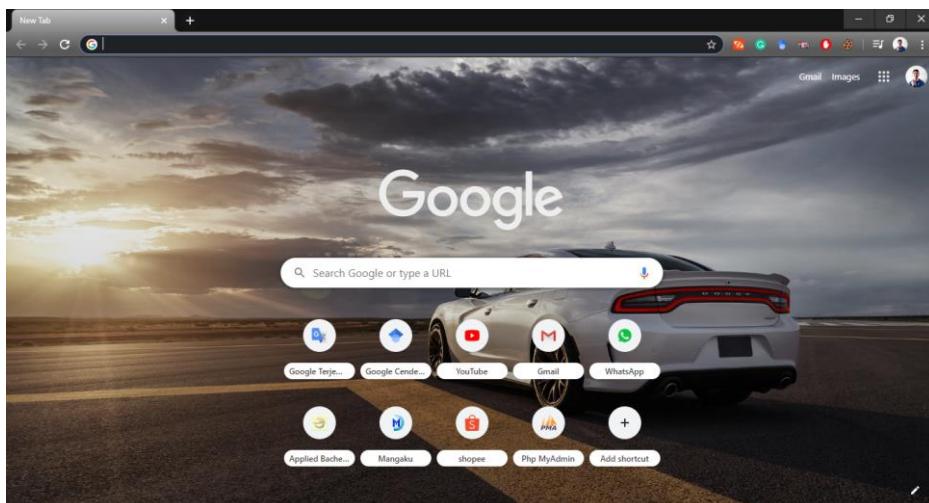
terlebih dahulu dengan menggunakan bantuan *software Xampp* yang sudah di *download* dengan mengikuti langkah dari gambar 3.1 – 3.11.

7. Buka *software Xampp* tersebut dan jalankan *Apache* serta *MySQL*.



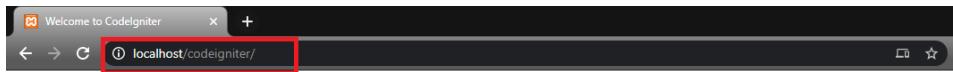
Gambar 3.32 CI Tahap 6

8. Buka *web browser* yang teman - teman gunakan.



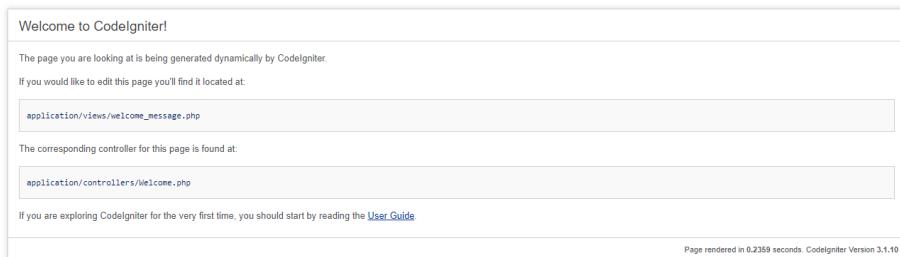
Gambar 3.33 CI Tahap 7

9. Ketikkan `http://localhost/codeigniter/` pada bagian pencarian.



Gambar 3.34 CI Tahap 8

10. Apabila tampilan yang muncul terlihat seperti pada gambar 3.35 maka *framework codeigniter* teman - teman sudah berhasil ter-*install*.



Gambar 3.35 CI Selesai

# BAB IV

## CODEIGNITER

### 4.1 Sejarah *CodeIgniter*

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

### 4.2 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*.

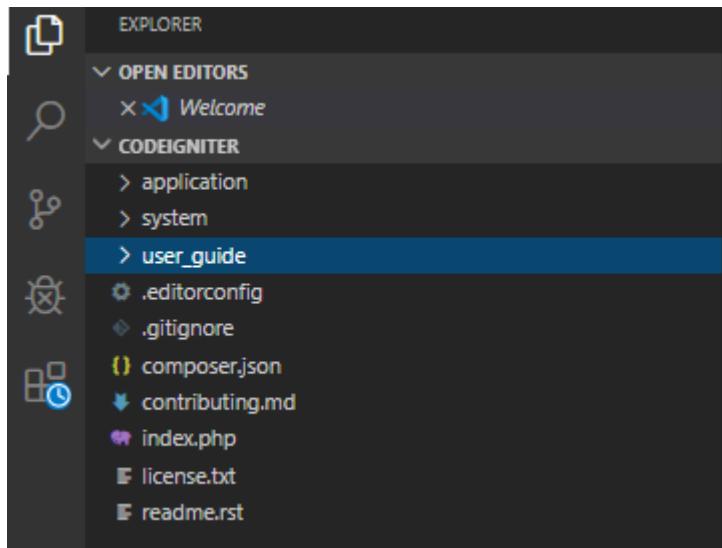


Gambar 4.1 User Guide Codeigniter

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

#### 4.3 Struktur Direktori *Codeigniter*

Sebelum memulai kegiatan *programming* teman - teman dengan *framework* *codeigniter*, kenali dulu yuk struktur direktori dari *framework* tersebut, perhatikan gambar 4.2.

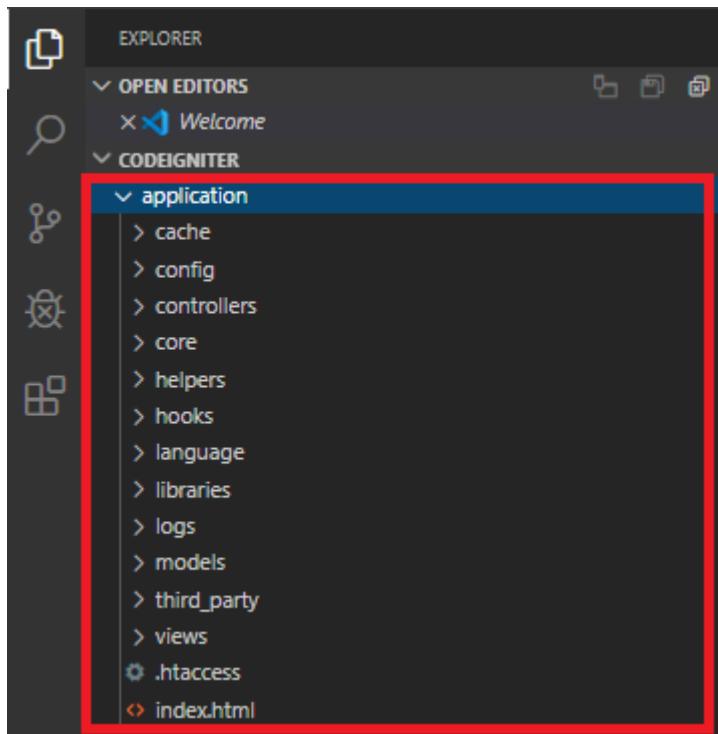


Gambar 4.2 Struktur Codeigniter

Pada gambar 4.2 dimana merupakan struktur direktori yang dimiliki oleh *framework codeigniter*. Dalam *codeigniter* terdapat dua direktori penting yaitu *application* dan juga *system*. Simak penjelasan berikut ini.

#### 4.3.1 Direktori *Application*

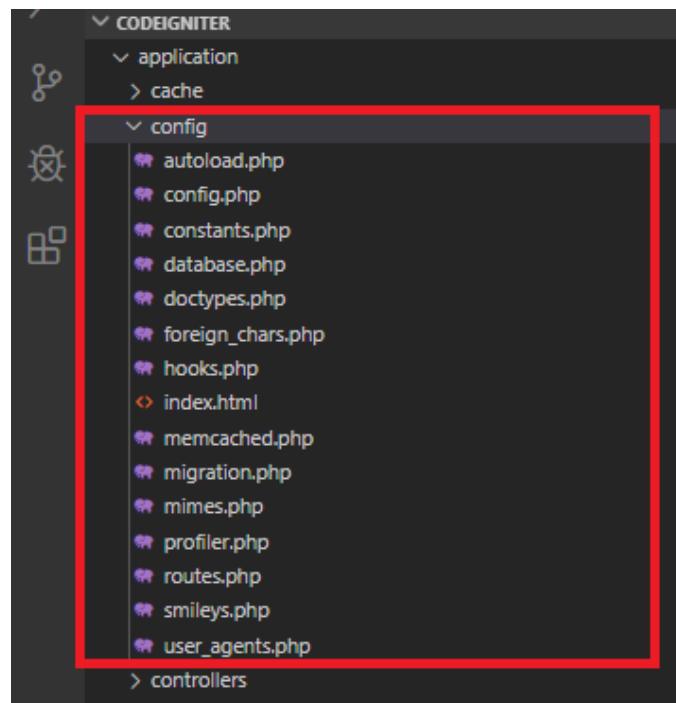
Direktori *application* merupakan direktori yang akan kita gunakan dalam pembuatan aplikasi kita nanti. Coba teman - teman klik pada direktori *application* tersebut dan lihat isinya, akan terlihat seperti gambar 4.3.



Gambar 4.3 Direktori Application

Dimana isi dari direktori *application* tersebut dapat dilihat pada gambar 4.3. Mari kita bedah satu persatu mengenai direktori *application* tersebut.

- **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 4.4 Direktori Config

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

```
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 4.5 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 4.6 Pemanggilan Fungsi Helpers

- **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 4.7 Script Code Model

Untuk menggunakan *model* tersebut dimana teman - teman harus melakukan pemanggilan terlebih dahulu pada *file autoload.php* seperti pada gambar 4.8.

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

Gambar 4.8 Pemanggilan Fungsi Model Pada Autoload

Apabila sudah di panggil melalui *file autoload.php* maka fungsi *model* tersebut sudah dapat digunakan, namun untuk menampilkannya pada *view* dimana teman - teman harus melakukan pemanggilan kembali pada *file controller*, perhatikan gambar 4.9.

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

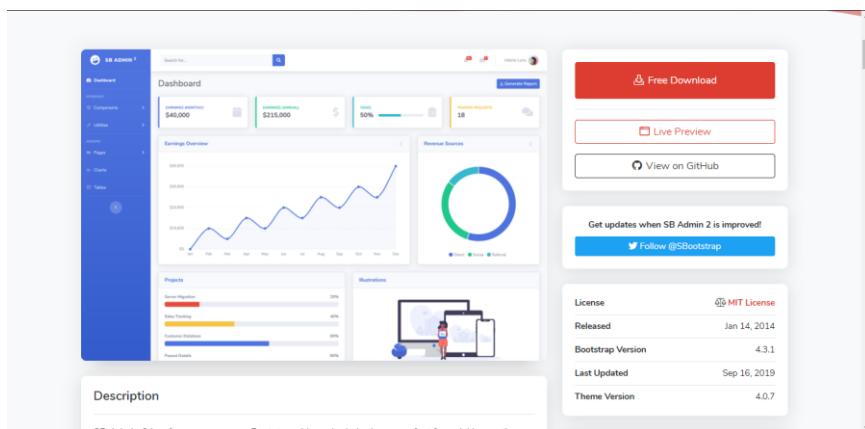
    $this->load->view('user/templates/header', $data);           *m_user merupakan nama dari model yang kita buat.
    $this->load->view('user/templates/sidebar');                  *.t_barang merupakan function dari m_user.
    $this->load->view('user/templates/topbar', $data);
    $this->load->view('user/halaman_produk',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

Gambar 4.9 Pemanggilan 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.

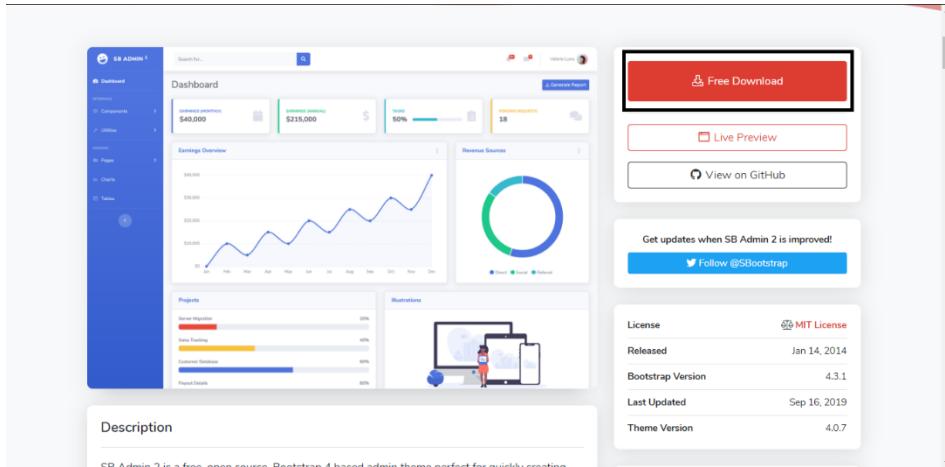
#### 4.4 Membuat CRUD Sederhana

Dengan teori – teori yang sudah diberikan yuk kita bekajar membuat *web site* CRUD sederhana denga *framework codeigniter*. Apa itu *website* CRUD ?, *web site* CRUD adalah dimana suatu sistem yang memiliki fungsi atau *action* “*cretae, read, update, dan delete*”. Sebelum membuat *web site* CRUD tersebut dimana kita perlu mempersiapkan komponen – komponen pendukung dalam membuat *web site* kita. Ada beberapa hal yang perlu teman - teman *download* terlebih dahulu. Karena kita akan membuat tampilan *web site* menjadi lebih menarik, maka di butuhkanlah sebuah *template bootstrap*. *Template bootstrap* yang akan kita gunakan adalah “ SB ADMIN 2 “ yang merupakan *template bootstrap version 4*. Untuk men-*download template* tersebut teman – taman dapat mengunjungi *link* berikut <https://startbootstrap.com/themes/sb-admin-2/>.



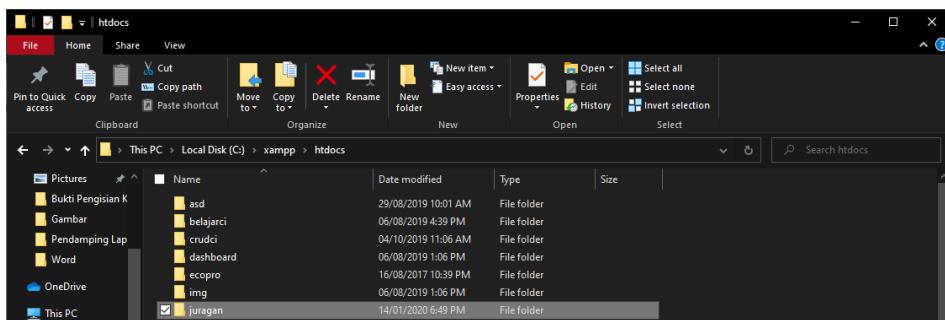
Gambar 4.10 Bootstrap SB Admin 2

*Download template bootstrap tersebut dengan mengklik button free download yang berwarna merah.*



*Gambar 4.11 Download Bootstrap SB Admin 2*

Tunggu proses *download* hingga selesai. Untuk latihan kali ini kita akan menggunakan atau merubah tampilan pada *codeigniter* yang sudah di *download* pada *tutorial* gambar 3.27 – 3.35. Langkah pertama silahkan teman – teman *rename folder* *codeigniter* yang ada pada *folder htdoc* dengan “juragan” seperti pada gambar 4.12.



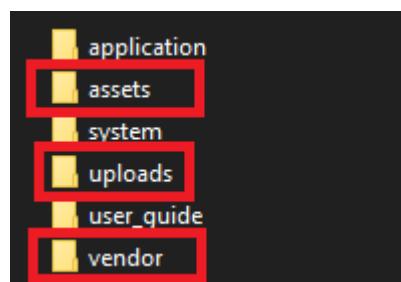
*Gambar 4.12 Konfigurasi Tahap 1*

Apabila *templates* SB Admin 2 sudah selesai teman – teman *download* jangan lupa lakukan *extract* telebih dahulu yah dan lakukan *rename* menjadi sbadmin2 agar tidak terlalu panjang, seperti pada gambar 4.13.

sbadmin2	31/01/2020 12:43 AM	File folder
Administrator-admin-dashboard-master....	04/10/2019 7:18 PM	WinRAR ZIP archive 1.669 KB
startbootstrap-sb-admin-2-gh-pages....	06/08/2019 1:49 PM	WinRAR ZIP archive 5.819 KB

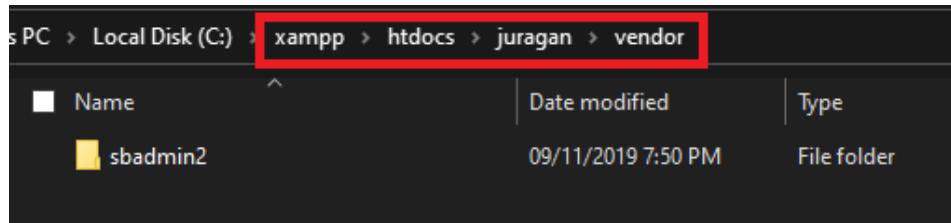
Gambar 4.13 Konfigurasi Tahap 2

Buka kembali *folder* juragan yang tersimpan di *xampp/htdocs/*. Langkah berikutnya dimana teman – teman harus membuat 3 buah *folder* baru yaitu *assets* yang dimana akan menampung *css* dan *javascript* dari *bootstrap*, *uploads* yang dimana akan menampung data gambar pada aplikasi yang kita buat, dan *vendor* yang dimana akan menampung komponen dari *bootstrap* SB Admin 2 kita. Perhatikan gambar 4.14 untuk lebih lengkapnya.



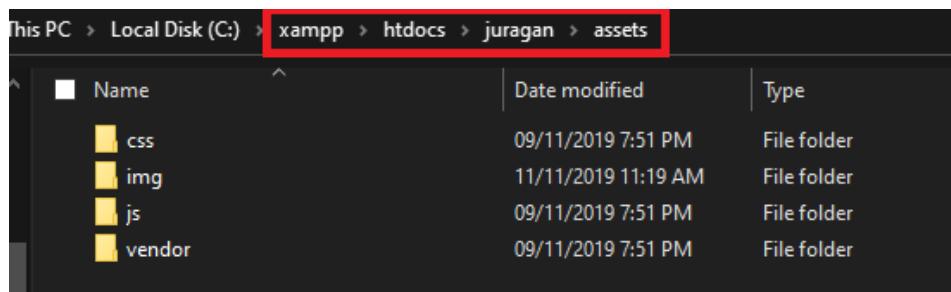
Gambar 4.14 Konfigurasi Tahap 3

Langkah berikutnya dimana teman – teman perlu memindahkan *folder templates* SD Admin 2 ke dalam *folder vendor* tersebut seperti pada gambar 4.15.



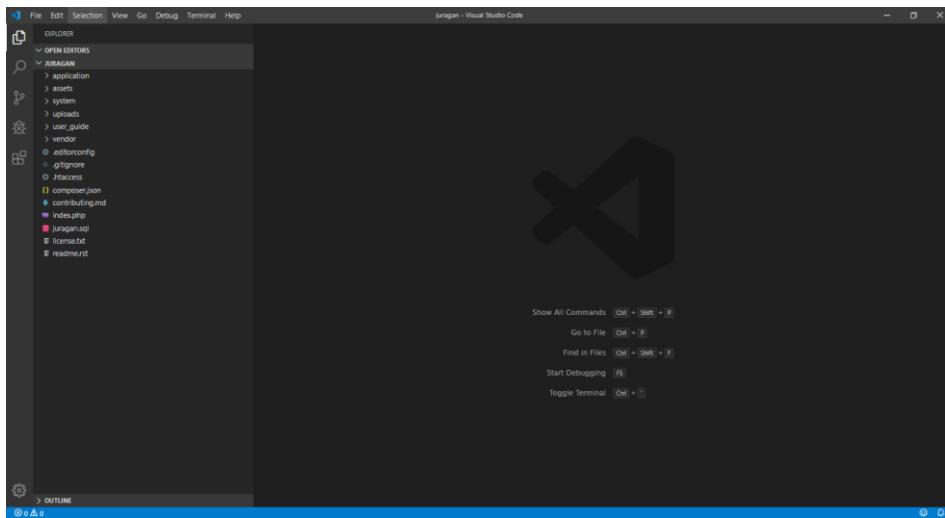
Gambar 4.15 Konfigurasi Tahap 4

Buka *folder* sbadmin2 tersebut, agar *css* yang diberikan oleh *templates bootstrap* dapat terbaca dimana teman – teman perlu memindahkan beberapa *folder* kedalam *folder assets* yang sudah teman – teman buat pada gambar 4.14. Apa saja *folder* tersebut ?, teman – teman perlu memindahkan *folder css, img, js*, dan *vendor* yang berada pada *folder* sbadmin2 tersebut, perhatikan gambar 4.16.



Gambar 4.16 Konfigurasi Tahap 5

Apabila teman – teman sudah mengikuti tahap 1 – 5 pada latihan CRUD dimana saatnya teman – teman mulai melakukan tahap pemrograman dengan *framework codeigniter* tersebut. Silahkan buka *software visual studio code* yang sudah teman – teman *download* dengan mengikuti *tutorial* pada gambar 3.12 – 3.25.



Gambar 4.17 Konfigurasi Tahap 6

Sebelum teman–teman melakukan tahap pemrograman dimana teman–teman perlu melakukan *setting* terhadap *framework codeigniter*. Silahkan teman–teman buka direktori *application/config/autoload.php* scroll hingga pada bagian *libraries*. Karena kita akan menggunakan kembali *folder* juragan tersebut untuk membuat sistem *e-commerce* yang akan kita bangun dimana kita akan melakukan konfigurasi terlebih dahulu dengan tujuan tidak mengulang kembali dan dapat menghemat waktu penggerjaan. Maka *libraries* yang akan kita gunakan adalah *database*, *email*, *sessions*, *form validation*, dan juga *cart*, perhatikan gambar 4.18.

```
61 $autoload['libraries'] = array('database', 'email', 'session', 'form_validation', 'cart');
```

Gambar 4.18 Konfigurasi Tahap 7

Kita juga memerlukan fungsi *model* untuk menghubungkan antara sistem dengan *database*, maka dari itu jangan lupa teman – teman lakukan konfigurasi terhadap *model*-nya juga yah, scroll kebawah hingga menemukan *model* pada file *autoload.php*.

```
135 $autoload['model'] = array('model_barang', 'model_admin', 'model_kategori');
```

Gambar 4.19 Konfigurasi Tahap 8

Selain *libraries* dan *model* dimana teman – teman juga perlu melakuan konfigurasi *helper* pada file *autoload.php*. Perhatikan gambar 4.20.

```
92 $autoload['helper'] = array('url', 'file', 'security');
```

Gambar 4.20 Konfigurasi Tahap 9

Langkah berikutnya dimana teman – teman perlu melakukan konfigurasi pada direktori *config/config.php*. Silahkan teman – teman scroll hingga ke bagian *base\_url*, dimana teman – teman perlu mengisinya dengan *url* dari sistem yang akan dibuat. Perhatikan gambar 4.21.

```
26 $config['base_url'] = 'http://localhost/juragan/';
```

Gambar 4.21 Konfigurasi Tahap 10

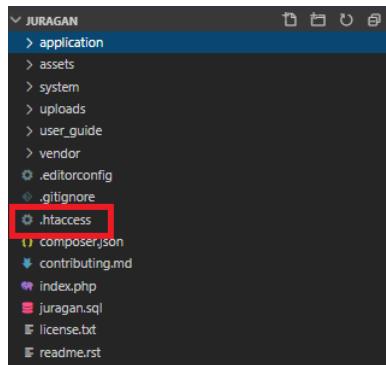
Agar teman – teman tidak perlu menuliskan *index.php* maka teman – teman perlu menghapus *index.php* tersebut pada bagian *index\_page*, scroll kebawah, perhatikan gambar 4.22.

```
32 | Typically this will be your index.php file, unless you've renamed it to
33 | something else. If you are using mod_rewrite to remove the page set this
34 | variable so that it is blank.
35 |
36 |
37 */
38 $config['index_page'] = '';
39
```

Gambar 4.22 Konfigurasi Tahap 11

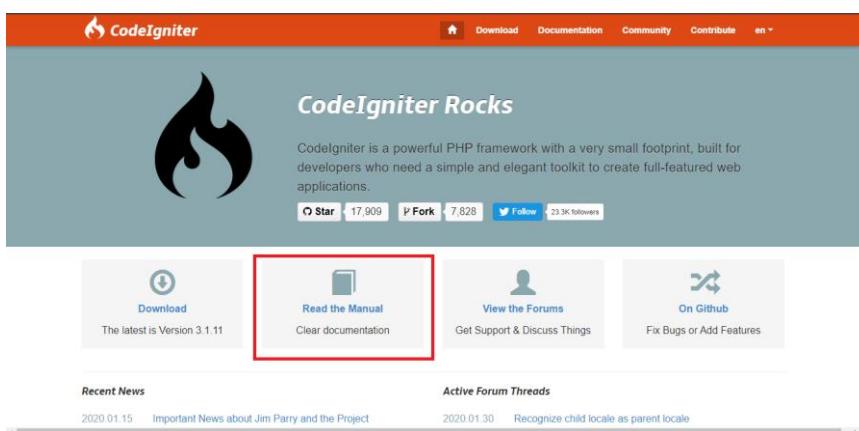
Apabila *index.php* sudah di hapus jangan lupa teman – teman perlu melakuakan *setting mod\_rewrite*-nya. Bagaimana sih cara melakukan *setting mod\_rewrite* ?, tenang disini penulis akan memberikan *tutorial*

secara lengkap. Silahkan teman – teman buat *file* baru di luar direktori *application*, beri nama “ *.htaccess* ”. Perhatikan gambar 4.23.



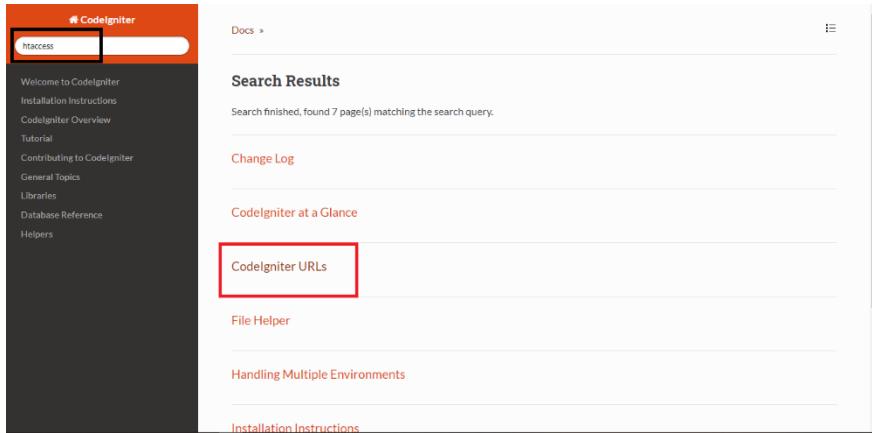
Gambar 4.23 htaccess 1

Apa sih isi dari *file .htaccess* ?, untungnya *framework codeigniter* sudah dilengkapi dengan *user guide* yang dimana merupakan dokumentasi lengkap dari *codeigniter*. Jadi teman – teman tidak perlu bingung, silahkan buka *web site codeigniter* untuk melihat dokumentasinya dan pilih *read the manual*.



Gambar 4.24 htaccess 2

Pada kolom pencarian teman – teman tinggal ketikkan saja *htaccess* lalu pilih *Codeigniter URLs*.



Gambar 4.25 htaccess 3

Lakukan *scroll* kebawah dimana teman – teman akan menemukan “*Removing the index.php file*”, *copy script code* yang diberikan kotak merah.

### 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
```

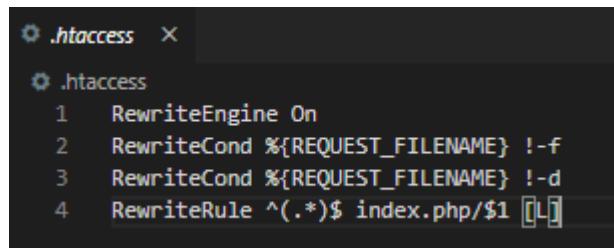
If your Apache server has *mod\_rewrite* enabled, you can easily remove this file by using a *.htaccess* file with some simple rules. Here is an example of such a file, using the “negative” method in which everything is redirected except the specified items:

```
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 4.26 htaccess 4

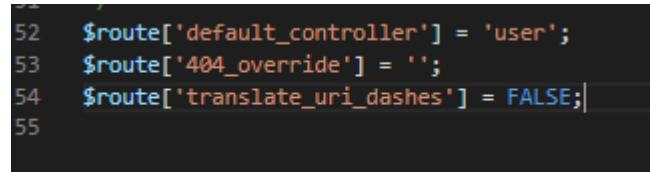
*Paste script code* tersebut ke *file .htaccess* yang sudah teman – teman buat pada gambar 4.23.



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

Gambar 4.27 htaccess 5

Selamat dimana teman – teman sudah berhasil melakukan *setting mod\_rewrite* pada *framework codeigniter* dan teman – teman tidak perlu lagi menuliskan *index.php* saat menjalankan *program*. Selanjutnya dimana teman – teman perlu melakukan konfigurasi terhadap *default controller*. Silahkan teman – teman buka *file routes.php* yang terdapat pada direktori *application/config*. *Scroll* kebagian paling bawah dan ganti *default controller welcome* dengan *user*.



```
51
52 $route['default_controller'] = 'user';
53 $route['404_override'] = '';
54 $route['translate_uri_dashes'] = FALSE;
55
```

Gambar 4.28 Konfigurasi Tahap 12

Apabila teman – teman sudah melakukan konfigurasi terhadap *framework codeigniter*-nya yuk kita mulai latihan membuat *crud* pada *framework codeigniter* kita. Silahkan buat *file* baru pada direktori *application/controller* dan beri nama *User.php*. Isikan *file User.php* dengan *scrip code* yang ada dibawah.

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

```
class User extends CI_Controller
{
}
```

Script Code 4.1 Controllers User

Langkah berikutnya dimana teman – teman perlu membuat *function index* yang merupakan *function default* dari *User.php*. Perhatikan *script code 4.2*.

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

class User extends CI_Controller
{
    public function index()
    {

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

Script Code 4.2 Function Default User.php

Sebelum teman – teman lanjut ke tahap berikutnya dimana teman – teman memerlukan *database*. Yuk kita *setting* terlebih dahulu *databasenya* melalui *phpmyadmin* dengan cara mengunjungi *link* *http://localhost/phpmyadmin/index.php* lalu pilih *new* dan berikan nama *juragan* lalu *create*. Pastikan *software Xampp* teman – teman sudah jalan yah.



Gambar 4.29 Membuat Database Juragan

Setelah berhasil membuat *database* dimana teman – teman membutuhkan sebuah tabel. Silahkan teman – teman buat tabel dengan nama *user* dan strikturnya seperti pada tabel 4.1.

Tabel 4.1 Struktur Tabel User

Nama	Tipe	Keterangan
Id	Integer(11)	Primary Key
Name	Varchar(25)	
Email	Varchar(25)	
Alamat	Text	
Image	Text	

Lakukan *input* data pada tabel *user* dengan menggunakan *script code* 4.3 pada *database* juragan.

```
INSERT INTO `user` (`id`, `name`, `email`, `image`,  
`alamat`) VALUES (NULL, 'YusniarNS',  
'yusniar33@gmail.com', 'default.jpg', 'asrama yon  
zipur 9');
```

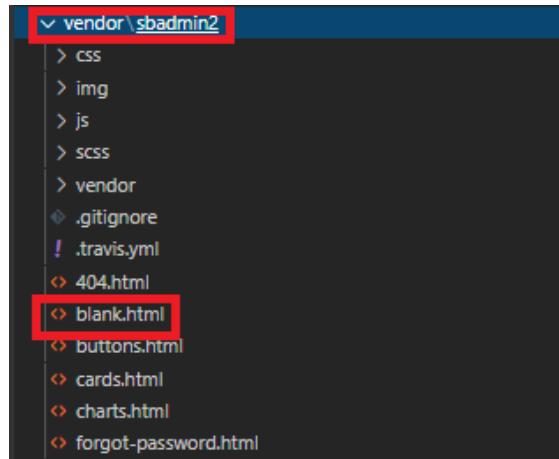
Script Code 4.3 Input Data Tabel User

Setelah itu teman – teman perlu melakukan konfigurasi pada direktori *application/config/database.php* seperti pada gambar 4.30.

```
73 $active_group = 'default';
74 $query_builder = TRUE;
75
76 $db['default'] = array(
77     'dsn' => '',
78     'hostname' => 'localhost', Server dari software Xampp
79     'username' => 'root', Username Software Xampp
80     'password' => '', Password Software Xampp, isi jika ada
81     'database' => 'juragan', 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
```

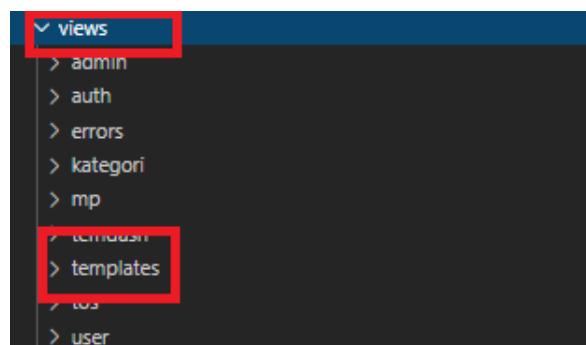
Gambar 4.30 Konfigurasi Database

Apabila *database* juragan teman – teman sudah siap coba kita tampilkan data yang ada pada *table user* tersebut pada sistem CRUD teman – teman. Tapi sebelumnya karena kita disini menggunakan *templates bootstrap* maka teman – teman perlu memecah *templates* tersebut menjadi 4 bagian yaitu *header*, *sidebar*, *topbar*, dan *footer*, hal ini sangat penting loh agar membuat *script code* teman – teman tidak jadi menumpuk. Bagaimana sih cara memecahkannya ?, yuk kita intip dulu *file blank.html* yang ada pada *folder vendor/sbadmin2*.



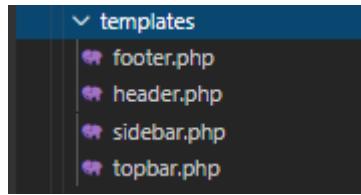
Gambar 4.31 File blank.html

Jika teman – teman perhatikan dimana *file blank.html* tersebut memiliki 5 bagian, yaitu *header*, *sidebar*, *topbar*, *index*, dan *footer*. Keempat *file* tersebut yaitu *header*, *sidebar*, *topbar*, dan *footer* merupakan sebuah *template css* pada *file blank.html* sedangkan *index* merupakan bagian utama dari *file blank.html*. Langkah berikutnya dimana teman – teman harus memisahkan keempat bagian *css* tersebut menjadi satu *view* tersendiri, bagaimana sih caranya ?, yuk ikutin tahap – tahap berikutnya. Silahkan teman – teman buat *folder* baru pada direktori *application/view* dengan nama *templates*.



Gambar 4.32 Membuat Folder Templates

Pada *folder templates* tersebut dimana teman – teman perlu membuat empat file yaitu *header.php*, *sidebar.php*, *topbar.php*, dan *footer.php*. Hal ini berguna untuk memisahkan keempat bagian *css* tersebut menjadi satu file tersendiri.



Gambar 4.33 Memisahkan File Css

Dimana hal pertama yang perlu teman – teman pisahkan adalah bagian *header* yang dimana merupakan bagian awal dari *css* tersebut. Pada dasarnya *header* digunakan untuk membuat judul atau dalam pemrograman biasa kita sebut dengan *tittle* dari suatu sistem tersebut. Untuk memisahkan *header* tersebut teman – teman hanya perlu melakukan *cut* dari *script code* baris 1 – 26 pada *file blank.html*. Lakukan *block* pada baris tersebut lalu *cut* dengan menakan tombol kombinasi “*CTRL+X*”. Pindahkan *script code* tersebut kedalam *file header.php* pada direktori *application/view/templates*.

```
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 4.34 Script Code Css Header

Langkah berikutnya dimana teman – teman perlu memindahkan bagian *sidebar* pada file *blank.html* tersebut. *Sidebar* merupakan sebuah *template* yang biasanya berada pada posisi sebelah kiri pada sistem dengan kolom – kolom yang diberikan fungsi *link navigasi*. Pada file *blank.html* dimana bagian *sidebar* terdapat pada baris 28 – 140. Lakukan hal yang sama dimana teman – teman perlu memindahkan bagian *sidebar* tersebut ke dalam file *sidebar.php* yang sudah teman – teman buat pada direktori *application/view/templates*.

```

28 <!-- Sidebar -->
29 <ul class="navbar-nav bg-gradient-primary sidebar sidebar-dark accordion" id="accordionSidebar">
30
31     <!-- Sidebar - Brand -->
32     <a class="sidebar-brand d-flex align-items-center justify-content-center" href="#">index.html>
33         <div class="sidebar-brand-icon rotate-n-15">
34             <i class="fas fa-laugh-wink"></i>
35         </div>
36         <div class="sidebar-brand-text mx-3">SB Admin <sup>2</sup></div>
37     </a>
38
39     <!-- Divider -->
40     <hr class="sidebar-divider my-0">
41
42     <!-- Nav Item - Dashboard -->
43     <li class="nav-item">
44         <a class="nav-link" href="#">index.html>
45             <i class="fas fa-fw fa-tachometer-alt"></i>
46             <span>Dashboard</span>
47         </a>
48
49     <!-- Divider -->
50     <hr class="sidebar-divider">
51
52     <!-- Heading -->
53     <div class="sidebar-heading">
54         Interface
55     </div>
56
57     <!-- Nav Item - Pages Collapse Menu -->
58     <li class="nav-item">
59         <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseTwo" aria-expanded="true" aria-controls="collapseTwo">
60             <i class="fas fa-fw fa-cog"></i>
61             <span>Components</span>
62         </a>

```

*Gambar 4.35 Script Code Css Sidebar*

Langkah berikutnya dimana teman – teman perlu memindahkan bagian *topbar* pada file *blank.html* tersebut. *Topbar* merupakan bagian atas dari sebuah sistem yang biasanya diisi oleh *form pencarian*, *cart*, *notification*, foto profil serta nama *user*, dan lain – lain. Pada file *blank.html* dimana *script code css topbar* terdapat pada baris 142 – 329. Lakukan hal yang sama dimana teman – teman perlu menindahkan bagian *topbar* tersebut kedalam file *topbar.php* pada direktori *application/view/templates*.

```

142    <!-- Content Wrapper -->
143    <div id="content-wrapper" class="d-flex flex-column">
144        <!-- Main Content -->
145        <div id="content">
146            <!-- Topbar -->
147            <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-top shadow">
148                <!-- Sidebar Toggle (Topbar) -->
149                <button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-circle mr-3">
150                    <i class="fa fa-bars"></i>
151                </button>
152
153                <!-- Topbar Search -->
154                <form class="d-none d-sm-inline-block form-inline mr-auto ml-3 my-2 my-md-0 mw-100 navbar-search">
155                    <div class="input-group group-append">
156                        <input type="text" class="form-control bg-light border-0 small" placeholder="Search for..." aria-label="Search" aria-describedby="basic-addon2">
157                        <div class="input-group-append">
158                            <button class="fas btn btn-primary" type="button">
159                                <i class="fas fa-search fa-sm"></i>
160                            </button>
161                        </div>
162                    </div>
163                </form>
164
165                <!-- Topbar Navbar -->
166                <ul class="navbar-nav ml-auto">
167
168                    <!-- Nav Item - Search Dropdown (Visible Only XS) -->
169                    <li class="nav-item dropdown no-arrow d-sm-none">
170                        <a class="nav-link dropdown-toggle" href="#" id="searchDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
171                            <i class="fas fa-search fa-fw"></i>
172                        </a>
173
174                    <!-- Dropdown - Messages -->
175                    <div class="dropdown-menu dropdown-menu-right p-3 shadow animated--grow-in" aria-labelledby="searchDropdown">
176
177

```

Gambar 4.36 Script Code Css Topbar

Langkah yang terakhir dimana teman – teman perlu memindahkan bagian *footer* pada file *balnk.html*. *Footer* merupakan *css* yang muncul pada bagian bawah yang dimana pada *bootstrap* SB Admin 2 ini diisikan dengan *teks copyriht*. Pada file *blank.html* dimana *script code css footer* tersebut terdapat pada baris 343 – 395. Teman – teman perlu memindahkan *script code* tersebut kedalam *file footer.php* yang sudah dibuat pada direktori *application/view/templates*.

```

343    <!-- Footer -->
344    <footer class="sticky-footer bg-white">
345        <div class="container my-auto">
346            <div class="copyright text-center my-auto">
347                <span>Copyright &copy; Your Website 2019</span>
348            </div>
349        </div>
350    </footer>
351    <!-- End of Footer -->
352
353    </div>
354    <!-- End of Content Wrapper -->
355
356    </div>
357    <!-- End of Page Wrapper -->
358
359    <!-- Scroll to Top Button-->
360    <a class="scroll-to-top rounded" href="#page-top">
361        <i class="fas fa-angle-up"></i>
362    </a>

```

Gambar 4.37 Scrip Code Css Footer

Ketika semua *css* sudah dipindahkan dimana isi dari *file blank.html* tersebut hanya menyisakan bagian *index*-nya saja. Bagian *index* tersebut terdapat pada baris 331 – 341. Biarkan saja *script code* tersebut yang dimana akan kita gunakan pada latihan CRUD pada tahap membuat tampilan profil.

```
331 | | | | | <!-- Begin Page Content -->
332 | | | | | <div class="container-fluid">
333 | | | | | |
334 | | | | | <!-- Page Heading -->
335 | | | | | <h1 class="h3 mb-4 text-gray-800">Blank Page</h1>
336 | | | | |
337 | | | | | </div>
338 | | | | | <!-- /.container-fluid -->
339 | | | | |
340 | | | | | </div>
341 | | | | | <!-- End of Main Content -->
```

Gambar 4.38 Script Code Index

Setelah semua *css* sudah dipisahkan dimana teman – teman perlu melakukan konfigurasi terhadap *file header.php* dan *footer.php* agar *css* tersebut dapat terpanggil. Silahkan buka *file header.php* lalu tambahkan fungsi *base url* seperti pada *script code* 4.4.

```
<title><?= $title; ?></title>

<!-- Custom fonts for this template-->
<link href=<?= base_url(); ?>assets/vendor/font
awesome-
free/css/all.min.css" rel="stylesheet" type="text/cs
s">
    <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,7
00i,800,800i,900,900i" rel="stylesheet">

<!-- Custom styles for this template-->
```

```
<link href="= base_url(); ?&gt;assets/css/sb-admin-2.min.css" rel="stylesheet"&gt;

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

Script Code 4.4 Memberikan Fungsi Base URL Pada Header

Langkah berikutnya dimana teman – teman perlu menambahkan fungsi *base url* tersebut pada file *footer.php*. Perhatikan *script code* 4.5 dibawah. Dimana *base url* tersebut merupakan salah satu fungsi *helpers* pada *codeigniter*.

```
<!-- 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;</pre
```

Script Code 4.5 Memberikan Fungsi Base URL pada Footer

#### 4.4.1 Membuat *Controller*

Apabila teman – teman sudah melakukan konfigurasi pada file *codeiginter*, membuat *database*, dan memisahkan *file css bootstrap* menjadi satu *file* dimana teman – teman menyiapkan *controller* terlebih dahulu. Dikarenakan kita sudah membuat *file controller User.php* pada *sctipt code* 4.1 mari kita modifikasi *script code* tersebut seperti pada *script code* 4.6.

```
<?php  
defined('BASEPATH') or exit('No direct script access  
allowed');  
  
class User extends CI_Controller  
{  
    public function index()  
    {  
        $data['title'] = 'Data User';  
  
        $this->load->view('templates/header', $data);  
        $this->load->view('templates/sidebar');  
        $this->load->view('templates/topbar');  
        $this->load->view('user/user_profile');  
        $this->load->view('templates/footer');  
    }  
}
```

Script Code 4.6 Merubah Code Controller User

Pada *codeigniter* dimana variabel ditulis dengan menggunakan logo “\$”. Dapat kita lihat pada *script code 4.6* dimana telah bertambah variabel data yang diisikan dengan *tittle*. Variabel tersebut akan kita panggil pada bagian *template header*. Selanjutnya dimana kita akan menambahkan variabel baru yang dimana fungsinya sebagai pemanggilan *model* dan akan kita panggil pada *view user\_profile*.

```

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

class User extends CI_Controller
{
    public function index()
    {
        $data['title'] = 'Data User';
        $data['user_data'] = $this->m_user->t_user()
        ->result();

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

```

*Script Code 4.7 Menambahkan Varibel Baru*

#### **4.4.2 Membuat *Model***

Langkah selanjutnya dimana kita akan membuat *model*. Model biasanya berhubungan erat dengan *database*. Silahkan teman – teman membuat *file* baru pada direktori *application/models* beri nama *m\_user.php*. Isi dari *file m\_user.php* dapat dilihat pada *script code 4.8*.

```

<?php

class M_user extends CI_Model
{
    //Merupakan model user
}

```

*Script Code 4.8 Membuat Model User*

Pada model *user* tersebut dimana teman – teman perlu membuat *function* yang berguna untuk memanggil data *user* dari tabel *user*. Buat

*function* dengan nama “*t\_user*” dan berikan *query read* kepada tabel user seperti pada *script code 4.9*.

```
<?php

class M_user extends CI_Model
{
    //Merupakan model user

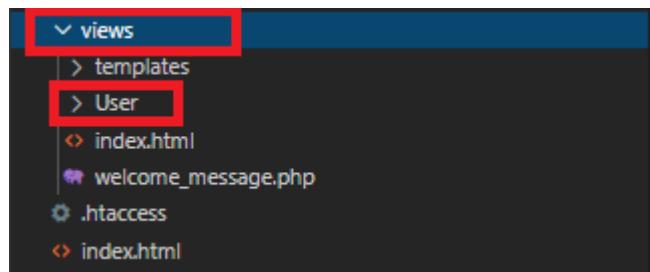
    public function t_user()
    {
        return $this->db->get('user');
    }

}
```

*Script Code 4.9 Membuat Function t\_user Pada Model*

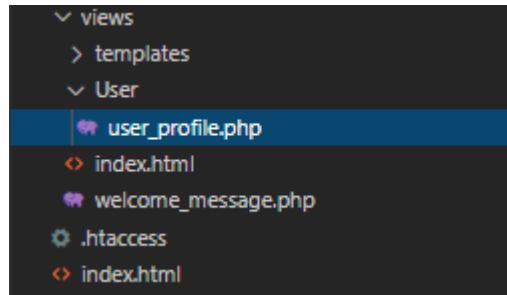
#### 4.4.3 Membuat View

Langkah selanjutnya dimana teman – teman perlu membuat *view* untuk menampilkan data yang ada pada *database* juragan. Untuk membuat *view* silahkan teman – teman masuk ke direktori *application/view*, lalu teman – teman buat *folder* baru dengan nama *user*.



*Gambar 4.39 Membuat Folder User Pada View*

Pada *folder* user tersebut teman – teman akan menyimpan semua *file* yang berkaitan dengan *user*. Buat *file* baru dengan nama *user\_profile.php* dan simpan pada *folder* user tersebut.



Gambar 4.40 Membuat file *user\_profile.php* Pada Foler User

Buka *file* *user\_profile.php* tersebut, sekarang coba teman – teman panggil *file* tersebut. Berikan *script code index* pada *file blank.html* pada *file* *user\_profile.php* lalu *save*.

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

    <!-- Page Heading -->
    <h1 class="h3 mb-4 text-gray-800">Blank Page</h1>

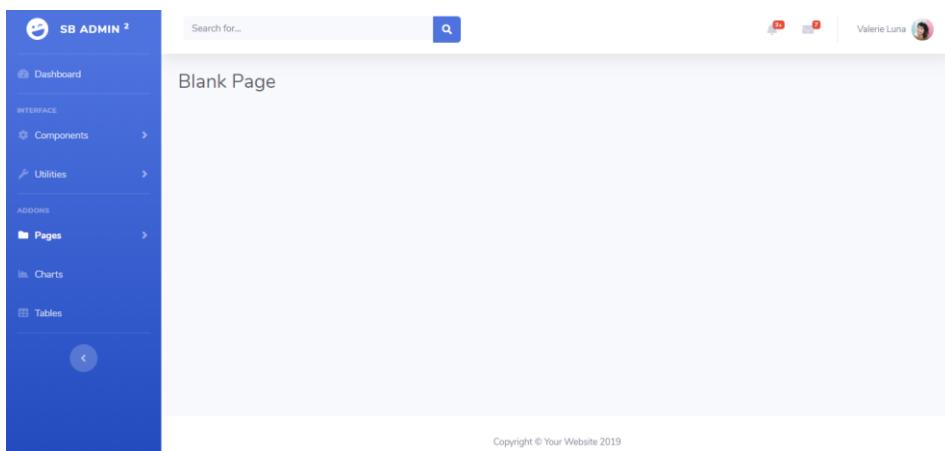
</div>
<!-- /.container-fluid -->

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

Script Code 4.10 Memindahkan File *index* Ke *user\_profile.php*

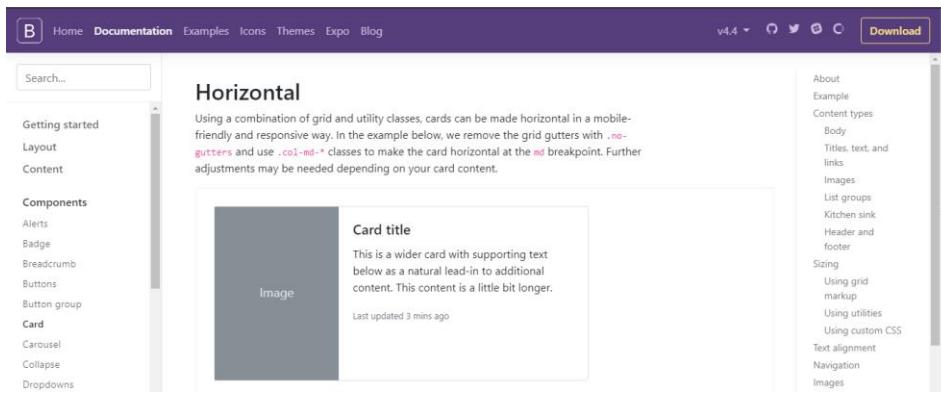
Buka *web broser* yang teman – teman gunakan sebagai contoh disini saya menggunakan *Chrome*. Panggil *controller* *User.php*, kok yang di panggil *controller* sih ?, karena untuk memanggil dimana teman – teman harus mendeskripsikan *file* *user\_profile.php* pada *controller*,

lihat *script code 4.6*. Bagaimana sih cara memanggilnya ?, pastikan terlebih dahulu *software Xampp* teman – teman sudah jalan yah, lalu untuk memanggilkan dimana teman – teman hanya perlu mengetikkan ***localhost/nama\_folder/nama\_controller/nama\_function*** pada kolom pencarian *web browser*. Agar lebih jelas dalam cara pemanggilan silahkan teman – teman ketik ***localhost/juragan/user/index*** maka *view* yang akan ditampilkan akan nampak seperti pada gambar 4.41.



Gambar 4.41 View *user\_profile* Menggunakan *Script Code Index*

Sekarang gimana kalau kita rubah *script code user\_profile.php* menjadi seperti pada *script code 4.11*. Untuk membuat *view* tersebut dimana teman – teman membutuhkan fungsi *card* pada *bootstrap* agar tampak lebih menarik. Silahkan teman – teman kunjungi *link* <https://getbootstrap.com/docs/4.4/components/card/> lalu scroll kebawah hingga menemukan “*Horizontal*”.



Gambar 4.42 Card Style Boostrap

Pada dokumentasi *bootstrap* sudah memberikan *script code* jadi teman – teman hanya tinggal meng-*copy* saja *script code* tersebut, gimana mudah kan ?.

```
<div class="card mb-3" style="max-width: 540px;">
  <div class="row no-gutters">
    <div class="col-md-4">
      
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">This is a wider card with supporting text below as a natural lead-in to ac</p>
        <p class="card-text"><small class="text-muted">Last updated 3 mins ago</small></p>
      </div>
    </div>
  </div>
</div>
```

Gambar 4.43 Script Code Style Card Boostrap

Paste *script code* style *card bootstrap* tersebut pada file *user\_profile.php* lalu lakukan modifikasi seperti pada *script code 4.11*.

```

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

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

    <div class="row">
        <div class="col-lg-8">
            <?php foreach ($user_data as $user) : ?>
            </div>
        </div>

        <div class="card mb-3 col-lg-8">
            <div class="row no-gutters">
                <div class="col-md-4">
                    
                </div>
                <div class="col-md-8">
                    <div class="card-body">
                        <h5 class="card-title">
                            <?= $user->name ?>
                        </h5>
                        <p class="card-text">
                            <?= $user->email ?>
                        </p>
                        <p class="card-text">
                            <?= $user->alamat ?>
                        </p>
                        <p class="card-text">
                            <small class="text-muted"> Terdaftar sejak
                            <?= date('d F Y', $user->date_created); ?>
                            </small>
                        </p>
                        <?= anchor('user/edit' . $user->$id , '<div class="btn btn-sm btn-info">Edit Profil</div>') ?>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

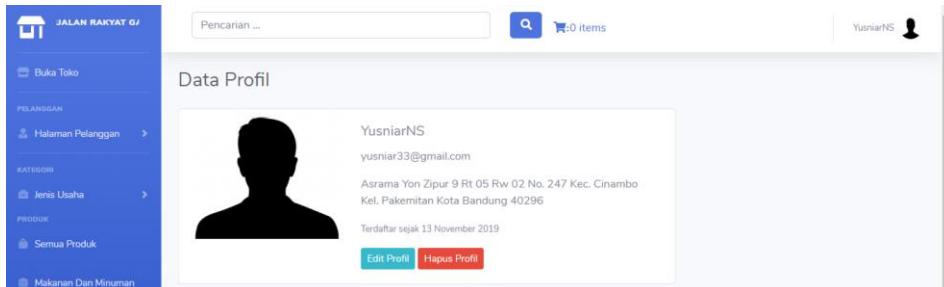
```

<?= anchor('user/hapus ,
    '<div class="btn btn-sm btn-info">
        Hapus Profil</div>')
?>
</div>
</div>
</div>
<?php endforeach; ?>
</div>
</div>
<!-- /.container-fluid →
</div>
<!--End of Main Content →

```

*Script Code 4.11 Mofikasi codingan view user\_profile*

Apabila teman – teman sudah melakukan modifikasi pada file *user\_profile.php* jangan lupa untuk di *save*. Lakukan *refresh browser* selamat teman – teman sudah berhasil mengaplikasikan fungsi *read*.



*Gambar 4.44 View user\_profile.php Setelah Modifikasi*

#### 4.4.4 Membuat Fungsi Hapus

Setelah berhasil membuat fungsi *read* dimana sekarang teman – teman belajar membuat fungsi *delete* atau *hapus* pada *framework codeigniter*. Dimana kita akan membuat fungsi hapus berdasarkan *id*, silahkan teman – teman kembali pada file *controller User.php*. Pada

*file controller User.php* dimana teman – teman perlu menambahkan *function* baru, beri nama hapus.

```
public function hapus($id)
{
    $where = array('id' => $id);
    $this->m_user->hapus_data($where, 'user');

    redirect('user');
}
```

*Script Code 4.12 Contolle Function Hapus*

Setelah menambahkan *function controller* dimana teman – teman perlu menambahkan *function* baru juga terhadap *file models/m\_user.php*. Silahkan buka terlebih dahulu *file m\_user.php*, lalu tambahkan *function* *hapus\_data* seperti pada *script code 4.13*.

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

*Script Code 4.13 Models Function hapus\_data*

#### **4.4.5 Membuat Fungsi *Update/Edit***

Setelah berhasil membuat fungsi *read* dan *delete* sekarang teman – teman akan belajar membuat fungsi *update* atau edit pada *framework codeigniter*. Hal pertama yang perlu teman – teman lakukan adalah menambahkan dua *funtion* baru pada *controller User.php*, beri nama *edit* dan *update*. Perhatikan *script code 4.14*.

```

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

    $data['title'] = 'Edit Profil';
    $data['user_data'] = $this->m_user
        ->edit_user($where, 'user')->result();

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

public function update()
{
    $data['user'] = $this->db->get_where('user');

    $name = $this->input->post('name');
    $email = $this->input->post('email');
    $alamat = $this->input->post('alamat');

    // cek jika ada gambar yang akan di upload
    $upload_image = $_FILES['image']['name'];

    if ($upload_image) {
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size']      = '2048';
        $config['upload_path']   = './assets/img/
            profile/';

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

        if ($this->upload->do_upload('image'))
        {
            $old_image = $data['user']['image'];
            if ($old_image != 'default.jpg')
            {
                unlink(FCPATH . 'assets/img/profile/' .

```

```

        $old_image);
    }

    $new_image = $this->upload
        ->data('file_name');
    $this->db->set('image', $new_image);

} else {

    echo $this->upload->display_errors();
}
}

$this->db->set('name', $name);
$this->db->set('alamat', $alamat);
$this->db->where('email', $email);
$this->db->update('user');

redirect('user');

}
}

```

*Script Code 4.14 Menambahkan Function Edit Dan Update*

Langkah selanjutnya dimana teman – teman perlu menambahkan *function* baru pada *file m\_user* dengan nama *edit\_user*. *Function* tersebut berfungsi untuk menampilkan isi pada tabel *user* dalam *database* berdasarkan *id*, perhatikan *script code 4.15*.

```

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

```

*Script Code 4.15 Menambahkan Function edit\_user Pada Models*

Langkah selanjutnya dimana teman – teman perlu membuat *view* agar halaman *edit\_profile* dapat di akses. Dimana Teman – Teman

perlu membuat sebuah file baru di dalam direktori *application/view/user/edit\_profile.php*. Pada file *edit\_profile.php* isikan codingan seperti pada *script code 4.16*.

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

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

    <div class="row">
        <div class="col-lg-9">

            <?php foreach ($user) : ?>

            <?= form_open_multipart('user/update'); ?>
            <div class="form-group row">
                <label for="email" class="col-sm-2
                    col-form-label">Email</label>
                <div class="col-sm-10">
                    <input type="text" class="form-control"
                        id="email" name="email"
                        value="<?= $user->email ?>" readonly>
                </div>
            </div>

            <div class="form-group row">
                <label for="name" class="col-sm-2
                    col-form-label">Nama Lengkap</label>
                <div class="col-sm-10">
                    <input type="text" class="form-control"
                        id="name" name="name"
                        value="<?= $user->name ?>">
                </div>
            </div>

            <div class="form-group row">
```

```

<label for="name" class="col-sm-2
    col-form-label">Alamat</label>
<div class="col-sm-10">
<textarea class="form-control"
    id="alamat" name="alamat">
<?= $user->alamat ?>
</textarea>
</div>
</div>

<div class="form-group row">
<div class="col-sm-2">Foto Profil</div>
<div class="col-sm-10">
<div class="row">
<div class="col-sm-3">

</div>
<div class="col-sm-9">
<div class="custom-file">
<input type="file" class="custom-file-input"
        id="image" name="image">
<label class="custom-file-label"
        for="image">Pilih file</label>
</div>
</div>
</div>
</div>
</div>

<div class="form-group row justify-content-end">
<div class="col-sm-10">
<button type="submit" class="btn btn-primary">
    Edit Profil
</button>
<?= anchor('user/user_profile' ,
    '<div class="btn btn-warning">Kembali</div>') ?>
</div>
</div>
</form>

```

```

<?php endforeach; ?>
</div>
</div>
</div>
<!-- /.container-fluid -->

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

```

*Script Code 4.16 View Edit Profil*

#### 4.4.6 Membuat Fungsi *Create*

Setelah teman – teman berhasil membuat fungsi *read*, *update*, dan *delete*, langkah berikutnya dimana teman – teman akan belajar membuat fungsi *create* pada *framework codeigniter*. Langkah awal untuk membuat fungsi *create* dimana teman – teman perlu menambahkan *function input* pada *controller User.php*, perhatikan *script code 4.17*.

```

public function input()
{
    $data['title'] = 'Input User';

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

    $name      = $this->input->post('nama_brg');
    $email     = $this->input->post('email');
    $alamat   = $this->input->post('alamat');
    $image     = $_FILES['image']['name'];
}

```

```

if ($image = '') {

} else {

$config['upload_path'] = './assets/img/profile';
$config['allowed_types'] = 'jpg|jpeg|png|gif';

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

$data = array(
    'name'      => $name,
    'email'     => $email,
    'alamat'    => $alamat,
    'image'     => $image
);

$this->m_user->inputdata($data, 'user');

redirect('user');
}

```

*Script Code 4.17 Function Input Pada Controller*

Setelah membuat *function input* pada *controller* dimana teman – teman perlu membuat *function inputdata* pada *m\_user* yang dimana berfungsi untuk mengeksekusi *query create* terhadap *database juragan*. Silahkan teman – teman buka kembali *file m\_user.php* pada direktori *application/models* dan tambahkan *funtion inputdata*, perhatikan *script code 4.18*.

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

Script Code 4.18 Function inputdata Pada Model

Langkah berikutnya dimana teman – teman perlu membuat *view input*. Silahkan teman – teman menuju ke direktori *application/view/user* lalu buat file baru dengan nama *input\_user.php* lalu tambahkan *script code* 4.19.

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

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

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

    <div class="form-group">
        <label>Nama</label>
        <input type="text" name="name"
               class="form-control" >
    </div>

    <div class="form-group">
        <label>Email</label>
        <input type="text" name="email"
               class="form-control" >
    </div>

    <div class="form-group">
        <label>Alamat</label>
        <textarea type="text" name="alamat"
                  class="form-control" >
        </textarea>
    </div>

```

```
<div class="form-group">
<label>Foto Profil</label><br>
    <input type="file" name="image"
        class="form-control">
</div>

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

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

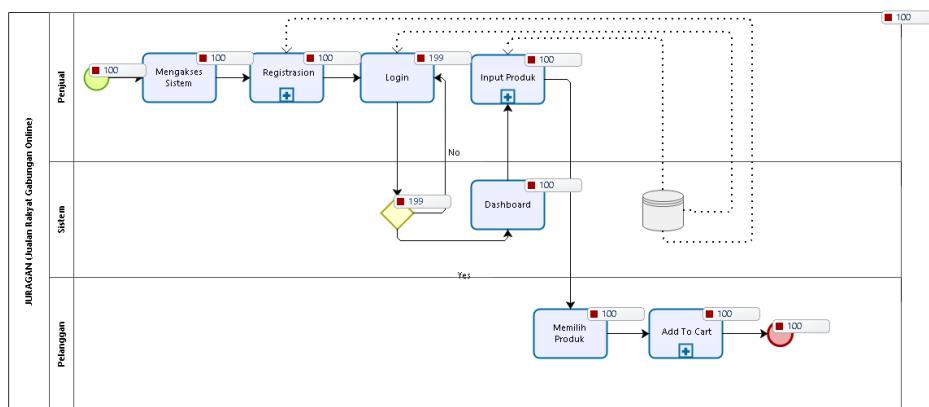
</div>
<!-- End of Main Content -->
```

## **BAB V**

## **PERANCANGAN**

### **4.1 BPMN Sistem JURAGAN**

Dalam membuat sebuah sistem dimana perlu dilakukan tahap perancangan terlebih dahulu. Perancangan sistem dibuat dengan tujuan untuk menghindari kesalahan dalam pembuatan sistem sehingga sistem dapat berjalan dengan baik. Dimana penulis telah membuat sebuah analisis *business process model and notation (BPMN)* pada sistem JURAGAN adalah sebagai berikut :



Gambar 4.1 BPMN Sistem JURAGAN

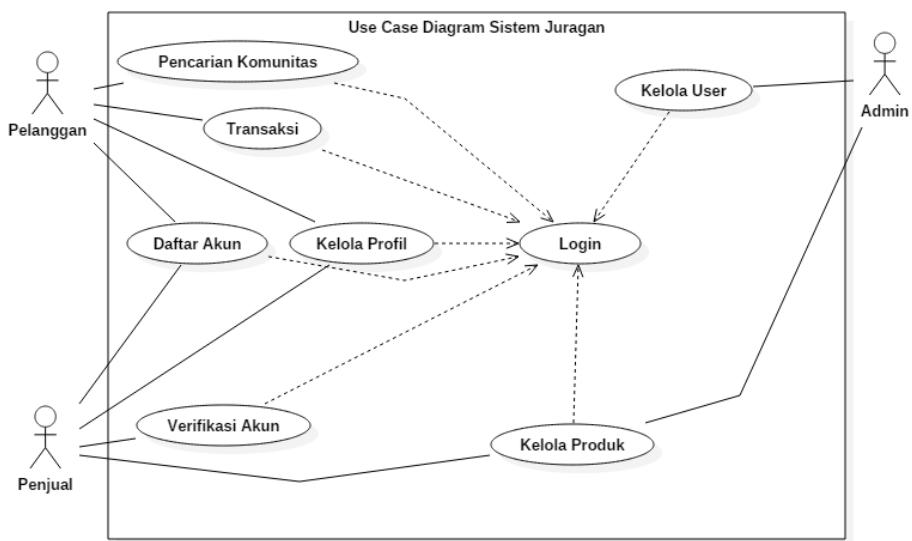
Tabel 4.1 Hasil Result BPMN Sistem JURAGAN

Name	Type	Instances Completed
Sistem JURAGAN	Process	100
NoneStart	Start Event	100

Mengakses Sistem	<i>Task</i>	100
<i>Login</i>	<i>Task</i>	199
<i>ExclusiveGateway</i>	<i>Gatewat</i>	199
<i>Dashboard</i>	<i>Task</i>	100
Memilih Produk	<i>Task</i>	100
<i>NoneEnd</i>	<i>End Event</i>	100
<i>Registration</i>	<i>Task</i>	100
Input Produk	<i>Task</i>	100
<i>Add To Cart</i>	<i>Task</i>	100

## 4.2 Use Case JURAGAN

*Use Case Diagram* merupakan salah satu diagram yang diklasifikasikan ke dalam aspek perilaku, dimana deskripsi perilaku dari setiap *use case* dijelaskan secara detail dan terpisah dengan menggunakan *document* secara *tekstual* yaitu *user case scenario*. Adapun *use case* dari sistem JURAGAN tersebut sebagai berikut :



Gambar 4.2 Use Case Sistem JURAGAN

#### 4.2.1 Definisi Aktor

Tabel 4.2 Definisi Aktor Use Case

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
---	-------	------------------------------------

#### 4.2.2 Definisi Use Case

Tabel 4.3 Definisi Use Case

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

		menginputkan No KTP dan <i>upload</i> KTP
7	Kelola Produk	<ul style="list-style-type: none"> <li>a. Melakukan pengelolaan produk terhadap akun level toko.</li> <li>b. Melakukan <i>view</i> produk terhadap akun level pelanggan.</li> </ul>
8	Kelola User	<ul style="list-style-type: none"> <li>a. Melakukan pengelolaan akun <i>user</i> oleh admin.</li> </ul>

#### 4.2.3 Use Case Scenario

Tabel 4.4 Use Case Scenario Daftar Akun

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

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

Tabel 4.5 Use Case Scenario Login

<b>Identifikasi</b>	
No.	JR2

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

*Tabel 4.6 Use Case Scenario Kelola Profil*

Identifikasi	
No.	JR3
Nama	Kelola Profil

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

*Tabel 4.7 Use Case Scenario Pencarian Komunitas*

Identifikasi	
No.	JR4
Nama	Pencarian Komunitas
Tujuan	Mencari produk berdasarkan komunitas.
Deskripsi	Melakukan pencarian produk dari <i>keyword</i> komunitas yang dikirim.

Aktor	Pelanggan
<b>Skenario</b>	
Kondisi Awal	<i>Topbar</i> halaman utama
<b>Aksi Aktor</b>	<b>Reaksi Sistem</b>
1. Melakukan <i>input keyword</i> komunitas.	a. Menampilkan produk berdasarkan <i>keyword</i> komunitas yang dikirim.

Tabel 4.8 Use Case Scenario Transaksi

<b>Identifikasi</b>	
No.	JR5
Nama	Transaksi
Tujuan	Menambahkan produk ke dalam keranjang belanja.
Deskripsi	Menambahkan produk ke dalam keranjang belanja untuk melakukan proses transaksi.
Aktor	Pelanggan
<b>Skenario</b>	
Kondisi Awal	Halaman Produk
<b>Aksi Aktor</b>	<b>Reaksi Sistem</b>

1. Memberikan aksi terhadap <i>button</i> tambah ke keranjang	a. Mengirimkan <i>id</i> produk yang ditambahkan.
2. -	b. Melakukan validasi
3. -	c. Menambahkan produk ke dalam keranjang.

Tabel 4.9 Use Case Scenario Verifikasi Akun

<b>Identifikasi</b>	
No.	JR6
Nama	Verifikasi Akun
Tujuan	Mengaktifkan akun toko.
Deskripsi	Mengaktifkan akun toko dengan menginputkan No KTP dan <i>upload</i> foto KTP.
Aktor	Penjual
<b>Skenario</b>	
Kondisi Awal	Halaman Verifikasi Akun
<b>Aksi Aktor</b>	
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 4.10 Use Case Scenario Kelola Produk Sub Penjual

<b>Identifikasi</b>	
No.	JR7
Nama	Kelola Produk
Tujuan	Mengelola data produk
Deskripsi	Melakukan pengelolaan terhadap data produk.
Aktor	Penjual
<b>Skenario</b>	
Kondisi Awal	Halaman Inventori
<b>Aksi Aktor</b>	
1. Memberikan aksi terhadap <i>button</i> tambah produk	a. Memunculkan <i>pop up form</i> <i>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</i> edit produk
7. Melakukan edit data produk.	g. -
8. Memberikan aksi pada <i>button edit produk</i>	h. Melakukan validasi terhadap data yang dikirim
9. -	i. Memberikan notifikasi produk berhasil di edit

Tabel 4.11 Use Case Scenario Kelola Produk Sub Admin

Identifikasi	
No.	JR8
Nama	Kelola Produk
Tujuan	Mengelola data produk
Deskripsi	Melakukan pengelolaan terhadap data produk.

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

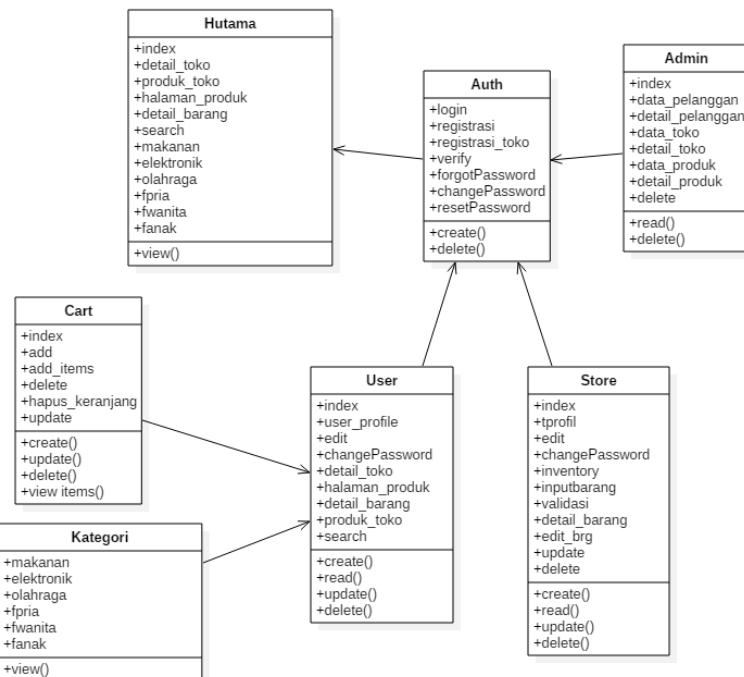
Tabel 4.12 Use Case Scenario Kelola User

<b>Identifikasi</b>	
No.	JR9
Nama	Kelola User
Tujuan	Mengelola user yang telah terdaftar
Deskripsi	Melakukan pengelolaan data terhadap <i>user</i> yang terdaftar
Aktor	Admin
<b>Skenario</b>	
Kondisi Awal	Halaman Admin
<b>Aksi Aktor</b>	<b>Reaksi Sistem</b>

1. Memilih menu <i>user</i> pada <i>sidebar</i>	a. Menampilkan halaman user
2. -	b. Mengambil data dari tabel <i>user</i>
3. -	c. Menampilkan data <i>user</i>

### 4.3 Class Diagram Sistem JURAGAN

*Class* diagram adalah sebuah spesifikasi yang jika tidak diinstansiasi akan menghasilkan sebuah obyek dan merupakan inti dari pengembangan dan desain berorientasi obyek. Adapun *class* diagram dari sistem JURAGAN adalah sebagai berikut :

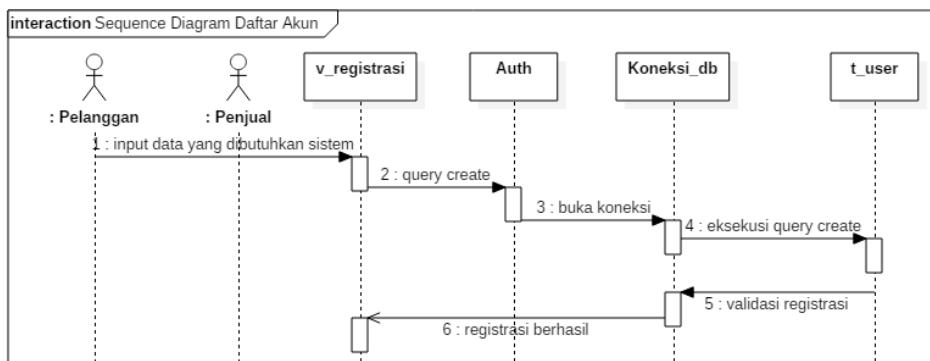


Gambar 4.3 Class Diagram Sistem JURAGAN

## 4.4 Sequence Diagram Sistem JURAGAN

*Sequence* diagram adalah suatu diagram yang memperlihatkan atau menampilkan interaksi – interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Adapun *sequence* diagram dari sistem JURAGAN adalah sebagai berikut:

### 1). Sequence Diagram Daftar Akun

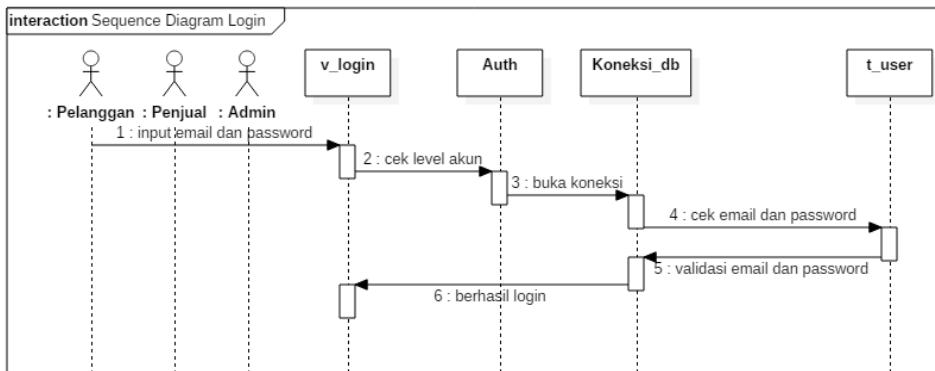


Gambar 4.4 Sequence Daftar Akun

Keterangan :

1. Pelanggan dan Penjual akan melakukan *input* data yang dibutuhkan.
2. *View registrasi* akan membuat *query create* melalui *controller Auth*.
3. *Controller Auth* akan membuka koneksi ke *database*.
4. Eksekusi *query create* akan di lakukan kepada tabel *user*.
5. Validasi *registrasi* akan di lakukan oleh tabel *user*.
6. *Registrasi* berhasil

## 2). Sequence Login

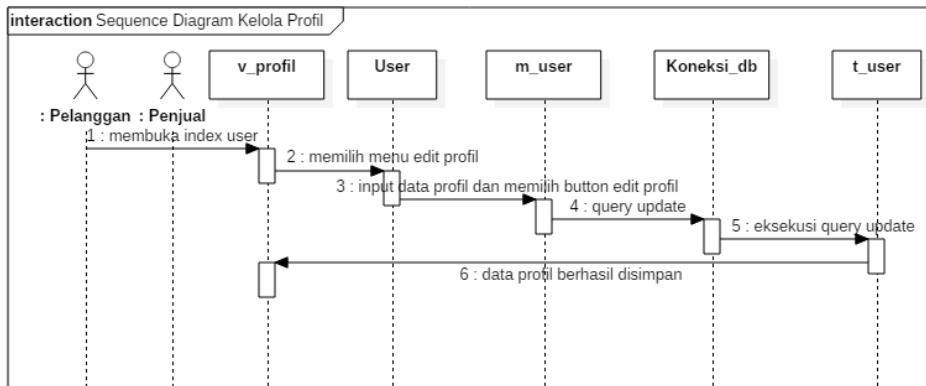


Gambar 4.5 Sequence Login

Keterangan :

1. Pelanggan, Penjual, dan admin akan melakukan *input email* dan *password*.
2. Sistem akan melakukan cek level akun pada *controller Auth*.
3. *Controller Auth* akan membuka koneksi *database*.
4. Di dalam *database* akan dilakukan proses pengecekan *email* atau *password* yang sudah terdaftar.
5. *Database* juga akan melakukan *validasi* terhadap tabel *user*.
6. Apabila lolos maka *user* berhasil melakukan *login*.

### 3). Sequence Kelola Profil

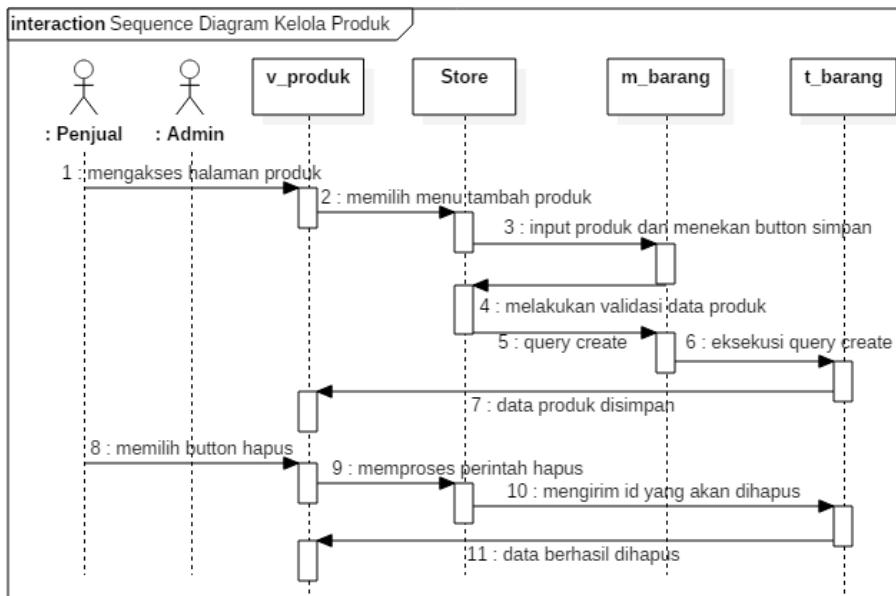


Gambar 4.6 Sequence Kelola Profil

Keterangan :

1. *User* akan mengakses halaman *index* nya masing - masing
2. *User* akan memilih menu edit profil
3. *User* melakukan *input* data profil dan apabila sudah *user* akan memilih *button* edit profil
4. *Model user* akan memberikan *query update* kepada *database*
5. *Database* akan mengeksekusi *query update*
6. Data profil berhasil disimpan.

#### 4). Sequence Kelola Produk



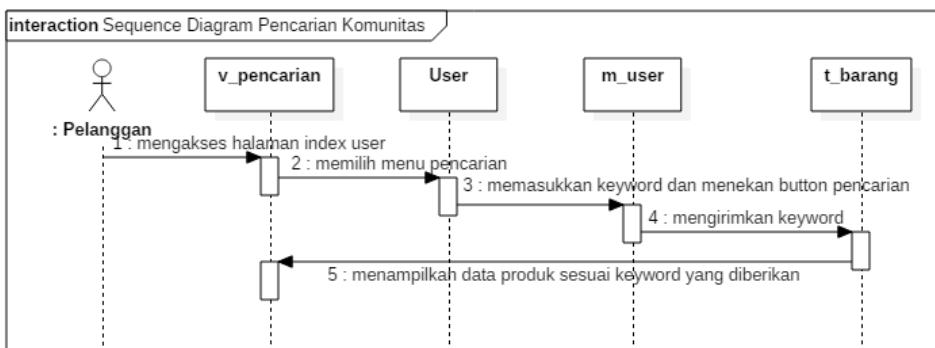
Gambar 4.7 Sequence Kelola Produk

Keterangan :

1. Penjual akan mengakses halaman produk
2. Penjual akan memilih menu tambah produk
3. Penjual akan melakukan *input* data produk dan menekan button simpan
4. *Model* barang akan memproses data yang dikirim dan akan di validasi
5. *Controller* mengirimkan *query create*
6. *Model* barang akan melakukan eksekusi terhadap *query create* tersebut kedalam tabel barang
7. Data produk berhasil di simpan admin dan penjualan dapat melihat pada halaman produk

8. Penjual dan Admin memilih *button* hapus
9. Perintah hapus akan di proses
10. *Controller* akan mengirimkan *id* yang akan di hapus
11. Data berhasil di hapus

### 5). Sequence Pencarian Komunitas

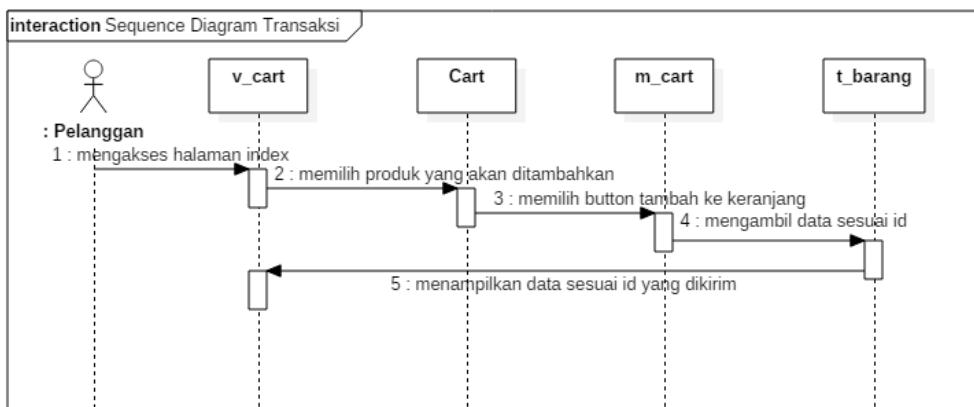


Gambar 4.8 Sequence Pencarian Komunitas

Keterangan :

1. Pelanggan akan mengakses halaman *index*
2. Pelanggan akan memilih menu pencarian
3. Pelanggan menginputkan *keyword* komunitasnya dan menekan *button* pencarian
4. *Model user* akan mengirimkan *keyword* kepada tabel barang
5. Tabel barang akan menampilkan data barang sesuai *keyword* yang diberikan

## 6). Sequence Transaksi

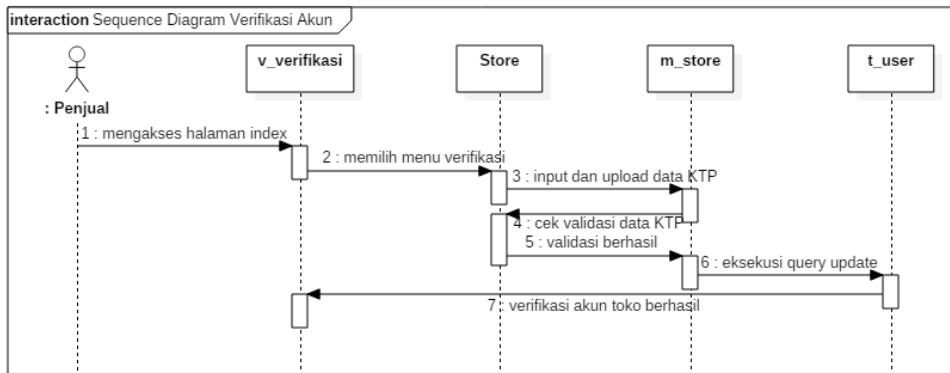


Gambar 4.9 Sequence Transaksi

Keterangan :

1. Pelanggan akan mengakses halaman *index*
2. Pelanggan akan memilih produk yang akan ditambahkan kedalam keranjang belanja
3. Pelanggan akan memilih *button tambah* ke keranjang berdasarkan *id* produk yang dikirim
4. *Mode cart* akan mengambil data dari tabel barang sesuai *id* yang dikirim
5. Produk berhasil ditambahkan kedalam *cart*

## 7). Sequence Verifikasi Akun

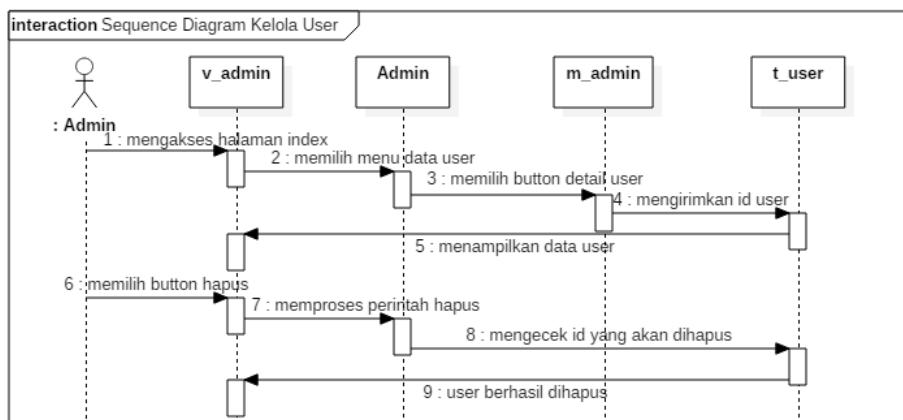


Gambar 4.10 Sequence Verifikasi Akun

Keterangan :

1. Penjual akan mengakses halaman *index*
2. Penjual akan memilih menu verifikasi
3. Penjual akan melakukan *input* nomor KTP dan melakukan *upload* KTP
4. *Controller store* aka melakukan validasi terhadap data KTP tersebut
5. Validasi berhasil akan dikirimkan kepada *model store*
6. *Model Store* akan mengeksekusi *query update* kepada tabel *user*
7. Verifikasi akan toko berhasil

## 8). Sequence Diagram Kelola User



Gambar 4.11 Sequence Diagram Kelola User

Keterangan :

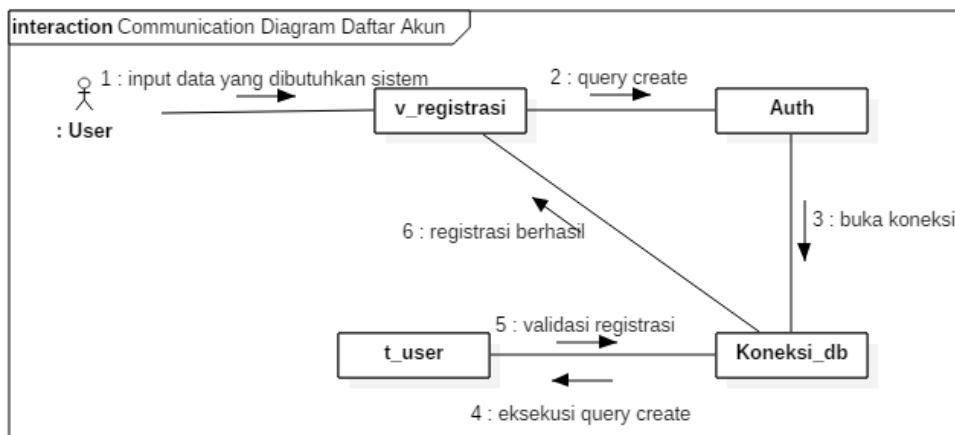
1. Admin akan mengakses halaman *index*
2. Admin akan memilih menu *data user*
3. Admin akan memilih *button detail user*
4. *Model admin* akan mengirimkan *id user* kepada tabel *user*
5. Tabel *User* akan menampilkan data *user*
6. Admin memilih *button hapus*
7. *Controller* akan memproses perintah hapus
8. *Controller* akan melakukan cek *id* yang akan dihapus
9. *User berhasil dihapus*

## 4.5 Collaboration Diagram Sistem JURAGAN

*Collaboration diagram* yaitu dimana sebuah pengelompokan pesan terhadap kumpulan diagram sekuen menjadi sebuah diagram. Pada *collaboration diagram* dimana terdapat *method* yang dijalankan antara objek yang satu dan objek lainnya. Dalam *collaboration diagram* tersebut diaman objek harus melakukan sinkronasi pesan dengan serangkaian pesan

– pesan lainnya. Adapun *Collaboration* diagram pada sistem JURAGAN adalah sebagai berikut :

### 1). *Collaboration* Daftar Akun

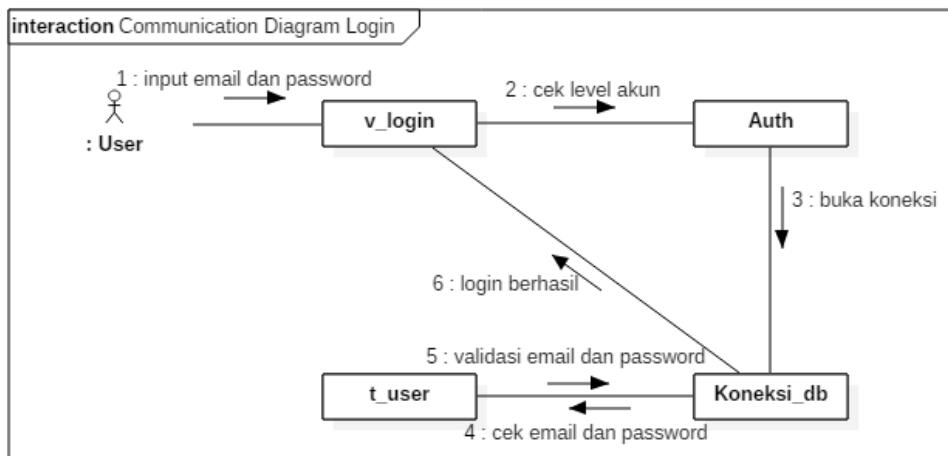


Gambar 4.12 *Collaboration* Daftar Akun

Keterangan :

1. Pelanggan dan Penjual akan melakukan *input* data yang dibutuhkan.
2. *View registrasi* akan membuat *query create* melalui *controller Auth*.
3. *Controller Auth* akan membuka koneksi ke *database*.
4. Eksekusi *query create* akan di lakukan kepada tabel *user*.
5. Validasi *registrasi* akan di lakukan oleh tabel *user*.
6. *Registrasi* berhasil.

## 2). Collaboration Login

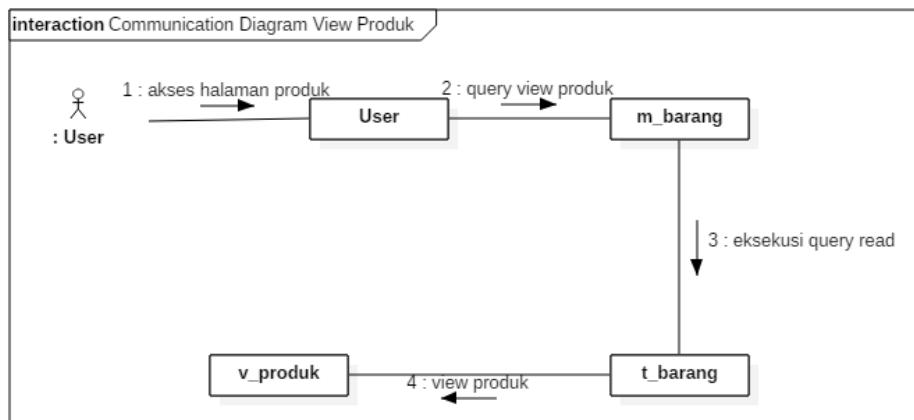


Gambar 4.13 Collaboration Login

Keterangan :

1. Pelanggan dan Penjual akan melakukan *input email* dan *password*.
2. Sistem akan melakukan cek level akun pada *controller Auth*.
3. *Controller Auth* akan membuka koneksi *database*.
4. Di dalam *database* akan dilakukan proses pengecekan *email* atau *password* yang sudah terdaftar.
5. *Database* juga akan melakukan *validasi* terhadap tabel *user*.
6. Apabila lolos maka *user* berhasil melakukan *login*.

### 3). Collaboration Kelola Produk

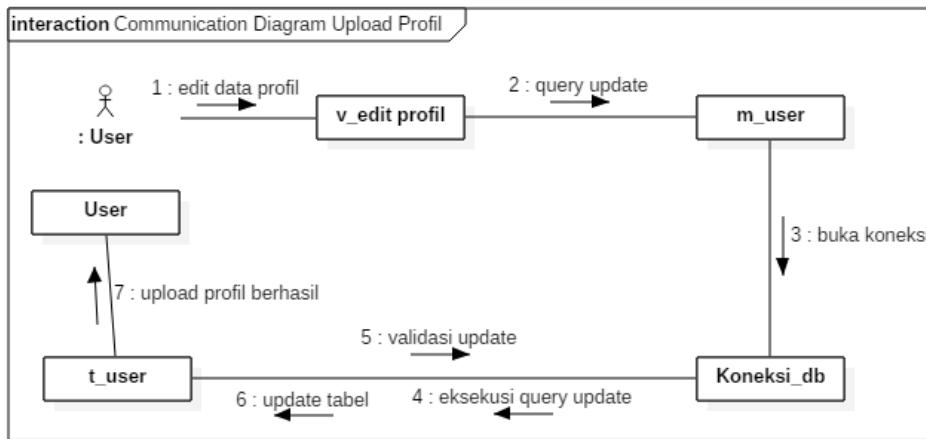


Gambar 4.14 Collaboration Kelola Produk

Keterangan :

1. *User* akan melakukan akses terhadap halaman produk.
2. *Controller auth* akan menampilkan halaman produk dan mengirimkan *query view* produk.
3. Pada model barang dimana *query view* tersebut di eksekusi kepada tabel barang.
4. Tabel barang akan melakukan *view produk* ke halaman *view produk*

#### 4). Collaboration Kelola Profil

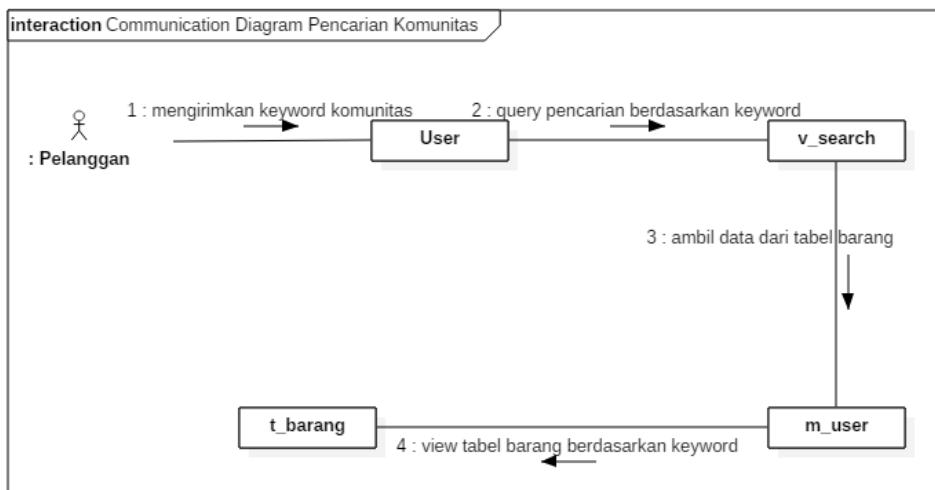


Gambar 4.15 Collaboration Kelola Profil

Keterangan :

1. *User* akan mengakses halaman *edit profil* dan merubah profil mereka pada halaman tersebut.
2. *User* akan menekan button *edit* dan *query update* akan dikirim kepada model *user*.
3. *Model user* akan melakukan akses terhadap *database*.
4. *Database* akan melakukan eksekusi *query update* kepada tabel *user*.
5. Tabel *user* akan melakukan validasi.
6. *Database* akan merubah data pada tabel *user* melalui *query update*.
7. Data akan tersimpan pada tabel *user* dan *update* profil berhasil.

## 5). Collaboration Pencarian Komunitas

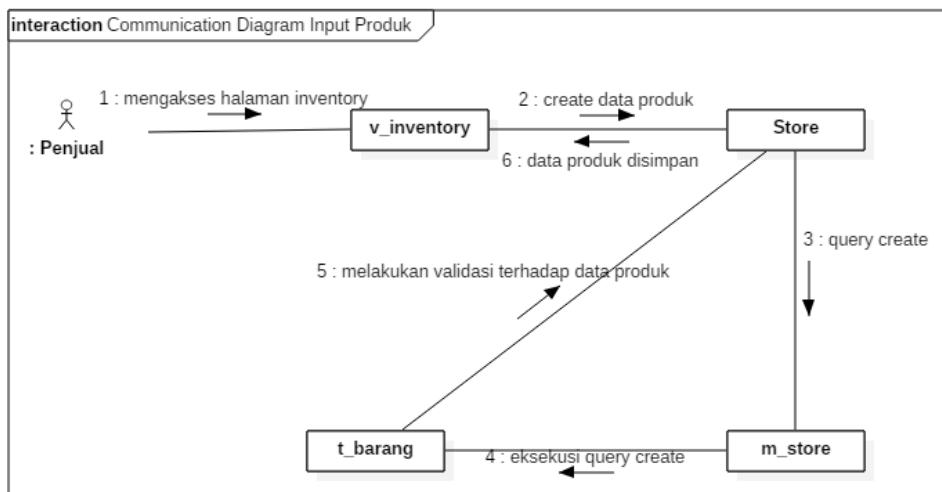


Gambar 4.16 Collaboration Pencarian Komunitas

Keterangan :

1. Pelanggan akan mengirimkan *keyword* komunitasnya.
2. Pada *controller user* dimana akan mengirimkan *query view* dengan menambahkan *keyword* yang di cari.
3. *Model user* akan mengambil data dari tabel barang.
4. Tabel barang akan melakukan *view* sesuai *keyword* yang diberikan.

## 6). Collaboration Kelola Produk

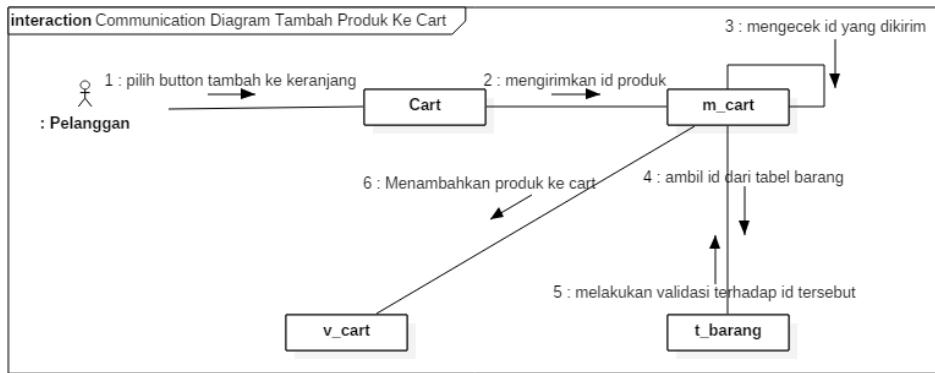


Gambar 4.17 Collaboration Kelola Produk

Keterangan :

1. Penjual akan melakukan akses terhadap halaman *inverntory*.
2. Pada halaman *inventory* tersebut pelanggan akan melakan *create* produk.
3. *Controller* akan melakukan mengirimkan *query create* kepada model *store*.
4. *Query create* tersebut akan dieksekusi oleh model *store* kepada tabel *barang*.
5. *Controller Store* akan melakukan validasi terhadap data produk yang dikirim.
6. Produk akan tersimpan di tabel barang dan ditampilkan pada halaman *inventory*.

## 7). Collaboration Transaksi

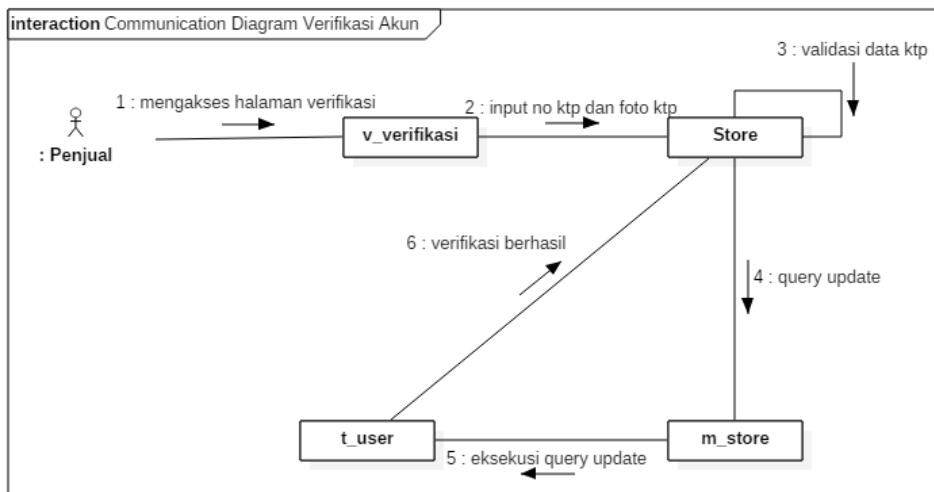


Gambar 4.18 Collaboration Transaksi

Keterangan :

1. Pelanggan akan menekan *button tambah ke keranjang* pada sebuah produk yang *id* dari produk tersebut akan dikirim.
2. *Controller cart* akan mengirimkan *id* ke dalam *model cart*.
3. *Model cart* akan mengecek *id* yang dikirim.
4. *Model cart* akan mengambil *id* tersebut dari tabel barang
5. Validasi pada *id* tersebut akan dilakukan.
6. Apabila lolos maka *model cart* akan menambahkan produk tersebut ke halaman *cart*.

## 8). Collaboration Verifikasi Akun



Gambar 4.19 Collaboration Verifikasi Akun

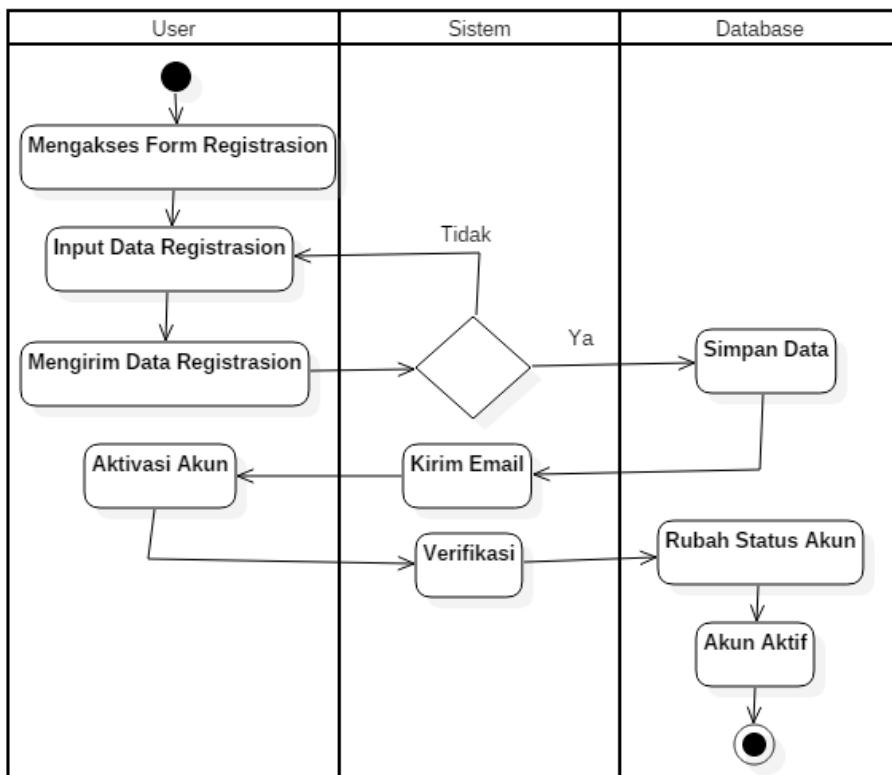
Keterangan :

1. Penjual akan mengakses halaman verifikasi.
2. Pada halaman verifikasi dimana penjual perlu mengisikan No KTP dan melakukan *upload* foto KTP.
3. Pada *controller store* dimana akan terjadinya proses validasi data KTP.
4. *Controller Store* akan mengirimkan *query update* kepada *model store*.
5. *Model store* akan melakukan eksekusi *query update* terhadap tabel *user*.
6. Verifikasi akun toko telah berhasil dilakukan.

## 4.6 Activity Diagram Sistem JURAGAN

Secara umum *activity* diagram merupakan gambaran alur dari suatu sistem yang dibuat, sehingga pengguna mengerti kegunaan sistem yang akan dibangun. *Activity* diagram merupakan gambaran *workflow* atau aktivitas dari sebuah sistem dalam proses bisnis.

### 1) *Activity Diagram Daftar Akun*

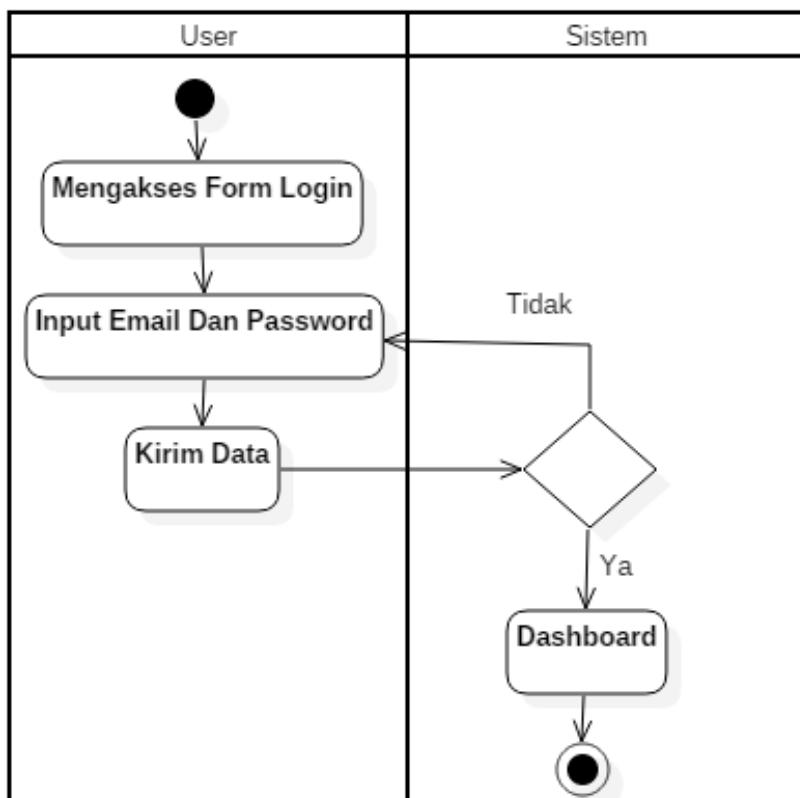


Gambar 4.20 *Activity Diagram Daftar Akun*

Pada *activity* diagram tersebut dimana dijelaskan bahwa *user* akan mengakses terlebih dahulu *form registrasian*. Setelah *form registrasian* terbuka maka langkah selanjutnya dimana *user* akan melakukan *input data registrasian* pada *form* tersebut sesuai dengan

ketentuan yang diberikan oleh sistem. Data yang di *input* kan tersebut akan dikirim kepada sistem oleh *user*. Sistem akan melakukan proses validasi dan apabila berhasil maka data akan di simpan kedalam *database* namun akun belum aktif sehingga *user* tidak dapat melakukan *login*. Untuk mengaktifkan akun tersebut dimana *user* perlu melakukan aktivasi akun pada *email* yang telah dikirimkan oleh sistem. Sistem akan melakukan verifikasi. Data yang berada pada *database* akan di *update* sehingga status akun berubah menjadi aktif.

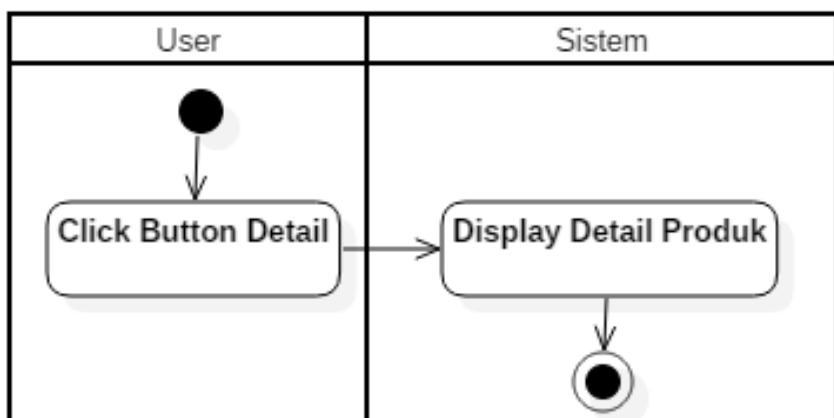
## 2) *Activity Diagram Login*



Gambar 4.21 *Activity Diagram Login*

Pada *activity diagram* tersebut dimana *step* pertama yang *user* lakukan adalah mengakses *form login*. Ketika *form login* sudah terakses maka *step* berikutnya *user* akan melakukan *input email* dan *password* yang sudah terdaftar pada sistem. Data *email* dan *password* yang sudah di *input* kan tersebut akan dikirimkan kepada sistem. Dimana sistem akan melakukan proses validasi dan apabila berhasil maka sistem akan mengarahkan *user* ke dalam *dashboard* yang menandakan bahwa *login* telah berhasil dilakukan.

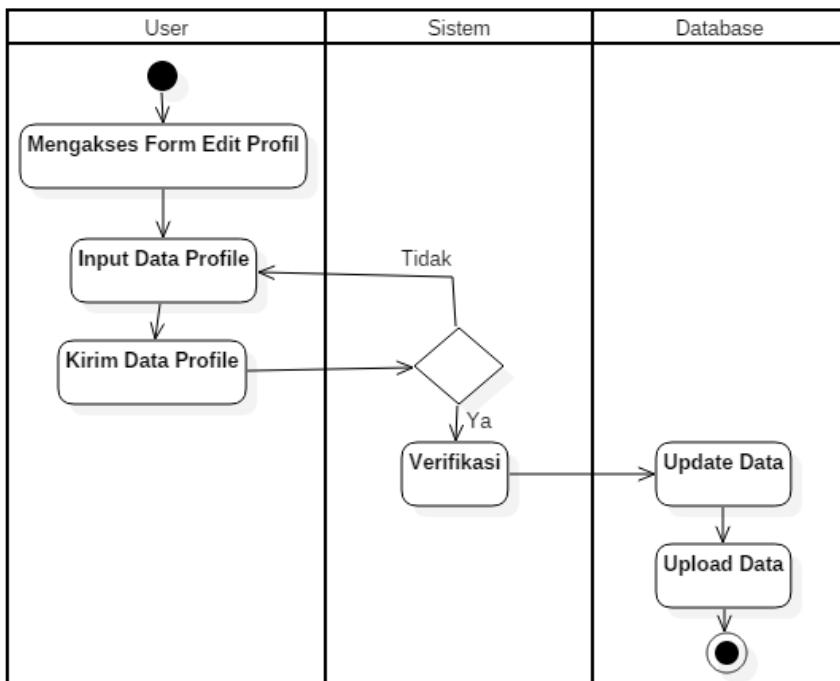
### 3) *Activity Diagram View Produk*



Gambar 5.26 Activity Diagram View Produk

Pada *activity diagram* tersebut dimana menunjukkan aktivitas *user* dalam melakukan *view produk*. Hal pertama yang harus *user* lakukan adalah memberikan *action* terhadap *button detail* dengan cara meng *click button* tersebut. Ketika *button* tersebut di *click* dimana sistem akan mengirim *id* dari produk tersebut dan memunculkan *display detail produk*. Dalam *display detail produk* dimana akan menampilkan informasi – informasi secara detail pada produk tersebut.

#### 4) Activity Diagram Edit Profile

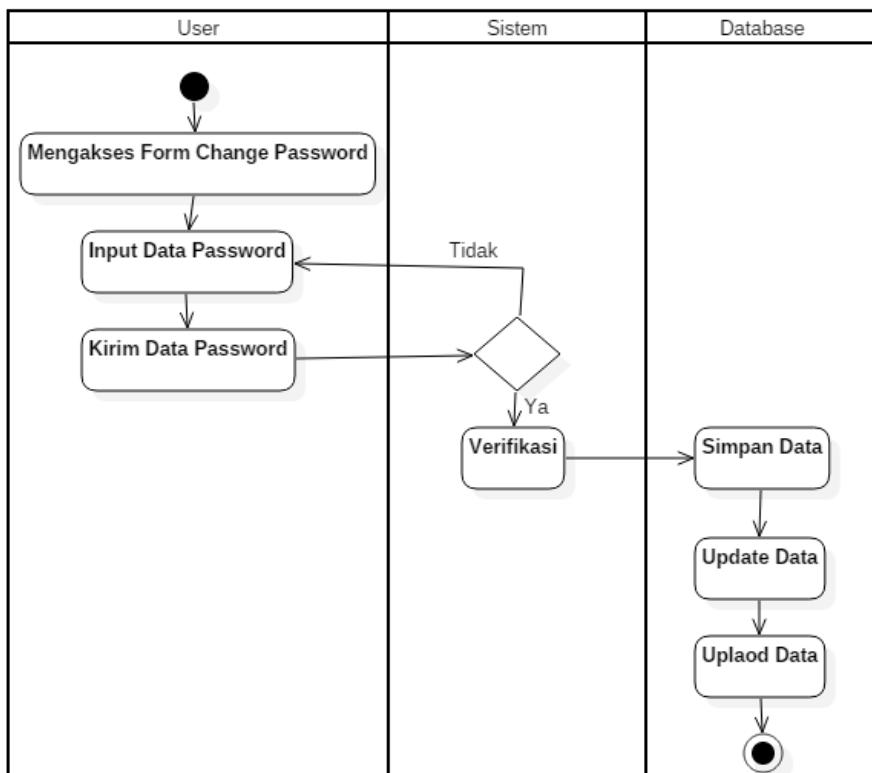


Gambar 5.27 Activity Diagram Edit Profile

Pada *activity diagram* tersebut dimana munjukkan sebuah aktifitas *edit profile*. Dimana *step* pertama dalam melakukan aktifitas *edit profile* *user* perlu mengakses *form edit profile*. Pada *form edit profile* tersebut *user* akan melakukan perubahan data *profile*-nya. Ketika data sudah dirubah maka *user* akan mengirimkan data tersebut kepada sistem. Sistem akan melakukan proses validasi dimana untuk mengecek data tersebut apakah sudah benar. Jika data sudah benar maka sistem akan melakukan verifikasi. Setelah data terverifikasi sistem akan mengirimkan data tersebut kepada *database*. Di dalam *database* dimana terjadi fungsi *get* yang dimana untuk melakukan

*update* data. Data yang sudah di *update* akan di *upload* agar *user* dapat melihat perubahannya.

### 5) Activity Diagram *Change Password*

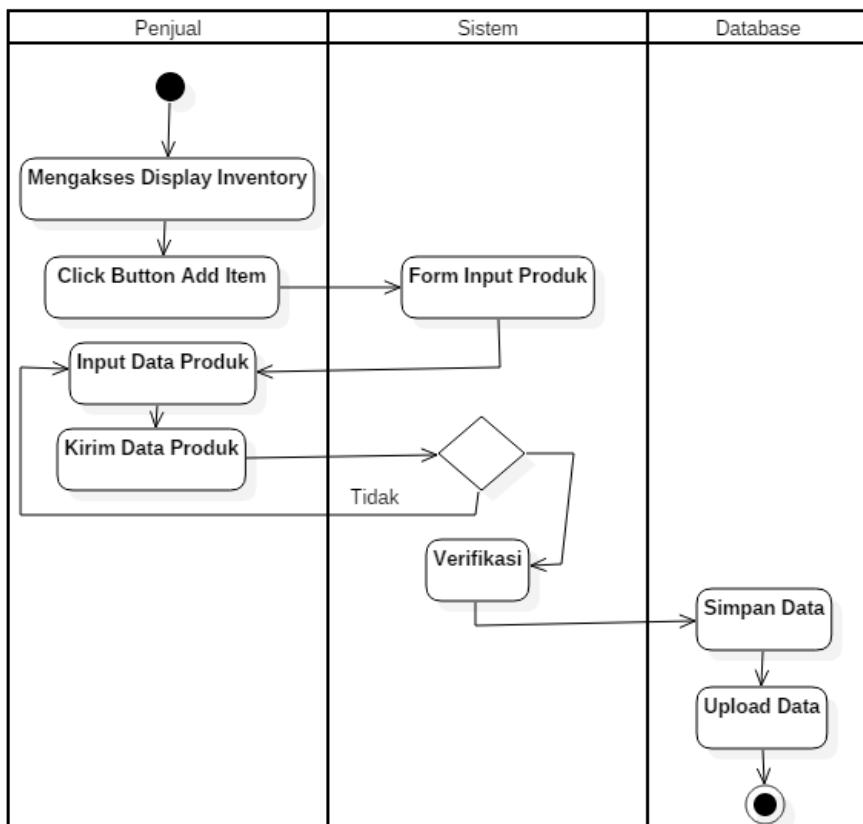


Gambar 5.28 Activity Diagram *Change Password*

Pada *activity* diagram tersebut dimana menjelaskan mengenai aktifitas dalam melakukan *change password*. Step pertama yang perlu *user* lakukan adalah mengakses *form change password*. Pada *form* tersebut diamana dibutuhkan data berupa *password* lama dan *password* baru yang dimana *user* perlu melakukan *input* data tersebut. Data yang di *input* kan tersebut akan dikirimkan kepada sistem oleh *user*. Sistem akan melakukan validasi untuk mengecek data apakah sudah benar.

Apabila sudah benar sistem akan memverifikasi data tersebut. Data yang sudah terverifikasi akan dikirimkan ke dalam *database*. Pada *database* akan dilakukan sebuah fungsi *get* yang berguna untuk melakukan *update* data. Data yang sudah di *update* akan di *upload* oleh *database*.

## 6) Activity Diagram Input Produk

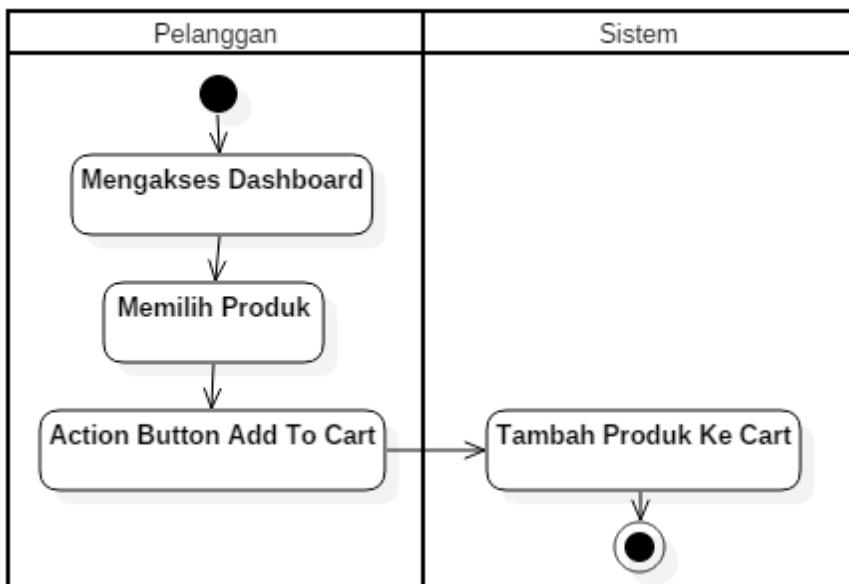


Gambar 5.29 Activity Diagram Input Produk

Pada *activity* diagram tersebut dimana menjelaskan bagaimana aktifitas pada proses *input* data produk dan aktor yang berperan adalah *user* dengan level penjual. Hal pertama yang harus dilakukan oleh penjual dimana perlu mengakses *display inventory*. Pada *display*

*inventory* terdapat satu *button* dengan nama *add item* dan apabila diberikan *action* akan memunculkan *form input produk* yang diberikan dengan fungsi modal. Pada *form* tersebut dimana penjual akan melakukan *input* data produk yang dibutuhkan oleh sistem. Data yang di *input* kan tersebut akan dikirim kepada sistem. Sistem akan melakukan proses validasi guna mengecek data tersebut apakah sudah benar. Apabila data yang dikirim tidak ada masalah maka sistem akan melakukan verifikasi. Data yang sudah terverifikasi akan disimpan kedalam *database*. Pada *database* akan berjalan fungsi *PUT* yang dimana berguna untuk meng-*input*-kan data. Data yang sudah ter *input* akan di *upload* oleh *database* agar tampil pada *Display My Inventory*.

## 7) Activity Diagram Tambah Produk Ke Cart



Gambar 5.30 Activity Diagram Tambah Produk Ke Cart

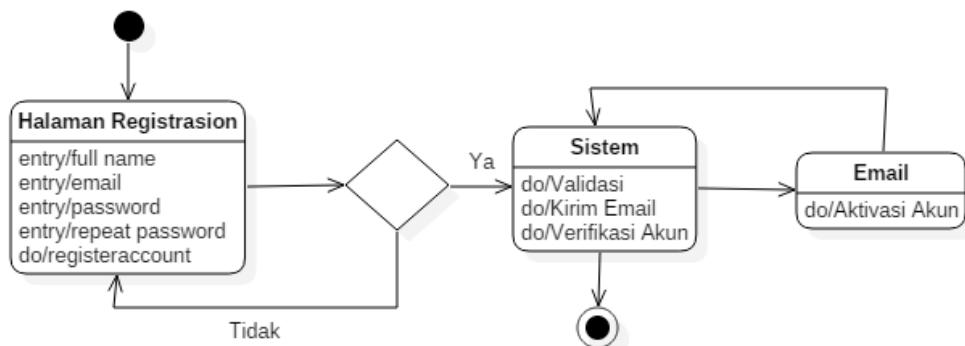
Pada *activity* diagram tersebut dimana menjelaskan bagaimana aktifitas dalam melakukan pemambahan data ke *cart* dan aktor yang

berperan adalah *user* dengan level *pelanggan*. Hal pertama yang harus dilakukan pelanggan adalah mengakses *dashboard* terlebih dahulu. Dengan begitu pelanggan dapat memilih produk yang diinginkan. Pelanggan juga dapat melihat informasi – informasi produk tersebut dengan menekan *button detail*. Sistem akan memunculkan *display* detail produk. Apabila pelanggan merasa cocok dengan produk tersebut maka pelanggan dapat menambahkan produk tersebut kedalam *cart* dengan cara meng *click button add to cart*. Sistem akan secara otomatis menambahkan produk tersebut kedalam *cart*.

#### 4.7 Statechart Diagram Sistem JURAGAN

*Statechart* diagram menggambarkan transisi dan perubahan keadaan (dari suatu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimulans yang diterima. *Statechart* diagram mendeskripsikan bagaimana suatu objek mengalami perubahan status adanya *trigger* dari *event-event*. Adapun *statechart* diagram pada sistem JURAGAN adalah sebagai berikut :

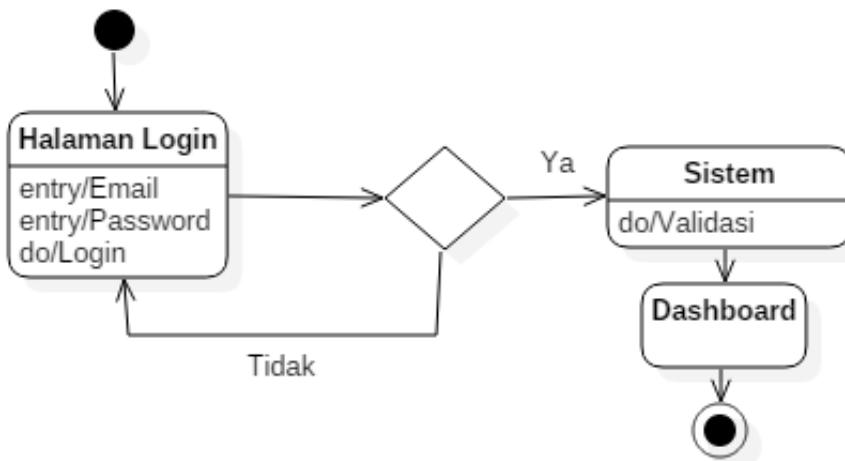
##### 1) *Statechart Diagram Registrasian*



Gambar 5.31 Statechart Diagram Registrasian

Pada *statechart* diagram tersebut dimana menjelaskan *state – state* yang dilakukan oleh *user* untuk *registrasian* akun. *State* pertama yang perlu dilakukan adalah dengan mengakses halaman *registrasian* terlebih dahulu. Pada halaman *registrasian* dimana *user* perlu melakukan *input* data berupa *full name*, *email*, *password*, dan *repeat password*. Ketika data sudah di *input* kan maka *user* perlu memberikan *action* terhadap *button register account*. *State* kedua dimana sistem akan melakukan proses validasi dan mengirimkan email kepada *user*. *State* ketiga dimana *user* akan mengakses *email* dan melakukan aktivasi akun. *State* keempat dimana sistem akan melakukan verifikasi akun sehingga akun menjadi aktif.

## 2) *Statechart Diagram Login*



Gambar 5.32 *Statechart Diagram Login*

Pada *statechart* diagram tersebut dimana menjelaskan *state – state* yang perlu dilakukan dalam proses *login*. Dimana untuk melakukan *login state* pertama ialah dengan mengakses halaman *login* terlebih dahulu. Pada halaman *login* dimana *user* perlu melakukan *input* *email*

dan *password* yang sudah terdaftar pada sistem. Ketika data sudah di *input* maka *user* perlu memberikan *action* terhadap *button login*. *State* kedua dimana sistem akan melakukan validasi guna melakukan pengecekan terhadap data yang dikirim. Apabila data yang dikirim tidak terjadi masalah maka sistem akan melanjutkan ke proses berikutnya. *State* ketiga dimana sistem akan memunculkan *dashboard* yang menunjukkan bahwa *login* telah sukses.

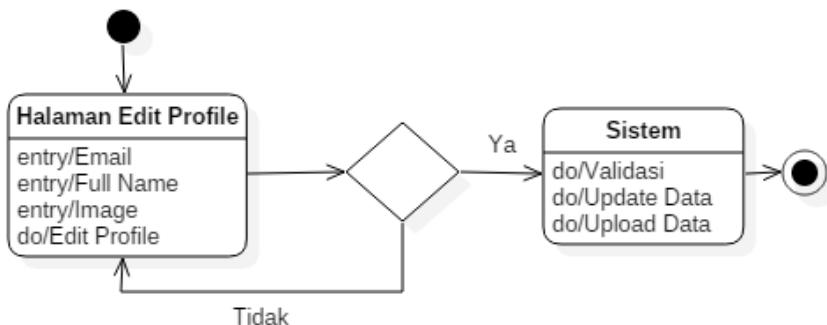
### 3) Statechart Diagram View Produk



Gambar 5.33 Statechart Diagram View Produk

Pada *statechart* diagram tersebut dimana menunjukkan *state – state* dalam melakukan *view* produk. *State* pertama yang perlu dilakukan adalah dengan memberikan *action* terhadap *button detail*. *State* kedua yaitu dimana sistem akan meberikan respon dengan cara memunculkan halaman detail produk.

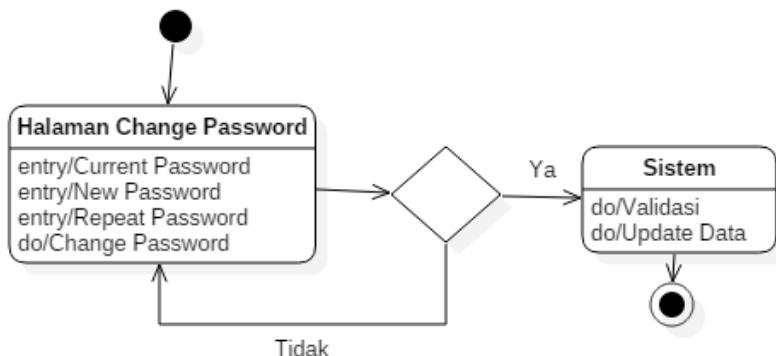
### 4) Statechart Diagram Edit Profile



Gambar 5.34 Statechart Diagram Edit Profile

Pada *statechart* diagram tersebut dimana menunjukkan *state – state* dalam melakukan *edit profile*. *State* pertama yang perlu dilakukan *user* adalah dengan mengakses halaman *edit profile*. Dalam halaman *edit profile* tersebut dimana *user* perlu menginputkan data *full name* dan *image*. *Email* disini bersifat *read only* sehingga tidak dapat diubah. Jika data tersebut sudah di *input* kan dimana *user* perlu memberikan *action* terhadap *button edit profile*. *State* kedua dimana sistem akan melakukan proses validasi guna mengecek data apakah sudah sesuai dengan ketentuan sistem. Apabila data sudah sesuai maka data akan dikirim ke dalam *database* untuk di *update*. Ketika sudah di *update* maka data tersebut akan di *upload* agar dapat dilihat perubahannya oleh *user*.

## 5) Statechart Diagram Change Password

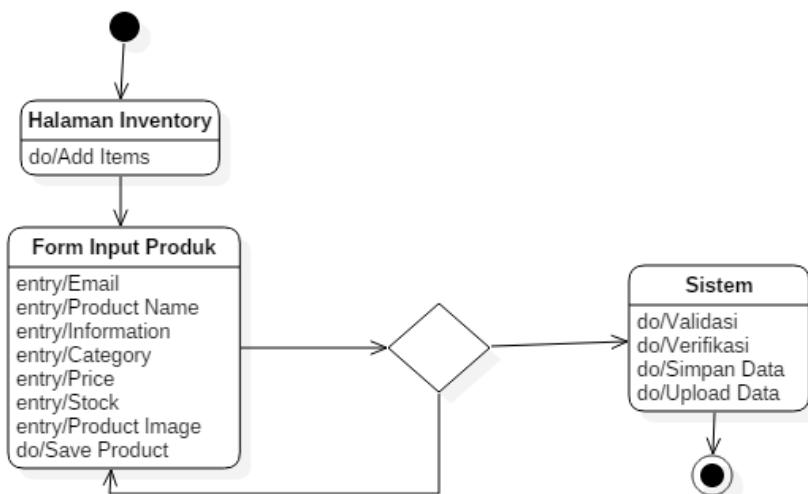


Gambar 5.35 Statechart Diagram Change Password

Pada *statechart* diagram tersebut dimana menunjukkan *state – state* dalam melakukan *change password*. *State* pertama yang perlu dilakukan adalah dengan mengakses halaman *change password*. Pada halaman tersebut dimana *user* perlu melakukan *input* data *current password*, *new password*, dan *repeat password*. Ketika data sudah di

*input* kan dimana *user* perlu memberikan *action* terhadap *button change password*. *State* kedua yaitu dimana sistem akan melakukan proses validasi guna mengecek data apakah sudah sesuai dengan ketentuan sistem. Apabila data sudah sesuai maka sistem akan mengirim data tersebut ke dalam *database*. *Database* akan melakukan proses *get* dimana berfungsi untuk *update* data.

## 6) Statechart Diagram Input Produk

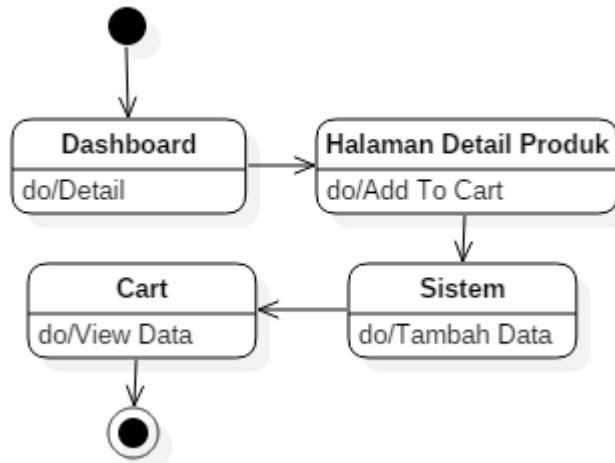


Gambar 5.36 Statechart Diagram Input Produk

Pada *statechart* diagram tersebut merupakan *state – state* dalam melakukan proses *input produk* dimana aktor yang berperan adalah *user* dengan level penjual. *State* pertama yang perlu dilakukan adalah dengan mengakses halaman *inventory* terlebih dahulu, dimana pada halaman tersebut terdapat satu *button add item* yang perlu di berikan *action*. *State* kedua yaitu sebuah modal berupa *form input product* akan muncul dan penjual perlu melakukan *input* data berupa *email*, *product name*, *information*, *category*, *price*, *stock*, dan *product image*. Ketika

data sudah di *inputkan* dimana penjual perlu memberikan *action* terhadap *button save produt*. *State* ketiga dimana sistem akan melakukan proses validasi guna mengecek data apakah sudah sesuai. Apabila data sudah sesuai maka sistem akan melakukan verifikasi data tersebut dan menyimpannya kedalam *database*. Pada *database* dimana akan berjalan fungsi *PUT* yang berperan dalam meng *input* kan data dan data yang sudah ter *input* akan di *upload* oleh *database*.

## 7) Statechart Diagram Tambah Produk Ke Cart



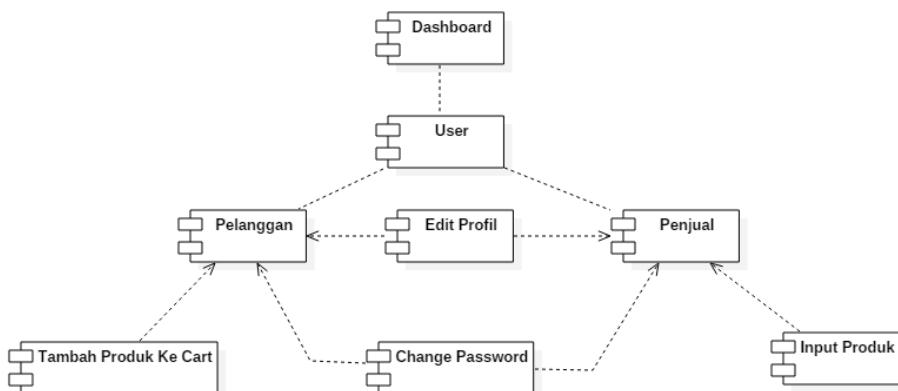
Gambar 5.37 Statechart Diagram Tambah Produk Ke Cart

Pada *statechart* diagram tersebut dimana akan menjelaskan *state – state* yang dilalui dalam proses menambahkan produk ke dalam *cart* yang dimana aktor disini adalah *user* dengan level pelanggan. *State* pertama dimana pelanggan perlu mengakses *dashboard* terlebih dahulu. Pada *dashboard* pelanggan dapat memilih produk – produk yang telah di *input* kan oleh penjual. Untuk melihat *informasian* produk tersebut dimana pelanggan dapat memberikan *action* terhadap

*button* detail. *State* kedua dimana sistem akan merespon dengan memunculkan halaman detail produk. Pada halaman detail produk dimana pelanggan dapat menambahkan produk ke *cart* secara langsung dengan cara memberikan *action* terhadap *button add to cart*. *State* ketiga dimana sistem akan melakukan proses tambah data ke dalam halaman *cart*. *State* keempat dimana produk yang sudah ditambahkan dapat di *view* pada halaman *cart*.

#### 4.8 Component Diagram

Berikut ini merupakan *component* diagram yang menjelaskan semua *component* yang terdapat pada sistem JURAGAN.

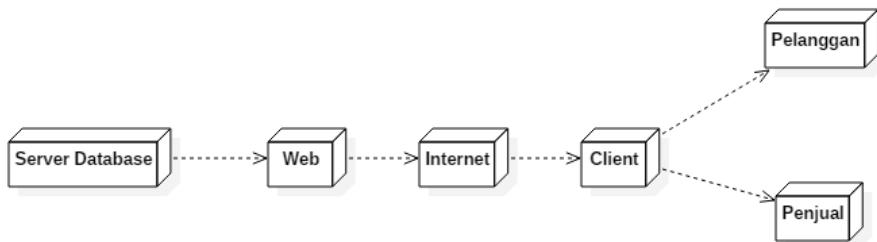


Gambar 5.38 Component Diagram

#### 4.9 Deployment Diagram

*Deployment* diagram menunjukkan tata letak sebuah sistem secara fisik, menampilkan bagian – bagian *software* yang berjalan pada bagian – bagian *hardware* yang digunakan untuk mengimplementasikan sebuah sistem dan keterhubungan antara komponen – komponen *hardware* tersebut. *Deployment* diagram dapat digunakan pada bagian – bagian awal

proses perancangan sistem untuk mendokumentasikan arsitektur fisik sebuah sistem. Berikut *deployment* diagram yang menggambarkan susunan fisik, perangkat lunak dan sistem.



*Gambar 5.39 Deployment Diagram*

#### 4.10 Perancangan Basis Data/Database

*Tabel 5.20 User*

No.	Field	Type	Wide	Information
1	<i>Id</i>	<i>Integer</i>	11	<i>Id user</i> yang merupakan <i>primary key</i>
2	<i>Name</i>	<i>Varchar</i>	128	Nama lengkap <i>user</i>
3	<i>Email</i>	<i>Varchar</i>	128	<i>Email</i> dari <i>user</i>
4	<i>Password</i>	<i>Varchar</i>	300	<i>Password</i> dari akun <i>user</i>
5	<i>Image</i>	<i>Varchar</i>	128	<i>Image profile</i> pada <i>user pelanggan</i>

6	<i>Role_id</i>	<i>Integer</i>	11	Merupakan id level <i>user</i> . 1. Admin 2. User
7	<i>Is_active</i>	<i>Integer</i>	11	Id akun yang aktif. 0. Belum Aktif 1. Aktif
8	<i>Date_created</i>	<i>Integer</i>	11	Tanggal <i>user</i> mendaftar akun
9	Kebijakan	<i>Varchar</i>	15	Validasi daftar toko
10	Tlpn	<i>Varchar</i>	15	Nomor telpon <i>user</i>
11	Alamat	<i>Varchar</i>	600	Alamat dari <i>user</i>
12	Kecamatan	<i>Varchar</i>	25	Kecamatan <i>user</i>
13	Kelurahan	<i>Varchar</i>	25	Kelurahan <i>user</i>
14	Kota	<i>Varchar</i>	25	Kota <i>user</i>
15	Kode_pos	<i>Varchar</i>	15	Kode pos alamat <i>user</i>
16	Status_toko	<i>Varchar</i>	25	Status verifikasi toko
17	Cat_toko	<i>Text</i>		Berisikan catatan toko

18	No_ktp	Varchar	20	No KTP user
19	Img_ktp	Varchar	25	Foto KTP user

Tabel 5.21 tb\_barang

No.	Field	Type	Wide	Information
1	Id_brg	Integer	11	Merupakan id dari barang dan merupakan <i>primary key</i>
2	Email	Varchar	128	Forgent key dari tabel user
3	Nama_brg	Varchar	120	Nama dari produk yang di input
4	Keterangan	Varchar	225	Keterangan dari sebuah produk
5	Kategori	Varchar	60	Kategori dari sebuah produk
6	Harga	Integer	11	Harga dari sebuah produk
7	Stok	Integer	4	Merupakan jumlah dari sebuah produk yang ada

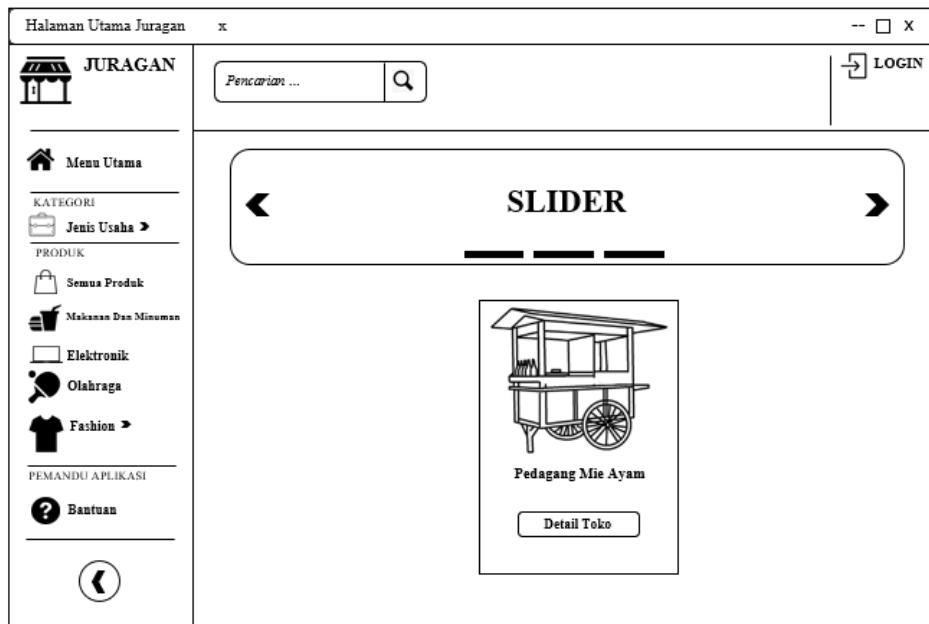
8	Gambar	Varchar	60	Merupakan gambar dari sebuah produk
---	--------	---------	----	-------------------------------------

Tabel 5. 22 User\_token

No.	Field	Type	Wide	Information
1	<i>Id</i>	<i>Integer</i>	11	Merupakan <i>id</i> dari token yang masuk dan sebuah <i>primary key</i>
2	<i>Email</i>	Varchar	128	Merupakan <i>forgent key</i> dari tabel <i>user</i>
3	<i>Token</i>	Varchar	128	Token yang akan dikirim saat aktivasi akun
4	<i>Date_created</i>	Varchar	128	Tanggal token di <i>input</i> kan

## 4.11 User Interface Sistem

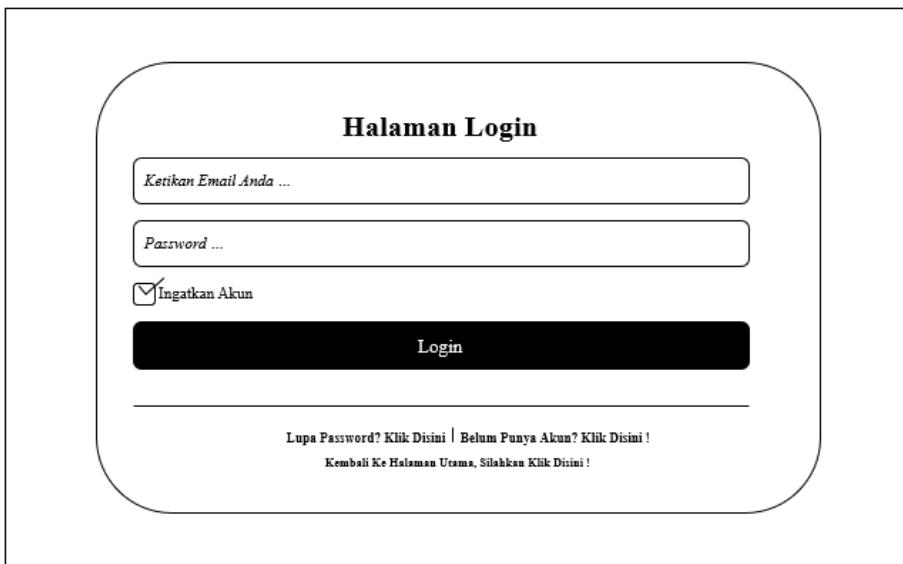
### 1). User Interface Halaman Utama (Sebelum Login)



Gambar 5.40 UI Halaman Utama Sebelum Login

Pada gambar 5.40 dimana merupakan UI halaman utama dengan kondisi sebelum melakukan *login*. Dalam *topbar* pada UI tersebut terdapat sebuah *form pencarian* yang berguna untuk mencari sebuah toko. Terdapat juga sebuah *icon login* yang disertai dengan *text* dengan fungsi *link*, dan apabila di klik akan menuju ke halaman *login*. Pada bagian *sidebar* terdapat *icon-icon* yang dibutuhkan dalam sistem JURAGAN tersebut yang diberikan dengan fungsi *link*. Pada bagian *index* nya terdapat *slider* yang akan berpindah – pindah secara otomatis. Terdapat juga *card* yang di isi dengan toko – toko yang sudah terdaftar pada sistem JURAGAN tersebut.

## 2). User Interface Login



Gambar 5.41 UI Login

Pada gambar 5.41 tersebut merupakan UI *login* dari sistem JURAGAN. Dimana UI *login* tersebut di masukkan kedalam *card* dan diberikan sebuah *form* guna mengimputkan *email* yang terdaftar dan *password*. Terdapat juga pengingat akun untuk melakukan save pada akun kita. Terdapat juga *button login* yang dimana apabila diklik akan memulai proses *login*.

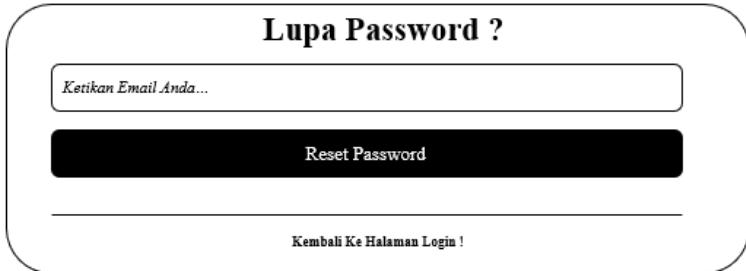
### 3). User Interface Daftar Akun

The diagram shows a user interface for account registration. It consists of a main container with rounded corners. Inside, there is a title 'Daftarkan Akun Anda!'. Below the title are four input fields: 'Nama Lengkap ...', 'Email Anda ...', 'Password Anda ...', and 'Ulangi Password Anda ...'. A large black button labeled 'Daftar Akun' is positioned below the input fields. At the bottom of the container, there are two links: 'Lupa Password ? Klik Disini !' and 'Sudah Punya Akun ? Silahkan Login !'.

Gambar 5.42 UI Daftar Akun

Pada *UI* daftar akun tersebut dimana hanya terdapat satu *layer* yang didalamnya terdiri dari 4 *textbox* yang diberikan fungsi *placeholder*, terdapat juga 1 *button* untuk melakukan *action registrasion*. Di bagian bawah *layer* tersebut terdapat 2 *link* yang yang terdiri dari *Forgot Password* dan *Already have as account ? Login !* yang dimana akan menuju ke halaman lain sesuai *link* tersebut apabila di *click*.

#### **4). User Interface Lupa Password**

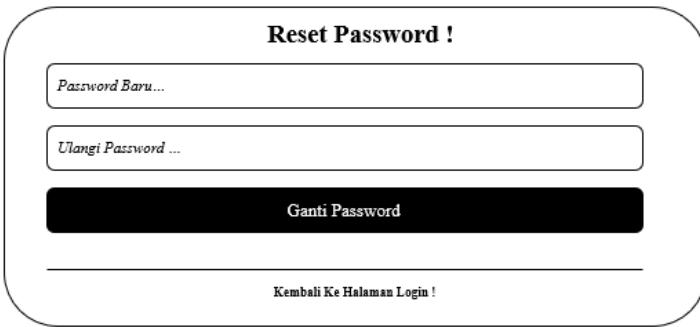


The image shows a wireframe-style user interface for a password recovery page. At the top center, it says "Lupa Password ?". Below that is a rectangular input field with the placeholder text "Ketikan Email Anda...". Underneath the input field is a black rectangular button with the white text "Reset Password". At the bottom of the page, there is a horizontal line and a link "Kembali Ke Halaman Login !". The entire form is enclosed in a rounded rectangular border.

*Gambar 5.43 UI Forgot Password*

Pada *UI Forgot Password* dimana hanya terdiri dari satu *layer* utama yang di dalamnya terdapat 1 *textbox* dengan menggunakan fungsi *placeholder*, terdapat juga 1 *button* untuk melakukan *action reset password*. Pada bagian bawah *layer* terdapat satu *link Back to login* yang dimana akan berpindah ke halaman *login* apabila di *click*.

#### **5). User Interface Reset Password**



The image shows a wireframe-style user interface for a password reset page. At the top center, it says "Reset Password !". Below that are two rectangular input fields: the top one has the placeholder "Password Baru..." and the bottom one has the placeholder "Ulangi Password ...". Underneath these input fields is a black rectangular button with the white text "Ganti Password". At the bottom of the page, there is a horizontal line and a link "Kembali Ke Halaman Login !". The entire form is enclosed in a rounded rectangular border.

*Gambar 5.44 UI Reset Password*

Pada *UI Reset Password* tersebut hanya terdapat satu *layer* utama yang dimana di dalamnya terdapat 2 *textbox* dengan fungsi *placeholder*, terdapat juga 1 *button* untuk melakukan *action changed password*. Di bagian bawah *layer* terdapat 1 *link* untuk menuju ke halaman *login* apabila di *click*.

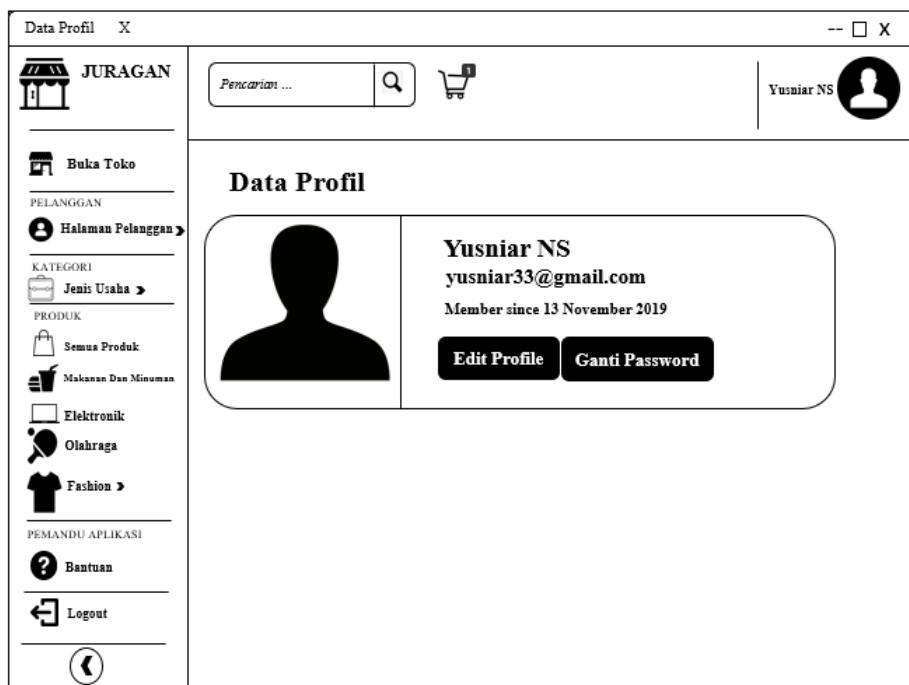
## 6). User Interface Halaman Utama Sesudah Login



Gambar 5.45 UI Halaman Utama Sesudah Login

Pada gambar 4.45 merupakan UI pada halaman utama sistem JURAGAN dengan kondisi sesudah *login*, dimana bedanya dengan gambar 5.40 UI pada gambar 4.45 ini memberikan akses yang lebih luas yaitu dengan adanya *icon-icon* yang bertambah seperti buka toko dan halaman pelanggan pada *sidebar* dan juga nama *user* beserta foto profil dan *icon cart* pada *topbar*.

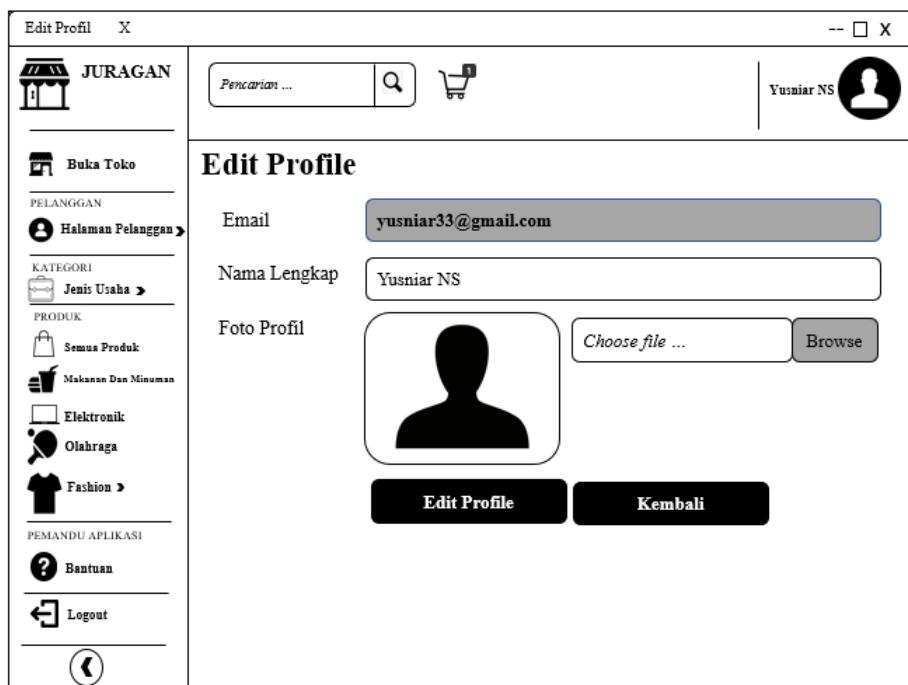
## 7). User Interface Data Profil



Gambar 5.46 UI Data Profil

Pada gambar 5.46 merupakan UI dari data profil dimana untuk bagian *sidebar* dan *topbar* akan terlihat sama dengan gambar 5.45. Pada bagian index nya terdapat sebuah *card* yang didalamnya diberikan informasi – informasi data profil *user*.

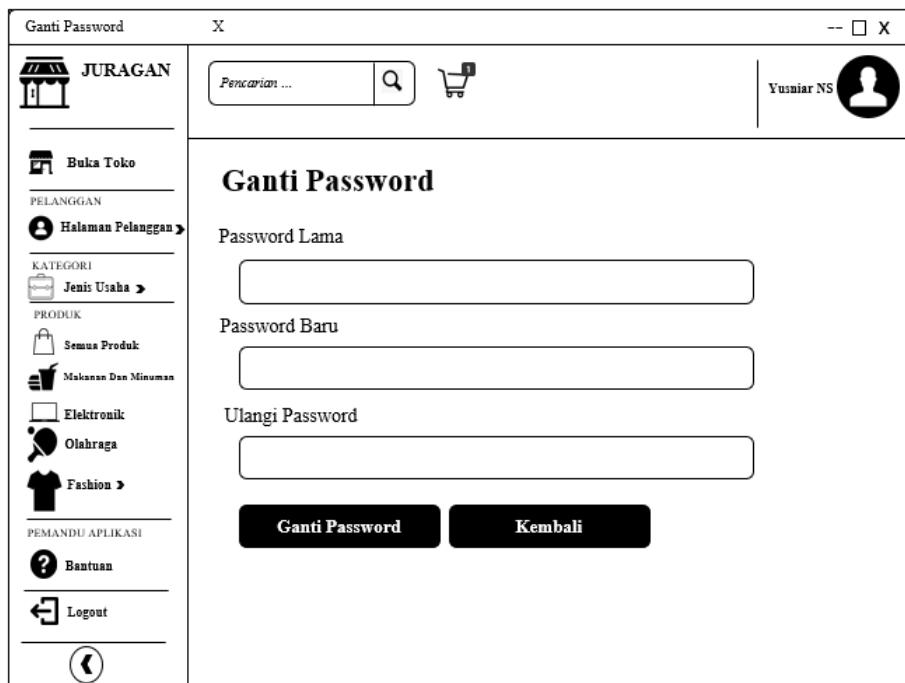
## 8). User Interface Edit Profil



Gambar 5.47 UI Edit Profil

Pada gambar 5.47 merupakan UI dari edit profil pelanggan dimana pada bagian *sidebar* dan *topbar* terlihat sama dengan gambar 5.45. Pada bagian *index* dimana terdapat sebuah *form* yang pada setiap *textbox* nya di ambil dari *database*. Pada bagian *email* dimana bersifat *readonly*. Terdapat dua buah *button* sebagai memulai aksi yang diberikan.

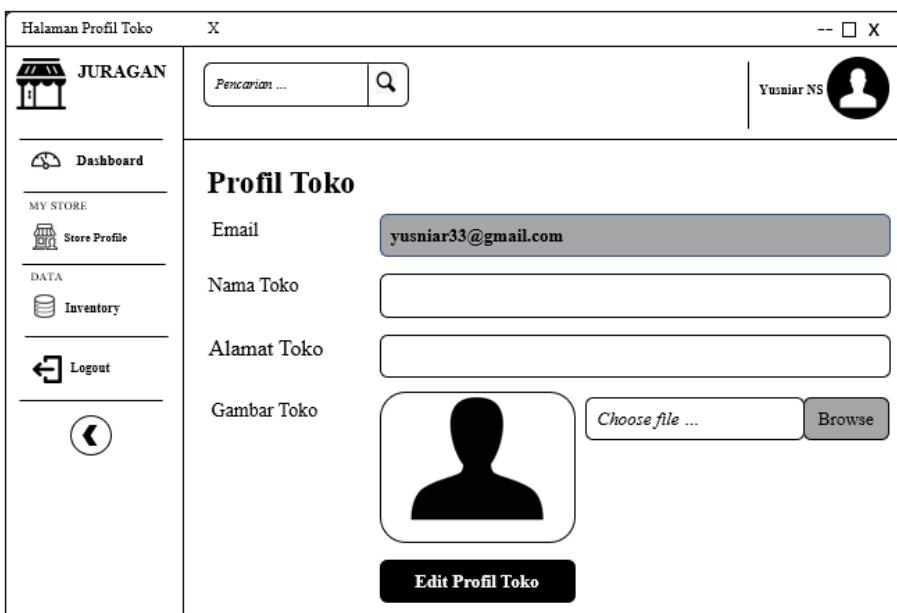
## 9). User Interface Ganti Password



Gambar 5.48 UI Ganti Password

Pada gambar 5.48 tersebut dimana merupakan UI dari ganti password bagi *user*. Pada bagian *sidebar* dan *topbar* akan terlihat sama dengan gambar 5.45. Pada bagian *index* terdapat sebuah *form* yang diberikan 3 *textbox* dan 2 *button* untuk memulai aksi ganti *password*.

## 10). User Interface Profil Toko



Gambar 5.49 UI Profil Toko

Pada gambar 5.49 merupakan UI dari profil toko bagi *user penjual*. Pada bagian *sidebar* terdapat beberapa *icon* yang nampak lebih sedikit dari pada gambar 5.45. Hal ini dikarenakan bedanya akses yang diberikan. Pada bagian *topbar* sekilas terlihat sama dengan gambar 5.45 akan tetapi yang membedakan adalah tidak adanya *icon cart* yang dimana *icon cart* merupakan sesuatu *fitur* yang hanya diberikan oleh *user* dengan level pelanggan. Pada bagian *index* nya terdapat satu *form* yang diberikan beberapa *textbox* serta *button edit* profil toko yang dimana akan melakukan pendaftaran pada toko tersebut.

## 11). User Interface Produk Toko

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 5.50 UI Produk Toko

Pada gambar 5.50 merupakan UI dari produk toko yang dimana akses tersebut diberikan kepada *user* dengan level pelanggan. Pada bagian *sidebar* dan *topbar* akan terlihat sama dengan gambar 5.49. Pada bagian *index* nya dimana terdapat tabel untuk melakukan *view* dari *database* berdasarkan *session* yang diberikan. Terdapat juga satu *button* yang berfungsi untuk mengakses halaman *input* produk.

## 12). User Interface Form Input Produk

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"/>
<input type="button" value="Kembali"/> <input type="button" value="Simpan Produk"/>	

*Gambar 5.51 UI Form Input Produk*

Pada gambar 5.51 merupakan UI dari *form input* produk. Dimana hanya *user* dengan level pelanggan yang dapat mengakses UI tersebut. UI tersebut dapat dibuka dengan cara mengklik pada *button* yang berada dalam gamabr 5.50. *form* tersebut berfungsi untuk menambahkan suatu produk baru yang akan di jual oleh pelanggan. Pelanggan diperlukan mengisi informasi – informasi yang dibutuhkan oleh sistem. Informasi tersebut akan di tampilkan dengan *textbox*.

Terdapat dua *button* yang dimana berfungsi untuk memulai aksi *input* produk.

### 13). User Interface Halaman Keranjang

The screenshot shows the 'Halaman Keranjang Saya' (Cart Page) of the JURAGAN application. The sidebar on the left includes links for 'JURAGAN', 'Buka Toko', 'PELANGGAN', 'Halaman Pelanggan', 'KATEGORI', 'PRODUK' (with sub-links for 'Semua Produk', 'Makanan Dan Minuman', 'Elektronik', 'Olahraga', and 'Fashion'), and 'PEMANDU APLIKASI' (with sub-links for 'Bantuan' and 'Logout'). The main content area features a search bar with placeholder 'Pencarian ...' and a magnifying glass icon, along with a shopping cart icon showing a quantity of 1. On the right, there is a user profile for 'Yusniar NS' with a circular profile picture. The central part of the screen displays a table titled 'Keranjang Saya' with the following data:

NO	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Laptop	1	Rp. 6.400.000	Rp. 6.400.000	
2	Sepatu	1	Rp. 500.000	Rp. 500.000	
					Rp. 6.900.000

At the bottom of the page are three buttons: 'Bersihkan Keranjang', 'Lanjut Belanja', and 'Pembayaran'.

Gambar 5.52 UI Halaman Keranjang

Pada gambar 5.52 merupakan UI dari halaman keranjang yang dimana hanya dapat di akses oleh *user* dengan level pelanggan. Pada bagian *sidebar* dan *topbar* terlihat sama dengan gambar 5.45. Pada bagian *index* nya terdapat tabel yang isinya di peroleh dari hasil menambahkan produk kedalam keranjang melalui *button*. Terdapat 2 *icon* di dalam tabel tersebut yaitu *refresh* dan *delete* yang sudah di fungsikan. Di luar tabel terdapat 3 *button* yang dimana untuk memulai aksi yang diberikan.

#### **4.12 Perancangan Arsitektur *Software* Dan *Hardware***

Pada perancangan perangkat lunak dan perangkat keras tersebut merupakan kebutuhan – kebutuhan untuk mengakses aplikasi JURAGAN tersebut yang dimana telah dicantumkan pada tabel 5.23 – 5.24.

##### **A. Aplikasi Kebutuhan Perangkat Lunak**

Perangkat lunak (*software*) yang dibutuhkan dalam aplikasi JURAGAN tersebut adalah sebagai berikut :

*Tabel 5.23 Software Requirement*

No.	Tool/Software	Description
1	<i>Windows 7/8/10</i>	<i>Operation System</i>
2	<i>Xampp, VS Code, CodeIgniter, Microsoft Visio, Bizagi Modeler, Start UML, Microsoft Word</i>	<i>Development Tools</i>
3	<i>Browser</i>	<i>Web Browser Application</i>
4	<i>MariaDB</i>	<i>Database</i>

## B. Aplikasi Kebutuhan Perangkat Keras

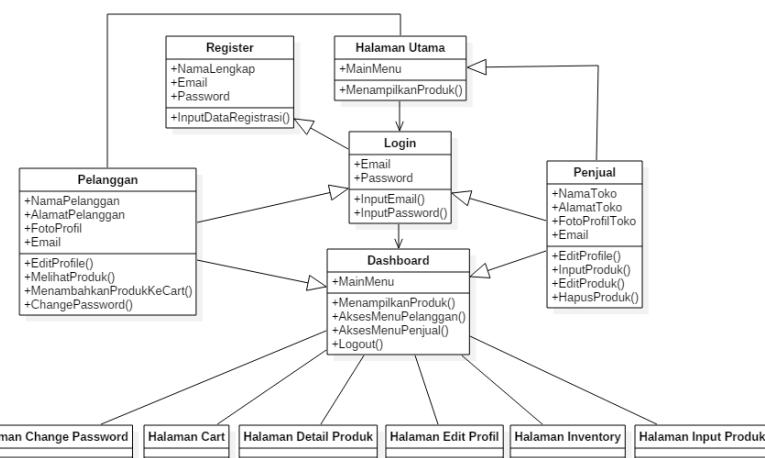
Perangkat keras (*hardware*) yang dibutuhkan dalam aplikasi JURAGAN tersebut adalah sebagai berikut :

*Tabel 5.24 Hardware Requirement*

No.	Dvice Name	Specification
1	<i>Processor</i>	<i>Core Inside (min)</i>
2	<i>Memory</i>	<i>DDR3 2 GB (min)</i>
3	<i>Harddisk</i>	<i>500 GB HD (min)</i>
4	<i>VGA</i>	<i>512 MB (min)</i>

### 4.13 Pemetaan Struktur Diagram User

Adapun pemetaan struktur diagram *user/aktor* pada sistem aplikasi JURAGAN tersebut adalah sebagai berikut :



*Gambar 5.53 Pemetaan Struktur Diagram User/Aktor Pada Sistem JURAGAN*

## **BAB VI**

## **EVALUASI**

### **6.1 Implementasi Dan Pengujian**

Implementasi merupakan sistem/aplikasi yang dibuat dengan merinci komponen – komponen pendukung berupa program, lingkungan implementasi, tampilan antarmuka, dan petunjuk penggunaan.

#### **6.1.1 Lingkungan Implementasi**

Perancang aplikasi ini dapat dilaksanakan dengan baik karena di dukung oleh perangkat pendukung yaitu perangkat lunak dan perangkat keras.

#### **A. Kebutuhan Perangkat Lunak**

Pada pembuatan aplikasi JURAGAN ini dimana perangkat lunak pendukung yang digunakan adalah sebagai berikut :

*Tabel 6.1 Kebutuhan Perangkat Lunak*

No.	Tool/Software	Fungsi
1	<i>Microsoft Windows 10</i>	<i>Operation System</i>
2	<i>MariaDB</i>	<i>Database</i>
3	<i>CodeIgniter</i>	<i>Framework</i>
4	<i>XAMPP For Windows 5.6.35</i>	<i>Web Server</i>
5	<i>PHP</i>	Bahasa Pemrograman
6	<i>Google Chrome</i>	<i>Web Browser</i>

7	<i>Bootstrap</i>	<i>Framework CSS</i>
---	------------------	----------------------

## B. Kebutuhan Perangkat Keras

Pada pembuatan aplikasi JURAGAN ini dimana perangkat keras pendukung yang digunakan adalah sebagai berikut :

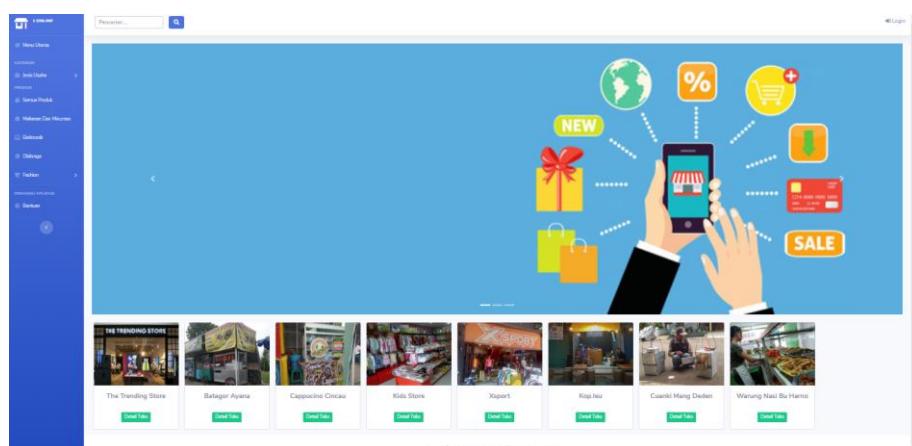
*Tabel 6.2 Kebutuhan Perangkat Keras*

No.	Nama Perangkat	Spesifikasi	Keterangan
1	<i>Processor</i>	<i>Intel Core i5-62000 2,8 GHz</i>	Media untuk menyimpan data aplikasi yang dibuat.
2	<i>Memory</i>	<i>8 GB</i>	<i>Memory system</i> yang digunakan.
3	<i>Hardisk</i>	<i>500 GB</i>	Untuk kecepatan <i>transfer</i> data dari sistem yang sangat bergantung pada kecepatan prosesor dan sebagai media penyimpanan data.
4	<i>Mouse</i> dan <i>Keyboard</i>	<i>Standart</i>	Alat pendukung.
5	<i>Monitor</i>	<i>All Device</i>	Menampilkan <i>user interface</i> .
6	Infrastruktur Jaringan		Merupakan media penghubung jaringan komputer.

## 6.2 Pembahasan Hasil Implementasi

Berdasarkan peranangan yang telah dibuat, didapat hasil dari implementasi yang menjadi tujuan pembuatan perangkat lunak ini yaitu dapat mengelola seluruh data Rancang Bangun Aplikasi JURAGAN (Jualan Rakyat Gabungan Online) Dengan *Platform E-commerce* Berbasis Komunitas.

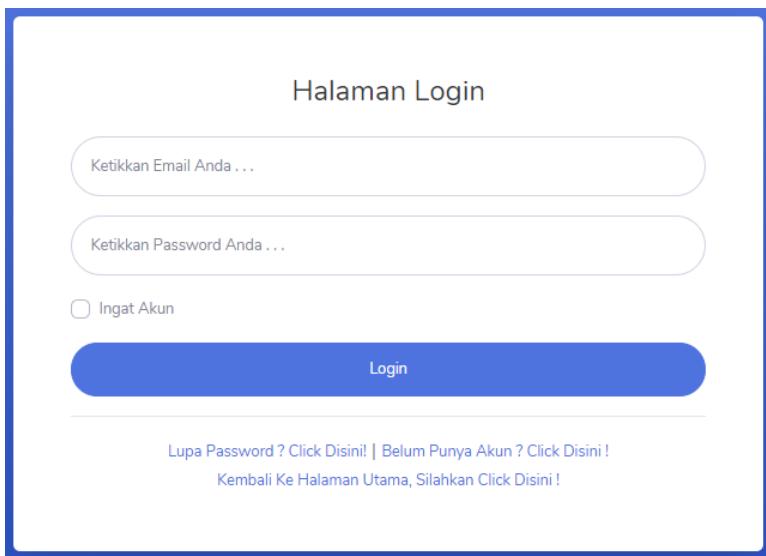
### A. Halaman Utama JURAGAN



Gambar 6.1 Halaman Utama JURAGAN

Pada halaman utama tersebut merupakan tampilan awal saat pertama kali *user* mengakses sistem JURAGAN. Pada halaman utama tersebut dimana *user* dapat melakukan *explore* terlebih dahulu terhadap sistem akan tetapi tidak bisa menambahkan produk ke dalam keranjang.

## B. Halaman *Form Login*



*Gambar 6.2 Halaman Form Login*

Pada gambar 6.2 merupakan halaman *form login* yang dimiliki oleh aplikasi JURAGAN tersebut. Prosedur yang diperlukan dalam melakukan *login* dimana *user* harus melakukan *input email* dan *password* yang telah terdaftar dan aktif di aplikasi JURAGAN.

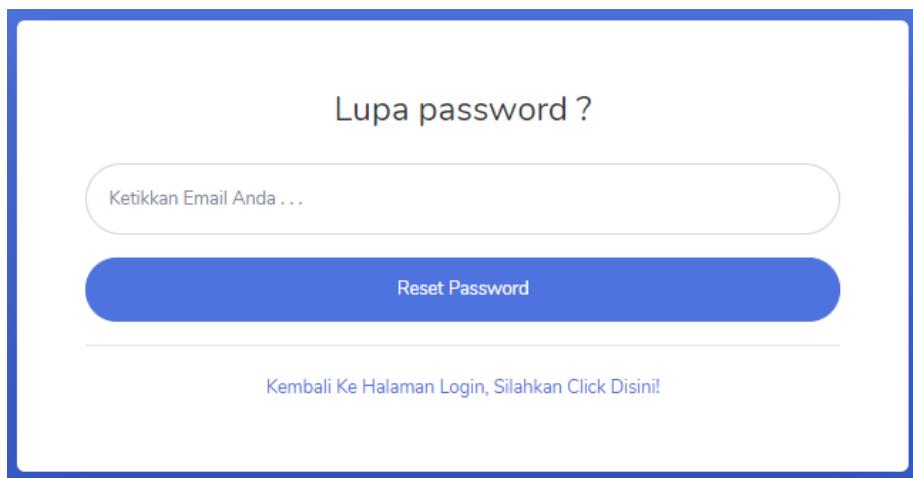
### C. Halaman Daftar Akun

The screenshot shows a registration form titled "Daftarkan Akun Anda!". It contains four input fields: "Nama Lengkap ...", "Email Anda ...", "Password Anda ...", and "Ulangi Password Anda ...". Below the form is a large blue "Daftar Akun" button. At the bottom, there are two links: "Lupa Password? Click Disini!" and "Sudah Punya Akun? Silahkan Login!".

*Gambar 6.3 Halaman Daftar Akun*

Pada gambar 6.3 tersebut merupakan halaman daftar akun yang dimiliki oleh aplikasi JURAGAN. Dimana prosedur yang perlu dilakukan adalah *user* akan melakukan *input* data sesuai dengan keterangan *form* tersebut. Jika sudah *user* akan menekan *button* daftar akun untuk mengirim data tersebut. Data akan terdaftar namun belum aktif, untuk mengaktifkannya dimana *user* perlu melakukan aktivasi melalui *email*.

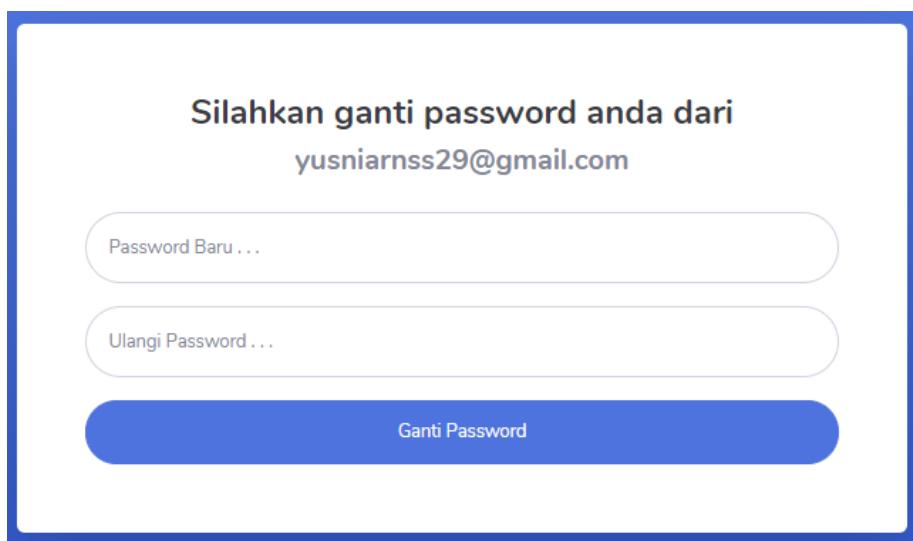
#### D. Halaman Lupa Password



*Gambar 6.4 Halaman Lupa Password*

Pada gambar 6.4 tersebut merupakan halaman lupa *password* pada aplikasi JURAGAN. Dimana prosedur dalam melakukannya *user* akan meng *input email* yang sudah terdaftar pada aplikasi JURAGAN. Sistem akan mengirimkan *email* kepada *user* dan apabila di *click* maka *password* akan dihapus, sistem akan membawa *user* ke halaman *reset password* pada gambar 6.5 untuk melakuakan *reset password*.

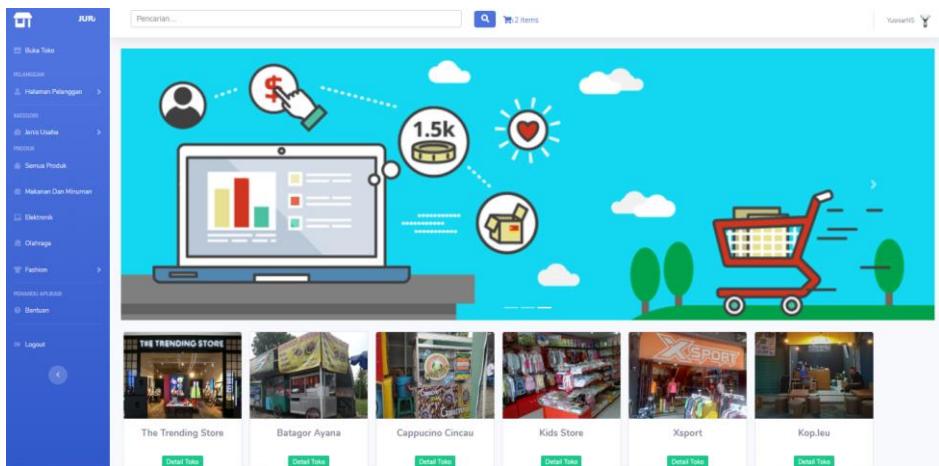
## E. Halaman Ganti Password



Gambar 6.5 Halaman Ganti Password

Pada gambar 6.5 tersebut merupakan halaman ganti *password* pada aplikasi JURAGAN. Dimana *user* akan melakukan *input password* dan memberikan *action* terhadap *button* ganti *password*, maka *password* akan diganti dengan yang baru.

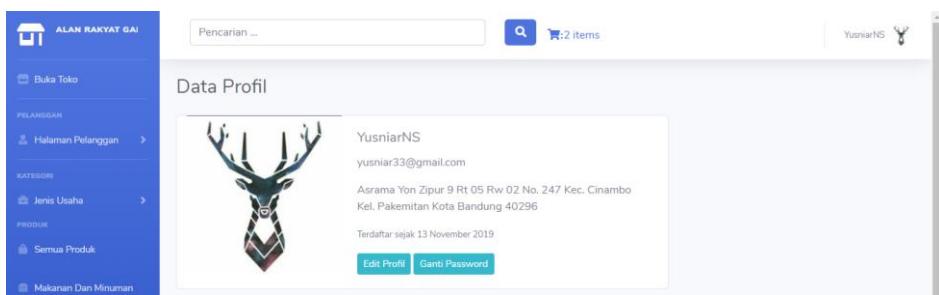
## F. Halaman Utama JURAGAN Kondisi Setelah Login



Gambar 6.6 Halaman Utama JURAGAN Kondisi Setelah Login

Pada gambar 6.6 merupakan halaman utama pada sistem JURAGAN dengan kondisi sudah melakukan *login*, dimana *user* memiliki akses yang lebih luas lagi. *User* juga dapat menambahkan produk ke dalam keranjang.

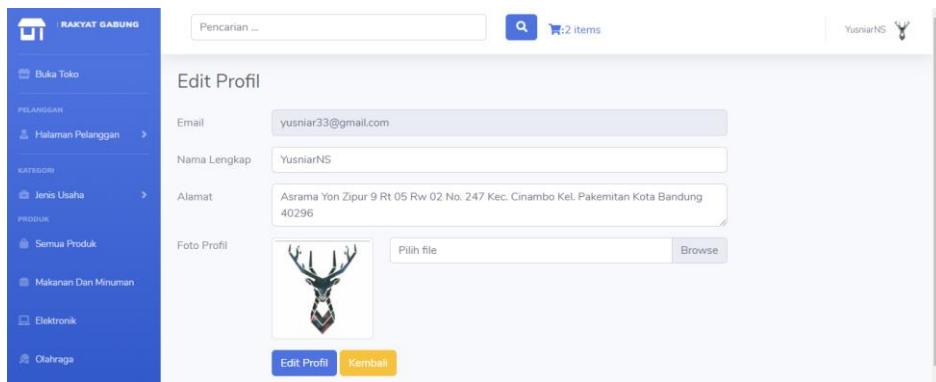
## G. Halaman Data Profil Pelanggan



Gambar 6.7 Halaman Data Profil

Pada gambar 6.7 tersebut merupakan halaman *profile* bagi *user* dengan keterangan pelanggan. Dimana *form* tersebut dapat menampilkan informasi *user* pelanggan.

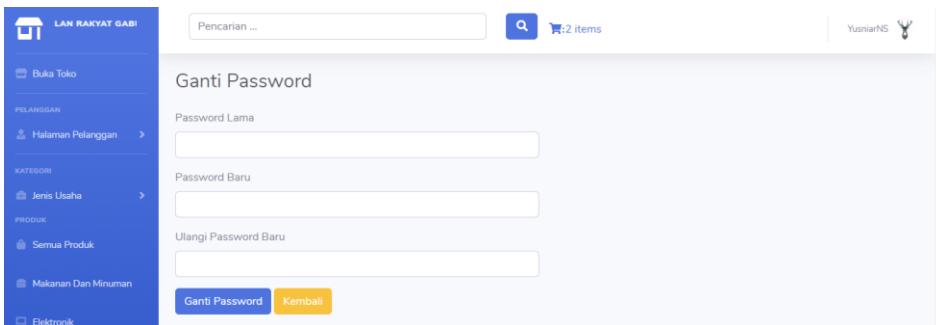
## H. Halaman *Edit Profile* Pelanggan



Gambar 6.8 Halaman *Edit Profile* Pelanggan

Pada gambar 6.8 tersebut merupakan halaman *edit profile* pelanggan yang diamana *user* dengan level pelanggan dapat melakukan *edit profile* pada halaman tersebut.

## I. Halaman Ganti Password



Gambar 6.9 Halaman Ganti Password

Pada gambar 6.9 tersebut merupakan halaman ganti *password* pada *user*. Dimana untuk melakukan pergantian *password* *user* perlu melakukan *input password* lama terlebih dahulu setelah itu *input password* yang baru.

## J. Halaman Keranjang

The screenshot shows the shopping cart section of the website. On the left sidebar, there are links for 'Buka Toko', 'PELANGGAN', 'Jenis Usaha', 'Semua Produk', 'Makanan Dan Minuman', and 'Elektronik'. The main content area has a search bar and a cart summary showing two items: 'Bola Basket' and 'Jam Tangan'. The total amount is Rp. 900.000. At the bottom, there are buttons for 'Bersihkan Keranjang', 'Lanjut Belanja', and 'Pembayaran'.

No.	Nama Produk	Jumlah	Harga	Sub-Total	Aksi
1	Bola Basket	1	Rp. 500.000	Rp. 500.000	[Edit, Delete]
2	Jam Tangan	1	Rp. 400.000	Rp. 400.000	[Edit, Delete]
				Rp. 900.000	

Gambar 6.10 Halaman Keranjang

Pada gambar 6.10 tersebut merupakan halaman keranjang yang dimana produk yang ditambahkan ke dalam keranjang akan ditampilkan pada *form* tersebut.

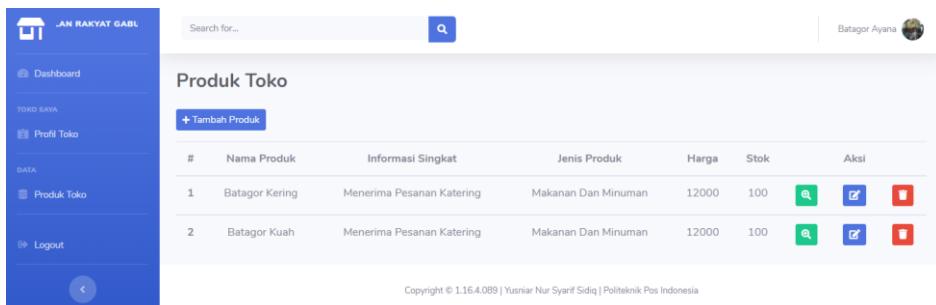
## K. Halaman Profil Toko

The screenshot shows the store profile section. The left sidebar includes 'Dashboard', 'TOKO SAYA', 'Profil Toko' (which is the active tab), 'Produk Toko', and 'Logout'. The main form contains fields for 'Email' (yusniar33@gmail.com), 'Nama Toko' (Batagor Ayana), 'Alamat Toko' (Asrama Yon Zipur 9 Rt 02 Rw 02 Kec. Cinambo Kel. Pakemitan Kota Bandung), 'Jenis Usaha' (Pengusaha Mikro), 'No. Telpon' (085723438131), and 'Gambar Toko' (a placeholder for a thumbnail image). There is also a 'Browse' button to upload a file. A blue 'Edit Profil Toko' button is at the bottom.

Gambar 6.11 Halaman Profil Toko

Pada gambar 6.11 merupakan halaman profil toko yang dimana apabila *user* akan membuka toko dan mengedit *profile* toko maka dapat melalui halaman tersebut.

## L. Halaman Produk Toko



Gambar 6.12 Halaman Produk Toko

Pada gambar 6.12 tersebut merupakan halaman produk toko yang dimana *item – item* pada akun toko akan ditampilkan pada halaman tersebut.

## M. Halaman *Form Input* Produk

Gambar 6.13 Halaman *Form Input* Produk

Pada gambar 6.13 tersebut merupakan halaman *form input* produk yang dimana penjual ingin menambahkan *item* maka dapat melalui *form* tersebut.

## 6.3 Pengujian Dan Hasil Pengujian

Pengujian merupakan hal terpenting yang bertujuan untuk menemukan kesalahan-kesalahan atau kekurangan yang ada pada perangkat lunak yang akan diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

### 6.3.1 Identifikasi Dan Rencana Pengujian

Pengujian yang dilakukan yaitu dengan pengujian *BlackBox*. Pengujian *BlackBox* digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang.

Tabel 6.3 Identifikasi Dan Rencana Pengujian

Kelas	Butir Uji	Identifikasi		Tingkat Pengujian	Jenis Pengujian	Jadwal
		SKP L	PDH UPL			
Lingkungan Aplikasi	• <i>Registration</i>	JR1	A_01	Pengujian Sistem	<i>Black Box</i>	09/12 /2019
	• <i>Login</i>	JR2	A_02		<i>Black Box</i>	09/12 /2019
	• <i>View Produk Pelangan</i>	JR3	A_03		<i>Black Box</i>	09/12 /2019

	<ul style="list-style-type: none"> <li>• <i>View Prod</i> <i>uk Penj</i> <i>ual</i></li> </ul>	JR4	A_04		<i>Black Box</i>	09/12 /2019
	<ul style="list-style-type: none"> <li>• <i>Edit Profi</i> <i>le Pela</i> <i>ngga</i> <i>n</i></li> </ul>	JR5	A_05		<i>Black Box</i>	09/12 /2019
	<ul style="list-style-type: none"> <li>• <i>Edit Profi</i> <i>le Penj</i> <i>ual</i></li> </ul>	JR6	A_06		<i>Black Box</i>	09/12 /2019
	<ul style="list-style-type: none"> <li>• <i>Cha</i> <i>nge</i> <i>Pass</i> <i>word</i></li> </ul>	JR7	A_07		<i>Black Box</i>	09/12 /2019
	<ul style="list-style-type: none"> <li>• <i>Inpu</i> <i>t Prod</i> <i>uk</i></li> </ul>	JR8	A_08		<i>Black Box</i>	09/12 /2019
	<ul style="list-style-type: none"> <li>• <i>Tam</i> <i>bah Prod</i></li> </ul>	JR9	A_09		<i>Black Box</i>	09/12 /2019

	uk Ke <i>Cart</i>					
Pengujian Antar Muka	<ul style="list-style-type: none"> <li>Bahasa Yang Diguangkan</li> </ul>		B_01	Pengujian Sistem	<i>Black Box</i>	09/12 /2019
	<ul style="list-style-type: none"> <li>Pengujian Pewarnaan</li> </ul>		B_02		<i>Black Box</i>	09/12 /2019
	<ul style="list-style-type: none"> <li>Pesan Kesalahan</li> </ul>		B_03		<i>Black Box</i>	09/12 /2019
	<ul style="list-style-type: none"> <li>Penataletak Menu</li> </ul>		B_04		<i>Black Box</i>	09/12 /2019