BAB 1

A. Definisi

1. PHP (*PHP Hypertext Preprocessor*)

Sejarah awal PHP adalah pendekatan dari *Personal Home Page* (situs personal), dulu PHP masih berbentuk *script* yang berfungsi sebagai pengolah data form dari *web server-side* yang memiliki sifat *open source*, dan memiliki nama *Form Interpreter* (FI). PHP (*Hypertext Preprocessor*) memiliki sintak yang mirip dengan ASP, Java, bahasa C, Perl dan memiliki kelebihan fungsi yang mudah dipahami dan spesifik. Versi terkini dari PHP adalah PHP7. Kelebihan PHP yaitu:

a. Bahasa pemrograman PHP tidak memerlukan *Compiler* dalam penggunaanya.
b. Memiliki sifat *open source*.
c. Banyak sekali aplikasi PHP yang gratis dan siap untuk digunakan seperti PrestaShop, WordPress, dll.
d. Bisa membuat web menjadi dinamis.
e. PHP memiliki banyak dukungan dari berbagai *web server* contohnya saja Apache.
f. PHP memiliki keunggulan lebih cepat dibandingkan dengan Java dan ASP.
g. MySQL merupakan paket aplikasi dengan PHP.

Kekurangan PHP yaitu:

a. PHP memiliki kelemahan keamanan.
b. Kode PHP bisa dibaca oleh semua orang.
c. Biaya untuk *encoding* membutuhkan biaya yang sangat mahal

2. CodeIgniter

CodeIgniter adalah sebuah kerangka kerja untuk *web* yang dibuat dalam format PHP. Format yang dibuat dapat digunakan untuk membuat sistem aplikasi *web* yang kompleks. CodeIgniter dapat mempercepat proses pembuatan *web*, karena *class* dan *module* yang dibutuhkan sudah tersedia dan *programmer* hanya tinggal menggunakannya kembali pada aplikasi *web* yang akan dibuat.

CodeIgniter memiliki banyak fitur yang membuatnya berbeda dengan *framework* lainnya. Tidak seperti beberapa *framework* PHP lainnya, dokumentasi untuk *framework* ini sangat lengkap, yang mencakup seluruh aspek dalam *framework*. CodeIgniter juga mampu berjalan pada lingkungan *Shared Hosting* karena memiliki ukuran yang sangat kecil, namun memiliki kinerja yang sangat luar biasa. Dari segi pemrograman, CodeIgniter kompatibel dengan PHP4 dan PHP5, sehingga akan berjalan dengan baik pada *web host* yang banyak dipakai saat ini. CodeIgniter menggunakan pola desain *Model-View-Controller* (MVC), yang merupakan cara untuk mengatur aplikasi web ke dalam 3 bagian yang berbeda. Pada intinya CodeIgniter juga membuat penggunaan ekstensif dari pola desain Singleton. Maksudnya adalah cara untuk me-*load class* sehingga jika *class* itu di panggil dalam beberapa kali, kejadian yang sama pada *class* tersebut akan digunakan kembali. Hal ini

sangat berguna dalam koneksi *database*, karena kita hanya ingin menggunakan satu koneksi setiap kali *class* itu digunakan.

3. Javascript

JavaScript dibuat dan didesain selama sepuluh hari oleh Brandan Eich, seorang karyawan Netscape, pada bulan September 1995. Awalnya bahasa pemrograman ini disebut Mocha, kemudian diganti ke Mona, lalu LiveScript sebelum akhirnya resmi menyandang nama JavaScript. Versi pertama dari bahasa ini hanya terbatas di kalangan Netscape saja. Fungsionalitas yang ditawarkan pun terbatas. Namun, JavaScript terus dikembangkan oleh komunitas developer yang tak henti-hentinya mengerjakan bahasa pemrograman ini.

JavaScript adalah salah satu bahasa pemrograman yang paling banyak digunakan dalam kurun waktu dua puluh tahun ini. Bahkan JavaScript juga dikenal sebagai salah satu dari tiga bahasa pemrograman utama bagi web developer:

- HTML: Memungkinkan Anda untuk menambahkan konten ke halaman web.
- CSS: Menentukan *layout, style,* serta keselarasan halaman *website*.
- JavaScript: Menyempurnakan tampilan dan sistem halaman web.

JavaScript dapat dipelajari dengan cepat dan mudah serta digunakan untuk berbagai tujuan, mulai dari meningkatkan fungsionalitas *website* hingga mengaktifkan permainan (*games*) dan *software* berbasis web. Selain itu,

terdapat ribuan template dan aplikasi JavaScript yang bisa Anda gunakan secara gratis dan semuanya ini berkat beberapa situs, seperti Github.

4. JQuery

jQuery adalah library JavaScript yang populer. Bahasa pemrograman ini dibuat oleh John Resig, tepatnya pada tahun 2006, untuk memudahkan para *developer* dalam menggunakan dan menerapkan JavaScript di *website*. jQuery bukanlah bahasa pemrograman yang berdiri sendiri, melainkan bekerja sama dengan JavaScript. Dengan menggunakan jQuery, Anda bisa melakukan banyak hal.

Seperti yang kita ketahui, menulis kode bukanlah pekerjaan yang mudah dan terkadang menyulitkan, terlebih lagi kalau ada banyak string kode yang harus ditambahkan dan diaktifkan. Di sinilah jQuery memainkan perannya. Fungsi jQuery adalah meng-*compress* berbagai baris atau line kode ke dalam satu buah fungsi sehingga Anda tidak perlu menulis kembali semua baris kode hanya untuk menyelesaikan satu task. Salah satu alasan mengapa jQuery sangat populer dan banyak digunakan adalah kemampuan lintas platformnya. Secara otomatis, jQuery akan memperbaiki error serta punya fungsi yang sama seperti ketika dijalankan di browser, seperti Chrome, Firefox, Safari, MS-Edge, IE, Anroid, dan iOS.

Adanya jQuery juga memudahkan penggunaan Ajax. Ajax menganut sistem kerja yang asinkron, dan umumnya dimulai dari kode yang tersisa. Hal ini berarti kode yang ditulis dengan Ajax dapat berkomunikasi dengan server dan memperbarui kontennya tanpa harus me-*load* kembali halaman terlebih dulu. Sayangnya, tidak selamanya penggunaan Ajax lancar dan mulus-mulus saja. Setiap browser mengaktifkan Ajax Api dengan cara yang berbeda-beda.

Karena itulah, sebisa mungkin kode harus diatur agar bisa dijalankan di semua jenis browser.

5. Ajax

AJAX adalah sebuah singkatan dari *Asynchronous Javascript and XML* dan mengacu pada sekumpulan teknis pengembangan web (*web development*) yang memungkinkan aplikasi web untuk bekerja secara *asynchronous* (tidak langsung) – memproses setiap request (permintaan) yang datang ke *server* di sisi background.

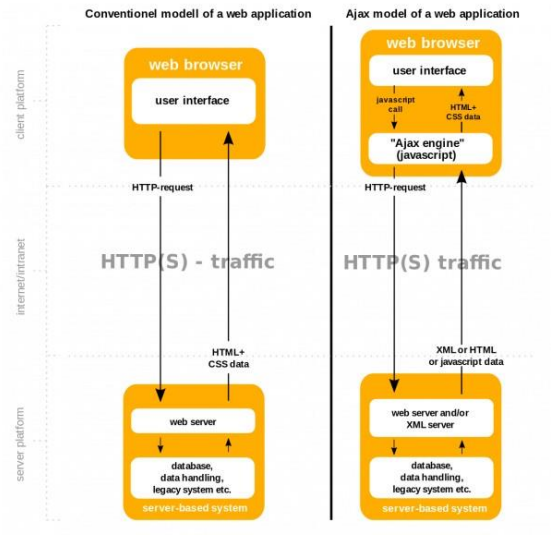
Layaknya HTML, XML atau *eXtensible Markup Language* adalah varian lain dari bahasa markup. Jika HTML dirancang untuk menampilkan data, maka XML dirancang untuk memuat dan membawa data. Baik JavaScript maupun XML bekerja secara *asynchronous* di dalam AJAX. Alhasil, aplikasi web yang menggunakan AJAX dapat mengirimkan dan menerima data dari *server* tanpa harus mereload keseluruhan halaman. Berikut adalah beberapa contoh dari penggunaan AJAX:

- Sistem *Voting* atau *Rating*
- *Chat Room*
- Notifikasi *Trending* di Twitter

AJAX bukanlah teknologi dan bukan pula bahasa pemrograman. Seperti yang telah dijelaskan sebelumnya, AJAX adalah sekumpulan teknik pengembangan web. Pada umumnya sistem ini terdiri atas:

- HTML/XHTML sebagai bahasa utama dan CSS untuk menampilkan data.
- *The Document Object Model (DOM)* untuk menampilkan data yang dinamis beserta interaksinya.
- XML untuk pertukaran data, sedangkan XSLT untuk manipulasi data. Sebagian besar *developer* mulai mengganti XML dengan JSON karena bentuknya yang mendekati JavaScript.
- Objek *XMLHttpRequest* untuk komunikasi tidak langsung (asynschronous).
- Bahasa pemrograman JavaScript untuk menyatukan semua teknologi ini.

Untuk memahami cara kerja AJAX secara keseluruhan, setidaknya Anda harus punya pemahaman teknis dasar terlebih dulu. Untungnya, prosedur umum dari cara kerja AJAX tidak begitu sulit. Lihat diagram dan tabel di bawah ini untuk perbandingannya.



Gambar 1 Diagram Alur AJAX

6. MySQL

MySQL adalah sebuah *database management system* (manajemen basis data) menggunakan perintah dasar SQL (*Structured Query Language*) yang cukup terkenal. *Database management system* (DBMS) MySQL multi pengguna dan multi alur ini sudah dipakai lebih dari 6 juta pengguna di seluruh dunia.

Gambar 2 Logo MySQL

MySQL adalah DBMS yang *open source* dengan dua bentuk lisensi, yaitu *Free Software* (perangkat lunak bebas) dan *Shareware* (perangkat lunak berpemilik yang penggunaannya terbatas). Jadi MySQL adalah *database server* yang gratis dengan lisensi GNU General Public License (GPL) sehingga dapat Anda pakai untuk keperluan pribadi atau komersil tanpa harus membayar lisensi yang ada.

Seperti yang sudah disinggung di atas, MySQL masuk ke dalam jenis RDBMS (Relational Database Management System). Maka dari itu, istilah semacam baris, kolom, tabel, dipakai pada MySQL. Contohnya di dalam MySQL sebuah database terdapat satu atau beberapa tabel. SQL sendiri merupakan suatu bahasa yang dipakai di dalam pengambilan data pada relational database atau database yang terstruktur. Jadi MySQL adalah database management system yang menggunakan bahasa SQL sebagai bahasa penghubung antara perangkat lunak aplikasi dengan database server.

Penggunaan dari MySQL sendiri tentu memiliki kelebihan dan kekurangannya, berikut adalah kelebihan dari MySQL:

- Mendukung integrasi dengan bahasa pemrograman lain
- Tidak membutuhkan RAM besar
- Mendukung *multi user*
- Bersifat *opensource*
- Struktur tabel yang fleksibel
- Tipe data yang bervariasi

Selain itu kekurangan dari penggunaan MySQL sebagai berikut:

- Kurang cocok untuk aplikasi *mobile* dan *game*
- Sulit mengelola *database* yang besar
- *Technical support* yang kurang bagus

BAB 2

B. Metode Penyelesaian Aplikasi

1. Definisi Scrum

Scrum yang masih merupakan bagian dari metodologi *agile*. *Scrum* merupakan sebuah proses pengembangan perangkat lunak dengan konsep *agile* yang dikembangkan oleh Jeff Sutherland dan Ken Schwaber pada tahun 1996. Model *scrum* biasanya digunakan karena menerapkan proses dengan siklus pendek berulang, secara aktif melibatkan pengguna untuk membangun, memprioritaskan, dan memverifikasi kebutuhan.

Kelebihan *scrum* adalah dapat menghasilkan produk sesuai dengan keinginan pengguna dan cocok untuk pengembangan sistem dengan skala kecil serta banyak perubahannya. Di sisi lain, *scrum* merupakan metodologi yang paling sering digunakan oleh banyak perusahaan besar karena merupakan salah satu metodologi yang menggunakan konsep *agile* dengan keunggulannya yaitu lebih cepat dan lebih efektif dalam pengembangan perangkat lunak.

Berikut merupakan beberapa aktifitas yang dilakukan dalam penerapan metodologi *scrum*:

1. *Backlog* atau yang lebih dikenal *product backlog* adalah daftar kebutuhan atau fitur yang memberikan nilai bisnis kepada klien. *Product backlog* dapat bertambah seiring dengan kebutuhan penggunanya.
2. *Scrum* membagi sebuah *project* ke dalam sebuah perulangan yang disebut sebagai *sprint*. Sebuah *sprint* biasanya memiliki durasi

pengerjaan hingga 4 minggu, dimana tim pengembang fokus dalam mencapai target yang jelas dan telah ditentukan.

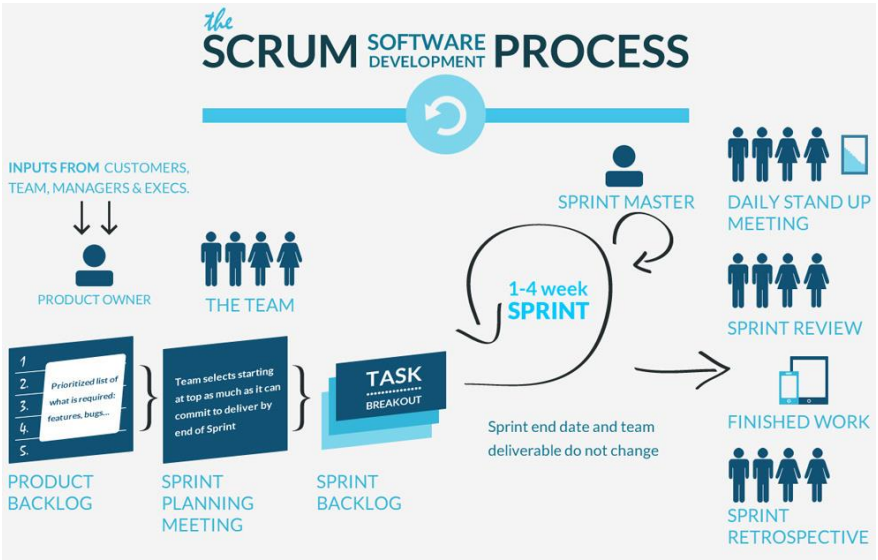
3. *Scrum meetings*, merupakan pertemuan rutin yang dilakukan perhari atau perminggu untuk mengevaluasi apa yang dikerjakan, hambatan, dan target penyelesaiannya. Dalam hal ini, penulis melakukan *scrum meetings* setiap minggu dengan membahas apa saja yang dikerjakan setiap minggunya serta hambatan apa yang dialami selama proses pengerjaan itu.
4. *Demo,* memberikan perubahan atau *update* kepada klien yang telah diimplementasikan agar dapat dilihat dan dievaluasi oleh klien.

Selain aktifitas yang telah disebutkan, ada beberapa aktor yang terlibat dalam pengembangan sistem menggunakan model *scrum*, yaitu:

1. *Product owner,* adalah orang yang bertanggung jawab atas keseluruhan sistem atau produk yang sedang dikembangkan
2. *Master scrum,* adalah orang yang mengeliminasi segala hambatan yang muncul dalam pengembangan proyeknya
3. *Team scrum,* adalah sekumpulan orang yang dapat terdiri dari *programmers*, *testers*, dan para ahli lainnya yang bertugas untuk mengembangkan produk dan memberikan kepuasan terhadap klien melalui produk yang dihasilkan.

2. Tahapan Metodologi *Scrum*

Untuk alur yang lebih jelas dari *Scrum*, maka lihatlah gambar 3 di bawah ini:



Gambar 3 Alur Scrum

Dengan diagram alur yang telah ditunjukkan pada gambar 3, alur tersebut dapat dibagi menjadi 3 tahapan utama yaitu tahap *pregame*, *game*, dan *post game*.

Berikut adalah tahapan – tahapan yang harus dilakukan:

1. Tahap *pregame* berfokus pada melakukan pengidentifikasian permasalahan yang terjadi, melakukan pengumpulan data yang relevan terkait dengan permasalahan yang terjadi serta melakukan perencanaan untuk membuat solusi dari masalah yang sedang dihadapi. Tahap *pregame* berisi aktifitas *scrum* seperti *backlog/product backlog*.

2. Tahap *game* dilakukan perancangan sesuai dengan perencanaan yang telah dibuat pada tahapan sebelumnya, biasanya pada tahapan ini berisi aktifitas *sprint* dan juga *scrum meetings*.
3. Sedangkan pada tahap *post game* dilakukan pengujian akhir guna melihat apakah sistem yang dikembangkan sudah memenuhi kebutuhan pengguna. Di dalam tahapan yang terakhir berisi aktifitas seperti *demo*.

Biasanya, untuk setiap *project* akan berbeda – beda *product backlog* dan *sprint* yang dilakukan tergantung bagaimana kompleksitas *project* itu atau faktor lainnya. Lalu untuk melakukan *scrum meetings* yang membahas kendala dan juga *milestone* dari *project* minimal satu minggu sekali yang seringkali harus menyesuaikan dengan jadwal dari *product owner*.

Untuk itu, sebelum dimulainya pengerjaan proyek perlu dilakukan pendefinisian yang jelas siapa saja *actor* yang terlibat dan juga *product backlog* apa saja yang akan dibuat. Selain itu *sprint* dari setiap *product backlog* juga perlu didefinisikan. Maka berikut adalah tabel yang mendefinisikan keseluruhan *actor, activity, product backlog,* dan juga *sprint* dari pengerjaan proyek.

Tabel 1 Tabel *Actor* dan *Activty Scrum*

Activity and Actor	*General*	
Team Scrum	M. Fauzan Aristyo	
	Muhamad Wahyunda	
	Dimas Haryoputra Maheswara	
	Riandaka Rizal	
Scrum Master	Riandaka Rizal	
Product Owner	M. Fauzan Aristyo	
	Product Backlog	*Sprint* (estimasi)
Backlog	*Create, update, delete, view* data peserta EPPM	1 minggu
	Create, update, delete, view data *workbook*	1 minggu
	Create, update, delete, view data *assessment* peserta EPPM	1 minggu
	Menu raport	3 hari
	Visualisasi data progres	4 hari
Total Estimasi Pengerjaan		4 minggu

BAB 3

C. Tutorial Pembuatan Aplikasi EPPM Go!

1. Inisiasi pengerjaan aplikasi

Untuk pembuatan aplikasi EPPM Go! ini akan dibagi menjadi 5 menu untuk admin dan juga 2 menu untuk *user*, menu yang ditujukan untuk admin adalah sebagai berikut:

- Menu Daftar Pekerja
- Menu *Workbook*
- Menu *Assessment*
- Menu Ruangkerja
- Menu Raport

Lalu untuk menu yang ditujukan untuk *user* adalah sebagai berikut:

- Menu *Workbook*
- Menu *Raport*

Untuk membuat aplikasi EPPM Go! Diperlukanlah sebuah logo sehingga aplikasi itu dapat dikenali oleh khalayak ramai, untuk logo harus dibuat dengan menarik dan atraktif yang dapat menarik perhatian dan minat orang untuk menggunakannya.

Gambar 4 Logo EPPM Go!

Ada tahapan tahapan yang harus dilalui mulai dari inisiasi sampai dengan *testing*. Agar lebih terstruktur maka harus didefinisikan terlebih dahulu tahapannya, yaitu:

1. *Download* CodeIgniter berupa zip *file*
2. Inisiasi CodeIgniter dengan *unzip file*
3. Membuat *login*
4. Buat CRUD (*Create, Read, Update, Delete*) menu daftar pekerja.
5. Buat CRUD (*Create, Read, Update, Delete*) menu *workbook* baik untuk admin dan *user*.
6. Bagi fungsi yang bisa diakses oleh admin, dan batasi fungsi yang hanya bisa diakses oleh *user*
7. Buat CRUD (*Create, Read, Update, Delete*) menu *assessment* untuk admin.
8. Di dalam menu *assessment*, buat pembobotan nilai untuk penilaian yang akan dilakukan. 40% untu nilai teori dan 60% untuk nilai praktek
9. Buat CRUD (*Create, Read, Update, Delete*) menu ruangkerja untuk admin.
10. Di dalam menu ruangkerja, buat fungsi *import* untuk mengimport *file* excel atau csv agar dapat efisien merubah data di dalam menu ruangkerja

11. Tampilkan hasil penilaian *assessment* di menu raport tanpa filter untuk admin dan filter sesuai dengan akun *user* jika digunakan oleh *user*.
12. Tambahkan fungsi *export* agar dapat diunduh di komputer masing masing.
13. Lalu pada halaman utama (*dashboard*) dari admin, tambahkan grafik untuk dapat memvisualisasikan data dari menu *workbook*.
14. Untuk grafik, gunakan ChartJS API agar dapat menggunakan grafik yang dinamis.

Pada halaman *user* atau pengguna, diperlukan sebuah gambar yang menarik agar mendapat kesan pertama dari pengguna itu sendiri berikut adalah gambar untuk halaman awal pengguna

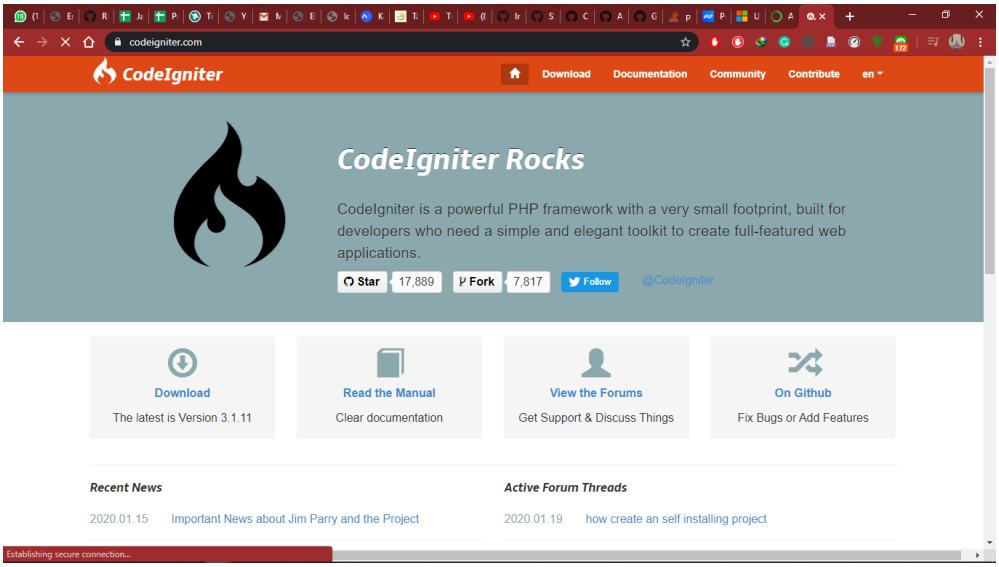
Gambar 5 Halaman Awal Pengguna

Dapat dilihat pada gambar 5, ada beberapa komponen dari gambar yang disesuaikan dengan kebiasaan dari pengguna sendiri. Hal ini merupakan suatu usaha untuk mendapatkan kesan pertama pengguna sendiri.

2. Pembuatan Aplikasi

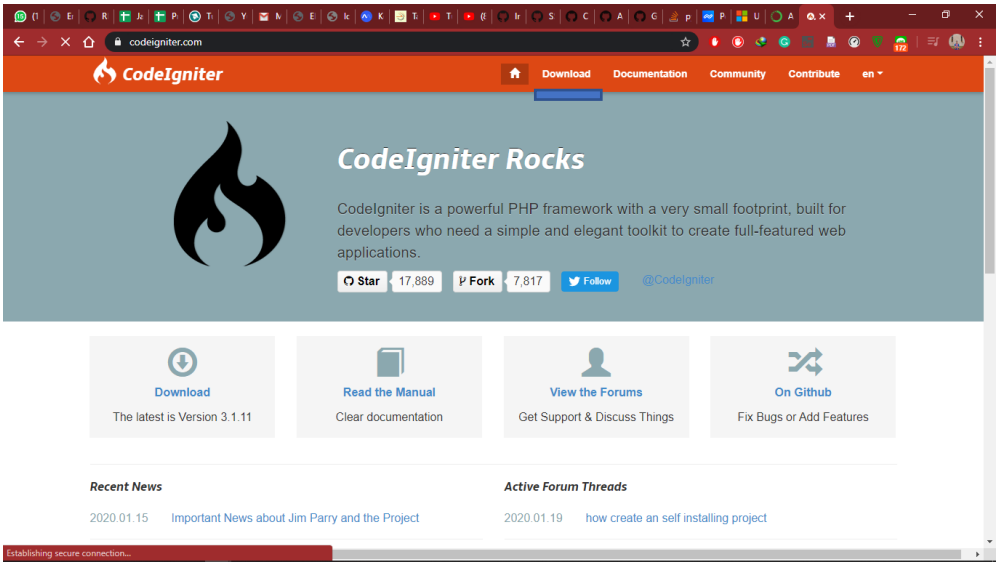
2.1. Inisiasi CodeIgniter

Untuk pembuatan aplikasi, maka harus disesuaikan dengan langkah yang telah diinisiasi sebelumnya. Maka hal pertama yang harus dilakukan adalah *download* CodeIgniter melalui *website* resminya yaitu www.codeigniter.com



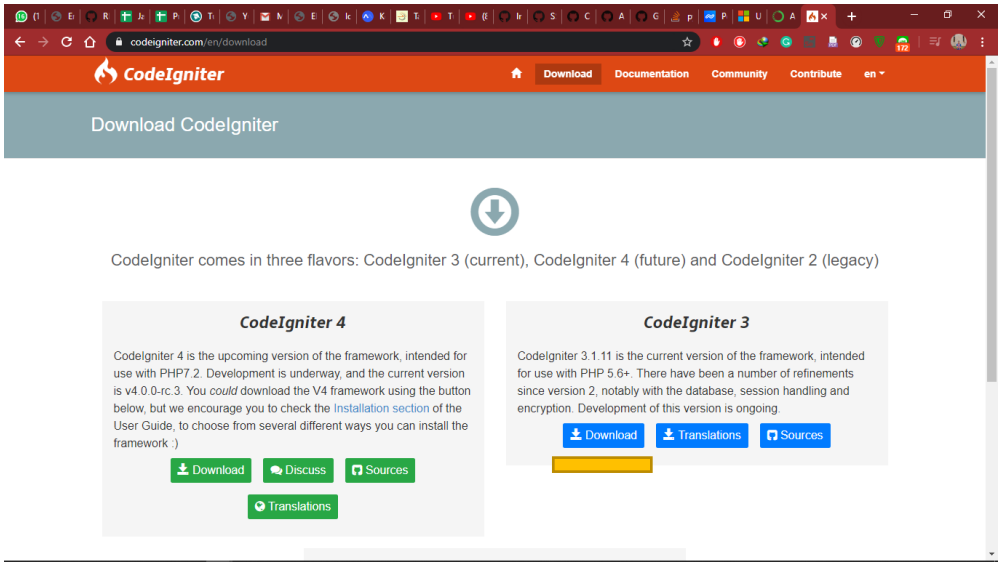
Gambar 6 Website Resmi CodeIgniter

Setelah membuka *website* CodeIgniter pilih menu *download* yang ada di atas menu bar.



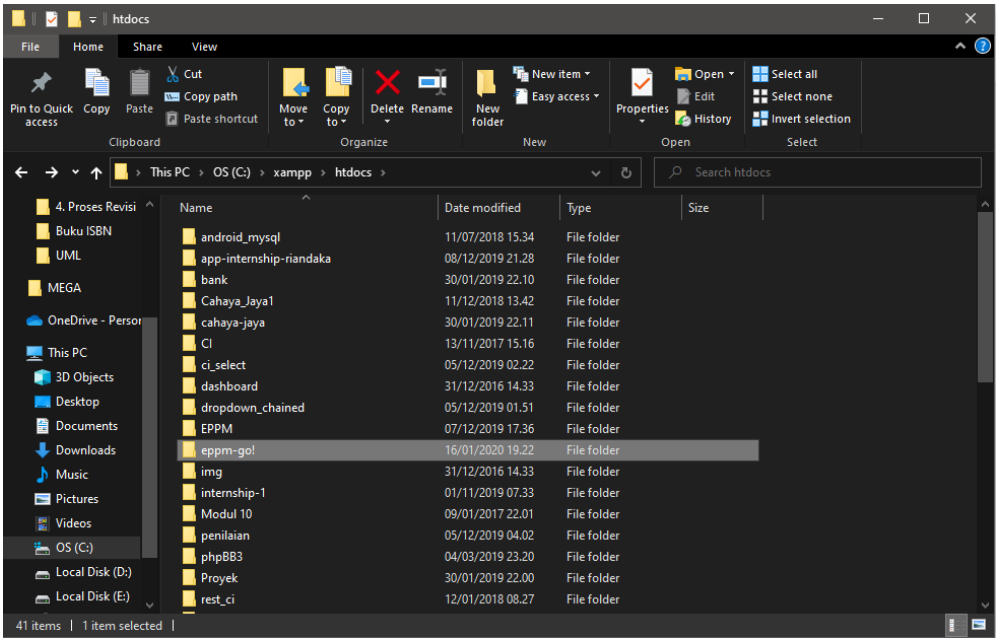
Gambar 7 Pilih Menu Download

Setelah berada di halaman *download* pilih python 3 untuk diunduh, karena saat ini python 3 adalah versi yang paling banyak digunakan oleh pengembang di dunia.



Gambar 8 CodeIgniter 3

Tunggulah beberapa saat untuk selesai mengunduh *file* zip dari CodeIgniter. Setelah itu *unzip file* CodeIgniter yang baru saja kita unduh ke dalam folder C://xampp/htdocs dan *rename* folder CodeIgniter menjadi eppm-go! Seperti yang ditunjukkan pada gambar 9.



Gambar 9 Folder eppm-go!

Setelah selesai membuat folder eppm-go! silakan lanjut ke langkah selanjutnya, yaitu membuat CRUD (*Create, Read, Update, Delete*) untuk menu daftar pekerja.

2.2. Login

Buat *file* dengan nama Login.php pada controller, nama *file* untuk controller selalu disarankan untuk berawalan huruf besar. Lalu buka *file* tersebut dan ketikkan seperti yang ditunjukkan pada gambar 10

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

class Login extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        //load library form validasi
        $this->load->library('form_validation');
        //load model admin
        $this->load->model('admin');
    }
```

Gambar 10 Controller Login.php

- Class Login extends CI_Controller = maksudnya adalah membuat *class* Login dengan memakai *library* CI_Controller
- Public function __construct() = maksudnya adalah menggunakan function construct
- $this->load->library('form_validation') = maksudnya adalah *load library* form validasi
- $this->load->mode('admin') = maksudnya adalah *load* model admin

Setelah itu kita lanjutkan kembali dengan mengetikkan sesuai dengan yang ditunjukkan oleh gambar 11

```php
public function index()
{

        if($this->admin->is_logged_in())
        {
            //jika memang session sudah terdaftar, maka redirect ke halaman dahsboard
            redirect("admin/dashboard");

        // }elseif($this->user->is_logged_in()){

        //       redirect("user/dashboard");

            }else{

            //jika session belum terdaftar

            //set form validation
            $this->form_validation->set_rules('username', 'Username', 'required');
            $this->form_validation->set_rules('password', 'Password', 'required');

            //set message form validation
            $this->form_validation->set_message('required', '<div class="alert alert-danger" style="margin-top: 3px">
                <div class="header"><b><i class="fa fa-exclamation-circle"></i> {field}</b> harus diisi</div></div>');

            //cek validasi
            if ($this->form_validation->run() == TRUE) {

            //get data dari FORM
            $username = $this->input->post("username", TRUE);
            $password = $this->input->post('password', TRUE);

            //checking data via model
            $checking = $this->admin->check_login('account', array('username' => $username), array('password' => $passw
```

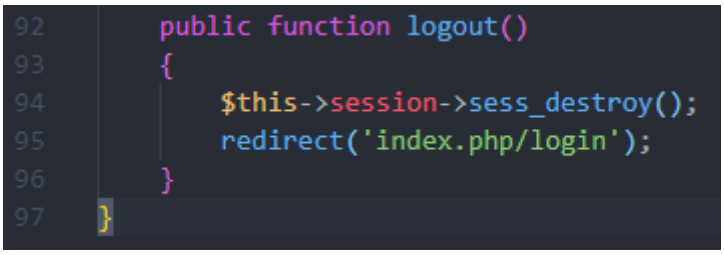
Gambar 11 Controller Login.php

Seperti yang dilihat pada gambar 11, untuk function index akan mengecek terlebih dahulu apakah sudah ada *session* yang dibuat apa belum, jika sudah maka akan langsung *redirect* ke halaman *dashboard* admin. Jika belum maka akan membuat form validasi dengan *username* dan *password* yang harus diisi lalu akan dicek apakah data *username* dan *password* yang kita masukkan sudah ada atau belum,

```
//jika ditemukan, maka create session
if ($checking != FALSE) {
    foreach ($checking as $apps) {

        $session_data = array(
            'employee_id'    => $apps->employee_id,
            'user_name'      => $apps->username,
            'user_pass'      => $apps->password,
            'user_email'     => $apps->user_email,
            'user_nama'      => $apps->employee_name,
            'directorate'    => $apps->directorate,
            'role'           => $apps->role
        );
        //set session userdata
        // print_r($session_data); die;
        $this->session->set_userdata($session_data);

        //redirect berdasarkan level user
        if($this->session->userdata("role") == "admin"){
            redirect('admin/dashboard');
        }else{
            redirect('user/dashboard');
        }

    }
}else{

    $error = 'Username atau Password Anda salah. Silakan ulangi Kembali';
    '<div class="alert alert-danger" style="margin-top: 3px">
    <div class="header"><b><i class="fa fa-exclamation-circle"></i> ERROR</b> username atau password salah!
    $this->load->view('login', $error);
}
```

Gambar 12 Controller Login.php

Jika sudah ada maka akan langsung dibuat *session*-nya dan akan *redirect* ke halaman awal sesuai dengan hak aksesnya masing masing. Jika itu admin maka akan diarahkan ke halaman *dashboard* admin, sedangkan jika *user* maka akan diarahkan ke halaman awal untuk pengguna.



```
92      public function logout()
93      {
94          $this->session->sess_destroy();
95          redirect('index.php/login');
96      }
97  }
```

Gambar 13 Controller Login.php

Lalu setelah itu, kita membuat function logout() yang berguna untuk keluar dari aplikasi dengan cara menghapus atau menghancurkan *session* yang sedang berjalan dan mengarahkannya ke halaman login. Lanjut lagi buat kembali *file* dengan nama Admin.php pada folder model

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

class Admin extends CI_Model
{
    //fungsi cek session logged in
    function is_logged_in()
    {
        return $this->session->userdata('id');
    }

```

Gambar 14 Models Admin.php

Seperti yang dapat dilihat pada gambar 14, ada function is_logged_in() yang berguna untuk mengecek apakah *session* telah ada dan dibuat. Lalu hasil yang dikeluarkan dari function ini berupa userdata dengan membawa id dari user yang sedang *login*. Lalu dilanjutkan dengan function is_role() yang ditunjukkan oleh gambar 15, dimana akan mengecek *role* dari setiap pengguna yang masuk sehingga akan diberikan hak akses sesuai dengan level atau *role*-nya.

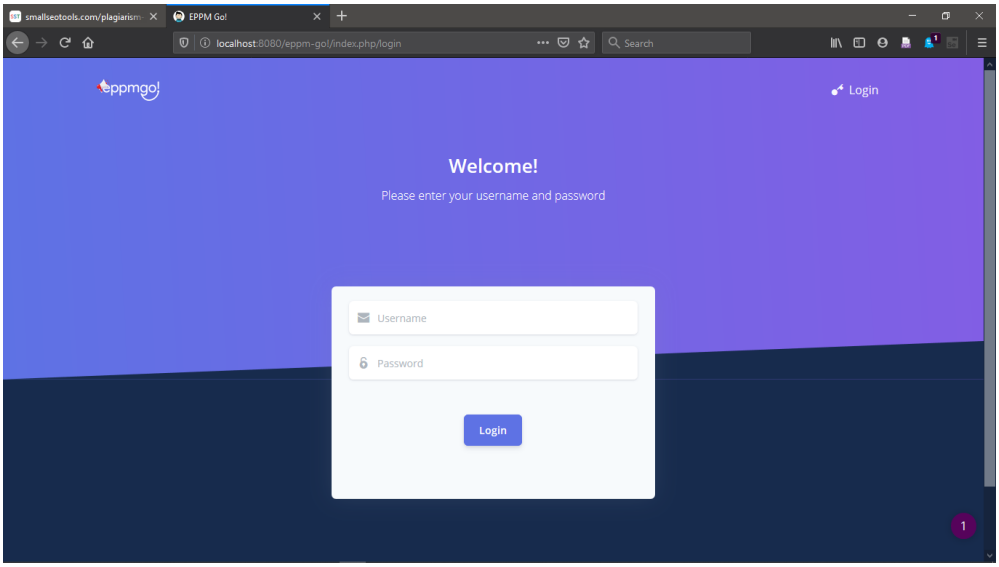
```php
//fungsi cek level
function is_role()
{
    return $this->session->userdata('role');
}
```

Gambar 15 Models Admin.php

Lalu untuk *view*-nya silakan berkreasi sendiri menggunakan bootstrap pilihan anda sendiri. Disini saya menggunakan bootstrap dari argon dashboard. Pastikan pada tombol login sudah diisikan form methodnya seperti yang ditunjukkan pada gambar 16 berikut.

```php
<form method="POST" action="<?=base_url();?>index.php/login">
```

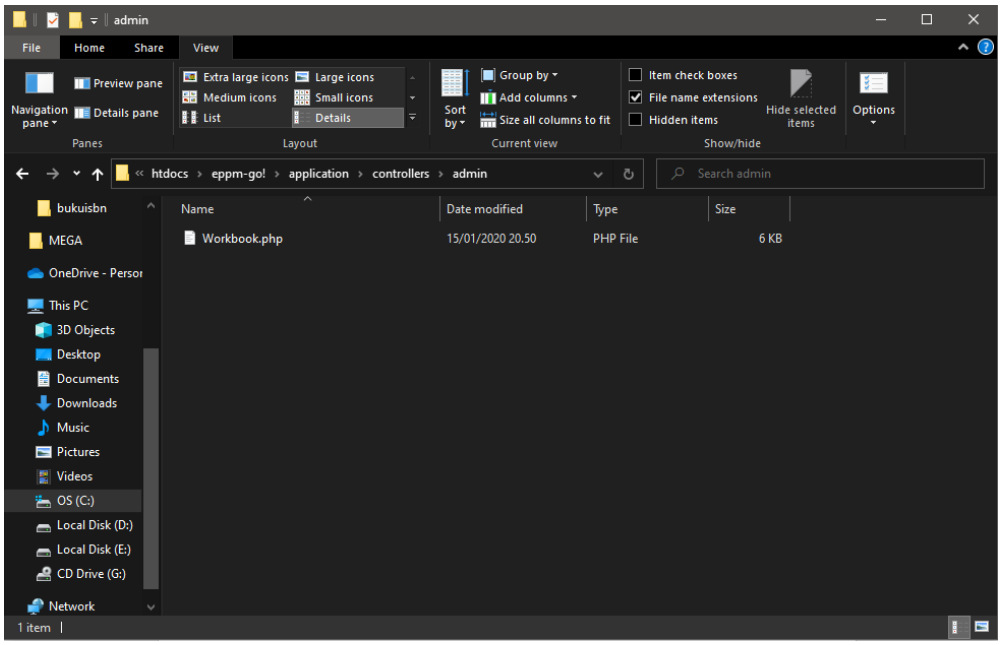
Gambar 16 Form Method Login



Gambar 17 Halaman Login

2.3. CRUD Menu *Workbook*

Untuk membuat CRUD menu daftar pekerja, mulai sekarang akan kita akan bekerja menggunakan HTML, PHP, dan juga *database*-nya. Untuk *database*-nya akan dijelaskan di lain bab. Pertama buatlah *file* bernama Workbook.php dan folder dengan nama admin, ini ditujukan untuk membedakan *file* untuk admin dan juga untuk *user*. Lalu tempatkan pada folder controller pada direktori eppm-go! Untuk penamaan *file* controller disarankan untuk selalu berawalan huruf kapital, agar penamaan *file* menjadi lebih rapi dan dapat dibedakan secara visual. Tetapi apabila lebih suka dibuat dengan berawalan huruf kecil maka sesuaikan saja dengan keinginan masing masing.



Gambar 18 Pembuatan Workbook.php

Setelah membuat file Workbook.php, maka selanjutnya yang akan kita lakukan adalah mulai untuk ngoding. Pada Workbook.php ketik kode berikut sesuai dengan yang ditunjukkan oleh gambar 19.

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

class Workbook extends CI_Controller
{
    public function __construct()
    {
        parent::__construct();
        $this->load->model("workbook_model"); //load model workbook
        $this->load->library('form_validation'); //load library form validation
        $this->load->helper(array('form', 'url'));
        if($this->workbook_model->is_role() != "admin"){
            redirect("index.php/login");
        }
    }
```

Gambar 19 Controller Workbook.php

Disini diawali dengan dengan me-*load* dari model workbook yang akan kita bahas selanjutnya, dan juga me-*load library* form validasi. Selain itu, pada controller Workbook.php akan terlebih dahulu mengecek *role* yang akan mengakses halaman *workbook* awal yang function-nya diambil dari model workbook, jika *role* yang masuk adalah admin maka akan langsung menuju dan mengakses halaman *workbook*. Sedangkan bila sebaliknya, maka akan langsung menuju halaman *login.*

Selanjutnya adalah membuat function index, yang di dalamnya terdapat sintaks yang berfungsi untuk me-*load* tampilan halaman *workbook* awal dengan mengambil data yang diinisiasikan melalui method getAll() dari model workbook_model seperti yang ditunjukkan oleh gambar 20.

```
17    public function index()
18    {
19        $data["workbook"] = $this->workbook_model->getAll(); //ambil data dari model
20        $this->load->view("admin/workbook/list_workbook", $data); //Load view data model ke workbook
21    }
22
```

Gambar 20 Controller Workbook.php

Kemudian, lanjut dengan membuat function add yang berfungsi untuk me-*load* tampilan halaman form input progres *workbook* dengan mengambil data *stages* dari modul pelatihan EPPM dari model workbook_model seperti yang ditunjukkan oleh gambar 21.

```
23    public function add()
24    {
25        $data["workbook"] = $this->workbook_model->get_data_stage();
26        $this->load->view("admin/workbook/new_form", $data); //Load isi form workbook
27    }
28
```

Gambar 21 Controller Workbook.php

Lalu dilanjutkan dengan membuat function aksi_add yang berfungsi untuk menyimpan data yang telah kita *input*-kan pada form *workbook* dengan menggunakan model workbook_model dan memberikan pesan apabila data telah sukses disimpan ke dalam *database* dan akan langsung mengarahkan ulang ke halaman *workbook* seperti yang ditunjukkan oleh gambar 22.

```php
public function aksi_add()
{
//    print_r('tes'); die;
    $this->workbook_model->save(); //objek modelredir
    $this->session->set_flashdata('Sukses', 'Data Anda Berhasil Disimpan'); //pesan berhasil
    redirect('index.php/admin/workbook');
}
```

Gambar 22 Controller Workbook.php

Jika sudah selesai membuat function aksi_add, maka selanjutnya adalah membuat function baru di dalam *file* Workbook.php dengan nama function edit yang berguna untuk menampilkan halaman edit workbook yang data form *input workbook* diambil dari *database* dan dilempar melalui model workbook_model berdasarkan id yang dipilih seperti yang ditunjukkan oleh gambar 23 berikut.

```php
public function edit($id=null)
{
    $data["workbook"] = $this->workbook_model->getById($id); //mengambil data berdasarkan id
    $data["status"] = ['Not Yet Started' , 'On Progress' , 'Acc. Done']; //mengambil data berdasarkan id
    if (!$data["workbook"]) show_404();// jika tidak ada show error
    $this->load->view("admin/workbook/edit_form", $data); //Load edit form workbook
}
```

Gambar 23 Controller Workbook.php

Selanjutnya sama seperti function add, kita harus membuat function aksi_edit untuk melakukan proses edit pada aplikasi. Function aksi_edit berfungsi untuk memberikan perubahan pada form yang telah kita simpan sebelumnya. Function aksi_edit dimulai dengan menginisiasikan *config* atau pengaturan tentang folder mana yang menjadi tempat penyimpanan gambar dan mana saja ekstensi untuk *file* gambar yang didukung. Lalu setelah itu akan me-*load library upload* dengan menggunakan *config* yang telah diinisiasi sebelumnya. Jika tidak ada gambar yang diubah atau di-*upload* maka hanya ubah data yang dimasukkan nilai yang baru dan bila sukses akan langsung diarahkan kembali menuju halaman *workbook* awal.

```
    public function aksi_edit()
    {
        $config['upload_path'] = './upload/workbook'; //path folder
        $config['allowed_types'] = 'gif|jpg|png|jpeg|bmp'; //type yang dapat diakses bisa anda sesuaikan


         $this->load->library('upload', $config);
         if ( ! $this->upload->do_upload('image')){
        $id = $this->input->post('workbook_id');
                    $data = array (
                        'workbook_id' => $this->input->post('workbook_id'),
                        'stage_id' => $this->input->post('stage_id'),
                        'modul_id' => $this->input->post('modul_id'),
                        'employee_name' => $this->input->post('employee_name'),
                        'progress' => $this->input->post('progress'),
                        'status' => $this->input->post('status'),
                        'image' => $this->input->post('old_image')
                    );

                        // print_r($data); die;
                    $this->workbook_model->update($id,$data);
                    $this->session->set_flashdata('success', 'Data Anda Berhasil Diupdate');
                    redirect('index.php/admin/workbook');
         }else{
```

Gambar 24 Controller Workbook.php

tetapi bila ada gambar yang diubah atau gambar yang baru saja di-*upload* maka masukkan seluruh data yang baru saja diubah ke dalam *database* dan bila sukses akan langsung diarahkan kembali ke halaman awal *workbook*

```
    }else{
    $id = $this->input->post('workbook_id');
        $image = $this->upload->data();
                $data = array (

                    'workbook_id' => $this->input->post('workbook_id'),
                    'stage_id' => $this->input->post('stage_id'),
                    'modul_id' => $this->input->post('modul_id'),
                    'employee_name' => $this->input->post('employee_name'),
                    'progress' => $this->input->post('progress'),
                    'status' => $this->input->post('status'),
                    'image' => $image['file_name']
                );
                    // print_r($data); die;
                $this->workbook_model->update($id,$data);
                $this->session->set_flashdata('success', 'Data Anda Berhasil Diupdate');
                redirect('index.php/admin/workbook');
            }
}
```

Gambar 25 Controller Workbook.php

Setelah membuat fungsi untuk *add* dan *edit*, maka kita juga harus membuat fungsi untuk menghapus data yang salah. Oleh karena itu, kita buat function delete yang berfungsi untuk menghapus data yang kita pilih atau sesuai dengan id-nya. Jika sudah maka akan diarahkan kembali ke halaman awal *workbook*.

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

        if ($this->workbook_model->delete($id)){
            redirect('index.php/admin/workbook');
        }
    }
}
```

Gambar 26 Controller Workbook.php

Setelah selesai membuat Workbook.php untuk admin, selanjutnya adalah kita harus membuat modelsnya yang berfungsi untuk melakukan manipulasi data yang kita inginkan. Silakan membuat *file* dengan nama Workbook_model.php di dalam folder models. Lanjut dengan mengetikkan *line* berikut sesuai dengan gambar 27.

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

class Workbook_model extends CI_Model
{
//    private $_table = "workbook";

    public $workbook_id;
    public $stage_id;
    public $modul_id;
    public $employee_id;
    public $employee_name;
    public $status;
    public $image = "default.jpg";
```

Gambar 27 Models Workbook_model.php

Pada *line* di atas, kita terlebih dahulu menginisiasikan variabel variabel yang akan digunakan, dengan membuat *class* Workbook_model dengan menggunakan *library* CI_Model. Jika sudah selesai, kita harus membuat aturan atau *rules* mana saja *field* yang harus diisi seperti yang ditunjukkan oleh gambar 28 berikut ini.

```php
public function rules()
{
    return [

        ['field' => 'stage_id',
        'rules'  => 'required'],

        ['field' => 'modul_id',
        'rules'  => 'required'],

        ['field' => 'employee_id',
        'rules'  => 'required'],

        ['field' => 'employee_name',
        'rules'  => 'required'],

        ['field' => 'progress',
        'rules'  => 'required'],

        ['field' => 'status',
         'rules' => 'required'],

        ['field' => 'image',
        'rules' => 'required'],
    ];
}
```

Gambar 28 Models Workbook_model.php

Lalu kita harus diatur pula hak akses, sehingga setiap pengguna hanya dapat menu yang sesuai dengan hak akses yang diberikan.

```php
//fungsi cek level
function is_role()
{
    return $this->session->userdata('role');
}
```

Gambar 29 Models Workbook_model.php

Jika sudah selesai mengatur hak aksesnya, maka langsung kita ambil datanya dari *database* di tabel workbook dengan membuat function getAll(). Lalu agar menampilkan data pada halaman *workbook* menjadi lengkap maka kita melakukan teknik *join* yang menggabungkan salah satu *field* di tabel *parent* ke dalam tabel *children*-nya sesuai dengan foreign key-nya.

```
48      public function getAll()
49      {
50          $this->db->select('workbook.*, account.employee_name as yaww');
51          $this->db->join('account', 'workbook.employee_id = account.employee_id');
52          $this->db->from('workbook');
53          return $this->db->get()->result();
54      }
```

Gambar 30 Models Workbook_model.php

Sama seperti function getAll(), kita membuat function getById() yang berfungsi untuk mengambil data berdasarkan id-nya pada tabel workbook di *database*. Dan kita juga melakukan teknik *join* yang menggabungkan field di tabel *parent* ke dalam tabel *children*-nya sesuai dengan foreign key-nya.

```
56      public function getById($id)
57      {
58          $this->db->select('workbook.*, account.employee_name as yaww');
59          $this->db->join('account', 'workbook.employee_id = account.employee_id');
60          $this->db->from('workbook');
61          $this->db->where('workbook_id', $id);
62          return $this->db->get()->row();
63      }
```

Gambar 31 Models Workbook_model.php

Lanjut lagi, kita buat function get_data_stage() yang berfungsi untuk mengambil data *stages* pada tabel workbook di kolom stage. Setelahnya kita buat function save() yang berfungsi untuk menyimpan *input*-an pada form dan memasukkan data tersebut ke dalam tabel *workbook* di dalam *database* seperti yang ditunjukkan pada gambar 32.

```
65      public function get_data_stage()
66      {
67          $query = $this->db->get('stage');
68          return $query;
69      }
70
71      public function save()
72      {
73          // $post = $this->input->post();
74          // print_r($post); die;
75          // $this->workbook_id = $post["workbook_id"];
76          $data = array (
77          "stage_id" => $this->input->post("stage_id"),
78          "modul_id" => $this->input->post("modul_id"),
79          "employee_id" => $this->input->post("employee_id"),
80          "employee_name" => $this->input->post("employee_name"),
81          "progress" => $this->input->post("progress"),
82          "status" => $this->input->post("status"),
83          "image" => $this->_uploadImage()
84          );
85
86          $this->db->insert("workbook", $data);
87      }
88
```

Gambar 32 Models Workbook_model.php

Jika sudah selesai, maka langkah selanjutnya adalah membuat function update, dimana data yang dimasukkan akan di*update* berdasarkan id yang sedang digunakan.

```
89      public function update($id, $data)
90      {
91          $this->db->where('workbook_id',$id);
92          $this->db->update("workbook", $data);
93      }
94
```

Gambar 33 Models Workbook_model.php

Jika sudah maka langkah selanjutnya adalah membuat function delete yang akan menghapus data yang dipilih dan akan dihapus sesuai dengan id-nya. Selanjutnya adalah membuat function _uploadImage(). Lalu kita memberikan *config* tentang *upload* gambar. Disimpan di direktori mana, lalu jenis ekstensi apa saja yang dibolehkan, lalu besar *file size* yang diperbolehkan. Setelahnya baru *load library upload* dan menggunakan *config* yang baru saja diberikan. Jika sedang melakukan *upload* dengan nama *file* aslinya, maka replace nama *file* menjadi default.jpg.

Langkah terakhir dalam membuat workbook_model.php adalah membuat function _deleteImage() yang akan dihapus sesuai dengan id yang dipilih, lalu hapus data yang ada di dalam tabel workbook di *database* seperti yang ditunjukkan oleh gambar 34 di bawah ini.

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

    private function _uploadImage()
    {
        $config['upload_path']          = './upload/workbook/';
        $config['allowed_types']        = 'gif|jpg|png';
        $config['file_name']            = $this->workbook_id;
        $config['overwrite']            = true;
        $config['max_size']             = 10240; // 10MB
        // $config['max_width']            = 1024;
        // $config['max_height']           = 768;

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

        if ($this->upload->do_upload('image')) {
            return $this->upload->data("file_name");
        }

        return "default.jpg";
    }

    private function _deleteImage($id)
    {
        $workbook = $this->getById($id);
            if ($workbook->image != "default.jpg") {
            $filename = explode(".", $workbook->image)[0];
        return array_map('unlink', glob(FCPATH."upload/workbook/$filename.*"));
        }
    }
}
```

Gambar 34 Models Workbook_model.php

Jika models dan controller telah kita buat, maka langkah selanjutnya adalah membuat view. Untuk view silakan mengunduh *bootstrap* sesuai dengan keinginan masing masing, disini saya masih menggunakan argon *dashboard bootstrap* dikarenakan *bootstrap* yang saya gunakan memiliki keseluruhan halaman yang dibutuhkan. Sehingga hanya diperlukan sedikit modifikasi dari argon *dashboard*-nya.

Buatlah folder admin di dalam folder views, lalu jika sudah membuat folder admin selanjutnya buat kembali sebuah folder dengan nama workbook. Nah pada folder workbook kita selanjutnya membuat list_workbook.php. Mengapa banyak sekali membuat folder pada folder views? Ini dimaksudkan untuk merapikan direktori kerja kita, selain itu dalam pengerjaan aplikasi akan menjadi lebih efisien. Sehingga direktori dari beberapa folder yang telah dibuat akan terlihat menjadi seperti ini C://xampp/htdocs/eppm-go!/application/views/admin/workbook.

Ingat! Hanya ubah beberapa komponen di dalam *bootstrap* sesuai dengan yang kita perlukan. Jangan mengubah semua komponen, karena akan berakibat *layout* yang telah diatur menjadi berantakan. Lalu pada *file* list_workbook.php jangan lupa tambahkan *link* atau URL (*Uniform Resource Locator*) menuju menu menu yang telah atau akan dibuat. *Link* akan ada di setiap halaman yang kita buat, jadi pastikan jangan lupa menaruh *link* pada tiap tiap halaman yang dibuat Seperti yang ditunjukkan oleh gambar 35 di bawah ini

```
<ul class="navbar-nav">
<li class="nav-item" >
  <a class=" nav-link" href="<?php echo base_url();?>index.php/admin"> <i class="ni ni-tv-2 text-primary"></i> Dashboard
  </a>
</li>
<li class="nav-item" >
  <a class="nav-link " href=" <?php echo base_url(); ?>admin/account">
    <i class="ni ni-bullet-list-67 text-red"></i> Daftar Peserta
  </a>
</li>
<li class="nav-item" class=" active" >
  <a class="nav-link active" href=" <?php echo base_url(); ?>admin/workbook">
    <i class="ni ni-bullet-list-67 text-red"></i> Workbook
  </a>
</li>
```

Gambar 35 Link/URL Menu

Setelah membuat *link* atau URL selanjutnya kita akan membuat tabel untuk menampilkan data yang kita punya. Pertama buat terlebih dahulu kolom untuk nama *field*-nya.

```
<table class="table align-items-center table-flush">
  <thead class="thead-light">
    <tr>
      <th scope="col">ID Stage</th>
      <th scope="col">ID Modul</th>
      <th scope="col">Nama Pekerja</th>
      <th scope="col">Progress</th>
      <th scope="col">Status</th>
      <th scope="col">Lembar Pengesahan</th>
      <th scope="col"></th>
    </tr>
  </thead>
  <tbody>
```

Gambar 36 Inisiasi Tabel Workbook

Setelah membuat kolom untuk nama *field*-nya maka kita selanjutnya harus membuat kolom untuk menempatkan atau mewadahi data yang akan ditampilkan. Agar memudahkan pembuatan tabel, kita akan menggunakan DataTables yang berguna untuk membuat tabel secara instan hanya dengan menggunakan sintaks jQuery yang masih merupakan *library* untuk Javascript.

```
<tbody>
  <?php foreach ($workbook as $workbook): ?>
  <tr>
      <td width="150">
          <?php echo $workbook->stage_id ?>
      </td>
      <td>
          <?php echo $workbook->modul_id ?>
      </td>
      <td>
        <?php echo $workbook->yaww ?>
      </td>
      <td>
        <?php echo $workbook->progress ?>
      </td>
      <td>
        <?php echo $workbook->status ?>
      </td>
      <td>
      <img src="<?php echo base_url('upload/workbook/'.$workbook->image) ?>" width="64" />
      </td>
      <td width="250">
          <a href="<?php echo site_url('admin/workbook/edit/'.$workbook->workbook_id)?>"
          class="btn btn-small"><i class="fas fa-edit"></i>Update</a>
      <a onclick="deleteConfirm('<?php echo site_url('admin/workbook/delete/'.$workbook->workbook_id) ?>')"
      href="#" class="btn btn-small text-danger"><i class="fas fa-trash"></i>Hapus</a>
      </td>
  </tr>
  <?php endforeach; ?>

</tbody>
</table>
```

Gambar 37 Views list_workbook.php

Disini kita menggunakan perulangan *foreach* yang mengambil data dari *models workbook* dan diinisiasikan sebagai variabel $workbook. Lalu memanggil *field* yang diperlukan ke dalam tabelnya. Karena ada perulangan *foreach* maka setiap variabel yang diinisiasikan akan diulang secara terus

menerus hingga semua datanya terpanggil. Lalu kita membuat sebuah tombol untuk melakukan perubahan atau *update* dengan mengambil id dari data yang kita pilih, begitupun dengan tombol hapus sama seperti tombol *update*. Dan jangan lupa kita akhiri perulangan *foreach*-nya.

Setelah itu, untuk sintaks jQuery dari DataTables kita taruh dibawah setelah sintaks pembuatan tabel. Jangan lupa untuk menyisipkan tag <script> karena masih satu bagian dari JavaScript. Selain itu, karena kita menggunakan *resources* dari luar ada beberapa cara yang bisa digunakan, yaitu dengan cara:

1. Mengunduh *library*-nya dan disimpan pada folder *assets* yang akan dipanggil di atas untuk CSS di bagian tag *head* dan di bawah untuk JavaScript di bagian tag *footer*. Kekurangan cara ini akan menggunakan *local resource* dari PC atau laptop kita.
2. Menyalin *link CDN* (*Content Delivery Network*) yang akan dipanggil di atas untuk CSS di bagian tag *head* dan di bawah untuk JavaScript di bagian tag *footer*. Kekurangan cara ini hanya memerlukan akses internet. *Link* CDN DataTables untuk CSS adalah //cdn.datatables.net/1.10.20/css/jquery.dataTables.min.css dan *link* CDN DataTables untuk JavaScript adalah //cdn.datatables.net/1.10.20/js/jquery.dataTables.min.js

```
<script>$(document).ready( function () {
  $('table').DataTable();
} );
</script>
```

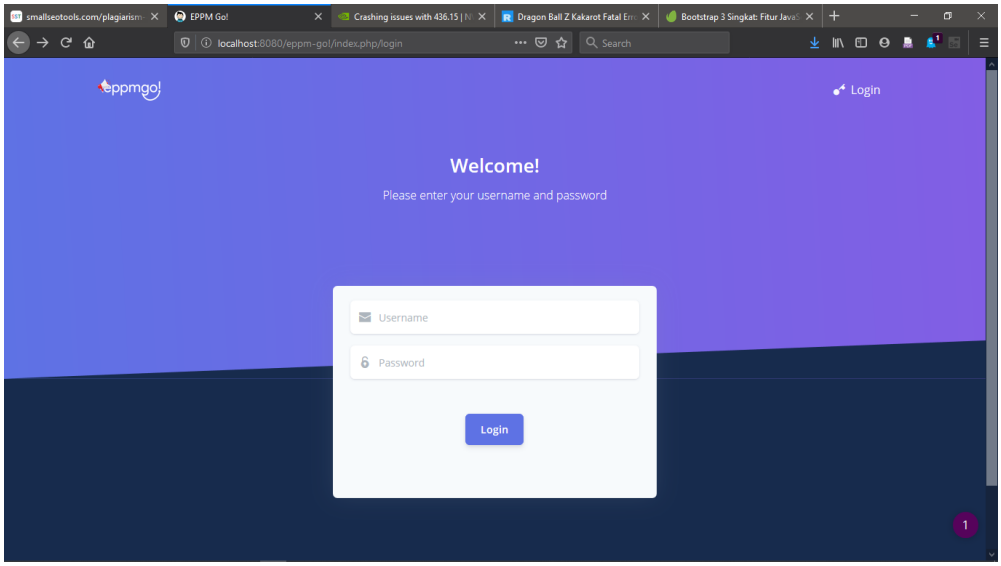
Gambar 38 Sintaks DataTables

Lalu kita akan menggunakan modals yang juga masih bagian dari JavaScript, yang akan digunakan untuk tombol hapus. Modals adalah sebuah kotak dialog kecil yang biasanya muncul di atas halaman yang sedang kita buka. Sehingga apabila tombol hapus tidak sengaja ditekan maka akan menampilkan dialog konfirmasi untuk menghapus sebuah data, seperti yang ditunjukkan oleh gambar 39 berikut.

Gambar 39 Modal Tombol Hapus

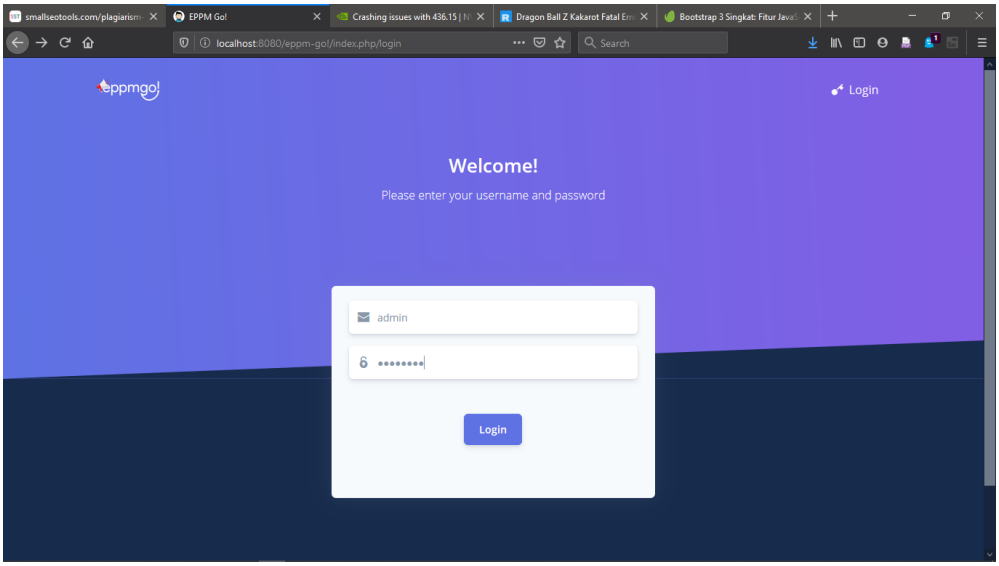
Jika controller, models, dan views telah kita buat maka langkah selanjutnya adalah mencoba untuk memanggil halaman web yang baru saja kita buat. Apakah terdapat error saat dibuka ataukan sudah sesuai dengan yang kita inginkan? Untuk itu silakan coba membuka *browser* atau peramban sesuai dengan preferensi kita. Dan coba ketikkan localhost:8080/eppm-go!/index.php/login.

Untuk pengaturan localhost masing masing pengguna pasti akan berbeda beda, dikarenakan *port* yang digunakan untuk mengakses *localhost*-nya. Karena saya menggunakan *port* 8080 untuk mengakses *localhost* makanya ada tambahan nama *port* di samping *localhost*.

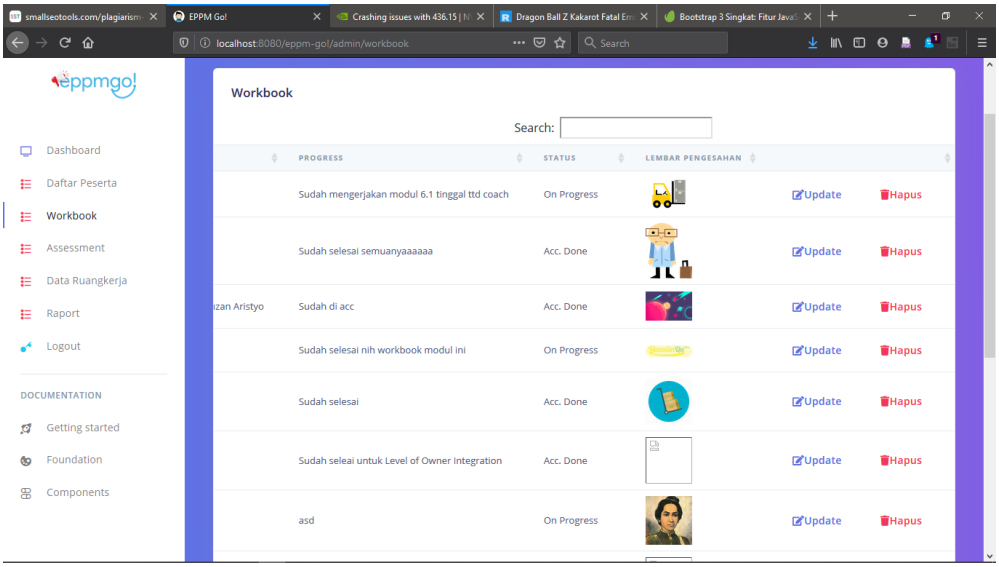


Gambar 40 Halaman Login

Berhasil masuk ke halaman *login*, maka kita harus mengisi *username* dan *password* agar bisa mengakses halaman berikutnya. Karena halaman *workbook* yang baru saja kita buat adalah halaman untuk admin maka kita akan menggunakan *username* admin dan juga *password* admin123. Karena admin yang mengakses maka semua menu dan fitur fitur yang disediakan dapat diakses oleh admin. Sedangkan untuk pengguna akan mendapatkan limitasi terhadap menu dan fitur yang dapat digunakan.



Gambar 41 Halaman Login



Gambatr 42 Halaman List Workbook

Setelah halaman *workbook* selesai dibuat, maka selanjutnya adalah membuat form *update* untuk dapat mengubah beberapa data yang kita masukkan sebelumnya. Untuk form *update* baru akan muncul setelah mengklik tombol *update*, maka langsung saja kita buat sebuah *file* dengan nama edit_form.php pada direktori yang sama yaitu C://xampp/htdocs/eppm-go!/application/views/admin/workbook.

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

      <form action="<?php echo base_url();?>index.php/admin/workbook/aksi_edit" method="post" enctype="multipart/form

      <input type="hidden" name="workbook_id" value="<?php echo $workbook->workbook_id?>" />

        <div class="form-group">
          <label for="name">Stage</label>
          <input class="form-control <?php echo form_error('stage_id') ? 'is-invalid':'' ?>"
           type="text" name="stage_id" placeholder="Stage" value="<?php echo $workbook->stage_id ?>" readonly/>
          <div class="invalid-feedback">
            <?php echo form_error('stage') ?>
          </div>
        </div>

        <div class="form-group">
          <label for="name">Modul</label>
          <input class="form-control <?php echo form_error('modul_id') ? 'is-invalid':'' ?>"
           type="text" name="modul_id" min="0" placeholder="Modul" value="<?php echo $workbook->modul_id ?>" readonly
          <div class="invalid-feedback">
            <?php echo form_error('modul_id') ?>
          </div>
        </div>
```

Gambar 43 Form edit_form.php

Sesuai dengan yang ditunjukkan oleh gambar 43, disini kita membuat sebuah form menggunakan sintaks bawaan yang disediakan oleh *bootstrap* argon *dashboard.* Pada tiap *textbox* pada form, kita memanggil kembali data data yang akan diubah sesuai dengan id yang dipilih, sama seperti yang kita lakukan sebelumnya pada halaman list_workbook.php

```
<div class="form-group">
  <label for="name">Nama Pekerja</label>
  <input class="form-control <?php echo form_error('employee_name') ? 'is-invalid':'' ?>"
  type="text" name="employee_name" min="0" placeholder="Nama Pekerja" value="<?php echo $workbook->yaww ?>"
  <div class="invalid-feedback">
    <?php echo form_error('employee_name') ?>
  </div>
</div>

<div class="form-group">
  <label for="name">Progress*</label>
  <input class="form-control <?php echo form_error('progress') ? 'is-invalid':'' ?>"
  type="text" name="progress" min="0" placeholder="Progress" value="<?php echo $workbook->progress ?>" requi
  <div class="invalid-feedback">
    <?php echo form_error('progress') ?>
  </div>
</div>

<div class="form-group">
  <label for="name">Status*</label>
  <select class="form-control <?php echo form_error('status') ? 'is-invalid':'' ?>" name="status">
    <?php foreach ( $status as $status ) : ?>
    <?php if ( $status == $workbook->status ) : ?>
      <option value="<?= $status; ?>" selected><?= $status; ?></option>
    <?php else : ?>
      <option value="<?= $status; ?>"><?= $status; ?></option>
    <?php endif; ?>
  <?php endforeach; ?>
  </select>
  <div class="invalid-feedback">
    <?php echo form_error('status') ?>
  </div>
</div>
```

Gambar 44 Form edit_form.php

Dapat dilihat pada gambar 43 sampai gambar 44, pada *textbox stage*, modul, nama pekerja, dan juga progres hanya memangil data berdasarkan variabel yang telah kita definisikan sebelumnya pada models. Sedangkan untuk status, pada gambar dicoba untuk *foreach* untuk menampilkan data status akan dipilih dalam bentuk *combo box*. Selanjutnya untuk *textbox* lembar pengesahan sama saja dengan *textbox* lainnya hanya memanggil data berdasarkan variabel yang telah didefinisikan sebelumnya. Lalu ada tombol simpan untuk menyimpan perubahan perubahan yang dilakukan di form ini ke dalam *database*. Seperti yang ditunjukkan oleh gambar 45 berikut ini.

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

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

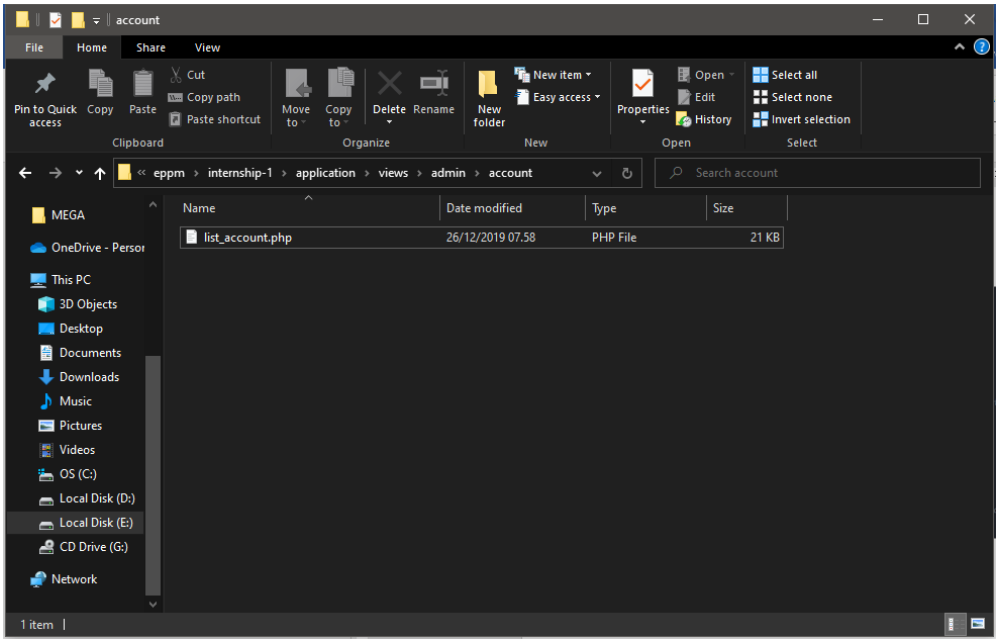
Gambar 45 Form edit_form.php

Dengan ini, selesailah sudah halaman *workbook* untuk admin. Mengapa tidak ada form untuk meng-*input* data *workbook* yang baru? Karena hal itu dilakukan untuk pengguna atau *user*. Jadi admin hanya untuk memantau apakah sudah melaksanakan *workbook* atau belum, nah untuk form tersebut baru akan kita buat pada tahapan *workbook* untuk *user* pada beberapa sub bab berikutnya. Lanjut terus ya bacanya!

2.4. CRUD Menu Daftar Pekerja

Untuk CRUD menu daftar pekerja, ini sebenarnya sama saja seperti CRUD menu *workbook*. Ada beberapa hal yang sama diulangi dan ada beberapa hal yang diganti menyesuaikan dengan kebutuhan pada menu yang akan dibuat. Maka langsung saja kita buat folder *account* pada direktori controller seperti berikut C://xampp/htdocs/eppm-go!/application/controller/admin/account.

Pada folder *account* buatlah sebuah *file* controller dengan nama Account.php lalu isikan di dalam *file* Account.php seperti yang ditunjukkan pada gambar 46 berikut ini.



Gambar 46 Controller Account.php

Seperti pada *file* controller Workbook.php maka kita terlebih dahulu membuat sebuah class dengan nama Account dengan menggunakan *library* CI_Controller. Lalu membuat function __construct() yang akan me-*load* model account_model dan juga me-*load library* form validasi. Karena sudah me-*load* model, maka kita gunakan function is_role() pada account_model yang akan mengecek apabila *role* yang masuk ke dalam aplikasi bukan admin maka akan diarahkan kembali ke halaman *login*.

Lalu setelah itu kita membuat function index() yang medefinisikan variabel $data yang datanya diambil melalui function getAll() pada account_model. Jika sudah maka akan menampilkan halaman list_account dengan membawa data yang sudah didefinisikan pada variabel $data sebelumnya. Seperti yang terlihat pada gambar 47 berikut ini.

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

class Account extends CI_Controller
{
    public function __construct()
    {
        parent::__construct();
        $this->load->model("account_model"); //Load model Account
        $this->load->library('form_validation'); //Load library form validation
        if($this->account_model->is_role() != "admin"){
            redirect("index.php/login");
        }
    }

    public function index()
    {
        $data["account"] = $this->account_model->getAll(); //ambil data dari model
        $this->load->view("admin/account/list_account", $data); //Load view data model ke account
    }
```

Gambar 47 Controller Account.php

Jika sudah maka kita lanjut untuk membuat function add() dengan mendefinisikan variabel $account pada account_model dan juga variabel $validation pada form validasi. Lalu variabel $validation tersebut kita terapkan *rules*. Setelahnya kita tampilkan halaman tambah data atau form tambah data.

```
    public function add()
    {
        $account = $this->account_model; //objek model
        $validation = $this->form_validation; //objek form validation
        $validation->set_rules($account->rules()); //terapkan rules

        if ($validation->run()){
            $account->save();
            $this->session->set_flashdata('Sukses', 'Data Anda Berhasil Disimpan'); //pesan berhasil
        }

        $this->load->view("admin/account/new_account"); //Load isi form account
    }
```

Gambar 48 Controller Account.php

Setelahnya kita tambahkan function edit() yang akan mengambil datanya berdasarkan id yang dipilih, lalu akan ditampilkan kembali halaman form edit dengan data dari id yang sudah dipilih.

```php
public function edit($id=null)
{
    if(!isset($id)) redirect('admin/account/list_account');

    $account = $this->account_model;
    $validation = $this->form_validation;
    $validation->set_rules($account->rules());

    if ($validation->run()){
        $account->update();
        $this->session->set_flashdata('success', 'Data Anda Berhasil Diupdate'); //pesan berhasil
    }

    $data["account"] = $account->getById($id); //mengambil data berdasarkan id
    if (!$data["account"]) show_404();// jika tidak ada show error
    $this->load->view("admin/account/edit_account", $data); //load edit form account
}
```

Gambar 49 Controller Account.php

Jangan lupa untuk membuat function delete() dengan mengambil data berdasarkan id yang kita pilih, sehingga hanya data yang dipilih saja yang terhapus. Jika sudah berhasil terhapus maka akan diarahkan kembali ke halaman daftar pekerja.

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

        if ($this->account_model->delete($id)){
            redirect('index.php/admin/account');
        }
    }
}
```

Gambar 50 Controller Account.php

Dengan ini, kita telah selesai membuat *file* controller Account.php. Maka langkah selanjutnya adalah kita haruss membuat *file* models account_model.php pada direktori models di localhost kita. Setelah selesai membuat *file* tersebut langsung saja kita isikan *file*-nya dengan isian seperti yang ditunjukkan pada gambar 51 berikut.

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

class Account_model extends CI_Model
{
    private $_table = "account";

    public $employee_id;
    public $directorate;
    public $employee_name;
    public $username;
    public $password;
    public $user_email;
    public $role;

    public function rules()
    {
        return [
            ['field' => 'employee_id',
            'label'  => 'No. Pekerja',
            'rules'  => 'required'],

            ['field' => 'employee_name',
            'label'  => 'Nama Pekerja',
            'rules'  => 'required'],

            ['field' => 'username',
             'label' => 'username',
             'rules' => 'required'],

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

Gambar 51 Models Account_model.php

Seperti yang ditunjukkan gambar 51 di atas, terlebih dahulu kita harus mendefinisikan variabel yang akan digunakan untuk menu daftar pengguna. Jika sudah didefinisikan, maka lanjut dengan membuat function rules() dimana ini merupakan aturan aturan yang akan kita gunakan untuk menu ini. Dapat dilihat hanya ada 4 *field* yang diberi aturan tidak boleh kosong alias wajib diisi!.

Jika sudah selesai, maka kita selanjutnya membuat function is_role() yang akan berfungsi untuk mengecek *role* yang masuk ke dalam aplikasi berdasarkan *session* yang aktif. Lalu kita juga membuat function getAll() yang berfungsi untuk mengambil semua data pada tabel account. Nah sama seperti function getAll() kita juga membuat function getById() yang berfungsi untuk mengambil data berdasarkan id-nya. Disini employee_id dijadikan *identifier* dari id yang akan diambil. Untuk lebih lengkapnya silakan lihat gambar 52 di bawah ini.

```php
//fungsi cek level
function is_role()
{
    return $this->session->userdata('role');
}

public function getAll()
{
    return $this->db->get_where($this->_table, array('role' => 'user'))->result();
}

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

Gambar 52 Models Account_model.php

Langkah selanjutnya adalah membuat function save() yang berfungsi untuk menyimpan data yang telah kita *input* kan ke dalam *database* pada tabel account. Begitupun dengan function edit() yang dibuat setelah function save() selesai, function edit() akan menampilkan data yang telah disimpan sebelumnya berdasarkan id yang dipilih. Lalu apabila telah mengubah data yang diperlukan maka akan langsung meng-*update* data tersebut sesuai dengan id yang dipilih.

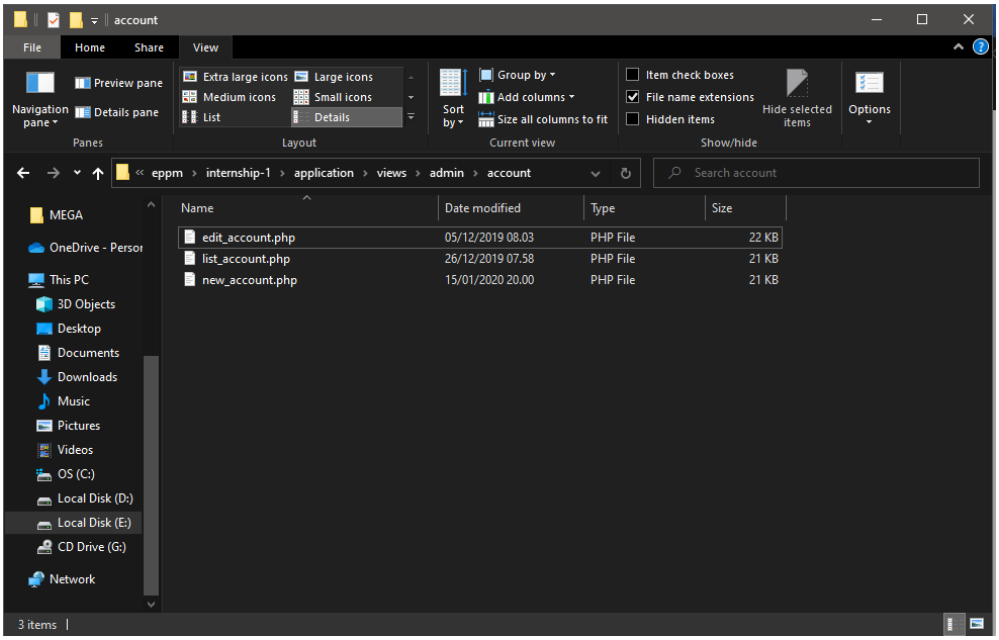
Sama seperti function edit(), function delete() juga mengambil data berdasarkan id yang dipilih akan tetapi peruntukkannya berbeda. Jika edit() untuk mengubah data, maka delete() untuk menghapus data yang kita pilih. Untuk lengkapnya dapat dilihat pada gambar 53 di bawah ini.

```
    public function save()
    {
        $post = $this->input->post();
        $this->employee_id = $post["employee_id"];
        $this->directorate = $post["directorate"];
        $this->employee_name = $post["employee_name"];
        $this->username = $post["username"];
        $this->password = $post["password"];
        $this->user_email = $post["user_email"];
        $this->role = $post["role"];
        $this->db->insert($this->_table, $this);
    }

    public function update()
    {
        $post = $this->input->post();
        $this->employee_id = $post["employee_id"];
        $this->directorate = $post["directorate"];
        $this->employee_name = $post["employee_name"];
        $this->username = $post["username"];
        $this->password = $post["password"];
        $this->user_email = $post["user_email"];
        $this->role = $post["role"];$this->db->update($this->_table, $this, array('employee_id' => $post['employee_id']));
    }

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

Gambar 53 Models Account_model.php

Mantap! Selesai sudah tahapan tahapan yang diperlukan untuk membuat *file* models Account_model.php. Nah bila temen temen yang lain sudah maka bisa langsung dilanjutkan untuk membuat *file* views-nya yang berfungsi untuk menampilkan datanya ke dalam sebuah *user interface* yang lebih nyaman untuk dilihat oleh pengguna. Untuk itu buat folder account pada direktori views dan admin, dan buat kembali sebuah *file* dengan nama list_account.php, edit_account.php dan juga new_account.php.

Sekarang fokusnya akan kita alihkan hanya kepada *file* list_account.php yang berguna untuk menampilkan seluruh data yang ada pada *database* di dalam tabel account.



Gambar 54 File Account

Jangan lupa, untuk setiap menu dan halaman yang akan dibuat kita harus selalu menaruh *link* atau URL yang akan mengarah pada 5 menu utama yang akan dibuat seperti yang ditunjukkan pada gambar 55 di bawah ini.

```
<ul class="navbar-nav">
<li class="nav-item" >
  <a class=" nav-link " href="<?php echo base_url();?>index.php/admin"> <i class="ni ni-tv-2 text-primary"></i> Das
    </a>
  </li>
  <li class="nav-item" class=" active" >
    <a class="nav-link active " href=" <?php echo base_url(); ?>admin/account">
      <i class="ni ni-bullet-list-67 text-red"></i> Daftar Peserta
    </a>
  </li>
  <li class="nav-item">
    <a class="nav-link " href=" <?php echo base_url(); ?>admin/workbook">
      <i class="ni ni-bullet-list-67 text-red"></i> Workbook
    </a>
  </li>
```

Gambar 55 Views Link atau URL