

Bab 4

IMPLEMENTASI DAN UJI COBA

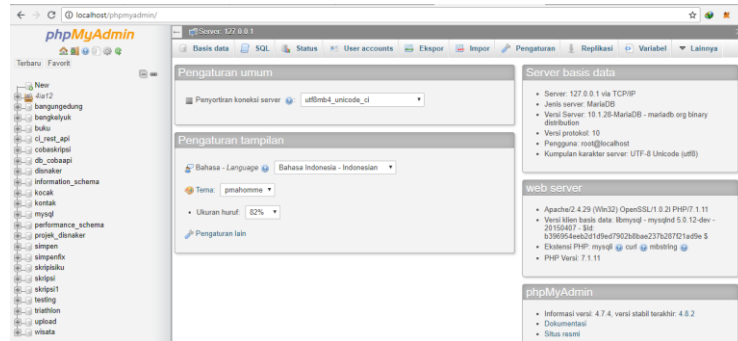
4.1 Implementasi

Implementasi adalah sebuah tahapan pembuatan android dengan menggunakan Bahasa pemrograman setelah melakukan perancangan aplikasi dengan UML dan struktur navigasi. Dalam pembuatan aplikasi ini, menggunakan Bahasa pemrograman Java untuk fungsi utama, XML untuk desain pada aplikasi, Codeigniter sebagai framework untuk website dan SQL sebagai *database* dengan Bahasa PHP. Implementasi android menggunakan Android Studio. Setelah tahapan implementasi, tahap pengujian dilakukan untuk menilai kesesuaian kinerja aplikasi dengan perancangan..

4.2 Pembuatan Basis Data (*Database*)

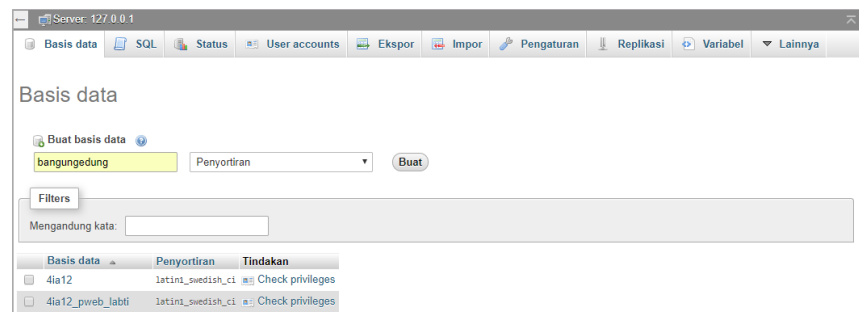
1. Basis data akan menyimpan semua data yang berhubungan dengan identitas bangun gedung dan yang akan ditampilkan di aplikasi survei bangun gedung. Baik berupa teks ataupun posisi pada *maps*. Berikut merupakan langkah-langkah dalam pembuatan *database* aplikasi :

- a. pada aplikasi ini menggunakan localhost pada komputer sebagai server aplikasi.
- b. akses alamat localhost dan masuk kedalam pembuatan basis data seperti gambar 4.1.



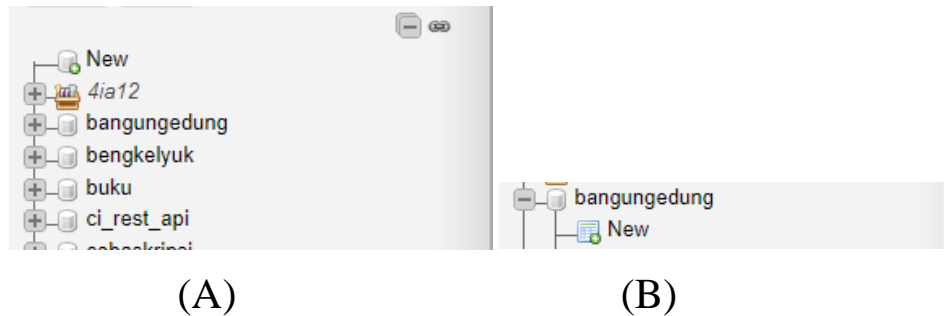
Gambar 4.1 Tampilan Home Basis data Localhost phpMyAdmin

c. Langkah selanjutnya memilih *new* yang terdapat di menu sebelah kiri untuk pembuatan basis data baru. Pembuatan basis data baru dapat dilihat pada gambar 4.2.



Gambar 4.2 Tampilan Pembuatan Basis Data Baru

2. Setelah pembuatan basis data, buat table baru ke dalam basis data yang telah dibuat. Berikut langkah-langkah dalam pembuatan tabel baru :
 - a. masuk ke halaman utama Localhost dan memilih *menu* phpMyAdmin pada gambar 4.1.
 - b. Pada halaman phpMyAdmin berisi daftar *database* yang telah tersimpan di dalam sistem. Halaman phpMyAdmin dapat dilihat pada gambar 4.3 (A), kemudian pilih *new* pada gambar 4.4 (B) untuk membuat tabel baru.



Gambar 4.3 (A) Daftar Basis Data, (B) Menu Menambahkan Tabel

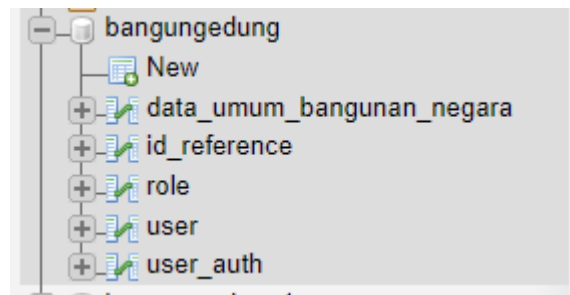
c. Kemudian membuat tabel baru dan mendefinisikan isi tabel. *Nama* Tabel merupakan *field* yang berisi nama tabel yang akan ditambahkan dan Tambah akan berisi jumlah kolom yang terdapat di dalam tabel. Tampilan pendefinisian struktur tabel dapat dilihat pada gambar 4.4.

Gambar 4.4 Pembuatan Tabel Baru

d. Setelah itu mengulang langkah 2a sampai dengan 2c untuk pembuatan tabel lainnya jika dibutuhkan. Pembuatan serta pendefinisian isi tabel sesuai dengan rancangan tabel pada sub bab 3.3.4.

3. Setelah membuat tabel baru, selanjutnya mengisi semua *record* sesuai dengan struktur tabel yang telah didefinisikan. Berikut ini langkah- langkah yang dilakukan :

- a. Kemudian memilih tabel yang telah dibuat, daftar tabel yang ada di dalam basis data yang telah dibuat yaitu tabel user, user_auth, data_bangun_gedung_negara, id_reference, dan role dapat dilihat pada gambar 4.5



Gambar 4.5 Tampilan Daftar Tabel

- b. Setelah itu seperti pada tampilan gambar 4.6 dan memilih *tab* Tambahkan. Pada halaman tersebut akan berisi semua *field* sesuai dengan tabel yang telah dibuat pada langkah 2. *Value* yang kosong akan diisi oleh semua *record* terkait. Tombol kirim dipilih untuk menyimpan perubahan yang telah dilakukan.

Kolom	Jenis	Fungsi	Kosong	Nilai
id_user	int(11)		<input type="text"/>	<input type="text"/>
user_id	int(11)		<input type="text"/>	<input type="text"/>
nama	int(11)		<input type="text"/>	<input type="text"/>
password	int(11)		<input type="text"/>	<input type="text"/>
user_role	int(11)		<input type="text"/>	<input type="text"/>
terakhir_login	int(11)		<input type="text"/>	<input type="text"/>
input_at	int(11)		<input type="text"/>	<input type="text"/>
update_at	int(11)		<input type="text"/>	<input type="text"/>

Gambar 4.6 Tampilan Memasukan *Record*

4.3 Pembuatan Aplikasi

Setelah rancangan tampilan dibuat, struktur navigasi, diagram uml dan *pseudocode* program tahap ini merupakan tahap pembuatan aplikasi berdasarkan perancangan yang telah dibuat.

4.3.1 Pembuatan Web Admin

1. Langkah pertama pada pembuatan admin dalam web ini adalah menggunakan bahasa pemrograman web, dengan menggunakan html dan codeigniter sebagai framework dan menggunakan konsep MVC(Model View Controller).

Berikut listing program untuk Controller pada website :

Admin.php

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
class Admin extends CI_Controller {
    public function __construct(){
        parent::__construct();
        $this->load->library('session');
        $this->load->model('Model_Web');
    }
    public function index()
    {
    }
    public function dashboard(){
        $this->load->library('session');
        if($this->session->userdata('logged_in')==true){
            $this->load->view('Templates/HeaderDash');
            $this->load->view('Main/Dashboard');
            $this->load->view('Templates/FooterDash');
        }else{
            redirect('LoginWeb');
        }
    }
}
```

Pada kode fungsi diatas yaitu *function dashboard* digunakan untuk memunculkan halaman *dashboard* dengan kondisi apabila *array* userdata bernilai *true* maka akan memunculkan *file* Headerdash, Dashboard, dan footerdash yang terdapat pada *folder view* di dalam codeigniter. Apabila *userdata* bernilai *false* maka *controller* akan mengarahkan kembali ke tampilan LoginWeb.

```
public function halaman_register(){
    $this->load->library('session');
    if($this->session->userdata('logged_in')==true){
```

```

        $data['option']='add';
        $this->load->view('Templates/HeaderDash');
        $this->load->view('Main/Register',$data);
        $this->load->view('Templates/FooterDash');
    }else{
        redirect('LoginWeb');
    }
}

```

Berdasarkan fungsi kode diatas yaitu *function* halaman_register digunakan digunakan untuk membuka halaman pendaftaran akun baru. Salah satu baris kode diatas `$data['option']='add'` berfungsi sebagai parameter yang menandakan bahwa halaman ini digunakan untuk menambah akun baru.

```

public function register(){
    $id=$this->input->post('id');
    $role=$this->input->post('role');
    $nama_depan=$this->input->post('nama_depan');
    $nama_belakang=$this->input->
    >post('nama_belakang');
    $user_id=$this->input->post('user_id');
    $name= "{ $nama_depan } { $nama_belakang }";
    $password_input=$this->input->post('password');
    $password=crypt($password_input,"");
    $data=array(
        'user_id'=>$user_id,
        'nama'=>$name,
        'password'=>$password,
        'user_role'=>$role,
        'input_at'=>date('Y-m-d H:i:s'),
    );
    $this->Model_Web->register_user($data);
    redirect('Admin/tabel_user');
}

```

Berdasarkan fungsi kode *register*, digunakan sebagai fungsi untuk memasukan data dengan method *POST* pada *view* halaman_register. Data yang dimasukan pada halaman tersebut diantaranya *id*, *role*, nama depan, nama belakang dan *password*. Fungsi pada baris kode diatas digunakan untuk

enskripsi kata atau kalimat yang telah di masukan. Fungsi *crypt* tersebut telah disediakan oleh codeigniter .

```
public function edit(){
    $id=$this->input->post('id');
    $role=$this->input->post('role');
    $user_id=$this->input->post('user_id');
    $name= $this->input->post('nama');
    $password_input=$this->input->post('password');
    $password=crypt($password_input,"");
    $data=array(
        'user_id'=>$user_id,
        'nama'=>$name,
        'password'=>$password,
        'user_role'=>$role,
        'update_at'=>date('Y-m-d H:i:s')
    );
    $this->Model_Web->update($id,$data);
    redirect('Admin/tabel_user');
}
```

Berikut kode diatas adalah untuk mengedit akun yang telah dibuat sebelumnya pada fungsi *register* dan pada halaman *_register*. Baris kode tersebut memiliki fungsi yang sama yaitu dengan menggunakan method *POST* untuk memasukan data pada basis data. Pada baris kode *\$this->Model_Web->update(\$id,\$data);* digunakan untuk memanggil model untuk *update* data akun yang telah terdaftar.

```
public function edit_user($id){
    if ($this->session->userdata('logged_in')==true){
        $data['option']='edit';
        $data['sql'] = $this->Model_Web->edit($id);
        $this->load->view('Templates/HeaderDash');
        $this->load->view('Main/Edit',$data);
        $this->load->view('Templates/FooterDash');
    }else {
        redirect('LoginWeb');
    }
}
```

Berdasarkan fungsi kode diatas yaitu *function edit* digunakan digunakan untuk membuka halaman edit akun. Salah satu baris kode diatas

`$data['option']='edit'` berfungsi sebagai parameter yang menandakan bahwa halaman ini digunakan untuk mengubah akun yang telah terdaftar.

```
public function delete_user($id){
    $data['sql'] = $this->Model_Web->delete($id);
    redirect('Admin/Tabel_User');
}
```

Pada kode fungsi *delete_user* digunakan untuk menghapus data akun yang telah terdaftar pada database, pada baris kode `Model_Web->delete($id);` digunakan untuk memanggil *model* yang terdapat pada *Model_Web* dan menggunakan method *DELETE* untuk menghapus data akun tersebut.

```
public function tabel_user(){
    $this->load->library('session'){
    if ($this->session->userdata('logged_in')==true){
    $data['sql'] = $this->Model_Web->get_user();
    $this->load->view('Templates/HeaderDash');
    $this->load->view('Main/Table_User',$data);
    $this->load->view('Templates/FooterDash');
    }else{
    redirect('LoginWeb');
    }
    }
}
```

Dari kode diatas yaitu fungsi *tabel_user* digunakan untuk menampilkan tabel data user yang telah dibuat pada halaman_register pada *website*. Pada kode baris kode `$data['sql'] = $this->Model_Web->get_user();` tersebut berfungsi untuk memanggil model *get_user* yang terdapat pada *Model_Web* dan menggunakan method *GET* untuk mengambil data akun yang terdapat pada basis data.

```
public function tabel_gedung(){
    $this->load->library('session');
    if ($this->session->userdata('logged_in')==true){
    $data['sql'] = $this->Model_Web->get_gedung();
    $this->load->view('Templates/HeaderDash');
    $this->load->view('Main/Table_Gedung',$data);
    $this->load->view('Templates/FooterDash');
```



```

    }else{
        redirect('LoginWeb');
    }
}
}

```

Dari kode diatas yaitu fungsi tabel_gedung digunakan untuk menampilkan tabel data user yang telah dibuat pada halaman_register pada *website*. Pada kode baris kode `$data['sql'] = $this->Model_Web->get_gedung();` tersebut berfungsi untuk memanggil model `get_gedung` yang terdapat pada `Model_Web` dan menggunakan method *GET* untuk mengambil gedung yang terdapat pada basis data.

LoginWeb.php

```

<?php
defined('BASEPATH') OR exit('No direct script access
allowed');
class LoginWeb extends CI_Controller {
    public function __construct(){
        parent::__construct();
        $this->load->model('Model_Web');
    }
}

```

Pada kode diatas berfungsi untuk memanggil model pada `Model_web` sebagai konsturktor yang berarti akan terus dijalan setiap program atau web dijalankan.

```

public function index()
{
    if ($this->session->userdata('logged_in'))
    {
        redirect('Admin/dashboard','refresh');
    }else{
        $this->load->helper(array('form'));
        $this->load->view("Templates/Header");
        $this->load->view('Main/Login');
        $this->load->view("Templates/Footer");
    }
}

```

Berdasarkan kode *function* index diatas, kode tersebut digunakan untuk pemanggilan halaman utama pada saat website dijalankan yaitu memanggil Main/Login sebagai index dari website.

```
public function login_check(){
    $user_id = $this->input->post('user_id',true);
    $password = $this->input->post('password',true);
    $this->Model_Web-
        >login_check($user_id,$password);
}
```

Pada kode diatas fungsi dari login_check digunakan untuk apakah *id* dan *password* yang dimasukan kedalam *form login* benar atau salah.

```
public function logout(){
    $this->session->unset_userdata('logged_in');
    redirect('LoginWeb','refresh');
}
```

Berdasarkan kode diatas fungsi *logout* digunakan untuk keluar dari sebuah akun pada website.

Controller digunakan sebagai penghubung antara *view* dengan model pada *website*, setelah *controller* semua sudah dibuat, berikut model untuk menampilkan basis data yang sudah dibuat dan yang ditampilkan oleh *view* pada *website*. Model ini menggunakan bahasa yang digunakan pada basis data yang digunakan

Berikut listing program untuk model untuk website :

Model_Web.php

Model_Web.php ini model yang hanya digunakan untuk akses pada *website*.

Berikut listing program pada Model_Web.php

```
<?php
defined('BASEPATH') OR exit('No direct script access
allowed');
class Model_Web extends CI_Model{
function login_check($user_id,$password)
{
```

```

        $q=$this->db->select('password,id_user')->from('User')-
        >where('user_id',$user_id)->get()->row();
        $hashed_password = $q->password;
        if(hash_equals($hashed_password,crypt($password,$hashed
        _password)) && $user_id=='50414797') {
        print_r(hash_equals($hashed_password,crypt($password,$ha
        shed_password)));
        $sess_array = array();
        $sess_array = array('username' => $q->user_id, 'logged_in'
        => true);
        $this->session->set_userdata($sess_array);
        $status = array('status' => true);
        }
        redirect('Admin/dashboard');
    }

```

pada kode model login_check digunakan untuk mengecek id dan *password* yang telah di enkripsi apakah benar atau salah dengan data pada basis data. Apabila *login* berhasil dan berhasil maka akan diarahkan kedalam halaman admin/dashboard.

```

    public function register_user($data){
        return $this->db->insert('user',$data);
    }

```

Pada kode model diatas yaitu *function* register_user digunakan untuk memasukan data akun kedalam basis data.

```

    public function edit($id){
        $this->db->where('id_user',$id);
        return $this->db->get('user');
    }

```

Berdasarkan kode model diatas yaitu *function* edit digunakan untuk mengambil data akun yang terdapat pada basisdata.

```

    public function update($id,$data){
        $this->db->where('id_user',$id);
        $this->db->update('user', $data);
        return $this->db->affected_rows();
    }

```

Kode function *update* diatas digunakan untuk merubah data akun yang telah ada pada basis data.

```
public function delete($id){
    $this->db->where('id_user',$id);
    return $this->db->delete('user');
}
```

Pada kode function *delete* diatas digunakan untuk menghapus akun user yang telah terdaftar pada basis data dengan menggunakan id sebagai parameternya.

```
public function get_user(){
    $sql = $this->db->query("select * from user");
    return $sql;
}
```

Berdasarkan kode dari *function* *get_user* diatas digunakan untuk mengambil seluruh data akun yang ada pada basis data.

```
public function get_gedung(){
    $sql=$this->db->query("select*from
    data_umum_bangunan_negara");
    return $sql;
}
```

Berdasarkan kode dari *function* *get_gedung* diatas digunakan untuk mengambil seluruh data gedung yang ada pada basis data.

MyModel.php

MyModel.php ini model yang digunakan untuk android dalam akses basis data yang telah dibuat pada *website*. Berikut listing kodenya :

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
class MyModel extends CI_Model {
    var $client_service = "frontend-client";
    var $auth_key      = "simplerestapi";
    public function check_auth_client(){
        $client_service = $this->input->get_request_header('Client-Service',
        TRUE);
        $auth_key = $this->input->get_request_header('Auth-Key', TRUE);
```

```

        if($client_service == $this->client_service && $auth_key == $this->auth_key){
            return true;
        } else {
            return json_output(401,array('status' => 401,'message' => 'Unauthorized.));
        }
    }
}

```

Kode diatas adalah memberikan *header* pada saat android ingin mengakses basis data survei bangun gedung. *Custom header* tersebut digunakan untuk memberikan hak akses secara unik kepada setiap user. Kode diatas adalah pengecekan apakah *header* client_service dan auth_key terdapat pada android pada saat mengakses data. Jika terdapat pada android maka client_service dan auth_key bernilai true. Jika tidak maka akan memberikan informasi bahwa akun tersebut *Unauthorized*.

```

public function login($username,$password)
{
    $q = $this->db->select('password,id_user,user_id,nama,user_role')->from('User')->where('user_id',$username)->get()->row();
    if($q == ""){
        return array('status' => 204,'message' => 'Username not found.');
```

```

    } else {
        $nama = $q->nama;
        $role = $q->user_role;
        $hashed_password = $q->password;
        $id = $q->id_user;
        $user_id = $q->user_id;
    }
}

```

Pada kode diatas digunakan untuk *login* pada android dengan menggunakan parameter username dan password. Kode diatas membandingkan antara nama, *role*, password yang telah di hash, id dan user id pada tabel user. Dan apabila pada saat *login* salah satunya kosong maka akan memberikan pesan username tidak ditemukan.

```

        if (hash_equals($hashed_password, crypt($password, $hashed_password))) {
            $last_login = date('Y-m-d H:i:s');
            $token = crypt(substr( md5(rand()), 0, 7),");

```

```

        $expired_at = date("Y-m-d H:i:s", strtotime('+12 hours'));
        $this->db->trans_start();
        $this->db->where('id_user',$id)-
>update('User',array('terakhir_login' => $last_login));
        $this->db->insert('user_auth',array('id_user' => $id,'token' =>

```

Lanjutan dari kode *function login* apabila kondisi *user* sudah terdapat *login* pada aplikasi, maka pada saat *splashscreen* akan di cek baris data user tersebut. mulai dari *last_login*, *token*, *Expired_at*, digunakan untuk memberikan *session* selama 12 jam kepada setiap *user* android yang telah *login* pada aplikasi.

```

$token,'expired_at' => $expired_at));
    if ($this->db->trans_status() === FALSE){
        $this->db->trans_rollback();
        return array('status' => 500,'message' => 'Internal server error.');
```

```

    } else {
        $this->db->trans_commit();
        return array('status' => 200,'message' => 'Successfully login.','id'
=> $id, 'token' => $token, 'nama'=> $nama, 'role'=> $role, 'user_id'=>
$user_id);
    }
} else {
    echo "Wrong password";
    exit();
    return array('status' => 204,'message' => 'Wrong password.');
```

```

}

```

Berdasarkan kode diatas apabila token sudah expired maka akan kembali rollback ke model *login*, dan apabila token masih kurang dari 12 jam dari terakhir login maka akan memberikan pesan *successfully login* dan memberikan data id, token, nama, role dan user_id pada saat login di android.

```

public function logout()
{
    $users_id = $this->input->get_request_header('User-ID', TRUE);
    $token = $this->input->get_request_header('Authorization', TRUE);
    $this->db->where('ID_USER',$users_id)->where('token',$token)-
>delete('user_auth');
```

```

    return array('status' => 200,'message' => 'Successfully logout.');
```

```

}

```

Pada kode diatas model *function logout* digunakan untuk *logout* pada android, pada kode diatas *user* akan meminta custom header yang telah dibuat, User-ID

yaitu *id* pengguna serta meminta header *Authorization* sebagai token pada tabel *user*. Apabila keduanya bernilai benar maka akan menghapus 1 baris pada tabel *user_auth*. Setelah terhapus akan memberikan pesan berupa array berisi “*Succesfully logout*”.

```
public function input_data_bangunan($data){
    $this->db->insert('data_umum_bangunan_negara',$data);
    return array('status' => 200,'message' => 'Data has been created.');
```

Kode diatas adalah model data bangunan yang digunakan untuk memasukan hasil survei kedalam data dengan menggunakan *method insert*, dan apabila data telah berhasil masuk kedalam basis maka akan memberikan sebuah pesan “*data has been created*”

```
public function hapus_data_umum_bangunan_negara($id_bangunan)
{
    $this->db->where('id_bangunan',$id_bangunan)-
    >delete('data_umum_bangunan_negara');
    return array('status' => 200,'message' => 'Data has been deleted.');
```

Fungsi kode diatas digunakan untuk menghapus data pada basis data, dengan menggunakan *method delete*, dan apabila data telah berhasil terhapus di android maka akan memberikan pesan “*data has been deleted*” pada *website*.

```
public function list_bangunan_user(){
    $this->db->select('u.nama,p.*');
    $this->db->from('data_umum_bangunan_negara as p');
    $this->db->join('user as u','u.user_id = p.user_id','inner');
    $this->db->order_by('nama_bangunan');
    return $this->db->get()->result();
}
```

Berdasarkan kode diatas adalah berfungsi sebagai model untuk menampilkan seluruh data yang terdapat pada *data_bangunan_umum_negara* dengan menggunakan *method get*. Model tersebut menggunakan *inner join* untuk menyatukan nama yang terdapat pada tabel *user*. Nama tersebut diambil agar

dapat ditampilkan nama petugas survei yang memasukan data bangunan gedung dengan menggunakan aplikasi survei bangun gedung.

```
public function list_bangunan_petugas_survei($user_id){
    $this->db->select('u.nama,p.*');
    $this->db->from('data_umum_bangunan_negara as p');
    $this->db->join('user as u','u.user_id = p.user_id','inner');
    $this->db->where('p.user_id', $user_id);
    $this->db->order_by('nama_bangunan');
    return $this->db->get()->result();
}
```

Berikutnya adalah kode dari *function* list_bangunan_survey, fungsi tersebut berfungsi untuk menampilkan hanya yang dimasukan oleh petugas survei yang bersangkutan dengan menggunakan *method* *get*., dengan menggunakan parameter *user_id* yang tentunya unik sehingga dapat dipisahkan dari seluruh data bangunan. Model tersebut menggunakan *inner join* dengan nama yang terdapat pada tabel *user*.

```
public function cari_bangunan($kata){
    $this->db->like('nama_bangunan',$kata);
    $query = $this->db->get('data_umum_bangunan_negara');
    return $query->result();}
```

Yang terakhir pada kode MyModel.php adalah fungsi cari_bangunan.. fungsi tersebut digunakan untuk mencari dan mendapatkan data menggunakan *methode* *get*, data bangun gedung pada android yang berupa *list* daftar dari nama departemen dan nama bangunan. Pencarian tersebut dapat dilakukan dengan mencari nama bangunan, fungsi bangunan dan nama departemen.

2. Langkah kedua pada pembuatan admin dalam web ini adalah membuat *REST API* yang berfungsi sebagai penghubung atau jembatan antara aplikasi android dan *website*, sehingga *website*, aplikasi android, basisdata saling terhubung atau saling terintegrasi. Pembuatan *REST API* ini masih menggunakan Codeigniter sebagai frameworknya.

Berikut listing program untuk REST API :

DataBangunan.php

```
<?php
defined('BASEPATH') OR exit('No direct script access
allowed');
class DataBangunan extends CI_Controller {
    public function __construct()
    {
        parent::__construct();
        $this->output->enable_profiler(TRUE);
    }
    public function index()
    {
        $method = $_SERVER['REQUEST_METHOD'];
        if($method != 'GET'){
            json_output(400,array('status'=> 400,'message'=> 'Bad
            request.));
        } else {
            $check_auth_client = $this->MyModel->check_auth_client();
            if($check_auth_client == true){
                $response = $this->MyModel->auth();
                if($response['status'] == 200){
                    $resp = $this->MyModel->semua_data_bangunan();
                    json_output($response['status'],$resp);
                }
            }
        }
    }
}
```

Berdasarkan kode diatas yaitu *function* databangunan digunakan sebagai pemanggilan data dari basis data kedalam android. Pada baris kode `if($response['status'] == 200){ $resp = $this->MyModel->semua_data_bangunan(); json_output($response['status'],$resp);`

digunakan sebagai pemanggilan data dengan method *GET* dari basis data. Sebelum mengambil data pada basis data website akan memeriksa terlebih dahulu kedalam `MyModel->check_auth_client();` dan apabila pemeriksaan tersebut benar maka android dapat mengambil data dari basis data berbentuk json.

```
public function masukandata(){
    $method=$_SERVER['REQUEST_METHOD'];
    if($method != 'POST'){
```

```

        json_output(400,array('status' => 400,'message' => 'Bad
        request.'));
    } else {
        $check_auth_client = $this->MyModel->check_auth_client();
        if($check_auth_client == true){
            $response = $this->MyModel->auth();
            $respStatus = $response['status'];
            if($response['status'] == 200){
                $params=json_decode(file_get_contents('php://input'),    TRUE);
                print_r($params);
                $resp=$this->MyModel-
                >input_data_umum_bangunan_negara($params);
                json_output($respStatus,$resp);
            }
        }
    }

```

Berdasarkan kode diatas *function* masukan digunakan untuk memasukan data pada android. Pada baris ini `if($response['status'] == 200)` `{ $params=json_decode(file_get_contents('php://input'), TRUE);` berfungsi sebagai kode input data berupa json yang *decode*. data di *decode* dikarenakan pada saat input atau memasukan data pada android telah di *encode* terlebih dahulu.

```

    public function hapusdata($id_bangunan)
    {
        $method = $_SERVER['REQUEST_METHOD'];
        if($method != 'DELETE' || $this->uri->segment(3) == "
        || is_numeric($this->uri->segment(3)) == FALSE){
            json_output(400,array('status' => 400,'message' => 'Bad
            request.'));
        } else {
            $check_auth_client      =      $this->MyModel-
            >check_auth_client();
            if($check_auth_client == true){
                $response = $this->MyModel->auth();
                if($response['status'] == 200){
                    $resp=$this->MyModel-
                    >hapus_data_umum_bangunan_negara($id_bangunan)
                    ;
                    json_output($response['status'],$resp);
                }
            }
        }
    }

```

Berdasarkan kode diatas digunakan untuk menghapus data pada basis data dengan menggunakan android. Sebelum menghapus data pada basis data *website* akan memeriksa terlebih dahulu kedalam MyModel->check_auth_client(); dan apabila pemeriksaan tersebut benar maka android dapat dapat menghapus data.

```

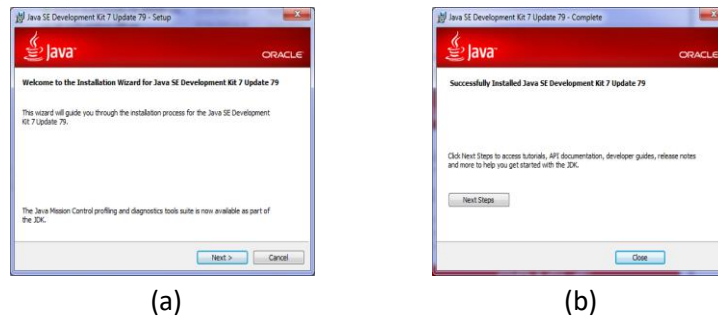
public function updatedata($id_bangunan)
{
    $method = $_SERVER['REQUEST_METHOD'];
    if($method != 'PUT' || $this->uri->segment(3) == " ||
    is_numeric($this->uri->segment(3)) == FALSE){
        json_output(400,array('status'=> 400,'message'=> 'Bad
        request.'));
    } else {
        $check_auth_client      =      $this->MyModel-
        >check_auth_client();
        if($check_auth_client == true){
            $response = $this->MyModel->auth();
            $respStatus = $response['status'];
            if($response['status'] == 200){
                $params= json_decode(file_get_contents('php://input'),
                TRUE);
                $params['date_input'] = date('Y-m-d');
                if($params['fungsi_bangunan']==""||$params['jenis_b
                angunan'] == "") {
                    $respStatus = 400;
                    $resp = array('status' => 400,'message' => `Data tidak
                    boleh kosong`
                } else {
                    $resp=$this->MyModel-
                    >update_data_umum_bangunan_negara($id_banguna
                    n,$params);
                }
                json_output($respStatus,$resp);
            }
        }
    }
}

```

Pada kode diatas digunakan untuk memperbarui data bangunan yang telah dimasukan pada basis data. Pada kode diatas menggunakan method *PUT* untuk mengambil data yang ada pada basis data dan memasukan kembali data yang telah diperbaharui.

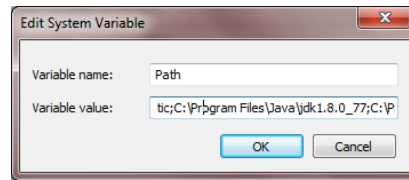
4.3.2 Pembuatan Aplikasi Android

1. Pada langkah pertama yaitu melakukan instalasi *software* JDK (*Java Development Kit*). JDK yang digunakan yaitu versi jdk1.8.0_77. Berikut adalah langkah-langkah instalasi *software* JDK:
 - a. Mengunduh *software* JDK di situs resmi oracle [8]
 - b. Pada langkah-langkah instalasi JDK, hanya menekan tombol *next*, seperti gambar 4.7.
 - c. Jika proses instalasi telah selesai dan berhasil, maka akan terdapat folder JDK dan JRE pada C:\Program Files\Java.



Gambar 4.7 (A) Instalasi Awal JDK, (b) Proses Instalasi JDK Selesai

- d. Selanjutnya melakukan pengaturan *PATH* Java, dengan cara memilih *Control Panel* kemudian memilih *System*, lalu melanjutkan dengan memilih *Advanced* dan mengklik *Environment Variables*, kemudian mengisi variabel *name* : *PATH* dan *variable value* : C:\Program Files\Java\jdk1.8.0_77\bin; (menunjuk pada folder \bin JDK yang diinstall). Gambar 4.8 adalah proses yang dilakukan dalam pengaturan *PATH* Java.



Gambar 4.8 Tampilan *System Variable*

2. Setelah instalasi JDK selesai dilakukan pada langkah pertama, selanjutnya menginstalasi Android Studio. Berikut adalah langkah-langkah instalasi Android Studio:
 - a. Mengunduh *software* Android Studio di situs resmi Android Studio[2]
 - b. Setelah selesai mengunduh, mencari *file* Android Studio instalasi *executable* (bernama **android-studio-bundle-<version>.exe**) di jendela *Windows Explorer* dan klik dua kali untuk memulai proses instalasi, setelah itu langkah-langkah instalasi Android Studio berikutnya hanya menekan tombol *next* seperti pada gambar 4.9.



(A)



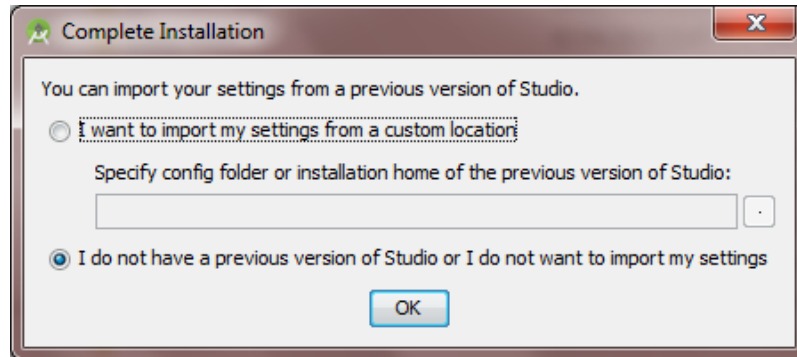
(B)

Gambar 4.9 (A) Proses Instalikasi Awal Android Studio,(B)

Instalasi Android Studio selesai

- c. Setelah langkah instalasi selesai seperti pada gambar 4.9 (B), kemudian akan muncul jendela yang menyediakan opsi untuk mengimpor pengaturan dari versi Android Studio sebelumnya seperti pada gambar 4.9. Jika

memiliki pengaturan dari versi sebelumnya dan ingin mengimpor mereka ke dalam instalasi terbaru, maka opsi yang sesuai dengan lokasi dipilih. Karena baru pertama kali menginstall Android Studio maka pernyataan *“I do not have a previous version of Android Studio or I do not want to import my settings”* dipilih dan tekan tombol *OK* untuk melanjutkan. Setelah memilih pernyataan tersebut, kemudian jendela untuk memulai Android Studio akan tampil seperti pada gambar 4.10.



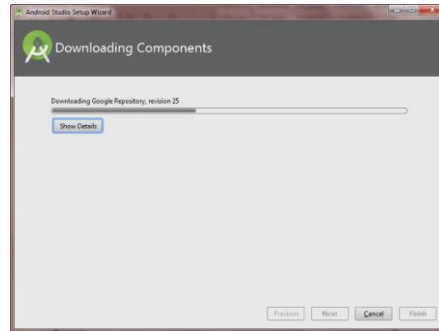
Gambar 4.10 Jendela Kelengkapan Instalasi.



Gambar 4.11 Jendela Memulai Android Studio

- d. Setelah Android Studio berhasil dimuat, selanjutnya akan tampil jendela *Android Studio Setup Wizard* seperti pada gambar 4.12. Pada jendela ini

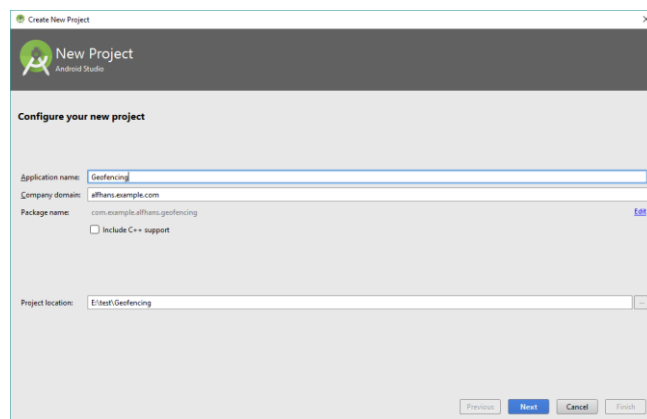
mengunduh dan menginstalasi komponen Android SDK Tools. Sebelumnya memastikan komputer terhubung dengan internet



Gambar 4.12 Proses Unduh dan Instalasi Android SDK tools

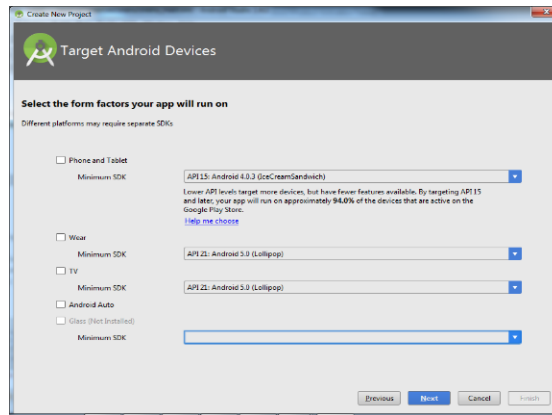
3. Setelah melakukan instalasi Android Studio pada langkah ke tiga, selanjutnya melakukan instalasi ADB Drivers pada komputer/PC, agar nantinya aplikasi dapat di uji coba dengan menggunakan *smartphone*. Berikut adalah langkah-langkah dalam melakukan instalasi ADB Drivers:
 - a. Hubungkan *Smartphone* Android dengan komputer menggunakan kabel USB.
 - b. Pada komputer yang terdapat di *desktop* atau *Windows Explorer*, tekan tombol klik pada bagian kanan *mouse* kemudian pilih *Manage*.
 - c. Selanjutnya memilih *Devices* yang terdapat di panel kiri.
 - d. Lalu mencari dan memperluas perangkat lain di panel kanan.
 - e. *Update Driver Software* dipilih setelah menekan tombol klik dibagian kanan *mouse* pada nama perangkat. Langkah ini akan meluncurkan *Hardware Update Wizard*.
 - f. Memilih *Browse my computer for driver software* kemudian menekan tombol *next*.
 - g. Kemudian mencari folder driver USB. (The Google USB Driver terletak di <sdk>\extras\google\usb_driver\.) dengan menekan *Browse*.
 - h. Tekan tombol *Next* untuk menginstalasi *driver*.

4. Setelah selesai menginstall android studio maka tahap selanjutnya adalah pembuatan aplikasi dengan memilih *Start an new Android Studio Project*.
 - a. Langkah pertama adalah mengisi nama proyek “*Survei Bangun Gedung*”, *Company Domain* (alfhan.example.com), kemudian menentukan direktori kerja atau *workspace* yang digunakan untuk menyimpan proyek Android. Lokasi direktori kerja yang ditetapkan yaitu “D:\Survei Bangun Gedung”. Lalu tekan tombol *Next* seperti pada gambar 4.13.



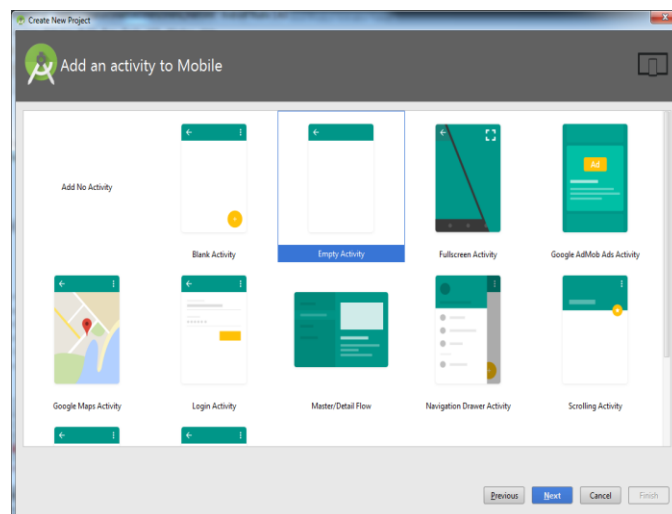
Gambar 4.13 Konfigurasi Proyek Baru

- b. Setelah membuat proyek pada gambar 4.13, lanjutkan dengan memilih target *run* aplikasi yang ada pada gambar 4.14, dalam hal ini memilih *Phone and Tablet*. Dibagian minimum SDK, pilih API 15: Android 4.0.3 (IceCreamSandwich). Kemudian tekan tombol *Next*.



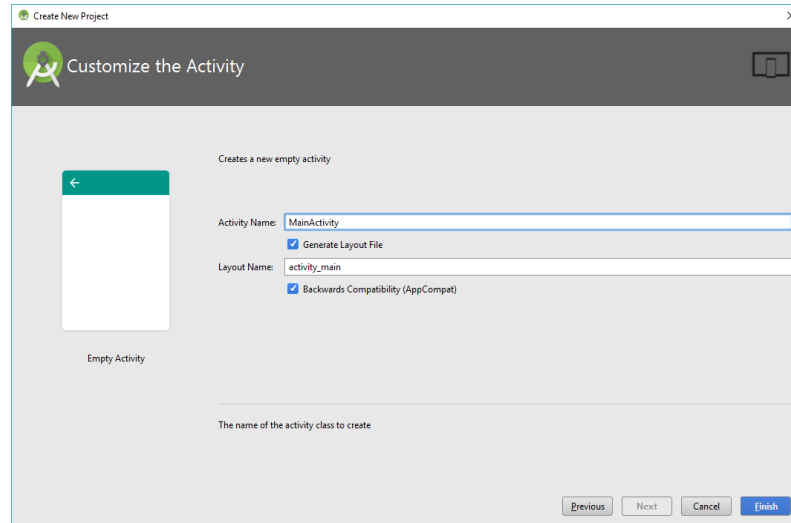
Gambar 4.14 Menentukan Target Run Aplikasi

- c. Dari gambar 4.14 kemudian akan muncul tampilan untuk pilihan *Layout* (*Activity*) aplikasi. Disini pengguna Android Studio bebas memilih *activity* yang disediakan tanpa harus membuat *layout* nya lagi dari awal. Untuk menyesuaikan dengan desain yang telah dibuat, maka memilih *Empty Activity* seperti pada gambar 4.15. Setelah itu tekan tombol *Next*.



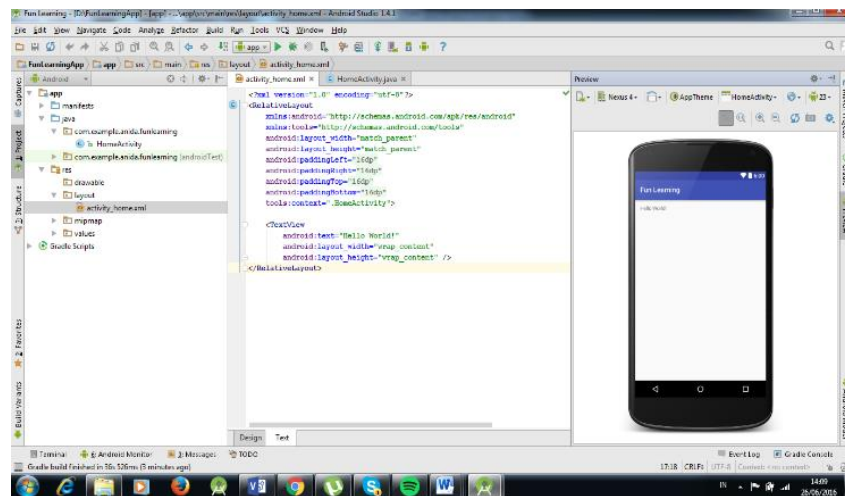
Gambar 4.15 Memilih Tampilan *Layout Activity*

- d. Pada tahap ini, tentukan nama dari *activity* yang telah dipilih. Beri nama *MainActivity* kemudian tekan tombol *Finish* seperti pada gambar 4.16.



Gambar 4.16 Mengatur Nama Activity

- e. Setelah menekan tombol *finish*, maka akan tampil lembar proyek Android Studio seperti pada gambar 4.17.



Gambar 4.17 Tampilan Lembar Kerja

4.3.2.1 Penambahan *Library* Pada Gradle

Gradle adalah suatu fitur build automation yang open source. *Gradle* didesain untuk membuat multi-project yang sedang berkembang menjadi project berskala besar. File Gradle terdiri dari file `settings.gradle` dan `build.gradle`. `build.gradle` berisi pengaturan global mengenai aplikasi pada Android Studio. Pada aplikasi survei bangun gedung ini dibutuhkan *library* tambahan pada *gradle* android seperti berikut :

```
android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "com.example.alfhanrf.skripsihehe"
        minSdkVersion 20
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        buildConfigField("String", "URI", "\"http://192.168.43.49/Surveygedung^\"")
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

Kode diatas adalah *gradle* yang digunakan dalam aplikasi, aplikasi survei bangun gedung menggunakan *compileSdkVersion 27* dan *BuildConfigField* digunakan sebagai *base url* atau *link website* yang telah dibuat sebagai *platform* basis data aplikasi.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support:design:27.1.1'
    implementation 'com.android.support:cardview-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'
    implementation 'com.google.android.gms:play-services-maps:15.0.1'
    implementation 'com.android.support:support-v4:27.1.1'
    testImplementation 'junit:junit:4.12'
    implementation 'com.jakewharton:butterknife:8.8.1'
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.8.1'
}
```

```

    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-
core:3.0.2'

    implementation 'com.squareup.retrofit2:retrofit:2.4.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
    implementation 'com.google.code.gson:gson:2.8.2'
    implementation 'com.squareup.okhttp3:okhttp:3.10.0'
    implementation 'com.amulyakhare:com.amulyakhare.textdrawable:1.0.1'
    implementation 'com.squareup.okhttp3:logging-interceptor:3.10.0'
}

```

Dependencies adalah *plugin* atau tambahan *library* yang digunakan untuk pembuatan aplikasi survei bangun gedung. *Library* yang sangat dibutuhkan pada aplikasi ini adalah 'com.squareup.retrofit2:retrofit:2.4.0' sebagai pembuat penghubung antara REST API *website* dan android. *Library* *annotationProcessor* 'com.jakewharton:butterknife-compiler:8.8.1' juga digunakan sebagai *butterknife* pada setiap activity android untuk menggantikan model deklarasi variabel.

4.3.2.2 Pembuatan Class APIInterface

Setelah membuat project baru android selanjutnya membuat kelas baru bernama APIInterface yang berfungsi sebagai REST API (*Application Programming Interface*) penghubung antara android dan *website*. Pada kelas APIInterface ini android memerlukan tambahan *library* baru bernama Retrofit. **Retrofit** adalah library Android yang dibuat oleh Square yang digunakan sebagai REST Client pada Android, yang berfungsi untuk memudahkan dalam programing. Karena tidak perlu lagi untuk membuat method-method sendiri untuk menggunakan REST Client API dari backend. Berikut kode aplikasi yang terdapat pada kelas APIInterface.Java :

```

public interface APIInterface {
    @POST("auth/login")
    Call<JsonObject> login(@Query("user_id")String user_id, @Query("password")
String password);
}

```

Kode diatas adalah API untuk *login* pada android dengan menggunakan method *POST* dan diberinama login untuk mendapatkan user_id dan mendapatkan password sebagai string.

```
@POST("auth/logout")
    Call<RequestResponse> logout(
        @Header("Authorization") String token,
        @Header("User-ID") String userid);
```

Kode diatas digunakan untuk *logout* pada aplikasi android. Dengan menggunakan method *POST* dan mendapatkan *Authorization* dan *user_id* sebagai string. User-ID dan *authorization* merupakan sebuah header yang dibutuhkan aplikasi android untuk masuk kedalam aplikasi dan menggunakan basis data pada android.

```
@POST("auth/authenticate")
    Call<RequestResponse> checkLogin(
        @Header("Authorization") String token,
        @Header("User-ID") String userid);
```

Kode diatas digunakan untuk cek validasi login. API tersebut digunakan untuk mengambil kembali userid dan token yang telah ada didalam basis data dan mencocokkan apabila *session login* pada database telah berakhir.

```
@GET("databangunan/list_bangunan_user")
    Call<BangunanList> getSemuaBangunan(
        @Header("Authorization") String token,
        @Header("User-ID") String userid);
```

Kode diatas digunakan mengambil seluruh data bangunan yang ada didalam database survei bangun gedung. API tersebut bernama *getSemuaBangunan* yang digunakan pada saat user membutuhkan seluruh data gedung di dalam tampilan *Mainactivity*. API tersebut memanggil kelas *BangunanList* sebagai kamus data yang akan mendapatkan data dari basis data dan akan disimpan dalam *BangunanList* tersebut.

```
@DELETE("databangunan/hapusdata/{id_bangunan}")
    Call<RequestResponse> deleteBangunan (
        @Header("Authorization") String token,
        @Header("User-ID") String userid,
        @Path("id_bangunan") String id_bangunan);
```

Pada kode diatas adalah API untuk menghapus data gedung yang telah dimasukan dengan android ke dalam basis data. API tersebut menggunakan method *DELETE* untuk menghapus pada android. Parameter yang ditambahkan pada API adalah

id_bangunan untuk mengarahkan id_bangunan sebagai *primary key* untuk menghapus data.

```
@GET("databangunan/list_bangunan_petugas_survei/{user_id}")
Call<BangunanList> getBangunanPetugasSurvei(
    @Header("Authorization") String token,
    @Header("User-ID") String userid,
    @Path("user_id") String user_id);
```

Pada kode diatas digunakan untuk mengambil data yang hanya dimasukan oleh *user* tertentu. Dengan tambahan parameter *user_id* sehingga data yang ditampilkan hanya data yang dimasukan oleh *user* yang bersangkutan.

```
@POST("databangunan/masukandata")
Call<RequestResponse> masukandata (
    @Header("Authorization") String token,
    @Header("User-ID") String userid,
    @Body Bangunan bangunan);
```

Berdasarkan kode diatas digunakan untuk memasukan data pada basis data dengan menggunakan method *POST* dengan android. Dalam memasukan data tersebut digunakan *@Body Bangunan* sebagai parameter body yang akan di masukan pada basis data.

```
@PUT("databangunan/updatedata/{id_bangunan}")
Call<RequestResponse> updateBangunan (
    @Header("Authorization") String token,
    @Header("User-ID") String userid,
    @Path("id_bangunan") String id_bangunan,
    @Body Bangunan bangunan);
```

Pada kode diatas digunakan untuk mengubah atau memperbarui data yang telah dimasukan kedalam basis data. Parameter yang digunakan adalah *id_bangunan* yang digunakan sebagai *primary_key* pada saat pengambilan data agar dapat mengubah data tersebut.

4.3.2.3 Pembuatan Halaman SplashScreen

Setelah membuat project baru android selanjutnya membuat activity XML dan java untuk mendesain tampilan awal pada saat program aplikasi dijalankan yaitu Splashscreen. Berikut tahapan-tahapan pembuatan splashscreen :

- a. Desain tampilan `activity_splashscreen.xml` dengan menggunakan `ConstraintLayout` agar widget terurut secara *vertical*. Didalam tag `ConstraintLayout` menggunakan android orientation *vertical* karena perangkat yang nantinya akan digunakan untuk aplikasi berorientasi *vertical*. Untuk mengatur tinggi dan lebar layer dengan menggunakan `match_parent` supaya ukuran mengikuti ukuran layer. Fungsi yang digunakan pada xml yaitu `android:gravity center` agar tampilan berada ditengah layer. menggunakan *ImageView*, *Progresbar* dan *TextView* untuk mengisi konten dalam layar. Berikut kode aplikasi pada `activity_splashscreen.xml` :

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
```

- b. Ditambahkan `ImageView` didalam tag `ConstraintLayout`. `ImageView` tersebut berfungsi untuk menampilkan gambar. Gambar yang digunakan untuk splashscreen adalah `sbg.png` kemudian untuk lebar dan tinggi dengan mengatur satuan `dp`. Berikut kode aplikasi pada `activity_splashscreen.xml` :

```
<ImageView
    android:id="@+id/imageView3"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="32dp"
    android:src="@drawable/ic_menu_camera"
    app:layout_constraintBottom_toTopOf="@+id/textView6"
    app:layout_constraintEnd_toEndOf="parent"
```

```

        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

```

- c. Ditambahkan 2 TextView untuk menuliskan nama aplikasi dan untuk menuliskan nama pembuat aplikasi. Berikut kode aplikasi yang masih dalam activity_splashscreen.xml :

```

<TextView
    android:id="@+id/textView6"
    android:layout_width="wrap_content"

    android:layout_height="wrap_content"
    android:layout_marginBottom="36dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:text="SISTEM  INFORMASI  SURVEI  BANGUN
GEDUNG"
    android:textAlignment="center"
    android:textColor="@android:color/black"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/progressBar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:textStyle="bold"
    android:text=""
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

```

- d. Ditambahkan 1 progressbar digunakan untuk *loading screen* dan pengecekan *session user* yang login di aplikasi survei bangun gedung. Berikut kode progress bar di activity_splashscreen.xml :

```

<ProgressBar
    android:id="@+id/progressBar"
    android:layout_width="wrap_content"

```



```

        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        app:layout_constraintBottom_toTopOf="@+id/textView4"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

```

- e. Setelah desain tampilan *Splashscreen* pada *activity_splashscreen.xml*, kemudian memasukan kode pada *activity* Java yang bernama *splashscreen.java* yang sudah dibuat pada langkah diatas. Kode *splashscreen.java* mempunyai *class splashscreen*. Kemudia tampilan pembuka ini akan memanggil layout dan *activity_splashscreen.xml*. lalu *activity_splashscreen* akan muncul selama aplikasi mencoba koneksi ke dalam server aplikasi. Setelah itu tampilan akan berpindah ke tampilan *login* dengan perintah *intent* dari kelas *Splashscreen.this* ke kelas *Login.class*. *Intent* merupakan sebuah *object* yang merupakan kunci berkomunikasi dengan *activity* lainnya. Di dalam *Splashscreen.java* juga terdapat method untuk pengecekan *userSession*. *UserSession* ini digunakan untuk mengambil data *login* yaitu *IdUser* dan *UserToken* untuk autentikasi mengakses basis data pada *website*. Apabila *user* belum *login* kedalam aplikasi maka aplikasi akan tertuju ke halaman *Login* apabila *user* sudah *login* maka aplikasi akan tertuju langsung kedalam *MainActivity*. Berikut kode skrip pada *Splashscreen.Java* :

```

public class Splashscreen extends BaseActivity {
    UserSession userSession;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splashscreen);
        userSession = new UserSession(Splashscreen.this);
        Thread timerThread = new Thread(){
            public void run(){
                try {
                    sleep(3000);
                }
            }
        };
        timerThread.start();
    }
}

```

Kode diatas digunakan deklarasi class UserSession dan insialisai class UserSession kedalam SplashScreen.Java. dan loading akan berhenti dan memberikan pesan gagal pada saat sudah cek koneksi setelah 3 detik.

```
catch (InterruptedException e){
e.printStackTrace(); }finally {
    if (userSession.isUserLoggedIn()){
        APIInterface apiInterface =
        APIClient.getClientAuthorize().checkLogin(APIInterface.class);
        Call<RequestResponse> call =
        apiInterface.checkLogin(userSession.getUserToken(),
        String.valueOf(userSession.getIdUser()));
        call.enqueue(new Callback<RequestResponse>() {
```

Kode diatas digunakan untuk memanggil API retrofit pada baris APIClient.getClientAuthorize().checkLogin(APIInterface.class); API tersebut digunakan untuk mengecek apakah *user* tersebut sudah *login* atau belum dan untuk pengecekan UserSession apakah *login user* sudah dalam waktu 12 jam terakhir.

```
try {
    if(response.isSuccessful())
    Intent intent = new Intent(Splashscreen.this,MainActivity.class);
    startActivity(intent);
    }else{
    sessionExpired();
    Intent intent = new Intent(Splashscreen.this,Login.class);
    startActivity(intent);
    }
}catch (Exception e){
    Toast.makeText(Splashscreen.this, "Password atau ID
    Salah",Toast.LENGTH_SHORT).show();
    }
}

@Override
public void onFailure(Call<RequestResponse> call, Throwable t) {
    showToast(t.toString());
    }
});
}else{
    Intent intent = new Intent(Splashscreen.this,Login.class);
```

```

        startActivity(intent);
    }
}
};
timerThread.start();
}

```

Pada kode diatas digunakan untuk pengecekan dan untuk menentukan apakah *session user* sudah berakhir atau belum. Dan untuk pengecekan apakah id dan *password* yang dimasukan pada saat *form login* pada android benar atau tidak.

4.3.2.4 Pembuatan Halaman Login

Pembuatan halaman *login* digunakan untuk membatasi akses siapa saja yang dapat menggunakan aplikasi survei bangun gedung. Halaman ini akan ditampilkan setelah halaman *splashscreen* apabila sudah melewati kondisi pengecekan pada halaman tersebut. Pada pembuatan halaman *login* ini terdapat *activity_login.xml* dan *login.java*. berikut kode pembuatan halaman *login*.

- a. Seperti *splashscreen* halaman *login* juga memiliki file xml dan halaman java. Langkah pertama yang dilakukan adalah membuat tampilan untuk halaman login. Berikut kode login.xml

```

<ImageView
    android:id="@+id/imageView3"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="32dp"
    android:src="@drawable/ic_menu_camera"
    app:layout_constraintBottom_toTopOf="@+id/guideline4"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<android.support.constraint.Guideline

```

```

        android:id="@+id/guideline4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.5" />
        <android.support.design.widget.TextInputEditText
            android:id="@+id/txtPassword"
            android:text="alfian123"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Password"
            android:inputType="textPassword" />
    </android.support.design.widget.TextInputLayout>

```

Pada halaman login terdapat 1 *ImageView*, 2 *EditText*, 1 *button* dan 2 *TextView*. *Imageview* digunakan untuk menampilkan gambar dari logo aplikasi survei bangun gedung. Dan 2 *EditText* digunakan untuk mengisi *user_id* dan *password*. *Button* digunakan untuk masuk kedalam aplikasi survei bangun gedung. *TextView* digunakan untuk menampilkan keterangan aplikasi survei bangun gedung dan menampilkan nama pembuat aplikasi.

- b. Setelah tampilan layout *login* telah dibuat maka masukan kode pada *login.java* untuk menjalankan fungsi pada saat *user* sedang *login*. Berikut potongan kode dan penjesalan pada *login.java* :

```

public class Login extends BaseActivity implements
View.OnClickListener {
    @BindView(R.id.btnLogin) Button btnLogin;
    @BindView(R.id.txtID) TextInputEditText txtID;
    @BindView(R.id.txtPassword) TextInputEditText txtPassword;

```

Berdasarkan kode diatas adalah awal dari kelas *login.java* kelas extends *BaseActivity* dan implement *ViewOncklickListener*. Dengan menggunakan ekstend pada header class maka *method* ata *function* yang terdapat pada *baseactivity* tidak perlu di inisialisasi pada kelas *login*. Untuk implement *View.OncklickListner* berfungsi untuk membuat *method* aksi kepada tombol yang akan diklik pada saat *login*. *@BindView* diatas adalah untuk deklarasi atribut atau widget yang terdapat pada *activity_login.xml*.

```

private void login(){
    mRegProgres.setTitle("Getting Data");
    mRegProgres.setMessage("Please Wait...");
    mRegProgres.setCanceledOnTouchOutside(false);
    mRegProgres.show();
    final String user_id = txtID.getText().toString().trim();
    String pass = txtPassword.getText().toString().trim();
    Map<String, Object> jsonParams = new ArrayMap<>();
    jsonParams.put("username", user_id);
    jsonParams.put("password", pass);

```

Kode diatas adalah *method login* yang dibuat pada *login.java*. *mRegProgres* adalah progress bar yang terdapat pada *baseactivity*. *Progress bar* tersebut memberikan pesan “Getting Data” setelah itu android akan mengambil *string* pada masukan *user_id* dengan *id* pada *activity_login.xml* yaitu *txtID* dan mengambil *string password* dengan *id* *txtPassword* pada *activity_login.xml*.

```

APIInterface.apiInterface=
APIClient.getClientAuthorize().create(APIInterface.class);
    Call<JsonObject> call = apiInterface.login(user_id, pass);
    call.enqueue(new Callback<JsonObject>() {
        @Override
        public void onResponse(Call<JsonObject> call,
        Response<JsonObject> response) {
            try {
                if(response.isSuccessful()){
                    Log.d("masuk", "asasas");
                    JsonElement jsonToken =
                    response.body().get("token");
                    JsonElement jsonId = response.body().get("id");
                    JsonElement jsonNama =
                    response.body().get("nama");
                    JsonElement jsonUserId =
                    response.body().get("user_id");
                    JsonElement jsonuserRole = response.body().get("role");

```

Setelah mendapatkan *user_id* dan *password* berupa json, maka android akan memanggil API login yang telah dibuat pada kelas *APIClient*. Pada saat *login* akan mendapatkan *response* dari *website* berupa json *UserId*, *UserToken*, *UserRole* dan *UserName*.

4.3.2.5 Pembuatan Halaman MainActivity

Pada pembuatan *mainactivity* sama seperti *splashscreen* namun sudah tidak memerlukan untuk pembuatan file baru *activity_main.xml* nya dan java class karena sudah secara default tersedia ketika membuat project baru. Sebelumnya tambahkan *user-permission* agar aplikasi Survei Bangun Gedung dapat berjalan. Berikut *user-permission* yang ditambahkan di dalam *AndroidManifest.xml* :

```
<uses-permission android:name="android.permission.INTERNET" />

<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />

<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

android.permission.ACCESS_FINE_LOCATION digunakan untuk memberikan perizinan untuk deteksi lokasi *smartphone* secara terperinci. *android:name=android.permission.ACCESS_NETWORK_STATE* digunakan untuk mengecek apakah *smartphone* terdapat koneksi internet. *android.permission.ACCESS_COARSE_LOCATION* digunakan untuk melakukan akses lokasi Anda melalui WIFI yang sedang digunakan. *android.permission.INTERNET* digunakan untuk mengakses internet ketika melakukan download API Google Maps.

- a. Langkah pertama dalam pembuatan *MainActivity* adalah mengatur tampilan awal, agar tampilan memenuhi ukuran layar ponsel, pada tampilan ini dibuat *width* dan *height* dengan *match_parent*, dan berikan *id* sebagai identitas tampilan nanti jika tampilan ini dipanggil kembali oleh sebuah fungsi. *id* tampilan *main_activity* adalah “*drawer*” dapat dilihat dari

`android:id="@+id/drawer_layout"`. `alignParentTop` berfungsi agar tampilan tepat berada dibawah header aplikasi. Dan `android.support.v4.widget.DrawerLayout` ini *library* yang digunakan pada menu aplikasi apabila menggunakan drawer atau tampilan yang bergeser pada tampilan `main_activity`. Dan menambahkan *drawer* dengan *id* `android:id="@+id/nav_view"` yang berfungsi untuk meletakkan menu yang disediakan aplikasi. Include `layout="@layout/app_bar_main"` digunakan untuk memasukan tampilan `app_bar_main` kedalam tampilan `MainActivity`

```
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />
</android.support.v4.widget.DrawerLayout>
```

- b. Selanjutnya kode dimasukan ke dalam *Mainactivity.class* untuk menjalankan fungsi-fungsi yang terdapat pada program yang terdapat pada tampilan `MainActivity`. Berikut kode yang terdapat pada `MainActiviy.java`

```
private void addMarkers(){

APIInterface apiInterface=
APIClient.getClientAuthorize().create(APIInterface.class);

Call<BangunanList>call=
apiInterface.getSemuaBangunan(userSession.getUserToken(),userSession.g
etIdUser());call.enqueue(new Callback<BangunanList>() {
```

Kode diatas adalah untuk memanggil API `getSemuaBangunan` pada *class* `APIInterface` untuk memanggil seluruh data yang ada didalam basis data. Dengan menggunakan refrensi dari *user* `UserToken` dan `IdUser` jika kedua refrensi tersebut valid maka *marker* akan muncul.

```
for (int i = 0;i< bangunan.size();i++){

    mLatlng                                =                                new
LatLng(Double.parseDouble(bangunan.get(i).getLatitude()),

                                Double.parseDouble(bangunan.get(i).getLongitude()));

    Marker marker = mMap.addMarker(new MarkerOptions()

                                .position(mLatlng)

                                .title(bangunan.get(i).getNama_bangunan())

                                .snippet("Cek Data Gedung"));

    detailMarker.put(marker,bangunan.get(i));}

showToast("Mendapatkan Posisi Gedung")}
```

Kode diatas menggunakan pengulangan untuk membaca semua data yang terdapat pada basis data, setelah itu data yang diambil hanya *latitude* dan *longitude* untuk menentukan posisi *marker*. Setelah *latitude* dan *longitude* sudah didapatkan apabila *marker* ditekan maka akan muncul nama bangunan dan keterangan untuk cek data gedung.


```

@Override

    public void onFailure(Call<BangunanList> call, Throwable t) {

        showToast("Gagal Mendapatkan Lokasi")
    }

});

mMap.setOnInfoWindowClickListener(new
GoogleMap.OnInfoWindowClickListener() {

    @Override

    public void onInfoWindowClick(Marker marker) {

        Intent i = new Intent(MainActivity.this,DetailBangun.class);

        i.putExtra("bangunan",detailMarker.get(marker));

        startActivity(i);

    }

});

}

```

Kode diatas apabila *marker* ditekan untuk kedua kalinya ditempat yang sama maka akan berpindah ke tampilan DetailBangun.class yang berisi keterangan tentang bangunan gedung yang telah di survei oleh petugas survei.

```

private void checkRole(){

    userSession = new UserSession(MainActivity.this);

    //Petugas survei = 2 , User Eksternal = 3

    String id = userSession.getUserRoleStr();

    Log.d("role", id);
}

```

```

        if(id.equals("3")){
            hideItemNavigation();
        }
    }

```

Berdasarkan kode diatas adalah kode untuk pemisahan akses *user*. Terdapat 2 *user* yaitu petugas survei dan user eksternal. Setelah *role user* diketahui kode diatas memanggil *method* `hideItemNavigation()`;

```

private void hideItemNavigation(){
    NavigationView navigationView = (NavigationView)
    findViewById(R.id.nav_view);

    Menu nav_Menu = navigationView.getMenu();

    nav_Menu.findItem(R.id.nav_history).setVisible(false);

    nav_Menu.findItem(R.id.nav_input_data).setVisible(false);
}

```

Kode diatas adalah *method* dari `hideItemNavigation()` yang berfungsi untuk menyembunyikan menu item `nav_history` dan `nav_input_data` apabila yang *login* ke dalam aplikasi adalah *role* user eksternal.

```

private void logout(){
    userSession = new UserSession(MainActivity.this);

    mRegProgres.setTitle("Logout Account");

    mRegProgres.setMessage("Please Wait...");

    mRegProgres.setCanceledOnTouchOutside(false);

    mRegProgres.show();

    APIInterface apiInterface =
    APIClient.getClientAuthorize().create(APIInterface.class);
}

```

```

        Call<RequestResponse> call =
        apiInterface.logout(userSession.getUserToken(),userSession.getIdUser());

```

Berdasarkan kode diatas digunakan untuk menampilkan progress bar dan melakukan *logout* dari aplikasi survei bangun gedung. Kode diatas memanggil API logout dengan menggunakan refrensi UserToken dan IdUser.

```

public boolean onNavigationItemSelected(MenuItem item) {

    // Handle navigation view item clicks here.

    int id = item.getItemId();

    if( id == R.id.posisi_bangunan)  }

    else if (id == R.id.nav_input_data) {

        Intent    surveibangun    =    new    Intent(MainActivity.this,
Spinner_fungsi.class);    surveibangun.putExtra(ConstantsUtils.ACTION,
ConstantsUtils.ACTION_NEW);

        MainActivity.this.startActivity(surveibangun);

    } else if (id == R.id.nav_list_survei) {

        Intent    listSurvei    =    new    Intent(MainActivity.this,
BangunanView.class);

        MainActivity.this.startActivity(listSurvei);

```

Berdasarkan kode diatas digunakan sebagai menu pilihan di dalam aplikasi survei bangun gedung, terdapat 5 menu untuk petugas survei dan 3 menu untuk user eksternal. Diantaranya adalah menu posisi gedung, input gedung, daftar bangun gedung, history input petugas survei dan *logout*.

```

} else if (id == R.id.nav_logout) {

    AlertDialog.Builder      alert      =      new
AlertDialog.Builder(MainActivity.this);

    alert.setTitle("Logout");

    alert.setMessage("Anda Yakin Akan Logout ? ");

    alert.setPositiveButton("Ya",                                new
DialogInterface.OnClickListener() {

        @Override

        public void onClick(DialogInterface dialog, int which) {

            dialog.dismiss();

            logout();

        }

    }).setNegativeButton("Tidak",                                new
DialogInterface.OnClickListener() {

        @Override

        public void onClick(DialogInterface dialog, int which) {

            dialog.dismiss();

        }

    }).create().show()}

```

kode diatas adalah salah satu menu yaitu *logout*, apabila menu *logout* ditekan maka akan menjalankan *method logout* yang telah dibuat sebelumnya. Setelah itu akan muncul Dialog untuk menyatakan *logout* atau tidak. Apabila menekan “Ya” maka akan kembali ke menu *Login*. Apabila menekan tidak maka akan kembali kedalam dalam MainActivity.

4.3.2.6 Pembuatan Halaman DetailBangun

Setelah pembuatan detailbangun yang berfungsi untuk menampilkan *maps* dan titik setiap bangun gedung yang telah didata oleh petugas survei, maka selanjutnya pembuatan detailbangun yang berfungsi untuk menampilkan data yang telah dimasukan oleh petugas survei. Pada pembuatan halaman detailbangun ini memiliki *java class* dan memiliki file *xml* untuk mengatur tampilan. Berikut listing kode untuk halaman detailbangun:

- a. Langkah pertama dalam pembuatan detailbangun adalah mengatur tampilan awal, agar tampilan memenuhi ukuran layar ponsel, pada tampilan ini dibuat *width* dan *height* dengan *match_parent*, pada tampilan ini menggunakan *ScrollView* agar tampilan layar dapat di *scroll* untuk melihat informasi dan menggunakan tag *LinearLayout* yang berguna untuk mengatur tampilan secara *linear*, dan menambahkan tag *cardview* agar tampilan akan membentuk persegi sesuai dengan kategori yang telah ditentukan. Berikut listing program *activity_detail_bangun.xml* :

```
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
```

Berdasarkan kode diatas adalah tag *Scrollview* yang berguna agar tampilan layar dapat di *scroll* dengan memberika ukuran *width* dan *height* dengan *match_parent* agar sesuai dengan tampilan layar pada android.

```
<LinearLayout
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="wrap_content">
<android.support.v7.widget.CardView
android:layout_margin="8dp"
android:elevation="8dp"
app:cardCornerRadius="16dp"
```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <LinearLayout
            android:layout_margin="16dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

```

Kode diatas adalah lanjutan pada tampilan desain menggunakan tag *LinearLayout* dan memberikan tag *CardView*, dan tag *cardCornerRadius* digunakan untuk memberikan efek melengkung pada setiap ujung *Cardview*.

```

<TextView
    android:drawableLeft="@drawable/ic_detail_bangun"
    android:drawablePadding="8dp"
    android:text="Fungsi Bangun Gedung"
    android:textColor="@color/colorPrimaryText"
    android:textSize="18dp"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<View
    android:layout_marginTop="8dp"
    android:layout_marginBottom="8dp"
    android:background="@color/colorAccent"
    android:layout_width="match_parent"
    android:layout_height="1dp"></View>

```

Lanjutan pada tampilan detailactivity menambahkan tag *TextView* yang berguna untuk memberikan text pada tampilan android. Tulisan tersebut terdapat pada atribut *android:text*. Untuk tag *View* hanya memberikan garis bawah yang berguna sebagai pembatas antar judul setiap kategori *cardview* dan isi data yang terdapat pada *cardview*.

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Penginput :"
    android:textStyle="bold" />
<TextView
    android:layout_marginTop="4dp"

```

```

        android:id="@+id/txtDetailPetugas survei"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="-"
        android:textColor="@color/colorSecondaryText" />

```

Pada TextView diatas diberikan id txtDetailPetugas survei yang berguna untuk menampilkan data yang akan didapatkan dari *website* dan nama petugas survei akan muncul pada textview diatas.

- b. Selanjutnya kode dimasukan ke dalam detailbangun.class untuk menjalankan fungsi-fungsi yang terdapat pada program yang terdapat pada tampilan detailbangun. Berikut kode yang terdapat pada detailbangun.java :

```

public class DetailBangun extends BaseActivity {
    @BindView(R.id.txtDetailPetugas survei)
    TextView txtpetugas survei;
    @BindView(R.id.txtFungsiGedung)
    TextView txtFungsiGedung;
    @BindView(R.id.txtJenisGedung)
    TextView txtJenisGedung;
    @BindView(R.id.txtJenisFungsi)
    TextView txtJenisFungsi;
}

```

Diatas adalah kode java untuk detailbangun.java, @BindView diatas adalah library *butterknife* untu menggantikan kata *findviewbyid* ketika deklarasi variabel memanggil atribut pada tampilan activity_detail_bangun.xml.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_detail_bangun);
    ButterKnife.bind(this);
    setTitle("Detail Bangunan");
    Bundle data = getIntent().getExtras();
    bangunan = data.getParcelable("bangunan");
}

```

Berdasarkan kode diatas adalah lanjutan dari kode detailbangun.java. kode diatas berfungsi sebagai pemanggil data pada class bangunan atau pada kamus data pada android di baris kode Bundle data = getIntent().getExtra().

Pada baris selanjutnya `bangunan = data.getParcelable("bangunan")` untuk memanggil satu bundle data pada class `bangunan`.

```
if (bangunan != null) {
    id_bangunan = bangunan.getId_bangunan();
    txtpetugas_survei.setText(bangunan.getNama());
    txtFungsiGedung.setText(bangunan.getFungsi_bangunan());
    txtJenisGedung.setText(bangunan.getJenis_bangunan());
    txtJenisFungsi.setText(bangunan.getJenis_fungsi_bangunan());
    txtNamaDepar.setText(bangunan.getNama_departemen());
    txtAlmtDepar.setText(bangunan.getAlamat_departemen());
    txtNoIkmn.setText(bangunan.getNo_ikmn());
}
```

Berdasarkan kode diatas digunakan untuk *set* data untuk pada atribut yang yang terdapat pada `activity_detail_bangunan.xml` yang telah dibuat. Pada saat data telah masuk kedalam android maka data akan masuk kedalam `bangunan.class` dan dimasukan kedalam variabel pada `textview`.

4.3.2.7 Pembuatan Halaman Form Survei

Halaman survei yang dibuat digunakan untuk petugas survei dalam memasukan data gedung. Halaman form survei dibuat dengan nama *class* `spinner_fungsi.java`. `spinner_fungsi` merupakan *activity* yang memiliki file berupa java *class* dan file berupa `xml`. Berikut kode form survei yang telah dibuat :

- a. Pada halaman form survei memiliki banyak *widget* atau atribut yang dimasukan kedalam tampilan dikarenakan kebutuhan data yang akan dimasukan oleh petugas survei bangun gedung. Berikut tampilan kode form survei yang terdapat pada form `xml` yaitu `activity_spinner_fungsi.xml`:

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/app_bar_main"
    tools:context="com.example.alfhanrf.skripsihehe.Spinner_fungsi">
```


Tampilan form survei bangun gedung menggunakan dasar layar *scrollview* sehingga tampilan dapat *discroll* sampai dengan konten terakhir pada layar. Menggunakan *width* dan *height* dengan ukuran *match_parent* agar ukuran sesuai dengan layar *smartphone* yang digunakan.

```
<android.support.v7.widget.CardView
    android:layout_margin="8dp"
    android:elevation="8dp"
    app:cardCornerRadius="12dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<Spinner
    android:id="@+id/Spinnerfungsigedung"
    android:layout_width="match_parent"
    android:entries="@array/fungsi_gedung"
    android:layout_height="50dp" />
```

Pada tampilan form survei menggunakan *widget* atau atribut berupa *cardview* agar masukan data sesuai dengan kategori pada saat dimasukan oleh petugas survei, sehingga petugas dapat membedakan setiap kategori pada saat memasukan data bangun gedung kedalam basis data. Kode diatas juga menggunakan *Spinner* sebagai pilihan yang telah ditentukan atau dikategorikan. *Spinner* tersebut memiliki *id* *Spinnerfungsigedung* yang akan dipanggil pada java class.

```
<android.support.design.widget.TextInputLayout
    android:layout_marginTop="8dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
<android.support.design.widget.TextInputEditText
    android:id="@+id/NoIkmn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Nomor IKMN"
    android:inputType="number" />
</android.support.design.widget.TextInputLayout>
```

Pada kode diatas menggunakan tambahan *library design widget texinput layout* agar tampilan untuk petugas layout lebih mudah. Kode diatas berfungsi memasukan text seperti diatas adalah *NoIkmn*. Tipe data masukan adalah

number sehingga pada saat petugas survei melakukan pendataan *keyboard smartphone* akan berubah menjadi angka.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="340dp"
    android:orientation="vertical">
    <fragment
xmlns:android="http://schemas.android.com/apk/res/android"

    android:name="com.google.android.gms.maps.SupportMapFragment"

        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    </LinearLayout>
```

Kode diatas masih berada pada file *spinner_fungsi.xml*, *tag* diatas memanggil *map fragment* agar *map* dapat ditampilkan dalam form survei gedung untuk menentukan lokasi atau titik bangun gedung dengan GPS (*Global Positioning Services*) pada saat petugas survei melakukan pendataan posisi bangun gedung.

- b. Selanjutnya adalah pembuatan *method* dan fungsi pada kelas *spinner_fungsi.java*. kode tersebut digunakan untuk memasukan data petugas survei kedalam basis data yang telah disediakan. Berikut kode *spinner_fungsi.java* :

```
public class Spinner_fungsi extends BaseActivity implements
    AdapterView.OnItemClickListener, OnMapReadyCallback {
    @BindView(R.id.Spinnerfungsigedung)
    Spinner fungsigedung;
    @BindView(R.id.Spinnerfungsigedung1)
    Spinner Spinnerfungsigedung1;
    @BindView(R.id.Spinnerfungsigedung2)
    Spinner Spinnerfungsigedung2;
    @BindView(R.id>NamaDepar)
    TextInputEditText>NamaDepar;
```

Pada kode diatas adalah kode header *class Spinner_fungsi* dengan *extends baseactivity* dan implement *AdapeterView.OnItemClickListener* yang digunakan untuk membuat *method menu spinner* dan *OnMapReadyCallback*

digunakan untuk memanggil *map* pada tampilan form survei. @BindView digunakan untuk mendeklarasikan *widget* yang terdapat file *spinner_fungsi.xml* kode diatas mendeklarasikan 3 spinner dan 1 TextInputEditText.

```

        ArrayAdapter<String> datafungsi = new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_item,          fungsigedunglist);
        datafungsi.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        fungsigedung.setAdapter(datafungsi);

```

kode diatas adalah adapter untuk spinner yang digunakan, fungsi tersebut memanggil array dari fungsigedunglist yang terdapat pada folder values dan file string agar kategori dari array gedung tersebut masuk kedalam spinner.

```

        fungsigedung.setOnItemClickListener(new
        AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View
            view, int position, long id) {
                if (position == 0) {
                    Spinnerfungsigedung1.setEnabled(true);
                    Spinnerfungsigedung1.setAdapter(datafungsibag1hunian);
                    Spinnerfungsigedung2.setEnabled(false);
                    refValue = 1;
                } else if (position == 1) {
                    Spinnerfungsigedung1.setEnabled(true);

                    Spinnerfungsigedung1.setAdapter(datafungsibag1keagamaan);
                    Spinnerfungsigedung2.setEnabled(false);
                    refValue = 2;
                }
            }
        });

```

Fungsi kode diatas adalah *spinner* fungsi gedung apabila pada saat dipilih atau akan memasukan nilai berupa angka, angka tersebut digunakan untuk memberikan *id_reference* kedalam basis data. Apabila dipilih salah nilai *spinner* maka akan terdapat kondisi dimana akan mengganti nilai *string-array* pada spinnerfungsi1 dan akan mengganti nilai pada string array spinnerfungsi2

```

        Button gpsGetPosisi = (Button) findViewById(R.id.getPosisi);
        gpsGetPosisi.setOnClickListener(new
        View.OnClickListener() {
            @Override
            public void onClick(View v) {
                locationManager=(LocationManager)
                getSystemService(Context.LOCATION_SERVICE);
                if(!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDE
                R)) {
                    buildAlertMessageNoGPS();
                }elseif
                (locationManager.isProviderEnabled(LocationManager.GPS_PROV
                IDER)) {
                    getLocation();
                    if
                    (ActivityCompat.checkSelfPermission(Spinner_fungsi.this,
                    Manifest.permission.ACCESS_FINE_LOCATION)           !=
                    PackageManager.PERMISSION_GRANTED                       &&
                    ActivityCompat.checkSelfPermission(Spinner_fungsi.this
                    Manifest.permission.ACCESS_COARSE_LOCATION)           !=
                    PackageManager.PERMISSION_GRANTED) {
                        return;
                    }
                    mMap.setMyLocationEnabled(true);
                }
            }
        });

```

Kode diatas adalah fungsi mendapatkan latitude dan longitude posisi dari petugas survei. Ketika menekan tombol dapatkan posisi gedung, maka android akan meminta izin untuk mengakses GPS serta memanggil *method* `getLocation()`. Apabila GPS petugas survei gedung tidak menyala maka akan menjalankan *method* `buildAlertMessageNoGPS()`;

```

private void getLocation()
{
    if (ActivityCompat.checkSelfPermission(Spinner_fungsi.this,
    Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED    &&
    ActivityCompat.checkSelfPermission(Spinner_fungsi.this,
        Manifest.permission.ACCESS_FINE_LOCATION)    !=
    PackageManager.PERMISSION_GRANTED) {
        mMap.setMyLocationEnabled(true);
    }
}

```

```

        ActivityCompat.requestPermissions(Spinner_fungsi.this,
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
        REQUEST_LOCATION);
    } else {
        Location location =
        locationManager.getLastKnownLocation(LocationManager.NETW
        ORK_PROVIDER);
        LatLng target = new LatLng(location.getLatitude(),
        location.getLongitude());

```

Kode diatas adalah *method* getLocation() yang berfungsi untuk mendapatkan posisi latitude longitude petugas survei bangun gedung. Kode diatas melakukan pengecekan apakah sudah mendapatkan perizinan untuk mendapatkan lokasi.

```

        if (location != null) {
            double latti = location.getLatitude();
            double longi = location.getLongitude();
            latitude = String.valueOf(latti);
            longitude = String.valueOf(longi);
            getLatitude.setText(latitude);
            getLongitude.setText(longitude);
            CameraPosition.Builder builder = new
            CameraPosition.Builder();
            builder.zoom(18);
            builder.target(target);
            this.mMap.animateCamera(CameraUpdateFactory.newCameraPosit
            ion(builder.build()));
            mMap.addMarker(new MarkerOptions().position(new
            LatLng(location.getLatitude(),
            location.getLongitude())).title("Posisi Anda"));
        } else
            Toast.makeText(this, "Tidak Mendapatkan Posisi",
            Toast.LENGTH_SHORT).show();
    }
}

```

Kode diatas adalah untuk mendapatkan latitude longitude serta memberikan *marker* pada titik petugas survei, kode diatas apabila *location* tidak bernilai null maka akan mengambil titik latitude dan longitude kedalam form masukan posisi latitude dan longitude. apabila GPS tidak dapat mendapatkan posisi maka akan memberikan pesan “Tidak Mendapatkan Posisi”.

```

private void buildAlertMessageNoGPS() {
    final AlertDialog.Builder builder = new
AlertDialog.Builder(this);
    builder.setMessage("Harap Menyalakan GPS")
        .setCancelable(false)
        .setPositiveButton("Nyalakan GPS", new
DialogInterface.OnClickListener() {
            public void onClick(final DialogInterface dialog, final
int id) {
                startActivity(new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
            }
        })
        .setNegativeButton("Tidak", new
DialogInterface.OnClickListener() {
            public void onClick(final DialogInterface dialog, final
int id) {
                dialog.cancel();
            }
        });
    final AlertDialog alert = builder.create();
    alert.show();
}

```

Kode diatas adalah *method* buildAlertMessageNoGPS() yang berfungsi apabila GPS *smartphone* petugas survei bangun gedung tidak menyala maka *method* ini akan menjalankan fungsinya yaitu membuka *setting smartphone* untuk menyalakan GPS.

```

InputData.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if
(getIntent().getStringExtra(ConstantsUtils.ACTION).equals(ConstantsUtils.ACTION_NEW)) {
            if (fieldCheck()) {
                final AlertDialog.Builder builder = new
AlertDialog.Builder(Spinner_fungsi.this);
                builder.setMessage("Apakah Data yang Di Input
Benar ?")
                    .setCancelable(false)

```

```

        .setPositiveButton("Ya", new
        DialogInterface.OnClickListener() {
            public void onClick(final DialogInterface
            dialog, final int id) {
                createData(parseData());
                dialog.dismiss();
                finish();
                final AlertDialog alert = builder.create();
                alert.show();
            } else {
                showToast("Ada Data Yang Belum diisi");
            }
        }
    }

```

Kode diatas adalah *Listener* atau aksi yang akan dilakukan apabila *tombol* yang ber id *InputData* ditekan atau diklik. Pada saat tombol ditekan akan menjalankan *Method* *fieldcheck*, apabila kondisi *fieldcheck* dapat dilewati maka akan memunculkan dialog “Apakah Data yang DiInput Benar ?” apabila menekan “Ya” maka akan menjalankan *method* *createData* dan *parseData*.

```

private void createData(Bangunan bangunan) {
    UserSession userSession = new
    UserSession(Spinner_fungsi.this);
    mRegProgres.setTitle("Sending Data");
    mRegProgres.setMessage("Please Wait...");
    mRegProgres.setCanceledOnTouchOutside(false);
    APIInterface apiInterface =
    APIClient.getClientAuthorize().create(APIInterface.class);
    Call<RequestResponse> call =
    apiInterface.masukandata(userSession.getUserToken(),
        userSession.getIdUser(), bangunan);
    call.enqueue(new Callback<RequestResponse>() {
        @Override
        public void onResponse(Call<RequestResponse> call,
        Response<RequestResponse> response) {
            if (response.isSuccessful()) {
                mRegProgres.dismiss();
                showToast("Data Berhasil Di Inputkan");
            }
        }
    })
}

```

Berdasarkan kode diatas adalah *method* *createData*, *method* *createData* berfungsi untuk memasukan data dari android kedalam basis data yang ada di

website. Pada saat menjalankan *method* ini akan memanggil API masuk data yang telah dibuat pada APIClient, apabila berhasil akan mendapatkan *response* yaitu IdUser dan UserToken dengan parameter yaitu bangunan. Jika data berhasil masuk kedalam basis data android akan memberikan pesan “Data Berhasil Di Inputkan”

```
@Override
    public void onFailure(Call<RequestResponse> call,
        Throwable t) {
        mRegProgres.dismiss();
        showToast("Data gagal Dikirim : " + t.toString());
    }
});
```

Apabila data gagal masuk akan memberikan pesan “Data gagal Dikirim dan memberikan pesan error pada android.

```
private Bangunan parseData() {
    Bangunan bangunan = new Bangunan();
    UserSession userSession = new UserSession(this);
    if (bangunan != null) {
        bangunan.setId_bangunan(bangunan.getId_bangunan());
    }
    bangunan.setUser_id(userSession.getUserId());

    bangunan.setNama_departemen>NamaBangun.getText().toString().trim());

    bangunan.setJenis_fungsi_bangunan(Spinnerfungsigedung2.getSelectedItem().toString().trim());
    bangunan.setId_reference(refValue);
    bangunan.setAlamat_departemen(AlmtDepartemen.getText().toString().trim());
    bangunan.setNo_ikmn(NoIkmn.getText().toString().trim());
    bangunan.setNo_hdno(NoHdno.getText().toString().trim());
    bangunan.setTelpon(TelponBgnGedung.getText().toString().trim());
    bangunan.setEmail>EmailBgnGedung.getText().toString().trim());
```

Kode diatas adalah *method* parseData, *method* tersebut digunakan untuk mengambil text dan pilihan *spinner* yang telah dimasukan kedalam *form* survei.

setelah data dimasukan data akan dirubah menjadi string dan akan menghapus spasi yang tidak terpakai pada teks.

4.3.2.8 Pembuatan *Class RecyclerViewAdapterBangunan*

RecyclerView ini berfungsi sebagai pengganti *listview* dan *gridview* pada android agar konten yang disediakan menjadi lebih rapih dan terstruktur. *RecyclerView* juga mempunyai animasi default sesuai standar Google Material Design saat menambahkan atau menghapus elemen. Sedangkan untuk mengatur posisi item pada list, *RecyclerView* menggunakan *LayoutManagers*. *RecyclerView* menggunakan *ViewHolder* untuk menyimpan refrensi data yang akan ditampilkan. Pada aplikasi survei bangun gedung dibuat *recycleview* untuk menampilkan data bangun gedung secara *list* selain dengan cara melihat titik pada *maps*. Dengan menggunakan *recycleview* ini *user* lebih mudah mencari informasi bangun gedung dan dapat mencari dengan menggunakan fitur *search* yang telah tersedia. Berikut kode *RecyclerView* pada aplikasi survei bangun gedung :

```
public class RecyclerViewAdapterBangunan extends
RecyclerView.Adapter<RecyclerViewAdapterBangunan.ViewHolder> implements
AdapterView.OnItemClickListener {

    private static final String TAG = RecyclerView.class.getSimpleName();
    private Context mContext;
    private List<Bangunan> mList;
    private List<Bangunan> bangunan;
    private AdapterView.OnItemClickListener mListener;
    private OnItemLongClickListener mLongListener;
    UserSession userSession;
    MainActivity mainActivity;
    boolean dialogLongEnabled ;
```

Pada kode diatas adalah *header class* dari *recycleviewadapterbangunan*, kelas tersebut digunakan untuk menambahkan *method* dari *adapteronclicklonglistener*. Pada kode diatas juga terdapat deklarasi variabel yang akan digunakan pada *class recycleviewadapterbangunan*.

```
public RecyclerViewAdapterBangunan(Context context, List<Bangunan>
bangunan,boolean dialogLongEnabled) {
```

```

    this.mContext = context;
    this.bangunan = bangunan;
    this.dialogLongEnabled = dialogLongEnabled;
}

```

Pada kode diatas adalah sebuah *method* yang bersifat public yang dapat dipanggil di lain kelas. *Method* tersebut menggunakan parameter context yang dideklarasikan halaman fragment itu sendiri, lalu list<Bangunan> adalah kamus data yang telah dibuat pada android, serta Boolean dialogLongEnabled sebagai parameter untuk menentukan aksi ketika item ditekan dan ditahan untuk beberapa waktu.

```

@Override
public void onBindViewHolder(final ViewHolder holder, final int position) {
    userSession = new UserSession(mContext);
    final Bangunan item = bangunan.get(position);
    holder.txtNamaBangunan.setText(item.getNama_bangunan());
    holder.txtNamaDepar.setText("Nama      Departemen      :      "      +
item.getNama_departemen());
    holder.txtNamaPemilik.setText("Nama      Pemilik      :      "      +
item.getNama_pemilik_tanah());
    final ColorGenerator generator = ColorGenerator.MATERIAL; // or use
    DEFAULT
    final int color2 = generator.getColor(item.getNama_bangunan());
    final TextDrawable.IBuilder builder = TextDrawable.builder()
        .beginConfig()
        .endConfig()
        .round();
    TextDrawable          ic2          =
builder.build(item.getNama_bangunan().toUpperCase().substring(0, 1), color2);
    holder.imgIconBangunan.setImageDrawable(ic2);
}

```

kode tersebut digunakan untuk menampilkan apa saja item pada recycleview, terdapat inisialisasi class UserSession yang digunakan untuk mengambil UserId dan UserToken agar data yang terdapat pada basis data dapat digunakan dengan menggunakan 2 kunci tersebut. setelah mendapatkan 2 kunci untuk mendapatkan data maka holder atau item akan menampilkan nama bangunan, nama departemen dan nama pemilik tanah pada holder atau pada item recycleview. Color generator digunakan untuk membuat image pada holder dengan menggunakan fungsi substring untuk mengambil huruf terdepan

dari nama bangunan setelah itu akan dibuat image dengan menggunakan huruf tersebut dan dengan warna secara acak.

```
holder.cvItemBangunan.setOnLongClickListener(new
View.OnLongClickListener() {
    @Override
    public boolean onLongClick(final View v) {
        userSession = new UserSession(mContext);
        //Surveyor = 2 , User Eksternal = 3
        String id = userSession.getUserRoleStr();
        Log.d("role", id);
        if(id.equals("2") && dialogLongEnabled == true) {
```

Kode diatas adalah pemisahan hak akses setiap role, kode diatas adalah pemisahan *user* yang dapat memanggil *delete* dan *edit* data. *OnLongClickListener* adalah *method* yang digunakan untuk memberikan aksi ketika *list* data atau holder pada recycleview. Apabila *user role* bernilai 2 maka *method* dari *OnLongClickListener* dapat digunakan.

```
holder.cvItemBangunan.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(mContext, DetailBangun.class);
        i.putExtra("bangunan", bangunan.get(position));
        mContext.startActivity(i);
    }
});
```

Berdasarkan kode diatas berfungsi apabila *list* item atau holder ditekan atau dipilih, maka akan berpindah kedalam *class* *DetailBangun* dan tampilan *activity_detail_bangun.xml* mengambil data bangunan dengan pilihan dari *id* position holder atau *list* data.

```
private void deleteData(Bangunan position) {
    final BangunanView bangunanView = new BangunanView();
    userSession = new UserSession(mContext);
    final ProgressDialog mRegProgres = new ProgressDialog(mContext);
    mRegProgres.setTitle("Processing Data");
    mRegProgres.setMessage("Please Wait...");
    mRegProgres.setCanceledOnTouchOutside(false);
    mRegProgres.show();
    userSession = new UserSession(mContext);
```

```

APIInterface                                apiInterface                                =
APIClient.getClientAuthorize().create(APIInterface.class);
    Call<RequestResponse>                                call                                =
apiInterface.deleteBangunan(userSession.getUserToken(),userSession.getIdUser(),
position.getId_bangunan());
    call.enqueue(new Callback<RequestResponse>() {

```

Kode diatas adalah *method* untuk menghapus data oleh petugas survei yang memasukan data. Dalam kode tersebut berfungsi untuk menggunakan *progressbar* dengan memberikan pesan “Processing Data” . *method* tersebut memanggil API yang telah dibuat pada APIClient yaitu deleteBangunan dengan menggunakan header UserToken dan UserId sebagai kuncinya dan memberikan parameter *id* bangunan untuk menghapus data bangunan.

```

@Override
    public void onResponse(Call<RequestResponse> call,
Response<RequestResponse> response) {
        mRegProgres.dismiss();
        Toast.makeText(mContext,"DataBerhasilDi
Hapus",Toast.LENGTH_SHORT).show();
        bangunanView.loadDataBangunan();
    }

```

setelah berhasil menghapus data maka akan memberikan pesan “Data Berhasil Dihapus” dan mengambil data terbaru dengan memanggil *method* yang terdapat pada bangunanView yaitu loadDataBangunan().

```

public class ViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
    @BindView(R.id.txtNamaBangunan)
    TextView txtNamaBangunan;
    @BindView(R.id.txtNamaDepar)
    TextView txtNamaDepar;
    @BindView(R.id.txtNamaPemilik)
    TextView txtNamaPemilik;
    @BindView(R.id.imgIconBangunan)
    ImageView imgIconBangunan;
    @BindView(R.id.cvItemBangunan)
    CardView cvItemBangunan;
    public ViewHolder(View itemView) {
        super(itemView);
        ButterKnife.bind(this, itemView);
    }

```

Kode datas adalah berguna untuk deklarasi *widget* atau atribut yang terdapat pada file `item_bangunan.xml` untuk membentuk *list* item atau holder dari `recycleview`.

4.3.2.9 Pembuatan Halaman BangunanView

Halaman ini digunakan untuk menampilkan data yang telah dibuat pada `RecyclerViewAdapterBangunan`. Pada *activity* ini memiliki file `BangunanView.java` dan `activity_bangunan_view.xml`. berikut listing kode yang terdapat pada halaman `BangunanView` :

- a. Pembuatan tampilan halaman `BangunanView` ini hanya memanggil *widget* `recycleview` untuk menampilkan data yang dibuat pada `recycleviewadapterbangunan`. berikut kode yang terdapat pada file `activity_bangunan_view.xml` :

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.example.alfhanrf.skripsihehe.BangunanFrag">
<!-- TODO: Update blank fragment layout -->
<android.support.v7.widget.RecyclerView
    android:id="@+id/rvBangunan"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</android.support.v7.widget.RecyclerView>
</FrameLayout>
```

Berdasarkan kode di atas pada tampilan `bangunanview` menggunakan ukuran *width* dan *height* menggunakan *match_parent*. Tampilan `bangunanview` ini menampilkan *fragment*. *Fragment* adalah sub-layar yang ada pada tampilan `bangunanview`. *Fragment* tersebut diberi nama *id* `rvBangunan` yang menggunakan ukuran *match_parent*.

- b. Setelah tampilan `bangunan view` dibuat, kemudian masukan kode java yang terdapat pada `BangunanView.java`. yang berfungsi untuk menjalankan *method*

yang dibutuhkan pada tampilan BangunanView. Berikut listing kode yang terdapat pada *class* BangunanView :

```
public class BangunanView extends BaseActivity{
    private List<Bangunan> bangunan = new ArrayList<>();
    private RecyclerViewAdapterBangunan viewAdapterBangunan;
    @BindView(R.id.recycleview)
    RecyclerView recyclerView;
```

Kode diatas adalah header kelas dari bangunanview yang menggunakan *extends* dari *baseactivity* sehingga tak perlu inisialisasi *class* pada saat menggunakan *widget* atau *method* yang terdapat pada *baseactivity*. Kode diatas adalah deklarasi variabel *RecycleVieAdapterBangunan* dan *Bangunan* sebagai *ArrayList*.

```
public void loadDataBangunan() {
    UserSession userSession = new
    UserSession(BangunanView.this);
    mRegProgres.setTitle("Mendapatkan Data");
    mRegProgres.setMessage("Tunggu Sebentar... ");
    mRegProgres.setCanceledOnTouchOutside(false);
    mRegProgres.show();
    APIInterface apiInterface =
    APIClient.getClientAuthorize().create(APIInterface.class);
    Call<BangunanList> call =
    apiInterface.getSemuaBangunan(userSession.getUserToken(),
    userSession.getIdUser());
    call.enqueue(new Callback<BangunanList>() {
        @Override
        public void onResponse(Call<BangunanList> call,
        Response<BangunanList> response) {
            mRegProgres.dismiss();
            bangunan = response.body().getSemuaBangunan();
            viewAdapterBangunan = new
            RecyclerViewAdapterBangunan(BangunanView.this,
            bangunan,false);
            recyclerView.setAdapter(viewAdapterBangunan);
        }
    })
}
```

Kode diatas adalah *method* untuk mengambil data bangunan dengan memanggil API yang terdapat pada *class* APIClient. API yang digunakan adalah *getSemuaDataBangunan* untuk mengambil seluruh data bangunan yang terdapat pada basis data bangungedung. Apabila data berhasil didapatkan dan masuk kedalam android, data tersebut akan diatur oleh holder dan *recycleviewadapterbangunan*. Pengambilan data dapat dilakukan dengan menggunakan kunci *UserToken*, dan *IdUser*.

4.3.2.10 Pembuatan Halaman *BangunanViewHistory*

Pembuatan halaman ini memiliki kesamaan dengan *BangunanView*, pada tampilan atau struktur yang ditampilkan pada layar, perbedaan pada tampilan ini adalah kategori data yang ditampilkan. Data yang ditampilkan adalah data yang telah dimasukan oleh petugas survei. Pada tampilan ini digunakan untuk menghapus dan mengubah data yang telah dimasukan oleh petugas survei. Berikut kode yang terdapat pada *BangunanViewHistory.java* :

```
private void loadDataBangunan() {
    final UserSession userSession = new UserSession(BangunanViewHistory.this);
    mRegProgres.setTitle("Mendapatkan Data Dirimu");
    mRegProgres.setMessage("Tunggu Sebentar... ");
    mRegProgres.setCanceledOnTouchOutside(false);
    mRegProgres.show();
    APIInterface apiInterface =
    APIClient.getClientAuthorize().create(APIInterface.class);
    Call<BangunanList> call =
    apiInterface.getBangunanSurveyor(userSession.getUserToken(),
    userSession.getIdUser(), userSession.getUserId());
}
```

Pada kode diatas adalah *method* *loadDataBangun* untuk tampilan *BangunanViewHistory*. Perbedaan data yang ditampilkan adalah terdapat parameter *UserId* pada tampilan *BangunanViewHistory* sehingga data yang akan ditampilkan pada *list* atau holder adalah data yang dimasukan berdasarkan *UserId* petugas survei.

4.3.2.11 Pembuatan *Class* *UserSession*

Class *UserSession* ini dibuat sebagai kamus untuk kunci yang dibutuhkan untuk mendapatkan data, menghapus data, mengedit data, dan memasukan data. *UserSession* ini menyimpan data *user* berupa nama, *role*, *user_id*, *id_user* dan token. Sehingga jika sebuah parameter membutuhkan data *user* tersebut dapat digunakan dengan inisialisasi *class* *UserSession* memanggil *method* yang terdapat dalam *class*. Berikut kode yang terdapat pada *class* *UserSession* :

```
public class UserSession {
    private SharedPreferences preferences;
    private SharedPreferences.Editor editor;
    private Context mContext;
    private static final String PREFER_NAME = "SurveyBangunGedung";
    private static final String IS_USER_LOGIN = "LoginStatus";
    private static final String KEY_ID_USER = "id_user";
    private static final String KEY_USER_ROLE = "user_role";
    private static final String KEY_USER_ID = "user_id";
    private static final String KEY_USER_TOKEN = "token";
    private static final String KEY_USER_NAME = "nama";
```

Kode diatas adalah deklarasi variabel bertipe *string* dan diberikan sebuah *key* untuk pemanggilan data *user*. Setiap data *user* yang di deklarasikan memiliki *key* yang berbeda seperti *KEY_ID_USER* yang digunakan untuk *id_user*, *KEY_USER_NAME* yangi digunakan untuk nama.

```
public void setLoginSession(String id, String token, String nama, String userRole,
String user_id ) {
    editor.putBoolean(IS_USER_LOGIN, true);

    editor.putString(KEY_USER_ID, user_id);
    editor.putString(KEY_ID_USER, id);
    editor.putString(KEY_USER_TOKEN, token);
    editor.putString(KEY_USER_NAME, nama);
    editor.putString(KEY_USER_ROLE, userRole);
    editor.commit();
}
```

Kode diatas adalah *method* untuk melakukan menyimpan informasi *user* pada saat melakukan *login* di android.


```

public int getUserID() {
    return preferences.getInt(KEY_ID_USER, 0);
}
public int getUserRole() {return preferences.getInt(KEY_USER_ROLE, 0);
}
public String getUserToken() {
    return preferences.getString(KEY_USER_TOKEN, null);
}
public String getUserId() {
    return preferences.getString(KEY_USER_ID, null);
}
public String getIdUser() {
    return preferences.getString(KEY_ID_USER, "null");
}
public String getUsername() {
    return preferences.getString(KEY_USER_NAME, null);
}
public String getUserRoleStr() {
    return preferences.getString(KEY_USER_ROLE, null);
}
public boolean isUserLoggedIn() {
    return preferences.getBoolean(IS_USER_LOGIN, false);
}
}

```

Kode diatas adalah variabel yang menyimpan data *user* sehingga jika API memerlukan data yang menggunakan informasi *user* dapat menggunakan *method* *getIdUser* atau *getUserToken* tergantung dari parameter yang dibutuhkan.

4.3.2.12 Pembuatan *Class BaseActivity*

BaseActivity digunakan sebagai dasar fungsi atau method yang sering digunakan pada *class* lain agar *method* tersebut tidak perlu lagi dibuat pada *class* yang membutuhkan. Berikut kode yang terdapat pada *BaseActivity* :

```

Public class BaseActivity extends AppCompatActivity {
    private Boolean connStatus;
    protected ProgressDialog mRegProgres;
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

```

checkInternet();
mRegProgres = new ProgressDialog(this);
mRegProgres.setTitle("Getting Data");
mRegProgres.setMessage("Please Wait...");
mRegProgres.setCanceledOnTouchOutside(false);
}

```

Berdasarkan kode diatas adalah header dari *class* BaseActivity dan deklarasi variabel yang yang digunakan pada *class* tersebut. `mRegProgres = new ProgressDialog(this)` digunakan untuk mendeklarasikan bahwa `mRegProgres` adalah *widget* ProgressDialog. Dan pada baris kode `mRegProgres.setTitle("Getting Data")` yaitu memberikan pesan “Getting Data “ ketika menjalankan ProgressDialog.

```

private void checkInternet(){
    ConnectivityManager ConnectionManager =(ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo=ConnectionManager.getActiveNetworkInfo();
    if(networkInfo != null && networkInfo.isConnected()==true ) {
        connStatus = true;
    }
    else {
        connStatus = false;
        showToast("No Internet Connection");
    }
}
protected void showToast(String message){
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
}
}

```

Kode diatas adalah *method* checkInternet, *method* tersebut digunakan untuk mengecek koneksi internet, apabila internet dalam keadaan mati, maka android akan memberikan pesan “No Internet Connection”. Kode *method* showToast digunakan untuk memberikan pesan tanpa perlu mengetikan kode *Toast*.

4.3.2.13 Pembuatan Class Bangunan

Class Bangunan digunakan sebagai penyimpan data yang didapatkan oleh android dari basis data yang disediakan oleh *website*. Pada *class* bangunan menggunakan *implement* Parcelable. Parcelable tersebut adalah fitur yang berfungsi

untuk mengirimkan secara bersamaan kepada *intent* atau *activity*. Berikut ini adalah kode *class* Bangunan.java :

```
public class Bangunan implements Parcelable {
    @SerializedName("user_id")
    @Expose
    private String user_id;
    @SerializedName("nama")
    @Expose
    private String nama;
    @SerializedName("id_bangunan")
    @Expose
    private String id_bangunan;
    @SerializedName("id_reference")
    @Expose
    private int id_reference;
```

Kode diatas adalah header dari *class* Bangunan yang mengimplementasikan *parcelable*. Fungsi kode diatas adalah untuk mendeklarasikan variabel apa saja yang diperlukan untuk data yang akan diolah oleh android.

```
public Bangunan (String nama, String id_bangunan, int id_reference,String
fungsi_bangunan, String jenis_bangunan, String jenis_fungsi_bangunan, String
nama_departemen, String alamat_departemen, String no_ikmn, String no_hdno,
String telpon, String email, String date_input, String nama_bangunan, String
alamat_bangunan,String klasifikasi_bangunan, String jumlah_lantai_bangunan,
String luas_lantai_bangunan, String luas_basement, String ketinggian_bangunan,
String date_selesai, String nama_pemilik_tanah, String no_ipt,
String no_bkt, String jenis_kepemilikan_tanah, String alamat_tanah,
String luas_tanah, String data_peruntukan_resmi,
String kdb, String klb, String kdh, String ktb, String
nama_departemen_dulu, String alamat_departemen_dulu, String no_ikmn_dulu,
String no_hdno_dulu,String telpon_dulu, String email_dulu, String
no_imb_terdahulu, String no_islf_terdahulu, String latitude, String longitude){

    this.nama = nama;
    this.id_bangunan = id_bangunan;
    this.id_reference = id_reference;
    this.fungsi_bangunan = fungsi_bangunan;
    this.jenis_bangunan = jenis_bangunan;
```

Kode diatas adalah *method* Bangunan yang menggunakan parameter data yang akan digunakan oleh android. Data tersebut akan dideklarasikan pada baris `this.nama = nama` dan seterusnya.

```
public String getNama() {
    return nama;
}
public void setNama(String nama) {
    this.nama = nama;
}
public String getId_bangunan(){return id_bangunan;}
public void setId_bangunan (String id_bangunan) {this.id_bangunan =
id_bangunan;}
```

Kode diatas adalah variabel yang menyimpan setiap data yang telah didapatkan android dari basis data. Variabel tersebut menggunakan parameter sesuai dengan nama yang disimpan dalam bentuk String.

4.3.2.14 Pembuatan *Class* Logout

class ini dibuat untuk keluar dari akun dalam aplikasi survei bangun gedung di android. Berikut kode yang terdapat pada `logout.java` :

```
APIInterface                                apiInterface                                =
APIClient.getClientAuthorize().create(APIInterface.class);
Call<RequestResponse>                       call                                     =
apiInterface.logout(userSession.getIdUser(),userSession.getUserToken());
call.enqueue(new Callback<RequestResponse>() {
    @Override
    public void onResponse(Call<RequestResponse> call,
Response<RequestResponse> response) {
        try {
            if (response.isSuccessful()) {
                Intent i = new Intent(activity, Login.class);
                i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                activity.startActivity(i);
                userSession.logoutUser();
                activity.finish();
            } else
                Toast.makeText(activity, "Gagal Login",
                Toast.LENGTH_SHORT).show();
```

```
}catch (Exception e) {
```

Kode diatas berguna untuk memanggil API logout yang terdapat pada APIClient.

Kode tersebut berfungsi untuk keluar dari akun yang sudah login pada android.

Table 4.1 merupakan *table file* Java dan .XML beserta keterkaitan antara *file* Java dan XML tersebut yang digunakan dalam membuat program aplikasi Survei Bangun Gedung.

Table 4.1 File Java dan XML

Activity (Java)	Layout (XML)	Fungsi
Splashscreen.java	Activity_splashscreen.xml	Menampilkan splashscreen
Login.java	Activity_login.xml	Menampilkan halaman login
MainActiviy.Java	Activity_main.xml	Menampilkan <i>Maps</i> dan posisi titik bangunan
BangunView.java	Activity_bangunan_view.xml	Menampilkan seluruh data bangunan menggunakan menggunakan list Recycleview
BangunViewHistory.java	Activity_bangunan_view_history.xml	Menampilkan data bangunan yang telah di dimasukan petugas survei menggunakan menggunakan list Recycleview

DetailBangun.java	Activity_detail_bangun.xml	Menampilkan seluruh informasi setiap bangun gedung
Spinner_fungsi.java	Activity_spinner_fungsi.xml	Menampilkan form yang digunakan untuk pendataan survei bangun gedung

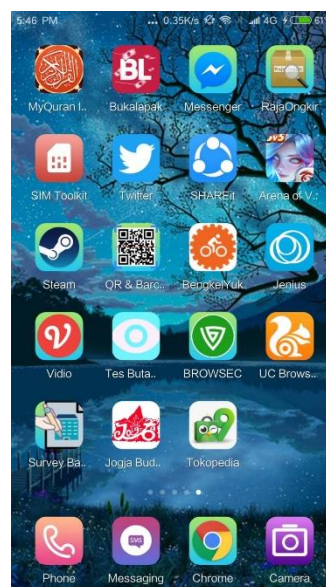
4.4 Uji Coba

Berikut uji coba yang dilakukan penulis dalam pengujian aplikasi :

4.4.1 Uji Coba Aplikasi

Pada skenario uji coba aplikasi, pengguna diminta untuk menjalankan aplikasi. Berikut skenario uji coba :

1. Pengguna membuka aplikasi dengan menekan icon pada smartphone seperti gambar 4.18



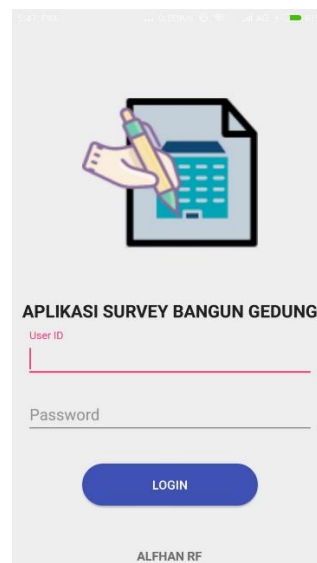
Gambar 4.18 Tampilan Icon Aplikasi Aplikasi

2. Kemudian akan tampil *Splashscreen* aplikasi seperti gambar 4.19



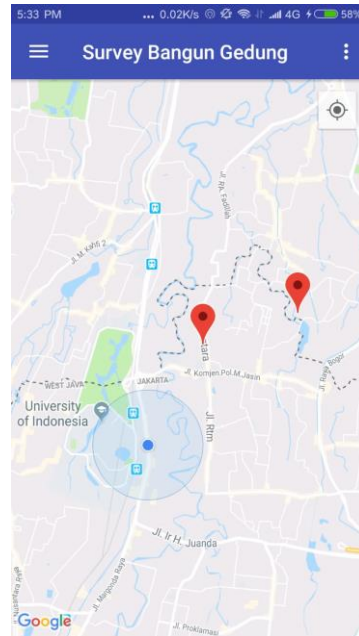
Gambar 4.19 Tampilan Splashscreen Aplikasi

3. Setelah tampilan splashscreen, maka muncul tampilan *login* seperti gambar 4.20



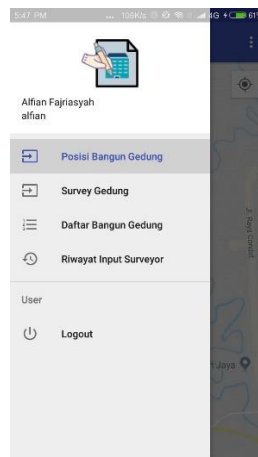
Gambar 4.20 Tampilan *Login* Aplikasi

4. Saat *user* berhasil *login* akan muncul menu utama yaitu *maps* dan titik bangun gedung yang telah di survei seperti pada gambar 4.21.

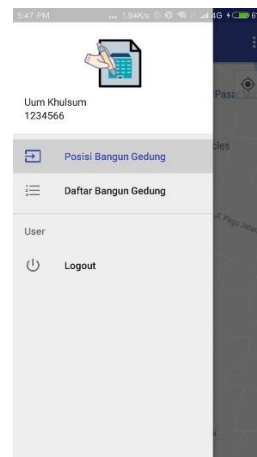


Gambar 4.21 Tampilan Utama Aplikasi

5. Apabila *user* menekan menu yang terdapat pada sebelah kiri atas maka akan muncul tampilan menu seperti pada gambar 4.22(A) dan 4.22(B).



(A)



(B)

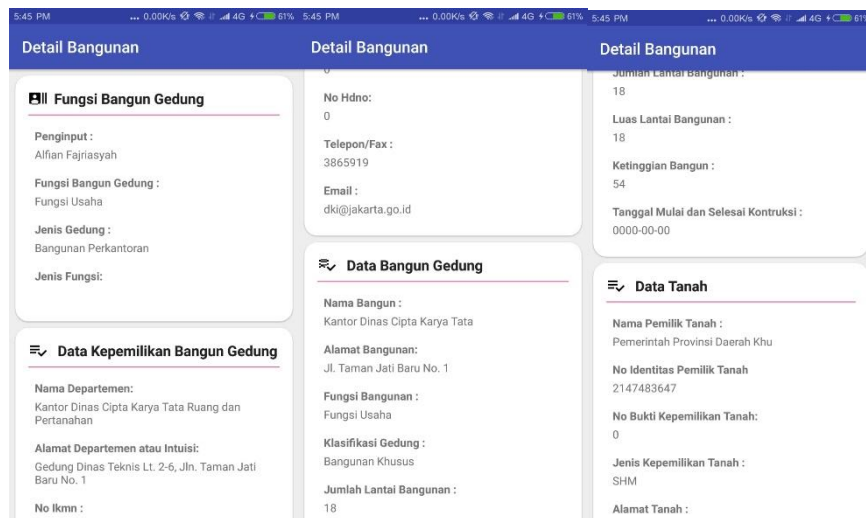
Gambar 4.22 (A) *Menu Navigasi Role Petugas Survei*, (B) *Menu Navigasi Role User Eksternal*

6. Apabila *user* memilih titik gedung pada *maps* maka akan memberikan informasi data nama bangunan gedung seperti pada gambar 4.23.



Gambar 4.23 Tampilan Utama Saat Titik Pada *Maps* Ditekan

7. Apabila *user* menekan pada titik bangun gedung yang sama maka akan memberikan tampilan informasi seperti pada gambar 4.24.



Gambar 4.24 Tampilan Detail Bangun Gedung

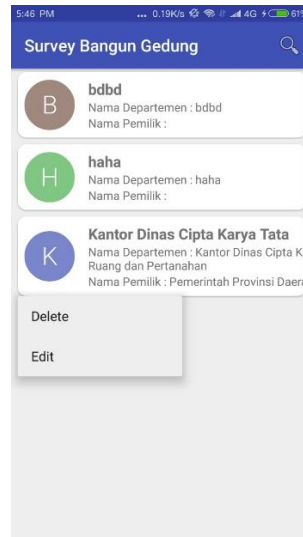
8. Tampilan apabila *user* memilih tampilan survey gedung pada menu seperti pada gambar 4.25.

Gambar 4.25 Tampilan Form Survey

9. Tampilan apabila *user* memilih *menu* daftar bangun gedung, seperti pada gambar 4.26.

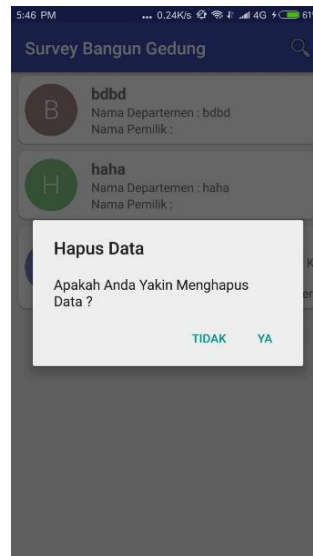
Gambar 4.26 Tampilan *Menu* Daftar Bangun Gedung

10. Tampilan apabila *user* memilih *menu* Riwayat Input Survei, seperti pada gambar 4.27.



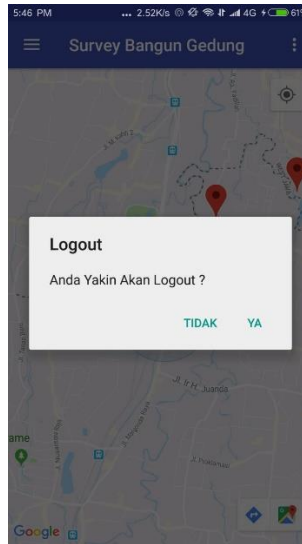
Gambar 4.27 Tampilan *Menu* Riwayat Input Petugas Survei

11. Tampilan apabila *user* menekan *menu* delete pada saat ingin menghapus data yang telah di masukan seperti pada gambar 4.28.



Gambar 4.28 Tampilan *Menu* Riwayat Input Petugas Survei Hapus Data

12. Tampilan apabila *user* ingin *logout* pada dari aplikasi survei bangun gedung seperti pada gambar 4.29.

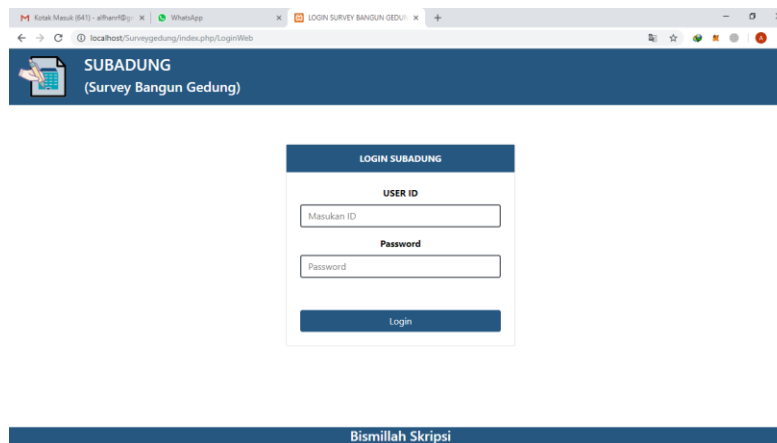


Gambar 4.29 Tampilan *Menu Logout*

4.4.2 Uji Coba Website

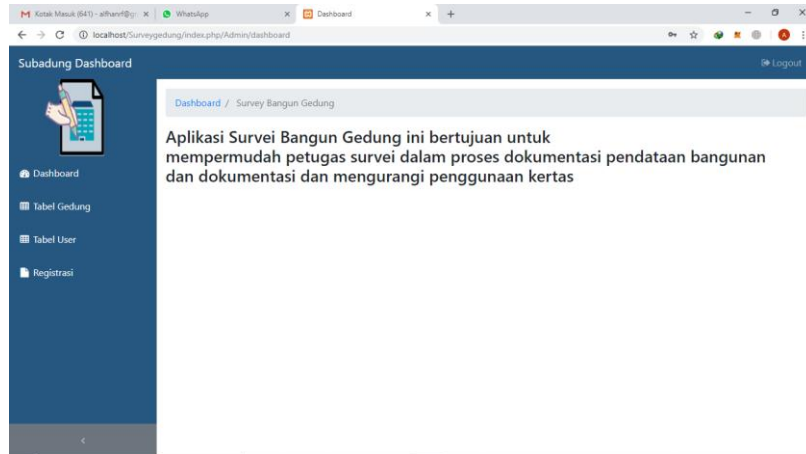
Pada skenario uji coba aplikasi, pengguna diminta untuk menjalankan aplikasi. Berikut skenario uji coba :

1. Membuka halaman login admin pada website dengan memasukan alamat *localhost* website



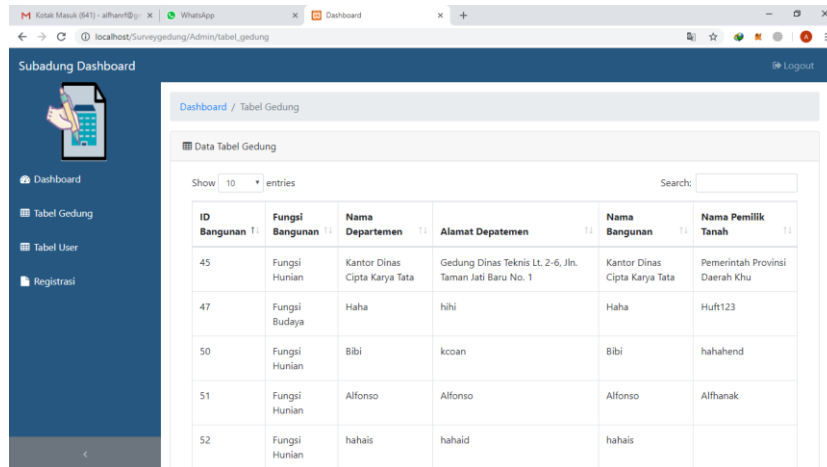
Gambar 4.30 Tampilan Login Admin Website

2. Kemudian akan tampil menu dashboard website survei bangun gedung dan menu yang terdapat pada sebelah kiri.



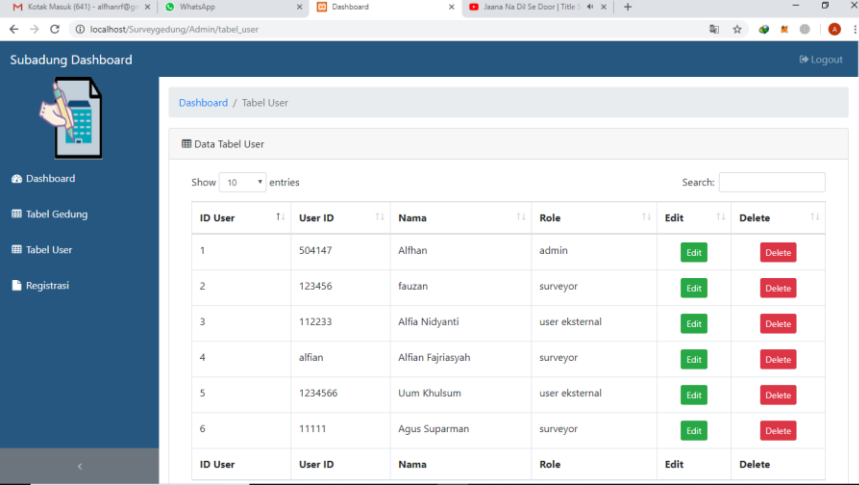
Gambar 4.31 Tampilan Dashboard Admin pada Website

3. Tampilan apabila admin memilih menu tabel gedung pada website survei bangun gedung dengan menekan menu Tabel Gedung.



Gambar 4.32 Tampilan Tabel Gedung Website

4. Tampilan apabila admin memilih menu tabel user pada website survei bangun gedung dengan menekan menu Tabel User.



Subadung Dashboard

Dashboard / Tabel User

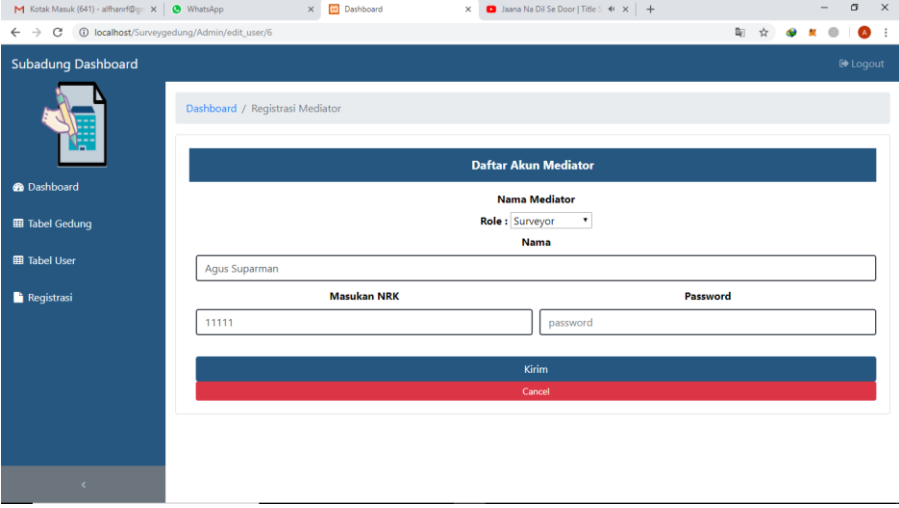
Data Tabel User

Show 10 entries

ID User	User ID	Nama	Role	Edit	Delete
1	504147	Alfhan	admin	Edit	Delete
2	123456	fauzan	surveyor	Edit	Delete
3	112233	Alfia Nidyanti	user eksternal	Edit	Delete
4	alfian	Alfian Fajriasyah	surveyor	Edit	Delete
5	1234566	Uum Khulsum	user eksternal	Edit	Delete
6	11111	Agus Suparman	surveyor	Edit	Delete

Gambar 4.33 Tampilan Tabel User Admin pada Website

5. Tampilan apabila admin memilih aksi edit user pada menu tabel user di website survei bangun gedung dengan menekan menu Table User.



Subadung Dashboard

Dashboard / Registrasi Mediator

Daftar Akun Mediator

Nama Mediator

Role : Surveyor

Nama

Agus Suparman

Masukan NRK

11111

Password

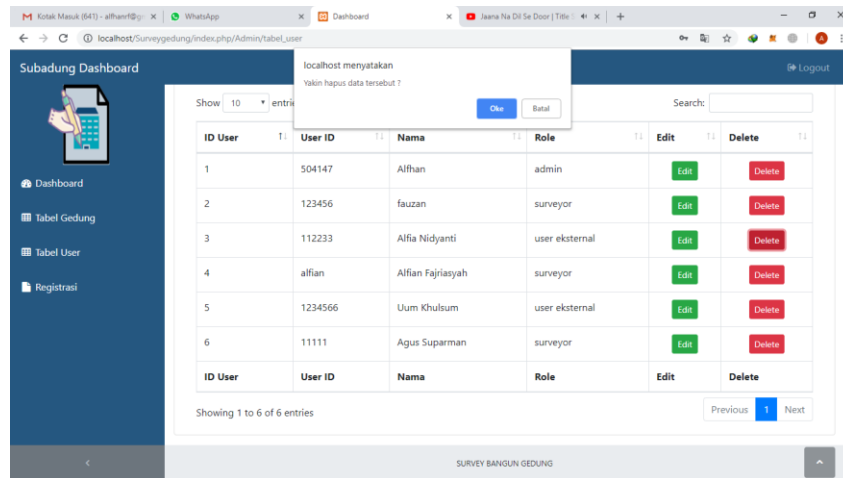
password

Kirim

Cancel

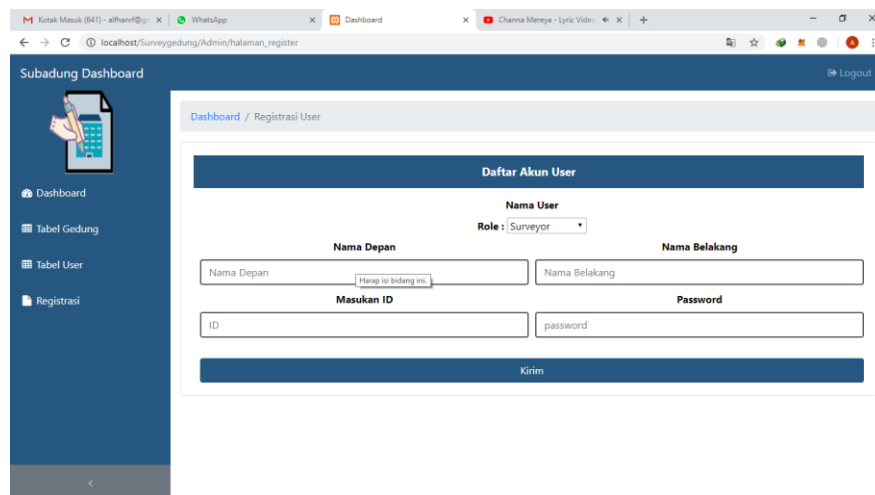
Gambar 4.34 Tampilan Edit User Admin Website

6. Tampilan apabila admin memilih aksi hapus user pada menu tabel user di website survei bangun gedung dengan menekan menu Table User.



Gambar 4.33 Tampilan Hapus User Website

7. Tampilan apabila admin memilih menu Registrasi pada website survei bangun gedung dengan menekan menu Registrasi.



Gambar 4.34 Tampilan Register User Website

4.4.3 Uji Coba dengan Ponsel Berbasis Android

Berikut uji coba pada beberapa *smartphone android* dengan spesifikasi perangkat yang berbeda :

Tabel 4.2 Spesifikasi Perangkat Uji Coba

No.	Nama Perangkat	Spesifikasi Perangkat Keras	Spesifikasi Perangkat Lunak
1.	Oppo A57	RAM 3GB Processore Octa-core 1.5 Ghz Layar 5.2 inches	Android OS Marshmallow v6.0
2.	Samsung Note FE	RAM 4 GB Processor Snapdragon 845 Layar 5.7 inches	Android OS Oreo v8.1
3.	Xiaomi Redmi Note 3 Pro	RAM 3 GB Processore Hexa-core 1.5 Ghz Layar 5.5 inches	Android OS Lollipop v5.1
4.	Xiaomi Redmi 4x	RAM 3 GB Processor Octa-Core 1.4 Ghz Layar 4.5 inches	Android OS Lollipop v5.1
5.	Xiaomi Redmi 4 Prime	RAM 3 GB Processore Octa-core 1.5 Ghz GHzLayar 5.0 inches	Android OS Marshmallow v6.0.1

Tabel 4.3 Hasil Uji Coba Pada Perangkat

No.	Fitur	Skenario Uji	Hasil yang di harapkan	Hasil Pengamatan
1.	Button Login	Pengguna menekan button <i>login</i>	Pengguna masuk kedalam tampilan main activity apabila userid dan password benar. Dan apabila terdapat kesalahn diantara keduanya maka terdatap pesan salah	Ketika dijalankan dengan menggunakan 5 perangkat tersebut, semua fitur berjalan sesuai harapan
2.	Menu Posisi Bangun Gedung	Pengguna menekan menu posisi bangun gedung pada aplikasi	Pengguna masuk ke dalam main activity yaitu tampilan berupa <i>maps</i> dan <i>marker</i> posisi bangun gedung.	Ketika dijalankan dengan menggunakan 5 perangkat tersebut, semua fitur berjalan sesuai harapan

No.	Fitur	Skenario Uji	Hasil yang di harapkan	Hasil Pengamatan
3.	Menu survey gedung	Petugas survei memilih menu survei gedung	Pengguna petugas survei dapat membuka form yang disediakan untuk pendataan bangun gedung	Ketika dijalankan dengan menggunakan 5 perangkat tersebut, semua fitur berjalan sesuai harapan
4.	Menu Daftar Bangun Gedung	Pengguna memilih menu daftar bangun gedung	Pengguna masuk ke tampilan data seluruh bangunan berupa list	Ketika dijalankan dengan menggunakan 5 perangkat tersebut, semua fitur berjalan sesuai harapan
5.	Menu Riwayat Input Petugas Survei	Petugas survei memilih menu riwayat input Petugas Survei	Pengguna petugas survei dapat membuka tampilan yang mengampilkan data apa saja yang tela di masukan oleh petugas survei ke basis data	Ketika dijalankan dengan menggunakan 5 perangkat tersebut, semua fitur berjalan sesuai harapan
6.	Marker posisi bangun gedung	Pengguna menekan posisi bangun gedung pada <i>maps</i>	Pengguna dapat melihat informasi tentang bangun gedung	Ketika dijalankan dengan menggunakan 5 perangkat tersebut, semua fitur berjalan sesuai harapan
7.	List Bangun Gedung	Pengguna memilih nama bangun gedung yang terdapat pada list	Pengguna dapat melihat informasi tentang bangun gedung	Ketika dijalankan dengan menggunakan 5 perangkat tersebut, semua fitur berjalan sesuai harapan
8.	<i>Button Logout</i>	Pengguna menekan <i>button logut</i>	Pengguna keluar dari akun yang sudah login pada aplikasi	Ketika dijalankan dengan menggunakan 5 perangkat tersebut, semua fitur berjalan sesuai harapan

4.4.4 Uji Coba Black Box

Pengujian aplikasi ini menggunakan teknik pengujian *black box*. Pengujian ini memperhatikan fungsional dari sistem yang dibangun. Pada tahap ini akan dilakukan pengujian pada fungsional dari fitur-fitur seperti menguji tombol-tombol apda aplikasi yang dibuat berfungsi sesuai dengan yang diharapkan. Berikut ini adalah proses sistemn aplikasi yang akan dilakukan pengujian. Table uji coba sistem dapat dilihat pada tabel 4.4.

Tabel 4.4 Uji Coba Sistem

Item Pengujian	Hasil yang diharapkan	Hasil Pengujian	Kesimpulan
<i>Install</i> APK	Proses instalasi terpasang pada <i>smartphone</i> dengan baik	Aplikasi terpasang dan muncul icon aplikasi pada <i>smartphone</i>	Valid
Menjalankan aplikasi yang terpasang	Aplikasi berjalan dan dapat terbuka dengan baik	Aplikasi dapat berjalan dengan baik tanpa <i>error</i>	Valid
Menu aplikasi	Menampilkan pilihan <i>menu</i> yang terdapat pada aplikasi, posisi bangun gedung, survey bangun gedung, daftar bangun gedung, riwayat input Petugas Survei	Aplikasi dapat menampilkan tampilan menu utama beserta pilihan menu yang ada di dalamnya	Valid
Menu Posisi Bangun Gedung	Menampilkan <i>maps</i> dan titik posisi bangun gedung	Pada saat menekan tombol menu posisi bangunan, aplikasi masuk ke halaman main activity	Valid
Menu survey gedung	Menampilkan form layout untuk pendataan bangun gedung,	Pada saat menekan tombol menu posisi bangunan, aplikasi masuk ke halaman form survei	Valid
Menu Daftar Bangun Gedung	Menampilkan data bangun gedung berupa list	Aplikasi dapat menampilkan list data bangun gedung	Valid
Menu Riwayat Input Petugas Survei	Menampilkan data bangun gedung yang telah di input kan petugas survei yang bersangkutan	Aplikasi menampilkan list data bangun gedung yang telah di masukan petugas survei	Valid
Tombol Hapus	Menghapus data pada tampilan menu riwayat petugas survei	Pada saat menekan tombol hapus, maka akan muncul dialogbox untuk hapus data dan data terhapus apabila dialog box tersebut terpilih “Ya” dan kembali pada tampilan menu riwayat petugas survei apabila	Valid

Item Pengujian	Hasil yang diharapkan	Hasil Pengujian	Kesimpulan
		memilih “Tidak” pada dialogbox	
Tombol Edit	Menampilkan Form Edit dengan mengambil kembali dan mengisi form dengan data sebelumnya	Membuka tampilan form survei dan form terisi dengan data sebelum berubah.	Valid
Tombol update data pada form edit.	Diarahkan kembali ke halaman utama dan data telah berubah.	Membuka tampilan Posisi Bangun gedung dan data telah berubah	Valid
Tombol Input Data pada form survey gedung.	Diarahkan kembali ke halaman utama dan data telah masuk kedalam basis data	Membuka tampilan posisi bangun gedung dan data telah ditambahkan serta <i>marker</i> bertambah sesuai dengan titik yang telah dimasukan petugas survei.	Valid
Tombol Logout	Keluar dari akun yang telah login dan menampilkan tampilan login	Keluar dari akun dan menampilkan tampilan login	Valid
Marker posisi bangun gedung	Ketika pengguna menekan marker berdasarkan akan muncul nama bangunan, dan apabila pengguna menekan ditempat yang sama untuk kedua kalinya maka akan tampil detail bangun dan menampilkan data detail bangunan	Menampilkan nama bangunan setelah marker ditekan satu kali, membuka tampilan detail bangun dan menampilkan data tentang bangunan yang dipilih	valid
List pada Menu Riwayat Input Survei	Apabila petugas survei menekan salah satu list pada data maka akan menampilkan detail bangunan gedung sesuai dengan data yang dipilih, apabila petugas menekan dan menahan	Menampilkan tampilan detail bangun gedung dan mengeluarkan popup apabila list ditekan dan ditahan beberapa waktu	valid

Item Pengujian	Hasil yang diharapkan	Hasil Pengujian	Kesimpulan
	salah satu list maka akan muncul menu edit dan hapus pada list.		
List pada Daftar Bangunan	Saat pengguna menekan salah satu list pada data maka akan menampilkan detail bangunan gedung sesuai dengan data yang dipilih.	Menampilkan tampilan detail bangun gedung dan menampilkan data bangunan yang dipilih.	valid
Login Website	Saat admin login kedalam website, maka akan menampilkan dashboard aplikasi survei bangun gedung.	Menampilkan dashboard survei bangun gedung.	valid
Lihat Data Bangunan pada Website	Saat admin memilih menu tabel bangunan maka akan menampilkan view tabel bangunan dan menampilkan tabel bangunan	Menampilkan tampilan tabel bangunan dan menampilkan tabel yang berisi data bangunan yang telah dimasukan petugas survei	valid
Lihat Tabel User pada Website	Saat admin memilih menu tabel user maka akan menampilkan view tabel bangunan dan menampilkan tabel bangunan, serta terdapat 2 action edit dan hapus user.	Menampilkan tampilan tabel user dan menampilkan tabel yang berisi data user dan menampilkan 2 action yang terdapat pada tabel	valid
Edit Data User pada Website	Saat admin memilih aksi edit pada tabel user maka akan menampilkan form edit dan dapat mengedit user	Menampilkan form edit user dan data berhasil di edit.	valid
Hapus Data User pada Website	Saat admin memilih aksi hapus pada tabel user maka akan menampilkan peringatan dan data berhasil dihapus apabila peringatan tersebut di validasi	Data user berhasil dihapus.	valid
Registrasi User pada Website	Saat dmin memilih menu registrasi user maka akan menampilkan form untuk registrasi user dan juga form	Menampilkan form registrasi dan auto generate password dapat berjalan.	valid

Item Pengujian	Hasil yang diharapkan	Hasil Pengujian	Kesimpulan
	password auto generate dengan 3 huruf depan role + user id.		