# LAMPIRAN

```php
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable
{
    /** @use HasFactory<\Database\Factories\UserFactory> */
    use HasFactory, Notifiable,HasApiTokens,HasRoles;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
    protected $fillable = [
        'name',
        'email',
        'no',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var list<string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
```

```
        ];
    }
}
```

```
<?php

namespace App\Models;

use
Illuminate\Database\Eloquent\M
odel;
use
Illuminate\Database\Eloquent\F
actories\HasFactory;

class Media extends Model
{
    use HasFactory;
    protected $fillable = [
        'judul',
        'deskripsi',
        'category_id',
        'harga',
        'tipe',
        'kamar_tidur',
        'kamar_mandi',
        'luas_bangunan',
        'luas_tanah',
        'subsidi',
        'status',
        'developer',
        'foto',
        'video',
    ];

    public function category()
    {
        return $this-
>belongsTo(Category::class);
    }
    public function
getFotoAttribute($value)
{
    return $value ?
url('storage/' . $value) :
null;
}
}
```

```
<?php

namespace App\Models;

use
Illuminate\Database\Eloquent\M
odel;
use
Illuminate\Database\Eloquent\F
actories\HasFactory;

class Category extends Model
{
    use HasFactory;
    protected $fillable =
['kategori','denah','user_id']
;

    public function media()
```

```
    {
        return $this->hasMany(Media::class);
    }

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Biodata extends Model
{
    protected $fillable = [
        'nama_pemohon',
        'tempat_tanggal_lahir',
        'jenis_kelamin',
        'status_pernikahan',
        'pendidikan_terakhir',
        'agama',
        'jenis_nasabah',
        'alamat',
        'desa_kelurahan',
        'kecamatan',
        'kota',
        'kodepos',
        'no_hp',
        'no_telpon',
        'pekerjaan',
        'jenis_perusahaan',
        'user_id',
        'ktp_pemohon',
        'ktp_pasangan',
        'npwp',
        'sk_tidak_memiliki_rumah',
        'sk_tidak_punya_subsidi',
        'slip_gaji',
        'rekening',
        'sk_lama',
        'foto_pemohon1',
        'foto_pemohon2',
    ];

    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }
}
```

```
<?php
```

```
namespace
App\Http\Controllers;


use App\Models\Biodata;

use Illuminate\Http\Request;

use
Illuminate\Support\Facades\Sto
rage;


class BiodataController
extends Controller

{

public function index()

{

    $user = auth()->user(); //
Ambil user yang sedang login



    if ($user->role ===
'admin') {

        // Admin bisa melihat
semua data

        $biodatas =
Biodata::all();

    } else {

        // Selain admin hanya
bisa melihat datanya sendiri

        $biodatas =
Biodata::where('user_id',
$user->id)->get();

    }



    return response()-
>json($biodatas);

}
```

```
public function store(Request
$request)

{

    $data = $request-
>validate([

        'nama_pemohon' =>
'required|string|max:255',

        'tempat_tanggal_lahir'
=> 'required|string|max:255',

        'jenis_kelamin' =>
'required|string|max:255',

        'status_pernikahan' =>
'required|string|max:255',

        'pendidikan_terakhir'
=> 'required|string|max:255',

        'agama' =>
'required|string|max:255',

        'desa_kelurahan' =>
'required|string|max:255',

        'kecamatan' =>
'required|string|max:255',

        'kota' =>
'required|string|max:255',

        'kodepos' =>
'required|string|max:255',

        'alamat' =>
'required|string',

        'no_hp' =>
'required|string|max:255',

        'no_telpon' =>
'required|string|max:255',

        'pekerjaan' =>
'required|string|max:255',

        'jenis_perusahaan' =>
'required|string|max:255',

        'jenis_nasabah' =>
'required|string|max:255',

        'ktp_pemohon' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',
```

```
        'ktp_pasangan' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

        'npwp' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',


'sk_tidak_memiliki_rumah' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',


'sk_tidak_punya_subsidi' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

        'slip_gaji' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

        'rekening' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

        'sk_lama' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

        'foto_pemohon1' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

        'foto_pemohon2' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

    ]);


    // Tambahkan user_id dari
user yang login

    $data['user_id'] = auth()-
>id();


    // Upload file jika ada

    foreach ($data as $key =>
$value) {

        if ($request-
>hasFile($key)) {

            $folder =
"uploads/biodata/{$key}";

            $data[$key] =
$request->file($key)-
>store($folder, 'public');

        }

    }


    $biodata =
Biodata::create($data);


    return response()-
>json($biodata);

}



    public function
show(Biodata $biodata)

    {

        return response()-
>json($biodata);

    }


    public function
update(Request $request,
Biodata $biodata)

    {

        $data = $request-
>validate([

        'nama_pemohon' =>
'required|string|max:255',

        'tempat_tanggal_lahir'
=> 'required|string|max:255',

        'jenis_kelamin' =>
'required|string|max:255',

        'status_pernikahan' =>
'required|string|max:255',
```

```
        'pendidikan_terakhir'
=> 'required|string|max:255',

        'agama' =>
'required|string|max:255',

        'desa_kelurahan' =>
'required|string|max:255',

        'kecamatan' =>
'required|string|max:255',

        'kota' =>
'required|string|max:255',

        'kodepos' =>
'required|string|max:255',

        'alamat' =>
'required|string',

        'no_hp' =>
'required|string|max:255',

        'no_telpon' =>
'required|string|max:255',

        'pekerjaan' =>
'required|string|max:255',

        'jenis_perusahaan' =>
'required|string|max:255',

        'jenis_nasabah' =>
'required|string|max:255',

            'ktp_pemohon' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

            'ktp_pasangan' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

            'npwp' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',


'sk_tidak_memiliki_rumah' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',


'sk_tidak_punya_subsidi' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

            'slip_gaji' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

            'rekening' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

            'sk_lama' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

            'foto_pemohon1' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

            'foto_pemohon2' =>
'nullable|file|mimes:jpg,jpeg,
png,pdf|max:5120',

        ]);


foreach ($data as $key =>
$value) {

    if ($request-
>hasFile($key)) {

        // Hapus file lama
jika ada

        if ($biodata->$key) {


Storage::disk('public')-
>delete($biodata->$key);

        }


        $folder =
"uploads/biodata/{$key}";

        $data[$key] =
$request->file($key)-
>store($folder, 'public');

    }

}
```

```
        $biodata->update($data);

        return response()->json($biodata);
    }

    public function destroy(Biodata $biodata)
    {
        // Hapus semua file
        foreach ($biodata->getAttributes() as $key => $value) {
            if (
                in_array($key, [
                    'ktp_pemohon',
                    'ktp_pasangan',
                    'npwp',
                    'sk_tidak_memiliki_rumah',
                    'sk_tidak_punya_subsidi',
                    'slip_gaji',
                    'rekening',
                    'sk_lama',
                    'foto_pemohon1',
                    'foto_pemohon2'
                ]) && $value
            ) {
                Storage::disk('public')->delete($value);
            }
        }

        $biodata->delete();

        return response()->json(['message' => 'Biodata deleted']);
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Models\Category;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Auth;

class CategoryController extends Controller
{
public function index()
{
    $categories = Category::with(['media', 'user'])->get();
```

```
    return response()->json($categories, 200);
}

public function store(Request $request)
{
    $request->validate([
        'kategori' => 'required|string|max:255',
        'denah' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
    ]);

    $user = Auth::user();

    if (!$user) {
        return response()->json(['message' => 'Unauthenticated'], 401);
    }

    // Simpan category terlebih dahulu
    $category = new Category();
    $category->kategori = $request->kategori;
    $category->user_id = $user->id;

    // Simpan denah jika ada
    if ($request->hasFile('denah')) {
        $denahPath = $request->file('denah')->store('uploads/denah', 'public');
        $category->denah = $denahPath; // Pastikan kolom 'denah' ada di tabel categories
    }

    $category->save();

    return response()->json([
        'message' => 'Kategori berhasil ditambahkan',
        'data' => $category,
    ], 201);
}

public function show(Category $category)
{
    return response()->json($category->load(['media', 'user']), 200);
}

public function update(Request $request, Category $category)
{
    $request->validate([
        'kategori' => 'required|string|max:255',
        'denah' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
    ]);
```

```
    $category->kategori = $request->kategori;


    // Jika ada file baru yang diunggah, simpan dan hapus file lama jika ada
    if ($request->hasFile('denah')) {
        // Hapus file lama jika ada
        if ($category->denah && Storage::disk('public')->exists($category->denah)) {

Storage::disk('public')->delete($category->denah);
        }


        $denahPath = $request->file('denah')->store('uploads/denah', 'public');
        $category->denah = $denahPath;
    }


    $category->save();


    return response()->json([
        'message' => 'Kategori berhasil diperbarui',
        'data' => $category,
        'denah_url' => $category->denah ? asset('storage/' . $category->denah) : null,
    ], 200);
}

    public function destroy(Category $category)
    {
        if ($category->media()->exists()) {
            return response()->json(['message' => 'Kategori memiliki data media dan tidak bisa dihapus'], 400);
        }
                if (!empty($category->denah) && Storage::disk('public')->exists($category->denah)) {

Storage::disk('public')->delete($category->denah);
        }


        $category->delete();
        return response()->json(['message' => 'Kategori berhasil dihapus'], 200);
    }
}


<?php


namespace App\Http\Controllers;


use App\Models\Media;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Storage;
```

```
class MediaController extends
Controller
{
    public function index()
    {
        return response()-
>json(Media::with('category')-
>get(), 200);
    }

    public function
store(Request $request)
    {
        $request->validate([
            'judul' =>
'required|string|max:255',
            'deskripsi' =>
'required|string',
            'category_id' =>
'required|exists:categories,id
',
            'harga' =>
'required',
            'tipe' =>
'required|string',
            'kamar_tidur' =>
'required',
            'kamar_mandi' =>
'required',
            'luas_bangunan' =>
'required',
            'luas_tanah' =>
'required',
            'subsidi' =>
'required',
            'status' =>
'required|string',
            'developer' =>
'required|string',
            'foto' =>
'nullable|image|mimes:jpeg,png
,jpg|max:10048',
            'video' =>
'nullable|mimes:mp4,mov,avi|ma
x:200240',


        ]);


        $data = $request-
>all();

        if ($request-
>hasFile('foto')) {
            $data['foto'] =
$request->file('foto')-
>store('uploads/foto',
'public');
        }

        if ($request-
>hasFile('video')) {
            $data['video'] =
$request->file('video')-
>store('uploads/video',
'public');
        }


        $media =
Media::create($data);
        return response()-
>json($media, 201);
```

```
    }


    public function show(Media
$media)
    {
        return response()-
>json($media-
>load('category'), 200);
    }


    public function
update(Request $request, Media
$media)
    {
        $request->validate([
            'judul' =>
'sometimes|required|string|max
:255',
            'deskripsi' =>
'sometimes|required|string',
            'category_id' =>
'sometimes|required|exists:cat
egories,id',
            'harga' =>
'required',
            'tipe' =>
'required|string',
            'kamar_tidur' =>
'required',
            'kamar_mandi' =>
'required',
            'luas_bangunan' =>
'required',
            'luas_tanah' =>
'required',
            'subsidi' =>
'required',
            'status' =>
'required|string',
            'developer' =>
'required|string',
            'foto' =>
'nullable|image|mimes:jpeg,png
,jpg|max:10048',
            'video' =>
'nullable|mimes:mp4,mov,avi|ma
x:200240',


        ]);


        $data = $request-
>except(['foto', 'video']); //
Ambil semua data kecuali
foto/video


        if ($request-
>hasFile('foto')) {
            if ($media->foto)
Storage::disk('public')-
>delete($media->foto);
            $data['foto'] =
$request->file('foto')-
>store('uploads/foto',
'public');
        }


        if ($request-
>hasFile('video')) {
            if ($media->video)
Storage::disk('public')-
>delete($media->video);
            $data['video'] =
$request->file('video')-
>store('uploads/video',
'public');
        }
```

```
        $media->update($data);

        return response()->json($media, 200);
    }

    public function destroy(Media $media)
    {
        // Periksa apakah file ada sebelum menghapus
        if (!empty($media->foto) && Storage::disk('public')->exists($media->foto)) {
            Storage::disk('public')->delete($media->foto);
        }

        if (!empty($media->video) && Storage::disk('public')->exists($media->video)) {
            Storage::disk('public')->delete($media->video);
        }

        $media->delete();

        return response()->json(['message' => 'Data deleted successfully'], 200);
    }
}
```

```
<?php

use Illuminate\Http\Request;
use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\MediaController;
use App\Http\Controllers\CategoryController;
use App\Http\Controllers\BiodataController;
use Illuminate\Support\Facades\Route;

Route::middleware(['auth:sanctum'])->get('/user', function (Request $request) {
```

```
    return response()->json([
        'user' => [
            'id' => $request->user()->id,
            'name' => $request->user()->name,
            'email' => $request->user()->email,
            'no' => $request->user()->no,
            'roles' => $request->user()->getRoleNames()->toArray() // Ubah ke array
        ]
    ]);
});

Route::middleware('auth:sanctum')->put('/profile', [ProfileController::class, 'update']);


Route::post('/register', [RegisteredUserController::class, 'store'])
    ->middleware('guest')
    ->name('register');


Route::post('/login', [AuthenticatedSessionController::class, 'store'])
    ->middleware('guest')
    ->name('login');


Route::post('/forgot-password', [PasswordResetLinkController::class, 'store'])
    ->middleware('guest')
    ->name('password.email');


Route::post('/reset-password', [NewPasswordController::class, 'store'])
    ->middleware('guest')
    ->name('password.store');


Route::get('/verify-email/{id}/{hash}', VerifyEmailController::class)
    ->middleware(['auth', 'signed', 'throttle:6,1'])
    ->name('verification.verify');


Route::post('/email/verification-notification', [EmailVerificationNotificationController::class, 'store'])
    ->middleware(['auth', 'throttle:6,1'])
    ->name('verification.send');


Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])
    ->middleware('auth')
    ->name('logout');

// Routes untuk kategori

Route::get('/categories', [CategoryController::class, 'index']);

Route::post('/categories', [CategoryController::class, 'store']);
```

```
Route::get('/categories/{category}', [CategoryController::class, 'show']);

Route::put('/categories/{category}', [CategoryController::class, 'update']);

Route::delete('/categories/{category}', [CategoryController::class, 'destroy']);


// Routes untuk media

Route::get('/media', [MediaController::class, 'index']);

Route::post('/media', [MediaController::class, 'store']);

Route::get('/media/{media}', [MediaController::class, 'show']);

Route::put('/media/{media}', [MediaController::class, 'update']);

Route::delete('/media/{media}', [MediaController::class, 'destroy']);


// List semua biodata

Route::get('/biodata', [BiodataController::class, 'index'])->name('biodata.index');


// Tampilkan detail satu biodata

Route::get('/biodata/{biodata}', [BiodataController::class, 'show'])->name('biodata.show');


// Simpan data baru

Route::post('/biodata', [BiodataController::class, 'store'])->name('biodata.store');


// Perbarui data

Route::post('/biodata/{biodata}', [BiodataController::class, 'update'])->name('biodata.update');


// Hapus data

Route::delete('/biodata/{biodata}', [BiodataController::class, 'destroy'])->name('biodata.destroy');


import axios from 'axios';

import AsyncStorage from '@react-native-async-storage/async-storage';


const API_URL = 'https://rumah.proyekkk.com/api';

// const API_URL = 'http://192.168.1.7/api'; // Jika ingin tes di lokal


// Konfigurasi instance axios

const api = axios.create({

    baseURL: API_URL,

    headers: {

        'Content-Type': 'application/json',
```

```
        'Accept': 'application/json'
    }
});


// Interceptor untuk menambahkan token otomatis di setiap request
api.interceptors.response.use(response => {
    if (response.data.token) {
        AsyncStorage.setItem('userToken', response.data.token);
        api.defaults.headers.common['Authorization'] = Bearer ${response.data.token};
    }
    return response;
}, error => Promise.reject(error));


// 🔹 Registrasi
export const register = async (data) => {
    try {
        const response = await api.post('/register', data);
        return response.data;
    } catch (error) {
        throw error.response?.data || { message: 'Registrasi gagal!' };
    }
};

// 🔹 Login
export const login = async ({ email, password }) => {
  const response = await api.post('/login', { email, password });
  const { token } = response.data;
  await AsyncStorage.setItem('token', token);
  return response.data;
};

// 🔹 Logout
export const logout = async () => {
    try {
        const token = await AsyncStorage.getItem('userToken');
        if (!token) throw { message: 'Tidak ada token untuk logout' };

        await api.post('/logout', {}, { headers: { Authorization: Bearer ${token} } });

        console.log('🔒 Token dihapus dari AsyncStorage');
        await AsyncStorage.removeItem('userToken'); // Hapus token setelah logout sukses
```

```
        return { message:
'Logout berhasil' };
    } catch (error) {
        throw
error.response?.data || {
message: 'Logout gagal!' };
    }
};



// 🔷 Mendapatkan data user
export const getUser = async
() => {
  try {
    const token = await
AsyncStorage.getItem('userToke
n');
    if (!token) throw new
Error('Token tidak
ditemukan');

    const response = await
api.get('/user', {
      headers: {
Authorization: Bearer ${token}
}
    });

    const user =
response.data?.user || null;
    const roles = user?.roles
&& user.roles.length ?
user.roles : ["user"];

    await
AsyncStorage.setItem('userRole
', JSON.stringify(roles));
    return { user, roles };
  } catch (error) {
    return { user: null,
roles: ["user"] }; // Fallback
agar aplikasi tetap berjalan
  }
};


// 🔷 Update profil user
(name, email, password)
export const updateProfile =
async (data) => {
    try {
        const token = await
AsyncStorage.getItem('userToke
n');
        if (!token) throw {
message: 'Token tidak
tersedia' };

        const response = await
api.put('/profile', data, {
            headers: {
                Authorization:
Bearer ${token}
            }
        });

        return response.data;
    } catch (error) {
        console.error('❌
Error saat update profile:',
error.response?.data ||
error);
        throw
error.response?.data || {
message: 'Gagal memperbarui
profil!' };
```

```
    }
};


// API untuk kategori
export const getCategories =
() => api.get('/categories');
export const createCategory =
async (formData) => {
  try {
    const response = await
api.post('/categories',
formData, {
      headers: {
        'Content-Type':
'multipart/form-data',
      },
    });
    console.log('Status:',
response.status, 'Response:',
response.data);
    return response.data;
  } catch (error) {
    if (error.response) {
      // Server mengembalikan
respon, tapi dengan status
error
      console.error("❌
Server error:", {
        status:
error.response.status,
        data:
error.response.data,
      });
    } else if (error.request)
{
      // Permintaan dibuat
tapi tidak ada respon
      console.error("❌ Tidak
ada respon dari server:",
error.request);
    } else {
      // Error saat menyusun
permintaan
      console.error("❌
Kesalahan saat mengirim
request:", error.message);
    }
    throw error;
  }
};


export const getCategory =
(id) =>
api.get(/categories/${id});
export const updateCategory =
async (id, formData) => {
  try {
    const response = await
api.post(/categories/${id}?_me
thod=PUT, formData, {
      headers: {
        'Content-Type':
'multipart/form-data',
      },
    });
    console.log('Status:',
response.status, 'Response:',
response.data);
    return response.data;
  } catch (error) {
    console.error('Gagal
memperbarui kategori:',
error.response?.data ||
error.message);
    throw error;
```

```
  }
};


export const deleteCategory =
(id) =>
api.delete(/categories/${id});


// API untuk media

export const getMedia = () =>
api.get('/media');

export const getMediaById =
(id) => api.get(/media/${id});

export const deleteMedia =
(id) =>
api.delete(/media/${id});


export const createMedia =
async (formData) => {

  try {

    const response = await
api.post('/media', formData, {

      headers: {

        'Content-Type':
'multipart/form-data',

      },

    });

    console.log('Status:',
response.status, 'Response:',
response.data);

    return response.data;

  } catch (error) {

    console.error('Gagal
mengirim data:',
error.response?.data ||
error.message);

    throw error;

  }
};


export const updateMedia =
async (id, data) => {

    return
api.put(/media/${id}, data, {

      headers: { "Content-
Type": "application/json" },
// Ubah ke JSON

    });

  };


  // API untuk Biodata


export const getAllBiodata =
() => api.get('/biodata');


export const createBiodata =
async (formData) => {

  try {

    const response = await
api.post('/biodata', formData,
{

      headers: {

        'Content-Type':
'multipart/form-data',

      },

    });

    return response.data;

  } catch (error) {

    console.error('✕ Gagal
membuat biodata:',
error.response?.data ||
error.message);

    throw error;
```

```
  }
};

export const getBiodataById =
(id) =>
api.get(/biodata/${id});

export const updateBiodata =
async (id, formData) => {
  try {
    const response = await
api.post(/biodata/${id},
formData, {
      headers: {
        'Content-Type':
'multipart/form-data',
      },
    });
    return response.data;
  } catch (error) {
    console.error(❌ Gagal
memperbarui biodata ID ${id}:,
error.response?.data ||
error.message);
    throw error;
  }
};


export const deleteBiodata =
async (id) => {
  try {
    const response = await
api.delete(/biodata/${id});
    return response.data;
  } catch (error) {
    console.error(❌ Gagal
menghapus biodata ID ${id}:,
error.response?.data ||
error.message);
    throw error;
  }
};

import { enableScreens } from
'react-native-screens';

enableScreens();

import { StatusBar } from
'expo-status-bar';

import { NavigationContainer }
from '@react-
navigation/native';

import {
createNativeStackNavigator }
from '@react-
navigation/native-stack';

import { createContext,
useEffect, useState } from
'react';

import { getUser } from
'./api';

import { View,
ActivityIndicator,Text } from
'react-native';

import tw from 'twrnc';

import LoginScreen from
'./Page/Auth/LoginScreen';

import RegisterScreen from
'./Page/Auth/RegisterScreen';

import ProfilScreen from
'./Page/Profile/ProfilScreen';

import PinScreen from
'./Component/PinScreen';

import BottomNavbar from
'./Component/BottomNavbar';

import AsyncStorage from
'@react-native-async-
storage/async-storage';

import Header from
'./Component/Header';
```

```
import WelcomeScreen from
'./Page/WelcomeScreen';

import GlobalErrorBoundary
from
'./Component/GlobalErrorBounda
ry';

import { AuthContext } from
'./Component/AuthContext';

import Indexkategori from
'./Page/Kategori/index';

import Createkategori from
'./Page/Kategori/Create';

import Editkategori from
'./Page/Kategori/Edit';

import Indexrumah from
'./Page/Rumah/indexx';

import Createrumah from
'./Page/Rumah/Createe';

import Detailrumah from
'./Page/Rumah/detail';

import Editrumah from
'./Page/Rumah/Editt';

import IndexBiodata from
'./Page/Biodata/indexxx';

import CreateBiodata from
'./Page/Biodata/Createeee';

import EditBiodata from
'./Page/Biodata/Edittt';

import DetailBiodata from
'./Page/Biodata/detaill';

import Info from
'./Page/Info';


const Stack =
createNativeStackNavigator();


export default function App()
{

    const [user, setUser] =
useState(null);

    const [loading,
setLoading] = useState(true);

    const [error, setError] =
useState(null);

    const [checkingToken,
setCheckingToken] =
useState(true);


    const checkAuth = async ()
=> {

        setLoading(true);

        setError(null);


        try {

            const token =
await
AsyncStorage.getItem('userToke
n');

            if (!token) {

                setUser(null);

            } else {

                const userData
= await getUser();


setUser(userData);

            }

        } catch (err) {


setError(err.message ||
'Terjadi kesalahan saat memuat
data.');

            setUser(null);

        } finally {

            setLoading(false);


setCheckingToken(false);

        }
```

```
    };

    // Gunakan useEffect hanya sekali saat aplikasi pertama kali dijalankan
    useEffect(() => {
        checkAuth();
    }, []);

    if (checkingToken) {
        return (
            <View style={tw`flex-1 justify-center items-center bg-gray-100`}>
                <ActivityIndicator size="large" color="#0000ff" />
                <Text style={tw`mt-4 text-lg text-gray-700`}>Memuat aplikasi...</Text>
            </View>
        );
    }

    return (
        <GlobalErrorBoundary>
            <AuthContext.Provider value={{ user, setUser }}>
                <NavigationContainer>
                    <Header />
                    <Stack.Navigator screenOptions={{ headerShown: false }}>
                        {!user ? (
                            <>
                                <Stack.Screen name="Welcome" component={WelcomeScreen} />
                                <Stack.Screen name="Login" component={LoginScreen} />
                                <Stack.Screen name="Register" component={RegisterScreen} />
                                <Stack.Screen name="Pin" component={PinScreen} />
                            </>
                        ) : (
                            <Stack.Screen name="Main" component={BottomNavbar} />
                        )}
                        <Stack.Screen name="DashboardCategory" component={Indexkategori} />
                        <Stack.Screen name="AddCategory" component={Createkategori} />
                        <Stack.Screen name="EditCategory" component={Editkategori} />
```

```
<Stack.Screen
name="Dashboardrumah"
component={Indexrumah} />

<Stack.Screen name="Addrumah"
component={Createrumah} />

<Stack.Screen name="Detail
rumah" component={Detailrumah}
/>

<Stack.Screen name="Edit
rumah" component={Editrumah}
/>

<Stack.Screen name="Dashboard
Biodata"
component={IndexBiodata} />

<Stack.Screen name="Tambah
Biodata"
component={CreateBiodata} />

<Stack.Screen name="Edit
Biodata"
component={EditBiodata} />

<Stack.Screen name="Detail
Biodata"
component={DetailBiodata} />

<Stack.Screen name="Info"
component={Info} />

<Stack.Screen name="Profile"
component={ProfilScreen} />

</Stack.Navigator>
                    <StatusBar
style="auto" />

</NavigationContainer>
</AuthContext.Provider>
        </GlobalErrorBoundary>
    );
}
```

```
import { useState, useContext
} from 'react';

import { View, Text,
TextInput, Button, Alert,
TouchableOpacity,
ActivityIndicator } from
'react-native';

import { login } from
'../../api';

import { AuthContext } from
'../../Component/AuthContext';

import tw from 'twrnc';

import { FontAwesome } from
'@expo/vector-icons';

import AsyncStorage from
'@react-native-async-
storage/async-storage';

import { useNavigation } from
'@react-navigation/native';

import { LinearGradient } from
'expo-linear-gradient';


export default function
LoginScreen() {

    const [email, setEmail] =
useState('');

    const [password,
setPassword] = useState('');

    const [isLoading,
setIsLoading] =
useState(false); // ← State
loading

    const { setUser } =
useContext(AuthContext);
```

```
    const [showPassword,
setShowPassword] =
useState(false);

    const navigation =
useNavigation();



    const handleLogin = async
() => {

        if (!email.trim() ||
!password.trim()) {


Alert.alert('Error', 'Email
dan password tidak boleh
kosong!');

            return;

        }



        setIsLoading(true); //
Mulai loading

        try {

            const response =
await login({ email, password
});



            await
AsyncStorage.setItem('userToke
n', response.token);


setUser(response.user);



Alert.alert('Berhasil', 'Login
sukses!');

        } catch (error) {

            console.error('❌
Login gagal:', error);

            let message =
'Login gagal!';

            const msg =
error?.message;


            if (msg === 'These
credentials do not match our
records.') {

                message =
'Email atau kata sandi Anda
salah';

            }



Alert.alert('Gagal', message);

        } finally {


setIsLoading(false); //
Selesai loading

        }

    };



    return (

        <LinearGradient


colors={['rgba(0,0,50,0.6)',
'rgba(0,0,100,0.6)']}

            style={tw`flex-1
items-center justify-center p-
6`}

        >

            <Text
style={tw`text-2xl font-bold
mb-6 text-white`}>Login</Text>


            <TextInput


placeholder="Email"

                value={email}
```

```
onChangeText={setEmail}
                style={tw`w-
full bg-white p-4 mb-4
rounded-xl shadow-sm`}
keyboardType="email-address"
autoCapitalize="none"
            />
            <View style={tw`w-
full bg-white flex-row items-
center p-4 mb-4 rounded-xl
shadow-sm`}>
                <TextInput
placeholder="Password"
value={password}
onChangeText={setPassword}
style={tw`flex-1`}
secureTextEntry={!showPassword
}
                />
<TouchableOpacity onPress={()
=>
setShowPassword(!showPassword)
}>
<FontAwesome
name={showPassword ? "eye-
slash" : "eye"} size={20} />
</TouchableOpacity>
            </View>
            <View style={tw`w-
full mt-2`}>
                {isLoading ? (
<ActivityIndicator
size="large" color="#4CAF50"
/>
                ) : (
<TouchableOpacity
onPress={handleLogin}
style={tw`bg-green-500 py-3
rounded-full`}
                    >
                        <Text
style={tw`text-white text-
center font-
bold`}>Login</Text>
</TouchableOpacity>
                )}
            </View>
            <TouchableOpacity
onPress={() =>
navigation.navigate('Register'
)}>
                <Text
style={tw`text-sm mt-4 text-
blue-400`}>Belum punya akun?
silahkan register</Text>
</TouchableOpacity>
        </LinearGradient>
    );
}
import { useState } from
'react';
```

```
import {
    View,
    Text,
    TextInput,
    Button,
    Alert,
    TouchableOpacity,
    ActivityIndicator
} from 'react-native';
import { register } from
'../../api';
import tw from 'twrnc';
import { FontAwesome } from
'@expo/vector-icons';


export default function
RegisterScreen({ navigation })
{
    const [name, setName] =
useState('');
    const [email, setEmail] =
useState('');
    const [no, setNo] =
useState('');
    const [password,
setPassword] = useState('');
    const
[passwordConfirmation,
setPasswordConfirmation] =
useState('');
    const [showPassword,
setShowPassword] =
useState(false);
    const
[showPasswordConfirmation,
setShowPasswordConfirmation] =
useState(false);
    const [isLoading,
setIsLoading] =
useState(false);


    const handleRegister =
async () => {
        if (!name || !email ||
!no || !password ||
!passwordConfirmation) {

Alert.alert('Peringatan',
'Semua field wajib diisi!');
            return;
        }


        if (password !==
passwordConfirmation) {

Alert.alert('Gagal', 'Password
dan Konfirmasi Password tidak
sama!');
            return;
        }


        setIsLoading(true);
        try {
            await register({
                name,
                email,
                no,
                password,

password_confirmation:
passwordConfirmation
            });

Alert.alert('Berhasil',
'Registrasi sukses!');
```

```
navigation.navigate('Login');
        } catch (error) {
Alert.alert('Gagal',
error.message || 'Registrasi
gagal!');
        } finally {
setIsLoading(false);
        }
    };

    return (
        <View style={tw`flex-1
items-center justify-center
bg-gray-100 p-6`}>
            <Text
style={tw`text-2xl font-bold
mb-6`}>Register</Text>

            <TextInput
placeholder="Nama"
                value={name}
onChangeText={setName}
                style={tw`w-
full bg-white p-4 mb-4
rounded-xl shadow-sm`}
            />

            <TextInput
placeholder="Email"
                value={email}
onChangeText={setEmail}
                style={tw`w-
full bg-white p-4 mb-4
rounded-xl shadow-sm`}
keyboardType="email-address"
autoCapitalize="none"
            />

            <TextInput
placeholder="Nomor Handphone"
                value={no}
onChangeText={(text) => {
                    //
Hilangkan semua karakter non-
digit
                    let
sanitized =
text.replace(/\D/g, '');

                    //
Pastikan diawali dengan '62'
                    if
(!sanitized.startsWith('62'))
{
sanitized = '62' +
sanitized.replace(/^0+/, '');
// hapus 0 jika ditulis di
awal
                    }

                    // Setelah
'62', angka berikut tidak
boleh '0'
                    const
afterPrefix =
sanitized.slice(2);
```

```
                    if
(afterPrefix.length > 0 &&
afterPrefix[0] === '0') {

Alert.alert("Nomor tidak
valid", "Angka setelah 62
tidak boleh 0");

return;

                    }



setNo(sanitized);

                }}

keyboardType="numeric"

                style={tw`w-
full bg-white p-4 mb-4
rounded-xl shadow-sm`}

autoCapitalize="none"

            />



            <View style={tw`w-
full bg-white flex-row items-
center p-4 mb-4 rounded-xl
shadow-sm`}>

                <TextInput

placeholder="Password"

value={password}

onChangeText={setPassword}

style={tw`flex-1`}

secureTextEntry={!showPassword
}

autoCapitalize="none"

                />

<TouchableOpacity onPress={()
=>
setShowPassword(!showPassword)
}>

<FontAwesome
name={showPassword ? "eye-
slash" : "eye"} size={20} />

</TouchableOpacity>

            </View>

            <View style={tw`w-
full bg-white flex-row items-
center p-4 mb-6 rounded-xl
shadow-sm`}>

                <TextInput

placeholder="Konfirmasi
Password"

value={passwordConfirmation}

onChangeText={setPasswordConfi
rmation}

style={tw`flex-1`}

secureTextEntry={!showPassword
Confirmation}

autoCapitalize="none"

                />

<TouchableOpacity onPress={()
=>
setShowPasswordConfirmation(!s
howPasswordConfirmation)}>
```

```
<FontAwesome
name={showPasswordConfirmation
? "eye-slash" : "eye"}
size={20} />


</TouchableOpacity>

            </View>


            <View style={tw`w-
full mt-2`}>

                {isLoading ? (


<ActivityIndicator
size="large" color="#4CAF50"
/>

                ) : (

                    <Button
title="Register"
onPress={handleRegister}
color="#4CAF50" />

                )}

            </View>


            <TouchableOpacity
onPress={() =>
navigation.navigate('Login')}>

                <Text
style={tw`text-sm mt-4 text-
blue-600`}>

                    Sudah
punya akun? Login di sini

                </Text>


</TouchableOpacity>

        </View>

    );

}
```

# KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

## SKRIPSI

| | |
|---|---|
| **Mahasiswa : ARMAYANI** | **Pembimbing I : WAHYUDDIN, S.Kom.,M.Kom** |
| **NIM : 218280161** | **Pembimbing II : MUGHAFFIR YUNUS, S.T.,MT** |
| **Judul Skripsi : RANCANG BANGUN APLIKASI PENJUALAN PERUMAHAN DI KABUPATEN SIDRAP BERBASIS ANDROID** | |

| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 1<br>- Perbaiki Aplikasinya | | Konsultasi 1<br>- Revisi Aplikasi<br>Revisi laporan | |
| Konsultasi 2 | | Konsultasi 2<br>- Revisi Aplikasi<br>- Revisi Laporan | |
| Konsultasi 3 | | Konsultasi 3<br>- Perbaiki Bab 4<br>- Perjelas Gambar | |
| Konsultasi 4 | | Konsultasi 4<br>Acc<br>seminar Hasil | |
| Konsultasi 5 | | Konsultasi 5 | |

*Lanjut ke halaman sebelah...*

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton A4 berwarna Hijau muda dan dicetak timbal balik

# KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

## SKRIPSI

| | |
|---|---|
| **Mahasiswa : ARMAYANI** | **Pembimbing I : Wahyuddin, S.Kom.,M.Kom** |
| **NIM : 218280161** | **Pembimbing II : Mughaffir, S.Kom.,M.Kom** |
| **Judul Skripsi : RANCANG BANGUN APLIKASI PENJUALAN PERUMAHAN DI KABUPATEN SIDRAP BERBASIS ANDROID (STUDI KASUS: PT.TIGA PUTRA BUGIS)** | |

| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 1<br>Acc | | Konsultasi 1<br>- Perbaiki metode penelitian | |
| Konsultasi 2 | | Konsultasi 2<br>- Perbaiki kerangka teori | |
| Konsultasi 3 | | Konsultasi 3<br>- Acc proposal | |
| Konsultasi 4 | | Konsultasi 4 | |
| Konsultasi 5 | | Konsultasi 5 | |

*Lanjut ke halaman sebelah...*

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik