LAMPIRAN

WelcomeScreen.jsx

```
import { View, Text,
TouchableOpacity,
ImageBackground } from 'react-
native';
import tw from 'twrnc';

export default function
WelcomeScreen({ navigation })
{
    return (
        <ImageBackground

source={require('../assets/log
oo.jpg')}
            style={tw`flex-1
justify-center items-center
bg-black`}
        >
            {/* Overlay manual
menggunakan View */}
            <View
style={tw`absolute inset-0 bg-
black opacity-50`} />

            <View
style={tw`flex-1 justify-
center items-center p-6`}>
                <Text
style={tw`text-yellow-400
text-4xl font-extrabold text-
center mb-4`}>
                    Temukan
Kelezatan Kue Khas Sulawesi
Selatannnn!
                </Text>

                <Text
style={tw`text-white text-lg
text-center mb-6`}>
                    Cita rasa
tradisional dengan sentuhan
khas budaya lokal.
                    Dari kue
yang lembut hingga renyah,
semua ada di sini!
                </Text>

                {/* Tombol
Masuk */}
<TouchableOpacity

style={tw`bg-yellow-500 px-8
py-3 rounded-full shadow-lg
mb-4`}

onPress={() =>
navigation.navigate('Login')}
                >
                    <Text
style={tw`text-black font-bold
text-lg`}>Masuk dan
Nikmati</Text>

</TouchableOpacity>

                {/* Tombol
Lihat Kue */}

<TouchableOpacity

style={tw`border border-
yellow-500 px-8 py-3 rounded-
full shadow-lg`}

onPress={() =>
navigation.navigate('Dashboard
Category')}
                >
                    <Text
style={tw`text-yellow-500
font-bold text-lg text-
center`}>

Jelajahi Ragam Kue Tradisional
                    </Text>

</TouchableOpacity>
            </View>
        </ImageBackground>
    );
}
```

LoginScreen.jsx

```
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';

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 (
        <View
            style={tw`flex-1
bg-yellow-300 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('Welcome')
}>
                <Text
style={tw`text-sm mt-4 text-
blue-400`}>Kembali ke halaman
awal</Text>

</TouchableOpacity>
        </View>
    );
}
```

RegisterScreen.jsx

```
import { useState } from
'react';
import { View, Text,
TextInput, Button, Alert,
TouchableOpacity } from
'react-native';
import { register } from
'../api';
import tw from 'twrnc';

export default function
RegisterScreen({ navigation })
{
    const [name, setName] =
useState('');
    const [email, setEmail] =
useState('');
    const [password,
setPassword] = useState('');
    const
[passwordConfirmation,
setPasswordConfirmation] =
useState('');
```

```
    const handleRegister =
async () => {
        if (password !==
passwordConfirmation) {

Alert.alert('Gagal', 'Password
dan Konfirmasi Password tidak
sama!');
            return;
        }
        try {
            await register({
                name,
                email,
                password,

password_confirmation:
passwordConfirmation
            });

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

navigation.navigate('Login');
        } catch (error) {

Alert.alert('Gagal',
error.message || 'Registrasi
gagal!');
        }
    };

    return (
        <View style={tw`flex-1
items-center justify-center
bg-gray-100 p-6`}>
      <TouchableOpacity
        style={tw`bg-gray-300
p-2 rounded mb-4 w-24`}
        onPress={() =>
navigation.goBack()}
      >
        <Text style={tw`text-
black text-center font-
bold`}>Kembali</Text>
      </TouchableOpacity>
            <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="Password"

value={password}

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

secureTextEntry
            />
            <TextInput

placeholder="Konfirmasi
Password"

value={passwordConfirmation}

onChangeText={setPasswordConfi
rmation}
                style={tw`w-
full bg-white p-4 mb-6
rounded-xl shadow-sm`}

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

```
                <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>
    );
}
```

DashboardScreen.jsx

```
import { View, Text, Button,
Alert } from 'react-native';
import { useEffect, useState,
useContext } from 'react';
import { getUser, logout }
from '../api';
import tw from 'twrnc';
import { AuthContext } from
'../Component/AuthContext';

export default function
DashboardScreen({ navigation
}) {
    const { user, setUser } =
useContext(AuthContext);

    useEffect(() => {
        const fetchUser =
async () => {
            try {
                const userData
= await getUser();

setUser(userData);
            } catch (error) {
                setUser(null);
            }
        };
        fetchUser();
    }, []);

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

            {user ? (
                <View
style={tw`bg-gray-900 p-4
rounded-xl shadow-sm mb-6`}>
                    <Text
style={tw`text-lg font-
semibold text-white`}>

Selamat datang, {user.name}!
                    </Text>
                    <Text
style={tw`text-sm text-gray-
500`}>Email:
{user.email}</Text>
                </View>
            ) : (
                <Text
style={tw`text-lg mb-4`}>Anda
belum login</Text>
            )}

        </View>
    );
}
```

ProfilScreen.jsx

```
import { useState } from
'react';
import { View, Text,
TextInput, Button, Alert,
TouchableOpacity } from
'react-native';
// import { login } from
'../api';
import tw from 'twrnc';

export default function
ProfilScreen({ navigation }) {

    return (
```

```
        <View style={tw`flex-1
items-center justify-center
bg-gray-100 p-6`}>
            <Text>Halaman
Profil</Text>
        </View>
    );
}
```

DashboardCategory.jsx

```
import React, { useEffect,
useState, useContext } from
"react";
import {
  View,
  Text,
  FlatList,
  TouchableOpacity,
  Alert,
  TextInput,
  ImageBackground
} from "react-native";
import tw from "twrnc";
import { getCategories,
deleteCategory } from
"../api";
import { FontAwesome } from
"@expo/vector-icons";
import { AuthContext } from
"../Component/AuthContext"; //
Import AuthContext

const DashboardCategory = ({
navigation }) => {
  const { user } =
useContext(AuthContext) || {
user: null };

  const [categories,
setCategories] = useState([]);
  const [search, setSearch] =
useState("");
  const [filteredCategories,
setFilteredCategories] =
useState([]);

  useEffect(() => {
    fetchCategories();
  }, []);

  const fetchCategories =
async () => {
    try {
      const response = await
getCategories();

setCategories(response.data);

setFilteredCategories(response
.data);
    } catch (error) {
      Alert.alert("Error",
"Gagal mengambil data
kategori");
    }
  };

  const handleSearch = (text)
=> {
    setSearch(text);
    if (text) {
      const filteredData =
categories.filter((item) =>

item.kategori.toLowerCase().in
cludes(text.toLowerCase())
      );

setFilteredCategories(filtered
Data);
    } else {

setFilteredCategories(categori
es);
    }
  };

  const handleDelete = async
(id) => {
    Alert.alert(
      "Konfirmasi",
      "Apakah Anda yakin ingin
menghapus kategori ini?",
      [
        { text: "Batal",
style: "cancel" },
        {
          text: "Hapus",
          onPress: async () =>
{
            try {
```

```
              await
deleteCategory(id);

Alert.alert("Sukses",
"Kategori berhasil dihapus");

fetchCategories();
            } catch (error) {

Alert.alert("Gagal", "Tidak
dapat menghapus kategori");
            }
          },
          style:
"destructive",
        },
      ]
    );
  };

  return (
            <ImageBackground

source={require('../assets/kue
.jpg')}
            style={tw`flex-1
justify-center items-center
bg-black`}
        >
      {/* Tombol Kembali */}
      <TouchableOpacity
        style={tw`flex-row
items-center bg-yellow-500 px-
4 py-2 rounded-full mb-4 self-
start`}
        onPress={() => {

navigation.navigate("Welcome")
        }}

      >
        <FontAwesome
name="arrow-left" size={20}
color="black" />
        <Text style={tw`text-
black font-bold ml-
2`}>Kembali</Text>
      </TouchableOpacity>

      {/* Judul Halaman */}
      <Text style={tw`text-
yellow-600 font-bold text-2xl
text-center mb-4`}>
        Kategori Kue
      </Text>

      {/* Input Pencarian */}
      <TextInput
        style={tw`border
border-yellow-500 text-white
p-3 rounded-lg mb-4 bg-gray-
800`}
        placeholder="Cari
kategori..."

placeholderTextColor="#FACC15"
        value={search}

onChangeText={handleSearch}
      />

      {/* Tombol Tambah
Kategori (Hanya jika user
login) */}
      {user && (
        <TouchableOpacity
          style={tw`bg-yellow-
500 px-6 py-3 rounded-full mb-
4`}
          onPress={() =>
navigation.navigate("AddCatego
ry")}
        >
          <Text
style={tw`text-black text-
center font-bold text-lg`}>
            + Tambah Kategori
          </Text>
        </TouchableOpacity>
      )}

      {/* Daftar Kategori */}
      <FlatList

data={filteredCategories}
        keyExtractor={(item)
=> item.id.toString()}
        renderItem={({ item })
=> (
          <View style={tw`bg-
gray-800 p-4 rounded-lg mb-
3`}>
```

```
            {/* Nama Kategori
*/}
            <Text
style={tw`text-yellow-500
text-lg font-semibold text-
center mb-2`}>
              {item.kategori}
            </Text>

            {/* Tombol Aksi
*/}
            <View
style={tw`flex-row justify-
center gap-2`}>
              {/* Tombol
Detail (Selalu ditampilkan)
*/}

<TouchableOpacity

style={tw`flex-row items-
center bg-yellow-500 px-4 py-2
rounded-full`}
                onPress={() =>

navigation.navigate("ShowCateg
ory", { id: item.id })
                }
              >
                <FontAwesome
name="eye" size={16}
color="black" />
                <Text
style={tw`text-black font-bold
ml-2`}>Detail</Text>

</TouchableOpacity>

              {/* Tombol Edit
(Hanya jika user login) */}
              {user && (

<TouchableOpacity

style={tw`flex-row items-
center bg-blue-500 px-4 py-2
rounded-full`}
                  onPress={()
=>

navigation.navigate("EditCateg
ory", { id: item.id })
                  }
                >
                  <FontAwesome
name="pencil" size={16}
color="white" />
                  <Text
style={tw`text-white font-bold
ml-2`}>Edit</Text>

</TouchableOpacity>
              )}

              {/* Tombol Hapus
(Hanya jika user login) */}
              {user && (

<TouchableOpacity

style={tw`flex-row items-
center bg-red-500 px-4 py-2
rounded-full`}
                  onPress={()
=> handleDelete(item.id)}
                >
                  <FontAwesome
name="trash" size={16}
color="white" />
                  <Text
style={tw`text-white font-bold
ml-2`}>Hapus</Text>

</TouchableOpacity>
              )}
            </View>
          </View>
        )}
      />
    </ImageBackground >
  );
};

export default
DashboardCategory;


AddCategory.jsx

import React, { useState }
from "react";
import { View, Text,
TextInput, TouchableOpacity,
Alert } from "react-native";
```

```
import tw from "twrnc";
import { createCategory } from
"../api";
import { FontAwesome } from
"@expo/vector-icons";

const AddCategory = ({
navigation }) => {
  const [kategori,
setKategori] = useState("");

  const handleSubmit = async
() => {
    if (!kategori.trim()) {
      Alert.alert("Error",
"Kategori tidak boleh
kosong!");
      return;
    }

    try {
      await
createCategory(kategori);
      Alert.alert("Sukses",
"Kategori berhasil
ditambahkan!");

navigation.navigate("Dashboard
Category");
    } catch (error) {
      Alert.alert("Gagal",
"Terjadi kesalahan saat
menambah kategori.");
    }
  };

  return (
    <View style={tw`flex-1 p-4
bg-black`}>
      {/* Tombol Kembali */}
      <TouchableOpacity
        style={tw`flex-row
items-center bg-yellow-500 px-
4 py-2 rounded-full mb-4 self-
start`}
        onPress={() =>
navigation.goBack()}
      >
        <FontAwesome
name="arrow-left" size={20}
color="white" />
        <Text style={tw`text-
white font-bold ml-
2`}>Kembali</Text>
      </TouchableOpacity>

      {/* Judul Halaman */}
      <Text style={tw`text-
yellow-500 font-bold text-2xl
text-center mb-6`}>
        Tambah Kategori
      </Text>

      {/* Input Kategori */}
      <TextInput
        style={tw`border
border-yellow-500 text-white
p-3 rounded-lg mb-4 bg-gray-
800`}
        placeholder="Masukkan
Nama Kategori"

placeholderTextColor="#FACC15"
        value={kategori}

onChangeText={setKategori}
      />

      {/* Tombol Simpan */}
      <TouchableOpacity
        style={tw`bg-yellow-
500 px-6 py-3 rounded-full`}
        onPress={handleSubmit}
      >
        <Text style={tw`text-
black text-center font-bold
text-lg`}>Simpan</Text>
      </TouchableOpacity>
    </View>
  );
};

export default AddCategory;
```

ShowCategory.jsx

```
import React, { useEffect,
useState, useContext } from
"react";
import {
  View,
  Text,
```

```
  TextInput,
  ActivityIndicator,
  Alert,
  FlatList,
  Image,
  TouchableOpacity,
} from "react-native";
import tw from "twrnc";
import { getCategory,
deleteMedia } from "../api";
import { FontAwesome } from
"@expo/vector-icons";
import { AuthContext } from
"../Component/AuthContext"; //
Import AuthContext

const ShowCategory = ({ route,
navigation }) => {
  const { user } =
useContext(AuthContext) || {
user: null };

  const id = route.params?.id;

  if (!id) {
    Alert.alert("Error",
"Kategori tidak ditemukan");
    navigation.goBack();
    return null;
  }

  const [category,
setCategory] = useState(null);
  const [loading, setLoading]
= useState(true);
  const [search, setSearch] =
useState("");
  const [filteredMedia,
setFilteredMedia] =
useState([]);

  useEffect(() => {
    fetchCategory();
  }, []);

  const fetchCategory = async
() => {
    try {
      const response = await
getCategory(id);

setCategory(response.data);

setFilteredMedia(response.data
.media || []);
    } catch (error) {
      Alert.alert("Error",
"Gagal mengambil data
kategori");
    } finally {
      setLoading(false);
    }
  };

  const handleSearch = (text)
=> {
    setSearch(text);
    if (text) {
      const filteredData =
category?.media?.filter((item)
=>

item.judul.toLowerCase().inclu
des(text.toLowerCase())
      );

setFilteredMedia(filteredData
|| []);
    } else {

setFilteredMedia(category?.med
ia || []);
    }
  };

  const handleDeleteMedia =
(mediaId) => {
    Alert.alert("Konfirmasi",
"Apakah Anda yakin ingin
menghapus media ini?", [
      { text: "Batal", style:
"cancel" },
      {
        text: "Hapus",
        onPress: async () => {
          try {
            await
deleteMedia(mediaId);

Alert.alert("Sukses", "Media
berhasil dihapus");

            // Update daftar
media setelah penghapusan
```

```
            const updatedMedia
= filteredMedia.filter(
              (item) =>
item.id !== mediaId
            );

setFilteredMedia(updatedMedia)
;
          } catch (error) {

Alert.alert("Error", "Gagal
menghapus media");
          }
        },
        style: "destructive",
      },
    ]);
  };

  if (loading) {
    return <ActivityIndicator
style={tw`flex-1`}
size="large" color="yellow"
/>;
  }

  return (
    <View style={tw`flex-1 p-4
bg-black`}>
      {/* Tombol Kembali */}
      <TouchableOpacity
        style={tw`flex-row
items-center bg-yellow-500 px-
4 py-2 rounded-full mb-4 self-
start`}
        onPress={() => {


navigation.navigate("Dashboard
Category");


        }}
      >
        <FontAwesome
name="arrow-left" size={20}
color="black" />
        <Text style={tw`text-
black font-bold ml-
2`}>Kembali</Text>
      </TouchableOpacity>

      {/* Judul Halaman */}
      <Text style={tw`text-
yellow-500 font-bold text-2xl
text-center mb-4`}>
        {category?.kategori ||
"Detail Kategori"}
      </Text>

      {/* Input Pencarian */}
      <TextInput
        style={tw`border
border-yellow-500 text-white
p-3 rounded-lg mb-4 bg-gray-
800`}
        placeholder="Cari
makanan..."

placeholderTextColor="#FACC15"
        value={search}

onChangeText={handleSearch}
      />

      {/* Tombol Tambah
Makanan (Disembunyikan jika
user tidak login) */}
      {user && (
        <TouchableOpacity
          style={tw`bg-yellow-
500 px-6 py-3 rounded-full mb-
4`}
          onPress={() =>
navigation.navigate("AddMedia"
)}
        >
          <Text
style={tw`text-black text-
center font-bold text-lg`}>
            + Tambah Makanan
          </Text>
        </TouchableOpacity>
      )}

      {/* Daftar Makanan */}
      {filteredMedia.length >
0 ? (
        <FlatList
          data={filteredMedia}
          keyExtractor={(item)
=> item.id.toString()}
          renderItem={({ item
}) => (
```

```
            <View
style={tw`mb-4 p-4 bg-gray-800
rounded-lg`}>
              {/* Gambar
Makanan */}
              {item.foto && (
                <Image
                  source={{
uri: item.foto }}
                  style={tw`w-
full h-40 rounded-lg mb-2`}

resizeMode="cover"
                />
              )}
              {/* Judul */}
              <Text
style={tw`text-yellow-500
text-lg font-bold`}>
                {item.judul}
              </Text>

              {/* Tombol Aksi
*/}
              <View
style={tw`flex-row justify-
between mt-3`}>
                {/* Tombol
Detail */}

<TouchableOpacity

style={tw`flex-row items-
center bg-yellow-500 px-4 py-2
rounded-full`}
                  onPress={()
=>

navigation.navigate("MediaDeta
il", { id: item.id })
                  }
                >
                  <FontAwesome
name="eye" size={16}
color="black" />
                  <Text
style={tw`text-black font-bold
ml-2`}>Detail</Text>

</TouchableOpacity>

                {/* Tombol
Edit (Disembunyikan jika user
tidak login) */}
                {user && (

<TouchableOpacity

style={tw`flex-row items-
center bg-blue-500 px-4 py-2
rounded-full`}

onPress={() =>

navigation.navigate("EditMedia
", { id: item.id })
                    }
                  >

<FontAwesome name="pencil"
size={16} color="white" />
                    <Text
style={tw`text-white font-bold
ml-2`}>Edit</Text>

</TouchableOpacity>
                )}

                {/* Tombol
Hapus (Disembunyikan jika user
tidak login) */}
                {user && (

<TouchableOpacity

style={tw`flex-row items-
center bg-red-500 px-4 py-2
rounded-full`}

onPress={() =>
handleDeleteMedia(item.id)}
                  >

<FontAwesome name="trash"
size={16} color="white" />
                    <Text
style={tw`text-white font-bold
ml-2`}>Hapus</Text>

</TouchableOpacity>
                )}
              </View>
            </View>
```

```
          )}
        />
      ) : (
        <Text style={tw`text-
gray-500 text-center`}>
          Tidak ada makanan
dalam kategori ini.
        </Text>
      )}
    </View>
  );
};

export default ShowCategory;
```

EditCategory.jsx

```
import React, { useState,
useEffect } from "react";
import { View, Text,
TextInput, TouchableOpacity,
Alert } from "react-native";
import tw from "twrnc";
import { getCategory,
updateCategory } from
"../api";
import { FontAwesome } from
"@expo/vector-icons";

const EditCategory = ({ route,
navigation }) => {
  const { id } = route.params;
// Ambil ID kategori dari
navigasi
  const [kategori,
setKategori] = useState("");
  const [loading, setLoading]
= useState(true);

  useEffect(() => {
    fetchCategory();
  }, []);

  const fetchCategory = async
() => {
    try {
      const response = await
getCategory(id);

setKategori(response.data.kate
gori);
    } catch (error) {
      Alert.alert("Error",
"Gagal mengambil data
kategori.");
      navigation.goBack();
    } finally {
      setLoading(false);
    }
  };

  const handleUpdate = async
() => {
    if (!kategori.trim()) {

Alert.alert("Peringatan",
"Kategori tidak boleh
kosong.");
      return;
    }

    try {
      await updateCategory(id,
kategori);
      Alert.alert("Sukses",
"Kategori berhasil
diperbarui.");

navigation.navigate("Dashboard
Category");
    } catch (error) {
      Alert.alert("Error",
"Gagal memperbarui
kategori.");
    }
  };

  if (loading) {
    return (
      <View style={tw`flex-1
items-center justify-center
bg-black`}>
        <Text style={tw`text-
yellow-500`}>Memuat...</Text>
      </View>
    );
  }

  return (
    <View style={tw`flex-1 p-4
bg-black`}>
      {/* Tombol Kembali */}
      <TouchableOpacity
```

```
        style={tw`flex-row
items-center bg-yellow-500 px-
4 py-2 rounded-full mb-4 self-
start`}
        onPress={() =>
navigation.goBack()}
      >
        <FontAwesome
name="arrow-left" size={20}
color="white" />
        <Text style={tw`text-
white font-bold ml-
2`}>Kembali</Text>
      </TouchableOpacity>

      {/* Judul Halaman */}
      <Text style={tw`text-
yellow-500 font-bold text-2xl
text-center mb-6`}>
        Edit Kategori
      </Text>

      {/* Input Kategori */}
      <TextInput
        style={tw`border
border-yellow-500 text-white
p-3 rounded-lg mb-4 bg-gray-
800`}
        value={kategori}

onChangeText={setKategori}
        placeholder="Masukkan
kategori"

placeholderTextColor="#FACC15"
      />

      {/* Tombol Simpan */}
      <TouchableOpacity
        style={tw`bg-yellow-
500 px-6 py-3 rounded-full`}
        onPress={handleUpdate}
      >
        <Text style={tw`text-
black text-center font-bold
text-lg`}>Simpan</Text>
      </TouchableOpacity>
    </View>
  );
};

export default EditCategory;
```

AddMedia.jsx

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

// const API_URL =
'https://kue.proyekk.com/api';
const API_URL =
'http://192.168.0.107/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('userToke
n', response.data.token);

api.defaults.headers.common['A
uthorization'] = `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
(data) => {
    try {
        console.log('🔍
Memulai proses login...',
data);

        const response = await
api.post('/login', data);
        const token =
response.data.token;

        if (!token) throw {
message: 'Token tidak diterima
dari server' };

        // Simpan token ke
AsyncStorage
        await
AsyncStorage.setItem('userToke
n', token);
        console.log('📦 Token
tersimpan:', token);

        return response.data;
    } catch (error) {
        console.error('❌
Error saat login:',
error.response?.data ||
error);
        throw
error.response?.data || {
message: 'Login gagal!' };
    }
};

// 🔹 Logout
export const logout = async ()
=> {
    try {
        const token = await
AsyncStorage.getItem('userToke
n');
        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('userT
oken'); // 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
(token = null) => {
    try {
        const headers = token
? { Authorization: `Bearer
${token}` } : {};
        const response = await
api.get('/user', { headers });
        return response.data;
    } catch (error) {
        throw
error.response?.data || {
message: 'Gagal mendapatkan
data user!' };
    }
};

// API untuk kategori
export const getCategories =
() => api.get('/categories');
export const createCategory =
(kategori) =>
api.post('/categories', {
kategori });
export const getCategory =
(id) =>
api.get(`/categories/${id}`);
```

```
export const updateCategory =
(id, kategori) =>
api.put(`/categories/${id}`, {
kategori });
export const deleteCategory =
(id) =>
api.delete(`/categories/${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
    });
  };

export default api;
```

Media2.jsx

```
import React, { useEffect,
useState } from "react";
import { View, Text,
ActivityIndicator, Alert,
ImageBackground,
TouchableOpacity, ScrollView,
Share } from "react-native";
import { WebView } from
"react-native-webview";
import tw from "twrnc";
import { getMediaById } from
"../api";

const Media2 = ({ route,
navigation }) => {
  const mediaId =
route.params?.id;
  const BASE_URL =
"http://192.168.0.107/storage/
";

  if (!mediaId) {
    Alert.alert("Error",
"Media tidak ditemukan");
    navigation.goBack();
    return null;
  }

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

  useEffect(() => {
    fetchMediaDetail();
  }, []);

  const fetchMediaDetail =
async () => {
    try {
      const response = await
getMediaById(mediaId);
      setMedia(response.data);
    } catch (error) {
      Alert.alert("Error",
"Gagal mengambil data media");
      navigation.goBack();
    } finally {
```

```
      setLoading(false);
    }
  };

  const handleShareText =
async () => {
    try {
      const videoLink =
media.video.startsWith("http")
? media.video :
`${BASE_URL}${media.video}`;
      const message =
`Resep:\n${media.judul}\n\n${m
edia.resep}\n\nTonton video
pembuatannya di sini:
${videoLink}`;
      await Share.share({
message });
    } catch (error) {
      console.error("Share
text error:", error);
      Alert.alert("Error",
"Gagal membagikan teks");
    }
  };

  if (loading) {
    return <ActivityIndicator
style={tw`flex-1`}
size="large" color="yellow"
/>;
  }

  const videoUrl =
media.video.startsWith("http")
? media.video :
`${BASE_URL}${media.video}`;

  return (
    <ImageBackground
source={require("../assets/kue
.jpg")} style={tw`flex-1 bg-
black`}>
      <View style={tw`absolute
inset-0 bg-black opacity-50`}
/>

      <ScrollView
contentContainerStyle={tw`flex
-grow justify-center items-
center p-6`}>
        <TouchableOpacity
          style={tw`bg-yellow-
500 px-6 py-3 rounded-full
shadow-lg mb-4`}
          onPress={() =>
navigation.goBack()}
        >
          <Text
style={tw`text-white font-bold
text-lg`}>Kembali</Text>
        </TouchableOpacity>

        {media ? (
          <View style={tw`w-
full`}>
            <Text
style={tw`text-3xl font-bold
text-white text-center mb-
4`}>{media.judul}</Text>

            {media.video && (
              <View
style={tw`w-full h-60 rounded-
lg overflow-hidden mb-4
shadow-lg`}>
                <WebView
source={{ uri: videoUrl }}
style={{ flex: 1 }}
allowsFullscreenVideo />
              </View>
            )}

            <View
style={tw`bg-white/80 p-4
rounded-lg shadow-lg w-full`}>
              <Text
style={tw`text-lg font-bold
text-gray-800 text-center mb-
2`}>Resep</Text>
              <View
style={tw`border-b border-
gray-300 my-2`} />
              <Text
style={tw`text-gray-700 text-
center`}>{media.resep}</Text>
            </View>

            <TouchableOpacity
              style={tw`bg-
yellow-500 px-6 py-3 rounded-
full shadow-lg mt-4`}

onPress={handleShareText}
```

```
            >
              <Text
style={tw`text-white font-bold
text-lg`}>Bagikan Cara
Pembuatan Kue</Text>

</TouchableOpacity>
          </View>
        ) : (
          <Text
style={tw`text-gray-300 text-
lg text-center`}>Media tidak
ditemukan.</Text>
        )}
      </ScrollView>
    </ImageBackground>
  );
};

export default Media2;
```

MediaDetail.jsx

```
import React, { useEffect,
useState, useCallback } from
"react";
import { View, Text,
ActivityIndicator, Alert,
Image, TouchableOpacity,
ImageBackground, ScrollView,
Share } from "react-native";
import tw from "twrnc";
import { getMediaById } from
"../api";
import { useFocusEffect } from
"@react-navigation/native";
import { Audio } from "expo-
av";

const MediaDetail = ({ route,
navigation }) => {
  const mediaId =
route.params?.id;

  const [media, setMedia] =
useState(null);
  const [loading, setLoading]
= useState(true);
  const [sound, setSound] =
useState(null);
  const [isPlaying,
setIsPlaying] =
useState(false);
  const BASE_URL =
"http://192.168.0.107/storage/
";

  useEffect(() => {
    if (!mediaId) {
      Alert.alert("Error",
"Media tidak ditemukan");
      setTimeout(() =>
navigation.goBack(), 100);
      return;
    }
    fetchMediaDetail();
  }, [mediaId]);

  useFocusEffect(
    useCallback(() => {
      fetchMediaDetail();
    }, [mediaId])
  );

  const fetchMediaDetail =
async () => {
    try {
      const response = await
getMediaById(mediaId);
      setMedia(response.data);
    } catch (error) {
      Alert.alert("Error",
"Gagal mengambil data media");
      setTimeout(() =>
navigation.goBack(), 100);
    } finally {
      setLoading(false);
    }
  };

  const playAudio = async ()
=> {
    if (!media?.audio) {
      Alert.alert("Error",
"Audio tidak tersedia");
      return;
    }

    const audioUri =
media.audio.startsWith("http")
? media.audio :
`${BASE_URL}${media.audio}`;
```

```
    try {
      if (sound) {
        await
sound.stopAsync();
        await
sound.unloadAsync();
        setSound(null);
        setIsPlaying(false);
        return;
      }

      const { sound: newSound
} = await
Audio.Sound.createAsync({ uri:
audioUri });
      setSound(newSound);


newSound.setOnPlaybackStatusUp
date((status) => {
        if
(status.didJustFinish) {
          setIsPlaying(false);
          setSound(null);
        }
      });

      await
newSound.playAsync();
      setIsPlaying(true);
    } catch (error) {
      Alert.alert("Error",
"Gagal memutar audio");
      console.error("Audio
Error: ", error);
    }
  };

  const handleShareText =
async () => {
    try {
      const message =
`${media.judul}\n\n${media.des
kripsi}\n\nLihat gambarnya di
sini:
${media.foto.startsWith("http"
) ? media.foto :
`${BASE_URL}${media.foto}`}`;
      await Share.share({
message });
    } catch (error) {
      console.error("Share
text error:", error);
      Alert.alert("Error",
"Gagal membagikan teks");
    }
  };

  useEffect(() => {
    return sound
      ? () => {
          sound.unloadAsync();
          setIsPlaying(false);
        }
      : undefined;
  }, [sound]);

  if (loading) {
    return <ActivityIndicator
style={tw`flex-1`}
size="large" color="yellow"
/>;
  }

  return (
    <ImageBackground
source={require("../assets/kue
.jpg")} style={tw`flex-1 bg-
black`}>
      <View style={tw`absolute
inset-0 bg-black opacity-50`}
/>

      <ScrollView
contentContainerStyle={tw`flex
-grow justify-center items-
center p-6`}>
        <TouchableOpacity
          style={tw`bg-yellow-
500 px-6 py-3 rounded-full
shadow-lg mb-4`}
          onPress={() =>
navigation.navigate("Dashboard
Category")}
        >
          <Text
style={tw`text-white font-bold
text-lg`}>Kembali</Text>
        </TouchableOpacity>

        {media ? (
          <>
```

```
            <Text
style={tw`text-3xl font-bold
text-white text-center mb-
4`}>{media.judul}</Text>

            {media.foto && (
              <Image source={{
uri: media.foto }}
style={tw`w-full h-60 rounded-
lg shadow-lg mb-4`}
resizeMode="cover" />
            )}

            <View
style={tw`bg-white p-4
rounded-lg shadow-lg w-full`}>
              <Text
style={tw`text-lg text-gray-
800 text-center font-semibold
mb-2`}>Tentang Kue Ini</Text>
               <Text
style={tw`text-sm text-gray-
800 text-center font-semibold
mb-2`}>Rekomendasi
Toko:</Text>
               <Text
style={tw`text-gray-600 text-
center`}>{media.lokasi}</Text>
              <View
style={tw`border-b border-
gray-300 my-2`} />
              <Text
style={tw`text-sm text-gray-
800 text-center font-semibold
mb-2`}>Deskripsi:</Text>
              <Text
style={tw`text-gray-600 text-
center`}>{media.deskripsi}</Te
xt>
            </View>

            <TouchableOpacity
onPress={playAudio}
style={tw`bg-yellow-500 p-3
rounded-full mt-4`}>
              <Text
style={tw`text-white text-
center font-bold`}>
                {isPlaying ?
"Sedang Diputar..." : "Putar
Audio"}
              </Text>
</TouchableOpacity>

            <TouchableOpacity
              style={tw`border
border-yellow-500 px-6 py-3
rounded-full shadow-lg mt-4`}
              onPress={() =>
navigation.navigate("Media2",
{ id: mediaId })}
            >
              <Text
style={tw`text-yellow-500
font-bold text-lg`}>Lihat
Video</Text>

</TouchableOpacity>

            <TouchableOpacity
              style={tw`bg-
yellow-500 px-6 py-3 rounded-
full shadow-lg mt-4`}

onPress={handleShareText}
            >
              <Text
style={tw`text-white font-bold
text-lg`}>Bagikan info
Kue</Text>

</TouchableOpacity>
          </>
        ) : (
          <Text
style={tw`text-gray-300 text-
lg`}>Media tidak
ditemukan.</Text>
        )}
      </ScrollView>
    </ImageBackground>
  );
};

export default MediaDetail;
```

EditMedia.jsx

```
import React, { useEffect,
useState } from "react";
import { View, Text,
TextInput, TouchableOpacity,
```

```
Alert, Image,
ActivityIndicator, ScrollView
} from "react-native";
import { Picker } from
"@react-native-picker/picker";
import tw from "twrnc";
import { updateMedia,
getMedia, getCategories } from
"../api";

const EditMedia = ({ route,
navigation }) => {
  const { id } = route.params
|| {};

  const [judul, setJudul] =
useState("");
  const [deskripsi,
setDeskripsi] = useState("");
  const [resep, setResep] =
useState("");
  const [lokasi, setLokasi] =
useState("");
  const [categoryId,
setCategoryId] = useState("");
  const [foto, setFoto] =
useState(null);
  const [video, setVideo] =
useState(null);
  const [categories,
setCategories] = useState([]);
  const [isLoading,
setIsLoading] =
useState(false);
  const [loading, setLoading]
= useState(true);

  useEffect(() => {
    fetchMedia();
    fetchCategories();
  }, []);

  const fetchMedia = async ()
=> {
    try {
      const response = await
getMedia(id);
      const media =
response.data.find((item) =>
item.id === id);

      if (media) {
        setJudul(media.judul
|| "");

setDeskripsi(media.deskripsi
|| "");
        setResep(media.resep
|| "");
        setLokasi(media.lokasi
|| "");

setCategoryId(media.category_i
d?.toString() || "");

        if (media.foto) {
          setFoto({ uri:
media.foto });
        }
        if (media.video) {
          setVideo({ uri:
`http://192.168.0.107/storage/
${media.video}` });
        }
      } else {
        Alert.alert("Error",
"Media tidak ditemukan");
      }
    } catch (error) {
      Alert.alert("Error",
"Gagal mengambil data media");
    } finally {
      setLoading(false);
    }
  };

  const fetchCategories =
async () => {
    try {
      const response = await
getCategories();

setCategories(response.data);
    } catch (error) {
      Alert.alert("Error",
"Gagal mengambil kategori");
    }
  };

  const handleUpdate = async
() => {
    if (!judul || !deskripsi
|| !resep || !categoryId) {
```

```
      Alert.alert("Error",
"Semua bidang harus diisi");
      return;
    }

    setIsLoading(true);

    const payload = {
      judul,
      deskripsi,
      resep,
      lokasi,
      category_id: categoryId,
    };

    try {
      await updateMedia(id,
payload);
      Alert.alert("Sukses",
"Media berhasil diperbarui");

navigation.navigate("Dashboard
Category");
    } catch (error) {
      Alert.alert("Error",
"Gagal memperbarui media");
    } finally {
      setIsLoading(false);
    }
  };

  if (loading) {
    return <ActivityIndicator
style={tw`flex-1`}
size="large" color="#FF5733"
/>;
  }

  return (
    <View style={tw`flex-1 bg-
gray-100`}>
      <ScrollView
contentContainerStyle={tw`p-
6`}>
        <Text style={tw`text-
[#FF5733] font-bold text-2xl
mb-6 text-center`}>Edit
Media</Text>

        <TextInput
          style={tw`border
border-gray-300 bg-white p-3
rounded-lg mb-4`}
          placeholder="Judul"
          value={judul}

onChangeText={setJudul}
        />
        <TextInput
          style={tw`border
border-gray-300 bg-white p-3
rounded-lg mb-4`}

placeholder="Deskripsi"
          value={deskripsi}

onChangeText={setDeskripsi}
        />
        <TextInput
          style={tw`border
border-gray-300 bg-white p-3
rounded-lg mb-4`}
          placeholder="Resep"
          value={resep}

onChangeText={setResep}
        />
        <TextInput
          style={tw`border
border-gray-300 bg-white p-3
rounded-lg mb-4`}
          placeholder="Lokasi"
          value={lokasi}

onChangeText={setLokasi}
        />
        <Picker

selectedValue={categoryId}

onValueChange={(itemValue) =>
setCategoryId(itemValue)}
          style={tw`border
border-gray-300 bg-white p-3
rounded-lg mb-4`}
        >
          <Picker.Item
label="Pilih Kategori"
value="" />

{categories.map((category) =>
(
```

```
            <Picker.Item
key={category.id}
label={category.kategori}
value={category.id} />
          ))}
        </Picker>

        {foto && <Image
source={{ uri: foto.uri }}
style={tw`w-full h-40 rounded-
lg mb-4`} />}
        {video && <Text
style={tw`text-gray-700 mb-4
text-center`}>Video telah
dipilih</Text>}

        <Text style={tw`text-
center text-red-500 mb-
6`}>Foto dan video tidak dapat
diedit.</Text>

        <TouchableOpacity
          style={[tw`bg-
[#FF5733] p-4 rounded-lg`,
isLoading && tw`bg-gray-400`]}

onPress={handleUpdate}
          disabled={isLoading}
        >
          <Text
style={tw`text-white text-
center font-bold text-lg`}>
            {isLoading ?
"Menyimpan..." : "Simpan
Perubahan"}
          </Text>
        </TouchableOpacity>
      </ScrollView>
    </View>
  );
};

export default EditMedia;


0001_01_01_000000_create_users
_table.php

<?php
use
Illuminate\Database\Migrations
\Migration;
use
Illuminate\Database\Schema\Blu
eprint;
use
Illuminate\Support\Facades\Sch
ema;

return new class extends
Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {

Schema::create('users',
function (Blueprint $table) {
            $table->id();
            $table-
>string('name');
            $table-
>string('email')->unique();
            $table-
>timestamp('email_verified_at'
)->nullable();
            $table-
>string('password');
            $table-
>rememberToken();
            $table-
>timestamps();
        });


Schema::create('password_reset
_tokens', function (Blueprint
$table) {
            $table-
>string('email')->primary();
            $table-
>string('token');
            $table-
>timestamp('created_at')-
>nullable();
        });
```

```
Schema::create('sessions',
function (Blueprint $table) {
            $table-
>string('id')->primary();
            $table-
>foreignId('user_id')-
>nullable()->index();
            $table-
>string('ip_address', 45)-
>nullable();
            $table-
>text('user_agent')-
>nullable();
            $table-
>longText('payload');
            $table-
>integer('last_activity')-
>index();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down():
void
    {

Schema::dropIfExists('users');

Schema::dropIfExists('password
_reset_tokens');

Schema::dropIfExists('sessions
');
    }
};
```

0001_01_01_000001_create_cache_table.php

```
<?php

use
Illuminate\Database\Migrations
\Migration;
use
Illuminate\Database\Schema\Blu
eprint;
use
Illuminate\Support\Facades\Sch
ema;

return new class extends
Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {

Schema::create('cache',
function (Blueprint $table) {
            $table-
>string('key')->primary();
            $table-
>mediumText('value');
            $table-
>integer('expiration');
        });


Schema::create('cache_locks',
function (Blueprint $table) {
            $table-
>string('key')->primary();
            $table-
>string('owner');
            $table-
>integer('expiration');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down():
void
    {

Schema::dropIfExists('cache');

Schema::dropIfExists('cache_lo
cks');
    }
};
```

001_01_01_000002_create_jobs_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('jobs', function (Blueprint $table) {
            $table->id();
            $table->string('queue')->index();
            $table->longText('payload');
            $table->unsignedTinyInteger('attempts');
            $table->unsignedInteger('reserved_at')->nullable();
            $table->unsignedInteger('available_at');
            $table->unsignedInteger('created_at');
        });

        Schema::create('job_batches', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->string('name');
            $table->integer('total_jobs');
            $table->integer('pending_jobs');
            $table->integer('failed_jobs');
            $table->longText('failed_job_ids');
            $table->mediumText('options')->nullable();
            $table->integer('cancelled_at')->nullable();
            $table->integer('created_at');
            $table->integer('finished_at')->nullable();
        });

        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('jobs');
        Schema::dropIfExists('job_batches');
```

```
Schema::dropIfExists('failed_jobs');
    }
};
```

2025_02_25_115025_create_personal_access_tokens_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {

Schema::create('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamp('expires_at')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {

Schema::dropIfExists('personal_access_tokens');
    }
};
```

2025_03_01_171735_create_categories_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {

Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('kategori');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
```

```
     */
    public function down(): 
void
    {

Schema::dropIfExists('categori
es');
    }
};


2025_03_03_125626_create_media
_table.php

<?php

use
Illuminate\Database\Migrations
\Migration;
use
Illuminate\Database\Schema\Blu
eprint;
use
Illuminate\Support\Facades\Sch
ema;

return new class extends
Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {

Schema::create('media',
function (Blueprint $table) {
            $table->id();
            $table-
>string('judul');
            $table-
>text('deskripsi');
            $table-
>text('resep');
            $table-
>string('lokasi')->nullable();
            $table-
>foreignId('category_id')-
>constrained('categories')-
>onDelete('cascade');
            $table-
>string('foto')->nullable();
            $table-
>string('video')->nullable();
            $table-
>string('audio')->nullable();
            $table-
>timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): 
void
    {

Schema::dropIfExists('media');
    }
};


User.php

<?php

namespace App\Models;

// use
Illuminate\Contracts\Auth\Must
VerifyEmail;
use
Illuminate\Database\Eloquent\F
actories\HasFactory;
use
Illuminate\Foundation\Auth\Use
r as Authenticatable;
use
Illuminate\Notifications\Notif
iable;
use
Laravel\Sanctum\HasApiTokens;

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

    /**
```

```
     * The attributes that are
mass assignable.
     *
     * @var list<string>
     */
    protected $fillable = [
        'name',
        'email',
        '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',
        ];
    }
}
```

Category.php

```
<?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'];

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

Media.php

```
<?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',
        'resep',
        'category_id',
        'foto',
        'lokasi',
        'video',
        'audio'
    ];

    public function category()
    {
        return $this-
>belongsTo(Category::class);
```

```
    }
    public function
getFotoAttribute($value)
{
    return $value ?
url('storage/' . $value) :
null;
}
}
```

Controller.php

```
<?php

namespace
App\Http\Controllers;

abstract class Controller
{
    //
}
```

MediaController.php

```
<?php

namespace
App\Http\Controllers;

use App\Models\Media;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use
Illuminate\Support\Facades\Sto
rage;

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',
            'resep' =>
'required|string',
            'lokasi' =>
'required|string',
            'category_id' =>
'required|exists:categories,id
',
            'foto' =>
'nullable|image|mimes:jpeg,png
,jpg|max:10048',
            'video' =>
'nullable|mimes:mp4,mov,avi|ma
x:200240',
           'audio' =>
'nullable|file|max:51200',

        ]);

        $data = $request-
>all();

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

        if ($request-
>hasFile('video')) {
            $data['video'] =
$request->file('video')-
>store('uploads/video',
'public');
        }
        if ($request-
>hasFile('audio')) {
            $audioPath =
 $request->file('audio')-
>storeAs('uploads/audio',
uniqid() . '.m4a', 'public');
            $data['audio'] =
$audioPath;
        }
```

```
        $media =
Media::create($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',
            'resep' =>
'sometimes|required|string',
            'lokasi' =>
'sometimes|required|string',
            'category_id' =>
'sometimes|required|exists:cat
egories,id',
            'foto' =>
'nullable|image|mimes:jpeg,png
,jpg|max:10048',
            'video' =>
'nullable|mimes:mp4,mov,avi|ma
x:200240',
            'audio' =>
'nullable|mimes:mp3,wav,m4a|ma
x:51200', // Maks 50MB

        ]);

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

CategoryController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Category;
use Illuminate\Http\Request;
use Illuminate\Http\Response;

class CategoryController extends Controller
{
    public function index()
    {
        return response()->json(Category::with('media')->get(), 200);
    }

    public function store(Request $request)
    {
        $request->validate([
            'kategori' => 'required|string|max:255',
        ]);

        $category = Category::create(['kategori' => $request->kategori]);
        return response()->json($category, 201);
    }

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

    public function update(Request $request, Category $category)
    {
        $request->validate([
            'kategori' => 'required|string|max:255,' . $category->id,
        ]);

        $category->update(['kategori' => $request->kategori]);
        return response()->json($category, 200);
    }

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

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

api.php

```
<?php

use Illuminate\Http\Request;
use App\Http\Controllers\Auth\AuthenticatedSessionController;
```

```
use
App\Http\Controllers\Auth\Emai
lVerificationNotificationContr
oller;
use
App\Http\Controllers\Auth\NewP
asswordController;
use
App\Http\Controllers\Auth\Pass
wordResetLinkController;
use
App\Http\Controllers\Auth\Regi
steredUserController;
use
App\Http\Controllers\Auth\Veri
fyEmailController;
use
App\Http\Controllers\MediaCont
roller;
use
App\Http\Controllers\CategoryC
ontroller;
use
Illuminate\Support\Facades\Rou
te;

Route::middleware(['auth:sanct
um'])->get('/user', function
(Request $request) {
    return $request->user();
});
Route::post('/register',
[RegisteredUserController::cla
ss, 'store'])
    ->middleware('guest')
    ->name('register');

Route::post('/login',
[AuthenticatedSessionControlle
r::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/verificati
on-notification',
[EmailVerificationNotification
Controller::class, 'store'])
    ->middleware(['auth',
'throttle:6,1'])
    -
>name('verification.send');

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

// Routes untuk kategori
Route::get('/categories',
[CategoryController::class,
'index']);
Route::post('/categories',
[CategoryController::class,
'store']);
Route::get('/categories/{categ
ory}',
[CategoryController::class,
'show']);
Route::put('/categories/{categ
ory}',
[CategoryController::class,
'update']);
Route::delete('/categories/{ca
tegory}',
[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']);
```

api.js

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

// const API_URL =
'https://kue.proyekk.com/api';
const API_URL =
'http://192.168.0.107/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('userToke
n', response.data.token);

api.defaults.headers.common['A
uthorization'] = `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
(data) => {
    try {
        console.log('
Memulai proses login...',
data);

        const response = await
api.post('/login', data);
        const token =
response.data.token;

        if (!token) throw {
message: 'Token tidak diterima
dari server' };

        // Simpan token ke
AsyncStorage
        await
AsyncStorage.setItem('userToke
n', token);
        console.log('
Token
tersimpan:', token);

        return response.data;
    } catch (error) {
        console.error('+
Error saat login:',
```

```
error.response?.data ||
error);
        throw
error.response?.data || {
message: 'Login gagal!' };
    }
};

// 🔹 Logout
export const logout = async ()
=> {
    try {
        const token = await
AsyncStorage.getItem('userToke
n');
        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('userT
oken'); // 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
(token = null) => {
    try {
        const headers = token
? { Authorization: `Bearer
${token}` } : {};
        const response = await
api.get('/user', { headers });
        return response.data;
    } catch (error) {
        throw
error.response?.data || {
message: 'Gagal mendapatkan
data user!' };
    }
};

// API untuk kategori
export const getCategories =
() => api.get('/categories');
export const createCategory =
(kategori) =>
api.post('/categories', {
kategori });
export const getCategory =
(id) =>
api.get(`/categories/${id}`);
export const updateCategory =
(id, kategori) =>
api.put(`/categories/${id}`, {
kategori });
export const deleteCategory =
(id) =>
api.delete(`/categories/${id}`
);

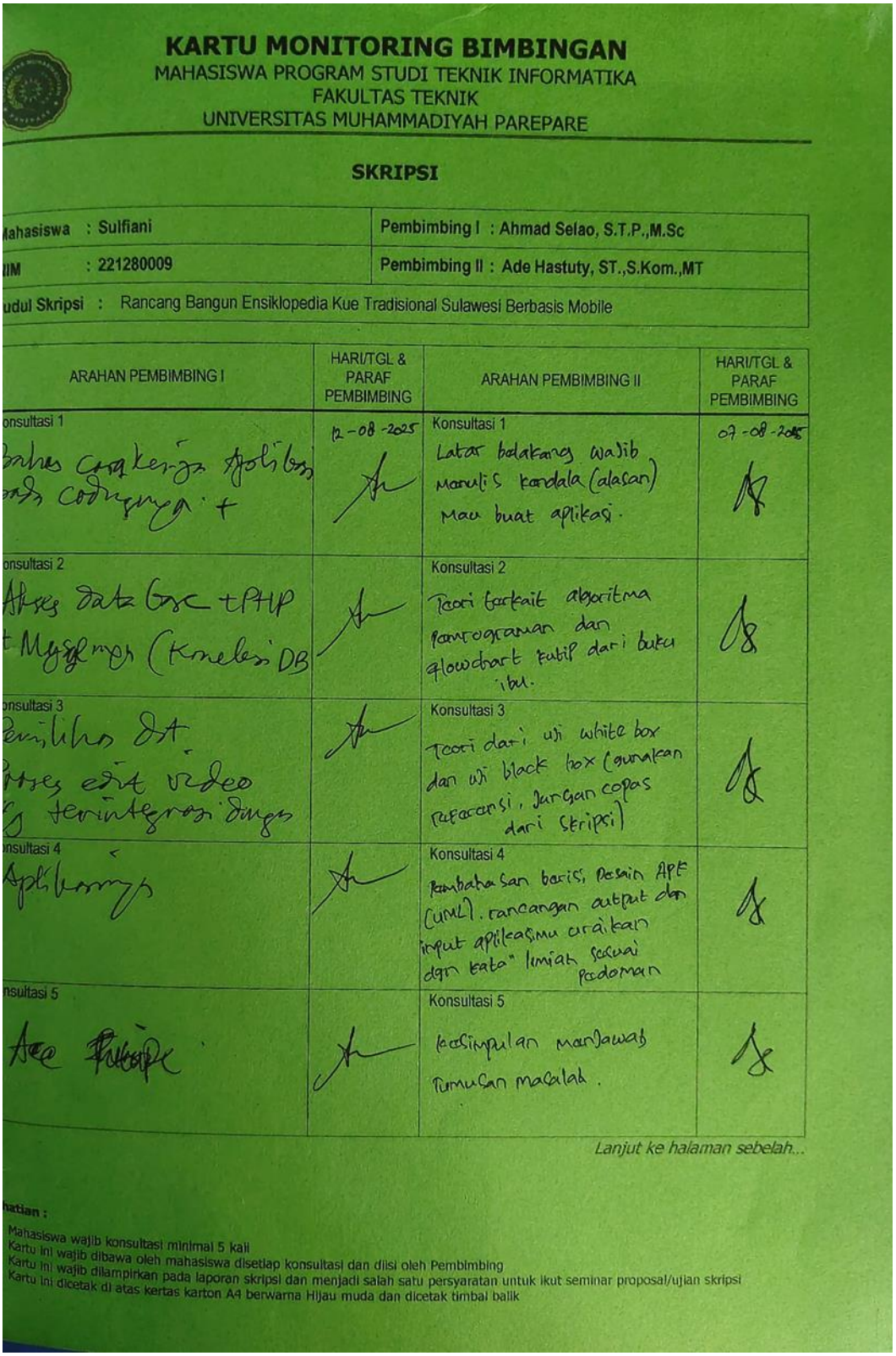
// API untuk media
export const getMedia = () =>
api.get('/media');
export const getMediaById =
(id) =>
api.get(`/media/${id}`);
export const deleteMedia =
(id) =>
api.delete(`/media/${id}`);

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
    });
  };


export default api;
```



KARTU MONITORING BIMBINGAN
MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : Sulfiani	Pembimbing I : Ahmad Selao, S.T.P.,M.Sc
NIM : 221280009	Pembimbing II : Ade Hastuty, ST.,S.Kom.,MT
Judul Skripsi : Rancang Bangun Ensiklopedia Kue Tradisional Sulawesi Berbasis Mobile	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Bahas cara kerja aplikasi ... codingnya :+	12-08-2025	Konsultasi 1 Latar belakang wajib Manulis kendala (alasan) Mau buat aplikasi.	07-08-2025
Konsultasi 2 Akses data Gsc + PHP + Mysql nya (Konelesi DB		Konsultasi 2 Teori terkait algoritma pemrograman dan flowchart tulis dari buku ibu.	
Konsultasi 3 Perbaiki DA. Akses edit video yg terintegrasi dengan		Konsultasi 3 Teori dari uji white box dan uji black box (gunakan referensi, jangan copas dari skripsi)	
Konsultasi 4 Aplikasinya		Konsultasi 4 Tambahkan bab isi, Desain APE (UML), rancangan output dan input aplikasimu uraikan dgn kata" lmiah sesuai pedoman	
Konsultasi 5 Acc Proposal		Konsultasi 5 Kesimpulan menjawab rumusan masalah.	

Lanjut ke halaman sebelah...

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

Lanjutan ...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 6		Konsultasi 6 Saran tulis fitur-fitur apa saja yang perlu ada di aplikasimu untuk pengembangan ke depan	
Konsultasi 7		Konsultasi 7 ABSTRAK = ringkasan isi skripsi.	
Konsultasi 8		Konsultasi 8 ACC - Seminar Tutup	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 14 - Agustus - 2025

Mengetahui
Plt.Ketua Program Studi

Mugaffir Yunus, ST.,MT
NBM. 1252 809

Mahasiswa

Sulfiani
NIM. 221280009

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