LAMPIRAN 3 Source Code Web

```
import Button from '@/components/Button'

import Input from '@/components/Input'

import Modal, { useModal } from '@/components/Modal'

import { CustomTableStyle } from
'@/components/table/CustomTableStyle'

import { CONFIG } from '@/config'

import axios from 'axios'

import { PencilIcon, PlusIcon, SaveAllIcon, Trash2Icon, TrashIcon
} from 'lucide-react'

import { useRouter } from 'next/router'

import React, { useEffect, useState } from 'react'

import DataTable from 'react-data-table-component'

import ReactSelect from 'react-select'

import Swal from 'sweetalert2'



export async function getServerSideProps(context: any) {

    try {

        const { page, limit, search } = context.query;

        const result = await axios.get(CONFIG.base_url_api +
`/datasets/list?page=${+page || 1}&limit=${+limit ||
10}&search=${search || ""}`)

        const diseases = await axios.get(CONFIG.base_url_api +
`/diseases/list?limit=999999`)

        const medicines = await axios.get(CONFIG.base_url_api +
`/medicines/list?limit=999999`)

        const symptoms = await axios.get(CONFIG.base_url_api +
`/symptoms/list?limit=999999`)

        console.log(result.data);

        return {

            props: {
```

```
                table: result?.data || [],
                diseases: diseases?.data?.items || [],
                symptoms: symptoms?.data?.items || [],
                medicines: medicines?.data?.items || []
            }
        }
    } catch (error: any) {
        console.log(error);
        return {
            props: {
                error: "Error internal server",
            }
        }
    }
}


export default function Medicine({ table, diseases, symptoms,
medicines }: any) {
    const router = useRouter();
    const [filter, setFilter] = useState<any>(router.query)
    const [show, setShow] = useState<boolean>(false)
    const [modal, setModal] = useState<useModal>()
    const [selected, setSelected] = useState<any>()
    const SYMPTOMS: any[] = [...symptoms?.map((v: any) => ({ ...v,
value: v?.id, label: v?.name?.toUpperCase() }))]
    const DISEASES: any[] = [{ value: "", label: "Pilih Penyakit"
}, ...diseases?.map((v: any) => ({ ...v, value: v?.id, label:
v?.name }))]
    const MEDICINES: any[] = [{ value: "", label: "Pilih
Rekomendasi Obat" }, ...medicines?.map((v: any) => ({ ...v, value:
v?.id, label: v?.name }))]
```

```
    useEffect(() => {
        if (typeof window !== 'undefined') {
            setShow(true)
        }
    }, [])
    useEffect(() => {
        const queryFilter = new
URLSearchParams(filter).toString();
        router.push(`?${queryFilter}`)
    }, [filter])
    const CustomerColumn: any = [
        {
            name: "Gejala",
            sortable: true,
            selector: (row: any) => row?.symptoms
        },
        {
            name: "Lama Sakit",
            sortable: true,
            selector: (row: any) => row?.period + " Hari"
        },
        {
            name: "Tingkat Kesakitan",
            sortable: true,
            selector: (row: any) => levels?.find((v:any) =>
v?.value == row?.level)?.label
        },
        {
            name: "Diagnosa",
```

```
            sortable: true,
            selector: (row: any) => row?.diagnose
        },
        {
            name: "Rekomendasi Obat",
            sortable: true,
            selector: (row: any) => row?.medicine
        },
        {
            name: "Aksi",
            right: true,
            selector: (row: any) => <div className='flex gap-2'>
                <Button title='Edit' color='primary' onClick={()
=> {
                    setModal({ ...modal, open: true, data: row,
key: "update" })
                }}>
                    <PencilIcon className='text-white w-5 h-5' />
                </Button>
                <Button title='Hapus' color='danger' onClick={()
=> {
                    setModal({ ...modal, open: true, data: row,
key: "delete" })
                }}>
                    <TrashIcon className='text-white w-5 h-5' />
                </Button>
            </div>
        },
    ]
```

```
    const levels = [
        { value: "", label: "Pilih Tingkat Kesakitan" },
        { value: 1, label: "Tidak Gawat Darurat" },
        { value: 2, label: "Gawat Darurat" },
        { value: 3, label: "Sangat Gawat Darurat" }
    ]

    const onSubmit = async (e: any) => {
        e?.preventDefault();
        const formData = Object.fromEntries(new
FormData(e.target))
        try {
            const payload = {
                ...formData,
                period: +formData?.period,
                level: +formData?.level,
                symptoms: selected?.symptoms?.map((v: any) =>
v?.name?.toUpperCase())?.join(" | "),
                diagnose: selected?.diagnose?.name,
                medicine: selected?.medicine?.name,
            }
            if (formData?.id) {
                const result = await
axios.patch(CONFIG.base_url_api +
`/datasets/update/${formData?.id}`, payload)
            } else {
                const result = await
axios.post(CONFIG.base_url_api + `/datasets/create`, payload)
            }
            Swal.fire({
```

```
                icon: "success",
                text: "Data Berhasil Disimpan"
            })
            setModal({ ...modal, open: false })
            router.push('')
        } catch (error) {
            console.log(error);
        }
    }
    const onRemove = async (e: any) => {
        try {
            e?.preventDefault();
            const formData = Object.fromEntries(new
FormData(e.target))
            const result = await axios.delete(CONFIG.base_url_api
+ `/datasets/delete/${formData?.id}`)
            Swal.fire({
                icon: "success",
                text: "Data Berhasil Dihapus"
            })
            setModal({ ...modal, open: false })
            router.push('')
        }
        catch (error) {
            console.log(error);
        }
    }
    return (
        <div>
```

```
            <h2 className='text-2xl font-semibold'>Dataset</h2>


            <div className='mt-5'>

                <div className='flex lg:flex-row flex-col justify-
between items-center'>

                    <div className='lg:w-auto w-full'>

                        <Input label='' type='search'
placeholder='Cari disini...' defaultValue={filter?.search}
onChange={(e) => {

                            setFilter({ ...filter, search:
e.target.value })

                        }} />

                    </div>

                    <div className='lg:w-auto w-full'>

                        <Button type='button' color='info'
className={'flex gap-2 px-2 items-center lg:justify-start justify-
center'} onClick={() => {

                            setModal({ ...modal, open: true, data:
null, key: "create" })

                        }}>

                            <PlusIcon className='w-4' />

                            Dataset

                        </Button>

                    </div>

                </div>

                <div className='mt-5'>

                    {

                        show &&

                        <DataTable

                            pagination

                            onChangePage={(pageData) => {
```

```
                                setFilter({ ...filter, page:
pageData })

                            }}

                            onChangeRowsPerPage={(currentRow,
currentPage) => {

                                setFilter({ ...filter, page:
currentPage, limit: currentRow })

                            }}

                            responsive={true}

                            paginationTotalRows={table?.total_item
s}

                            paginationDefaultPage={1}

                            paginationServer={true}

                            striped

                            columns={CustomerColumn}

                            data={table?.items}

                            customStyles={CustomTableStyle}

                        />

                    }

                </div>

                {

                    modal?.key == "create" || modal?.key ==
"update" ? <Modal open={modal.open} setOpen={() => setModal({
...modal, open: false })}>

                        <h2 className='text-xl font-semibold text-
center'>{modal.key == 'create' ? "Tambah" : "Ubah"} Dataset
Penyakit</h2>

                        <form onSubmit={onSubmit}>

                            {

                                modal.key == "update" &&

                                <input type="hidden" name="id"
value={modal?.data?.id || null} />
```

```
                                }

                                <div className='mt-2'>

                                    <label htmlFor="symptoms"
className='text-gray-500'>Gejala</label>

                                    <ReactSelect

                                        options={SYMPTOMS}

                                        id='symptoms'

                                        name='symptoms'

                                        maxMenuHeight={150}

                                        defaultValue={

                                            modal?.data?.symptoms ?

                                                SYMPTOMS?.filter((v:
any) => modal?.data?.symptoms?.split(" |
")?.includes(v?.name?.toUpperCase()))

                                                : ""

                                        }

                                        placeholder="Pilih Gejala"

                                        isMulti

                                        onChange={(e) => {

                                            setSelected({ ...selected,
symptoms: e })

                                        }}

                                    />

                                </div>

                                <Input label='Lama Sakit (Hari)'
placeholder='Masukkan Lama Sakit' type='number' name='period'
defaultValue={modal?.data?.period || ""} required />

                                <div className='mt-2'>

                                    <label htmlFor="level"
className='text-gray-500'>Tingkat Kesakitan</label>

                                    <ReactSelect
```

```
                                    options={levels}
                                    id='level'
                                    name='level'
                                    maxMenuHeight={100}
                                    defaultValue={
                                        modal?.data?.level ?
                                            { value:
modal?.data?.level, label: levels?.find((v: any) => v?.value ==
modal?.data?.level)?.label }
                                            : levels[0]
                                    }
                                />
                            </div>
                            <div className='mt-2 relative'>
                                <label htmlFor="disease_id"
className='text-gray-500'>Diagnosa</label>
                                <ReactSelect
                                    options={DISEASES}
                                    id='disease_id'
                                    name='disease_id'
                                    maxMenuHeight={150}
                                    onChange={(e) => {
                                        setSelected({ ...selected,
diagnose: e })
                                    }}
                                    defaultValue={
                                        modal?.data?.disease_id ?
                                            { value:
modal?.data?.disease_id, label: modal?.data?.diagnose }
                                            : DISEASES?.[0]
```

```
                                    }
                                    required
                                    menuPlacement='top'
                                />
                            </div>
                            <div className='mt-2 relative'>
                                <label htmlFor="medicine_id"
className='text-gray-500'>Rekomendasi Obat</label>
                                <ReactSelect
                                    options={MEDICINES}
                                    id='medicine_id'
                                    name='medicine_id'
                                    maxMenuHeight={150}
                                    onChange={(e) => {
                                        setSelected({ ...selected,
medicine: e })
                                    }}
                                    required
                                    defaultValue={
                                        modal?.data?.medicine_id ?
                                            { value:
modal?.data?.medicine_id, label: modal?.data?.medicine }
                                            : MEDICINES?.[0]
                                    }
                                    menuPlacement='top'
                                />
                            </div>
                            <div className='flex lg:gap-2 gap-0
lg:flex-row flex-col-reverse justify-end'>
                                <div>
```

```
                                    <Button color='white'
type='button' onClick={() => {
                                        setModal({ open: false })
                                    }}>
                                        Kembali
                                    </Button>
                                </div>

                                <div>
                                    <Button color='info'
className={'flex gap-2 px-2 items-center justify-center'}>
                                        <SaveAllIcon className='w-
4 h-4' />
                                        Simpan
                                    </Button>
                                </div>

                            </div>
                        </form>
                    </Modal>
                    : ""
                }
                {
                    modal?.key == "delete" ? <Modal
open={modal.open} setOpen={() => setModal({ ...modal, open: false
})}>
                        <h2 className='text-xl font-semibold text-
center'>Hapus Dataset Penyakit</h2>
                        <form onSubmit={onRemove}>
                            <input type="hidden" name="id"
value={modal?.data?.id} />
```

```
                                <p className='text-center my-2'>Apakah
anda yakin ingin menghapus data {modal?.data?.name}?</p>

                                <div className='flex gap-2 lg:flex-row
flex-col-reverse justify-end'>

                                    <div>

                                        <Button color='white'
type='button' onClick={() => {

                                            setModal({ open: false })

                                        }}>

                                            Kembali

                                        </Button>

                                    </div>


                                    <div>

                                        <Button color='danger'
className={'flex gap-2 px-2 items-center justify-center'}>

                                            <Trash2Icon className='w-4
h-4' />

                                            Hapus

                                        </Button>

                                    </div>


                                </div>

                            </form>

                        </Modal>

                            : ""

                    }

                </div>

            </div>

    )
}
```

LAMPIRAN 4 Source Code Aplikasi Mobile

```
import 'package:aplikasi_sistem_pakar/app.dart';
import
'package:aplikasi_sistem_pakar/features/home/cubit/home_cubit.dart';
import
'package:aplikasi_sistem_pakar/widget/custom_input_field.dart';
import 'package:aplikasi_sistem_pakar/widget/dialog_progress.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:google_fonts/google_fonts.dart';

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return BlocProvider(
      create: (context) => HomeCubit()..initCubit(),
      child: BlocConsumer<HomeCubit, HomeState>(
        listener: (context, state) {
          if (state is HomeLoading) {
            showProgressDialog(context: context);
          }
          if (state is HomeLoaded) {
            Navigator.pop(context);
          }
        },
        builder: (context, state) {
          final cubit = context.read<HomeCubit>();
          return Scaffold(
            appBar: AppBar(
              backgroundColor: Colors.blue,
              title: Text(
                "Home Page",
                style: GoogleFonts.urbanist(
                  color: Colors.white,
                ),
              ),
              actions: [
                IconButton(
                  onPressed: () =>
Navigator.pushNamedAndRemoveUntil(
                      context, Routes.login, (route) => false),
                  icon: const Icon(
                    Icons.logout,
                    color: Colors.white,
                  ),
                )
              ],
            ),
            bottomNavigationBar: Container(
              padding: const EdgeInsets.all(10),
```

```
                height: 250.h,
                child: Column(
                  children: [
                    Expanded(
                      child: SingleChildScrollView(
                        child: Column(
                          children: [
                            Text(
                              "Hasil Diagnosa :",
                              style: GoogleFonts.poppins(
                                color: Colors.black,
                                fontWeight: FontWeight.w600,
                              ),
                            ),
                            SizedBox(
                              height: 8.h,
                            ),
                            Text(
                              cubit.diagnoseResult ?? '-',
                              style: GoogleFonts.poppins(
                                color: Colors.black,
                                fontWeight: FontWeight.w600,
                                fontSize: 14.sp,
                              ),
                            ),
                            SizedBox(
                              height: 18.h,
                            ),
                            Text(
                              "Rekomendasi Obat ${cubit.medicineResult
?? '-'}",
                              style: GoogleFonts.poppins(
                                color: Colors.black,
                                fontWeight: FontWeight.w300,
                                fontSize: 14.sp,
                              ),
                            ),
                          ],
                        ),
                      ),
                    ),
                    GestureDetector(
                      onTap: () {
                        if (cubit.lamaGejala.text.isEmpty &&
                            cubit.optionDarurat == 0) {
                          showMessageNotification(context);
                        } else {
                          cubit.postCubit();
                        }
                        primaryFocus?.unfocus();
                      },
                      child: Container(
                        height: 40.h,
                        width: MediaQuery.of(context).size.width.w,
```

```
                        decoration: BoxDecoration(
                            color: const Color(0xffB311FF),
                            borderRadius:

BorderRadius.all(Radius.circular(20.r))),
                        child: Center(
                          child: Text(
                            "Diagnose",
                            style: GoogleFonts.poppins(
                              color: Colors.white,
                              fontWeight: FontWeight.w600,
                            ),
                          ),
                        ),
                      ),
                    ),
                  ],
                ),
              ),
              body: SingleChildScrollView(
                child: Container(
                  margin: const EdgeInsets.all(8),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Container(
                        margin: const EdgeInsets.all(8),
                        child: PriorityDropdown(
                          onChanged: (value) {
                            cubit.updateOptions(value);
                          },
                        ),
                      ),
                      CustomInputField(
                        keyboardType: TextInputType.number,
                        controller: cubit.lamaGejala,
                        hintText: "Berapa lama anda mengalami penyakit
?",
                      ),
                      const SizedBox(
                        height: 14,
                      ),
                      Text(
                        "Pilihlah Gejala dibawah ini",
                        style: GoogleFonts.urbanist(
                          color: Colors.black,
                          fontSize: 20,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      const SizedBox(
                        height: 4,
                      ),
                      Column(
```

```
                          crossAxisAlignment:
CrossAxisAlignment.start,
                          children:
List.generate(cubit.allItems.length, (index) {
                            return Column(
                              children: [
                                CheckboxListTile(
                                  title: Text(
                                    cubit.allItems[index]['name'],
                                    style: GoogleFonts.urbanist(
                                      color: Colors.black,
                                    ),
                                  ),
                                  value: cubit.selectedItems

.contains(cubit.allItems[index]['id']),
                                  onChanged: (bool? value) {
                                    if (value == true) {

cubit.addItem(cubit.allItems[index]['id']);
                                    } else {
                                      cubit.removeItem(

cubit.allItems[index]['id']);

                                    }
                                  },
                                ),
                                Padding(
                                  padding: const EdgeInsets.symmetric(
                                      horizontal: 16.0),
                                  child: TextField(
                                    controller: cubit.textControllers[
                                        cubit.allItems[index]['id']],
                                    keyboardType:
TextInputType.number,

                                    decoration: const InputDecoration(
                                      labelText: "Periode gejala

berapa lama ?",

                                      border: OutlineInputBorder(),
                                    ),
                                    onChanged: (text) {},
                                    onEditingComplete: () {
                                      cubit.updateItemComment(
                                        cubit.allItems[index]['id'],
                                        cubit

.textControllers[cubit.allItems[index]
                                                    ['id']]!
                                                .text,
                                      );
                                      primaryFocus?.unfocus();
                                    },
                                  ),
                                ),
```

```
                                const SizedBox(height: 12),
                              ],
                            );
                          })),
                    ],
                  ),
                ),
              ),
            );
          },
        ),
      );
    }

    void showMessageNotification(
      BuildContext context,
    ) {
      final snackBar = SnackBar(
        content: const Text("Harap lengkapi data terlebih dahulu !"),
        backgroundColor: Colors.blue[400],
        behavior: SnackBarBehavior.floating,
        action: SnackBarAction(
          label: 'Tutup',
          textColor: Colors.white,
          onPressed: () {
            // Aksi saat notifikasi ditutup
          },
        ),
      );

      ScaffoldMessenger.of(context).showSnackBar(snackBar);
    }
}

class PriorityDropdown extends StatefulWidget {
  final Function(int) onChanged;

  const PriorityDropdown({super.key, required this.onChanged});

  @override
  _PriorityDropdownState createState() => _PriorityDropdownState();
}

class _PriorityDropdownState extends State<PriorityDropdown> {
  int? _selectedValue; // Nilai default

  @override
  Widget build(BuildContext context) {
    return DropdownButton<int>(
      hint: const Text("Pilih Tingkat Kesakitan"),
      value: _selectedValue,
      items: const [
        DropdownMenuItem<int>(
          value: 1,
```

```
            child: Text("Tidak Gawat Darurat"),
          ),
          DropdownMenuItem<int>(
            value: 2,
            child: Text("Gawat Darurat"),
          ),
          DropdownMenuItem<int>(
            value: 3,
            child: Text("Sangat Darurat"),
          ),
        ],
        onChanged: (int? value) {
          if (value != null) {
            setState(() {
              _selectedValue = value;
            });
            widget.onChanged(value);
          }
        },
      );
    }
}
```

LAMPIRAN 5 Tempat Studi Kasus