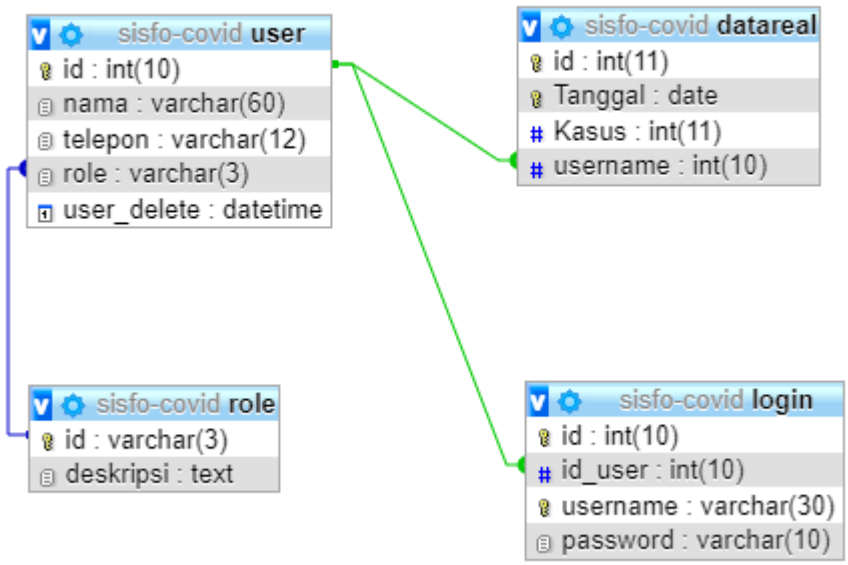
BAB V. IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi *Database*

Implementasi *database* yang dilakukan merupakan hasil dari perancangan pada Bab IV. *Database* ini memiliki 4 tabel yaitu yang pertama adalah user, datareal, dataprevaksin, dan datapascavaksin. Tabel user berisi tentang data admin pengguna aplikasi sedangkan ketiga tabel yang lain berisi data keseluruhan, data pre vaksin dan data pasca vaksin. Gambar … berikut ini menunjukkan hasil implementasi *database*



Gambar 5.1 Implementasi Database

5.1.1 Implementasi Tabel User

Tabel user adalah tabel yang berfungsi untuk menyimpan data admin atau user. Tabel ini terdiri dari beberapa kolom antara lain id, nama, telepon, username, role, dan user_delete Pada gambar … berikut adalah tabel user dari *database* sisfo-covid

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(10)			No	None		AUTO_INCREMENT	Change Drop More
2	nama	varchar(60)	latin1_swedish_ci		No	None			Change Drop More
3	telepon	varchar(12)	latin1_swedish_ci		No	None			Change Drop More
4	role	varchar(3)	latin1_swedish_ci		No	None			Change Drop More
5	user_delete	datetime			Yes	NULL			Change Drop More

Gambar 5.2 Implementasi Tabel User

5.1.2 Implementasi Tabel Login

Tabel Login digunakan untuk menyimpan *username* dan *password* pengguna. Atribut *username* dan *password* adalah tabel yang pertama kali dicek pada saat pengguna melakukan *login*. Kemudian apabila data pengguna ditemukan maka akan dilanjutkan untuk mencari role dan status pengguna. Gambar 5.3 adalah implementasi dari tabel *login*

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(10)			No	None		AUTO_INCREMENT	Change Drop More
2	id_user	int(10)			No	None			Change Drop More
3	username	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
4	password	varchar(10)	latin1_swedish_ci		No	None			Change Drop More

Gambar 5.3 Implementasi Tabel Login

5.1.3 Implementasi Tabel Role

Tabel *Role* adalah tabel yang menyimpan id dan deskrpsi mengenai *role* atau peran yang dimiliki setiap pengguna aplikasi. Setiap pengguna pada aplikasi memiliki peran dalam menggunakan aplikasi, peran tersebut dibagi menjadi 2 yaitu *admin* dan *user*. Gambar 5.4 adalah implementasi dari tabel *role*

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	varchar(3)	latin1_swedish_ci		No	None			Change Drop More
2	deskripsi	text	latin1_swedish_ci		No	None			Change Drop More

Gambar 5.4 Implementasi Tabel Role

5.1.4 Implementasi Tabel Datareal

Tabel datareal adalah tabel yang berfungsi untuk menyimpan data kasus positif secara keseluruhan dari tanggal pertama kali COVID-19 masuk ke Kota

Probolinggo hingga data paling baru. Tabel ini terdiri dari beberapa kolom antara lain id, Tanggal, dan Kasus. Pada gambar 5.5 berikut adalah tabel datareal dari *database* sisfo-covid.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	Tanggal	date			No	None			Change Drop More
3	Kasus	int(11)			No	None			Change Drop More
4	username	int(10)			No	None			Change Drop More

Gambar 5.5 Implementasi Tabel Datareal

5.2 Implementasi Fitur Login

Pada sub bab ini merupakan implementasi dari Bab IV yaitu perancangan halaman login. Pengguna aplikasi diharuskan login terlebih dahulu sebelum menggunakan aplikasi. Berikut ini menunjukkan source code dari fitur login

```
from tkinter import *
from tkinter import messagebox
from config import Config
import os

root = Tk()
root.title("Prediksi Covid v.1.0")
root.resizable(0, 0)
root.iconbitmap("Polinema.ico")

class Login(Config):
    def __init__(self, toplevel):
        self.toplevel = toplevel
        super(Login, self).__init__()
        lebar = 350
        tinggi = 500
        setTengahX = (self.toplevel.winfo_screenwidth()-
lebar)//2
        setTengahY = (self.toplevel.winfo_screenheight()-
tinggi)//2
        self.toplevel.geometry("%ix%i+%i+%i" % (lebar, tinggi,
```

```
setTengahX, setTengahY))
        self.komponen()

    def komponen(self):
        #atur frame
        self.top_level = Frame(self.toplevel,
background='#cedfe0')
        self.top_level.pack(side='top', fill='both',
expand='true')
        label_frame = Frame(self.top_level,
background='#808080', padx='10', pady='20')
        label_frame.pack(anchor='center', side='top', fill='x')
        header_frame = Frame(self.top_level,
background='#cedfe0', padx='10', pady='20')
        header_frame.pack(anchor='center', side='top', fill='x')
        input_frame = Frame(self.top_level,
background='#cedfe0', padx='20', pady='30')
        input_frame.pack(side='top', fill='both')
        button_frame = Frame(self.top_level,
background='#cedfe0', padx='20', pady='20')
        button_frame.pack(side='top', fill='x')

        #atur label
        Label(label_frame, background='#808080', text='Login',
font='{Segoe UI Semibold} 14 {}').pack(side='top')
        Label(header_frame, background='#cedfe0', text='Silahkan
lakukan login \n terlebih dahulu', font='{Segoe UI Semibold} 16
{}').pack(side='top')
        Label(input_frame, background='#cedfe0',
text='USERNAME', font='{Segoe UI Semibold} 12
{}').grid(column='0', row='0')
        Label(input_frame, background='#cedfe0',
text='PASSWORD', font='{Segoe UI Semibold} 12
{}').grid(column='0', row='2')

        #atur input
        self.inputUsername = Entry(input_frame, width='50')
        self.inputUsername.grid(column='0', row='1', pady='10')
        self.inputPassword = Entry(input_frame, width='50',
```

```
show='•')
        self.inputPassword.grid(column='0', row='3', pady='10')

        #button
        self.loginButton = Button(button_frame, font='{Segoe UI
Semibold} 11 {}', text='Login', width='33',
command=self.proses_login)
        self.loginButton.grid(column='0', row='4')

    def login_gagal(self):
        pass

    def proses_login(self):
        cursor = self._Config__db.cursor()
        verifikasi_username = self.inputUsername.get()
        verifikasi_password = self.inputPassword.get()
        if len(verifikasi_username or verifikasi_password) > 0:
            sql = "SELECT id_user FROM login WHERE username =
'{}' AND password = '{}'".format(verifikasi_username,
verifikasi_password)
            cursor.execute(sql)
            results = cursor.fetchone()
            if results:
                user_query = "SELECT role, user_delete FROM user
WHERE id = {}".format(results[0])
                cursor.execute(user_query)
                hasil_user = cursor.fetchone()
                if hasil_user[0] == 'ADM':
                   if str(hasil_user[1]) == 'None':
                       root.destroy()
                       os.system('halaman_utama_admin.py')
                   else:
                       messagebox.showerror(title='Error',
message='Pengguna ini tidak lagi aktif, kontak admin untuk
informasi lebih lanjut')
                elif hasil_user[0] == 'USR':
                    if str(hasil_user[1]) == 'None':
                        root.destroy()
                        os.system('halaman_utama_user.py')
```

```
                    else:
                        messagebox.showerror(title='Error',
message='Pengguna ini tidak lagi aktif, kontak admin untuk
informasi lebih lanjut')
            else:
                messagebox.showerror(title='Error',
message='Username atau Password Anda salah')
        else:
            messagebox.showerror(title='Error',
message='Username atau Password tidak boleh kosong')


Login(root)
root.mainloop()
```

5.3 Implementasi Fitur CRUD

Fitur CRUD digunakan untuk menambahkan data admin atau pengguna yang akan akan menggunakan aplikasi. Pengguna yang akan didaftarkan harus memasukkan data antara lain nama, username, password dan pengelola bagian. Berikut ini adalah source code dari fitur kelola data pengguna

```
from tkinter import *
from tkinter import ttk, messagebox
from config import Config
import os

root = Tk()
root.title("Prediksi Covid v.1.0")
root.resizable(0, 0)
root.iconbitmap("Polinema.ico")
# config = Config()
judul_kolom = ("ID", "Nama", "Username", "Password", "Bagian")

class Admin(Config):
    def __init__(self, parent):
        super(Admin, self).__init__()
```

```
        self.parent = parent
        lebar = 750
        tinggi = 740
        setTengahX = (self.parent.winfo_screenwidth()-lebar)//2
        setTengahY = (self.parent.winfo_screenheight()-
tinggi)//2
        self.parent.geometry("%ix%i+%i+%i" % (lebar, tinggi,
setTengahX, setTengahY))
        #self.parent.overrideredirect(1)
        self.komponen()


    def komponen(self):
        #atur frame
        top_level = Frame(self.parent, background='#cedfe0')
        top_level.pack(side=TOP, fill=X)
        Label(top_level, background='#808080', font='{Segoe UI
Semibold} 14 {}', text='Halaman Kelola Pengguna', padx='10',
pady='20').pack(fill='x', side='top')
        input_frame = Frame(top_level, background='#cedfe0')
        input_frame.pack(side=TOP, fill=X)
        self.frame_button = Frame(top_level,
background='#cedfe0')
        self.frame_button.pack(side=TOP, fill=X)
        self.frame_tabel = Frame(top_level)
        self.frame_tabel.pack(side=TOP, fill='both', expand=YES,
padx=10, pady=10)
        self.frame_menu = Frame(top_level, background='#cedfe0')
        self.frame_menu.pack(side=TOP, fill='y', expand=YES)

        #input frame_label
        Label(input_frame, background='#cedfe0', font='{Segoe UI
Semibold} 12 {}', text='Kode').grid(column='0', padx='30',
pady='10', row='0', sticky='w')
        Label(input_frame, background='#cedfe0', font='{Segoe UI
Semibold} 12 {}', text='Nama').grid(column='0', padx='30',
pady='10', row='1', sticky='w')
        Label(input_frame, background='#cedfe0', font='{Segoe UI
Semibold} 12 {}', text='Username').grid(column='0', padx='30',
```

```
pady='10', row='2', sticky='w')
        Label(input_frame, background='#cedfe0', font='{Segoe UI
Semibold} 12 {}', text='Password').grid(column='0', padx='30',
pady='10', row='3', sticky='w')
        Label(input_frame, background='#cedfe0', font='{Segoe UI
Semibold} 12 {}', text='Konfirmasi Password').grid(column='0',
padx='30', pady='10', row='4', sticky='w')
        Label(input_frame, background='#cedfe0', font='{Segoe UI
Semibold} 12 {}', text='Pengelola Bagian').grid(column='0',
padx='30', pady='10', row='5', sticky='w')

        #input frame entry
        readonlytext = StringVar()
        readonlytext.set('Dikembangkan Otomatis')
        self.inputKode = Entry(input_frame, width='25',
state='normal', textvariable=readonlytext)
        self.inputKode.grid(column='1', padx='20', row='0',
sticky='w')
        self.inputKode.config(state='disabled')
        self.inputNama = Entry(input_frame, width='50')
        self.inputNama.grid(column='1', padx='20', row='1',
sticky='w')
        self.inputUsername = Entry(input_frame, width='30')
        self.inputUsername.grid(column='1', padx='20', row='2',
sticky='w')
        self.inputPassword = Entry(input_frame, width='25',
show='•')
        self.inputPassword.grid(column='1', padx='20', row='3',
sticky='w')
        self.inputKonfirmasi = Entry(input_frame, widt='25',
show='•')
        self.inputKonfirmasi.grid(column='1', padx='20',
row='4', sticky='w')
        self.inputBagian = Entry(input_frame, width='25')
        self.inputBagian.grid(column='1', padx='20', row='5',
sticky='w')

        #button frame
        self.saveButton = Button(self.frame_button,
```

```
command=self.onSave, font='{Segoe UI Semibold} 10 {}',
relief='groove', text='Simpan', width='8')
        self.saveButton.grid(column='0', padx='30', pady='20',
row='0', sticky='w')
        self.updateButton = Button(self.frame_button,
font='{Segoe UI Semibold} 10 {}', relief='groove',
state='disabled', text='Update', width='8',
command=self.onUpdate)
        self.updateButton.grid(column='1', padx='30', pady='20',
row='0', sticky='w')
        self.deleteButton = Button(self.frame_button,
command=self.onDelete, state='disabled', font='{Segoe UI
Semibold} 10 {}', relief='groove', text='Hapus', width='8')
        self.deleteButton.grid(column='2', row='0', padx='30',
pady='20', sticky='w')
        self.clearButton = Button(self.frame_button,
font='{Segoe UI Semibold} 10 {}', command=self.onClear,
relief='groove', text='Clear', width='8')
        self.clearButton.grid(column='3', row='0', sticky='w',
padx='30')
        self.backButton = Button(self.frame_menu, font='{Segoe
UI Semibold} 10 {}', relief='groove', text='Kembali', width='8',
command=self.onKembali)
        self.backButton.grid(column='0', row='0', padx='30',
pady='10', sticky='w')

        #tabel
        self.trvTabel = ttk.Treeview(self.frame_tabel,
columns=judul_kolom, show='headings')
        self.trvTabel.bind("<Double-1>", self.onDoubleClick)
        sbVer = Scrollbar(self.frame_tabel, orient='vertical',
command=self.trvTabel.yview)
        sbVer.pack(side=RIGHT, fill=BOTH)
        self.trvTabel.pack(side=TOP, fill=BOTH, padx=10,
pady=10)
        self.trvTabel.configure(yscrollcommand=sbVer.set)
        self.table()

    def onSave(self):
```

```
        entNama = self.inputNama.get()
        entUsername = self.inputUsername.get()
        entPassword = self.inputPassword.get()
        entKonfirmasi = self.inputKonfirmasi.get()
        entBagian = self.inputBagian.get()

        if len(entNama or entUsername or entPassword or
entKonfirmasi or entBagian) > 0 :
            if entPassword != entKonfirmasi :
                messagebox.showwarning(title='Error',
message='Konfirmasi password tidak sama!')
            else:
                self.create(entNama, entUsername, entPassword,
entBagian)

self.trvTabel.delete(*self.trvTabel.get_children())
                self.frame_tabel.after(0, self.table())
                self.clear()
        else:
            messagebox.showerror(title='Error', message='Field
harus lengkap')

    def onDelete(self):
        konfirmasi_hapus = messagebox.askquestion(title='Hapus
data', message='Apakah Anda yakin ingin menghapus?',
icon='warning')
        if konfirmasi_hapus == 'yes':
            selected_item = self.trvTabel.selection()[0]
            get_id =
self.trvTabel.item(selected_item)['values'][0]
            self.delete(get_id)
            self.trvTabel.delete(*self.trvTabel.get_children())
            self.frame_tabel.after(0, self.table())
            self.clear()
            self.updateButton.config(state='disabled')
            self.deleteButton.config(state='disabled')
            self.saveButton.config(state='normal')
        else:
            pass
```

```
    def onUpdate(self):
        entKode = self.inputKode.get()
        entNama = self.inputNama.get()
        entUsername = self.inputUsername.get()
        entPassword = self.inputPassword.get()
        entKonfirmasi = self.inputKonfirmasi.get()
        entBagian = self.inputBagian.get()

        if entPassword != entKonfirmasi:
            messagebox.showwarning(title='Error',
message='Konfirmasi password tidak sama!')

        else:
            self.update(entNama, entUsername, entPassword,
entBagian, entKode)
            self.trvTabel.delete(*self.trvTabel.get_children())
            self.frame_tabel.after(0, self.table())
            self.clear()
            self.updateButton.config(state='disabled')
            self.deleteButton.config(state='disabled')
            self.saveButton.config(state='normal')

    def table(self):
        for kolom in judul_kolom:
            self.trvTabel.heading(kolom, text=kolom)
            self.trvTabel["displaycolumns"]=("0", "1", "2", "4")

        self.trvTabel.column("ID", width=50)
        self.trvTabel.column("Nama", width=250, anchor="w")
        self.trvTabel.column("Username", width=150)
        self.trvTabel.column("Password", width=50)
        self.trvTabel.column("Bagian", width=180)

        result = self.read()
        i=0

        for data in result:
            baris = "genap" if i%2 == 0 else "ganjil"
```

```
            self.trvTabel.insert('', 'end', values=data,
tags=(baris, ))
            i+=1
        self.trvTabel.tag_configure("ganjil",
background="#FFFFFF")
        self.trvTabel.tag_configure("genap",
background="whitesmoke")

    def clear(self):
        self.inputKode.config(state='normal')
        self.inputKode.delete(0, END)
        self.inputKode.config(state='readonly')
        self.inputNama.delete(0, END)
        self.inputUsername.delete(0, END)
        self.inputPassword.delete(0, END)
        self.inputKonfirmasi.delete(0, END)
        self.inputBagian.delete(0, END)

    def onClear(self):
        readonlytext = StringVar()
        readonlytext.set('Dikembangkan Otomatis')
        self.clear()
        self.inputKode.config(state='normal',
textvariable=readonlytext)
        self.inputKode.config(state='disabled')
        self.updateButton.config(state='disabled')
        self.deleteButton.config(state='disabled')
        self.saveButton.config(state='normal')

    def onKembali(self):
        root.destroy()
        os.system('python halaman_utama_admin.py')



    def onDoubleClick(self, event):
        self.saveButton.config(state='disabled')
        self.updateButton.config(state='normal')
        self.deleteButton.config(state='normal')
```

```
        self.clear()

        selected = self.trvTabel.focus()
        item = self.trvTabel.item(selected, "values")

        self.inputKode.config(state="normal")
        self.inputKode.insert(END, item[0])
        self.inputKode.config(state='readonly')
        self.inputNama.insert(END, item[1])
        self.inputUsername.insert(END, item[2])
        self.inputPassword.insert(END, item[3])
        self.inputBagian.insert(END, item[4])


Admin(root)
root.mainloop()
```

5.4 Implementasi Fitur Tambah Kasus Positif

Fitur tambah kasus positif digunakan untuk menambahkan data kasus positif COVID-19 yang kemudian disimpan didalam database. Pengguna yang akan memasukkan kasus positif harus memasukkan tanggal dan jumlah kasus positif. Berikut ini adalah source code dari fitur tambah kasus positif

```
from tkinter import *
from tkinter import messagebox
import os
from config import Config
import datetime

root = Tk()
root.title("Prediksi Covid v.1.0")
root.resizable(0, 0)
root.iconbitmap("Polinema.ico")

class InputPositif(Config):

    def __init__(self, toplevel):
```

```
        super(InputPositif, self).__init__()
        self.toplevel = toplevel
        lebar = 450
        tinggi = 300
        setTengahX = (self.toplevel.winfo_screenwidth() - lebar)
// 2
        setTengahY = (self.toplevel.winfo_screenheight() -
tinggi) // 2
        self.toplevel.geometry("%ix%i+%i+%i" % (lebar, tinggi,
setTengahX, setTengahY))
        self.komponen()

    def komponen(self):
        #frame
        self.top_level = Frame(self.toplevel,
background='#cedfe0')
        self.top_level.pack(side='top', fill='both',
expand='true')
        self.header_frame = Frame(self.top_level,
background='#808080', padx='10', pady='20')
        self.header_frame.pack(side='top', fill='x',
anchor='center')
        self.input_frame = Frame(self.top_level,
background='#cedfe0')
        self.input_frame.pack(side='top')
        self.button_frame = Frame(self.top_level,
background='#cedfe0', pady='10')
        self.button_frame.pack(side='top')

        #label
        self.header_label = Label(self.header_frame,
background='#808080', text='Halaman Input Kasus Positif',
font='{Segoe UI Semibold} 14 {}')
        self.header_label.pack(side='top')
        Label(self.input_frame, background='#cedfe0',
font='{Segoe UI Semibold} 12 {}',
text='Tanggal').grid(column='0', padx='10', pady='20', row='0',
sticky='w')
        Label(self.input_frame, background='#cedfe0',
```

```
font='{Segoe UI Semibold} 12 {}', text='(YYYY-MM-
DD)').grid(column='2', row='0', sticky='w')
        Label(self.input_frame, background='#cedfe0',
font='{Segoe UI Semibold} 12 {}', text='Jumlah
Kasus').grid(row='2', column='0', padx='10', pady='20',
sticky='w')

        #entry field
        self.entryTanggal = Entry(self.input_frame)
        self.entryTanggal.grid(row='0', column='1', padx='10')
        self.entryKasus = Entry(self.input_frame)
        self.entryKasus.grid(row='2', column='1')

        #button
        self.saveButton = Button(self.button_frame, font='{Segoe
UI Semibold} 10 {}', relief='groove', text='Go!',
command=self.onSave)
        self.saveButton.grid(row='0', column='1', padx='10',
pady='10')
        self.kembaliButton = Button(self.button_frame,
font='{Segoe UI Semibold} 10 {}', relief='groove',
text='Kembali', command=self.onKembali, width='8')
        self.kembaliButton.grid(row='0', column='0', padx='10',
pady='10')

    def onSave(self):
        entTanggal = self.entryTanggal.get()
        entKasus = self.entryKasus.get()
        #konversi tanggal dan kasus
        tanggalString = str(entTanggal)
        format = "%Y-%m-%d"

        if len(entTanggal or entKasus) > 0 :
            if len(entTanggal) > 0 :
                if len(entKasus) > 0 :
                    try:

datetime.datetime.strptime(tanggalString, format)
                        if isinstance(entKasus, int) == TRUE:
```

```
                            self.insertKasus(entTanggal,
entKasus)
                            self.onClear()
                            messagebox.showinfo(title='Sukses',
message='Data berhasil dimasukkan')
                        else:
                            messagebox.showerror(title='Error',
message='Kasus harus angka')
                    except ValueError:
                        messagebox.showerror(title='Error',
message='Format tanggal salah!')
                else:

messagebox.showerror(title='Error',message='Kasus tidak boleh
kosong')
            else:
                messagebox.showerror(title='Error',
message='Tanggal tidak boleh kosong')
        else:
            messagebox.showerror(title='Error', message='Field
tidak boleh kosong')

    def onClear(self):
        self.entryTanggal.delete(0, END)
        self.entryKasus.delete(0, END)

    def onKembali(self):
        root.destroy()
        os.system('python halaman_utama_admin.py')

InputPositif(root)
root.mainloop()
```

5.5 Implementasi Fitur Peramalan

Pada fitur peramalan, data yang telah disediakan kemudian dilakukan perhitungan untuk mengetahui jumlah kasus positif pada periode berikutnya.

Peramalan yang digunakan adalah dengan menggunakan metode *exponential smoothing*. Berikut ini adalah source code dari fitur peramalan

```
import pandas as pd
from tkinter import *
from matplotlib import pyplot as plt
import numpy as np
import os
from config import Config
from sklearn.metrics import mean_absolute_percentage_error

root = Tk()
root.title("Prediksi Covid v.1.0")
root.resizable(0,0)
root.iconbitmap("Polinema.ico")

class Peramalan(Config):

    def __init__(self, toplevel):
        super(Peramalan, self).__init__()
        self.toplevel = toplevel
        lebar = 600
        tinggi = 500
        setTengahX = (self.toplevel.winfo_screenwidth() - lebar)
// 2
        setTengahY = (self.toplevel.winfo_screenheight() -
tinggi) // 2
        self.toplevel.geometry("%ix%i+%i+%i" % (lebar, tinggi,
setTengahX, setTengahY))
        self.komponen()

    def komponen(self):
        #atur frame
        self.top_level = Frame(self.toplevel,
background='#cedfe0')
        self.top_level.pack(side='top', fill='both',
expand='true')
        Label(self.top_level, background='#808080',
text='Halaman Prediksi', font='{Segoe UI Semibold} 14 {}',
padx='10', pady='20').pack(side='top', fill='both')
```

```
        self.nilai_frame = Frame(self.top_level,
background='#cedfe0')
        self.nilai_frame.pack(side='top', fill='both')
        self.button_frame = Frame(self.top_level,
background='#cedfe0', pady='20', padx='10')
        self.button_frame.pack(side='top')



        Label(self.nilai_frame, background='#cedfe0',
font='{Segoe UI Semibold} 12 {}', text='Mean
Error\t:').grid(column='0', row='0', padx='30', pady='10',
sticky='w')
        Label(self.nilai_frame, background='#cedfe0',
font='{Segoe UI Semibold} 12 {}', text='Nilai MSE\t\t:').grid(
            column='0', row='1', padx='30', pady='10',
sticky='w')
        Label(self.nilai_frame, background='#cedfe0',
font='{Segoe UI Semibold} 12 {}', text='Prediksi
Besok\t:').grid(
            column='0', row='2', padx='30', pady='10',
sticky='w')
        Label(self.nilai_frame, background='#cedfe0',
font='{Segoe UI Semibold} 12 {}', text='MAPE\t\t: ').grid(
            column='0', row='3', padx='30', pady='10',
sticky='w')

        self.inputError = Entry(self.nilai_frame,
state='readonly')
        self.inputError.grid(column='1', row='0')
        self.inputMSE = Entry(self.nilai_frame,
state='readonly')
        self.inputMSE.grid(column='1', row='1')
        self.inputPrediksi = Entry(self.nilai_frame,
state='readonly')
        self.inputPrediksi.grid(column='1', row='2')
        self.inputAkurasi = Entry(self.nilai_frame,
state='readonly')
        self.inputAkurasi.grid(column='1', row='3')
```

```
        Button(self.button_frame,
command=self.GenerateGraph,font='{Segoe UI Semibold} 10 {}',
relief='groove', text='Generate Graph',
width='20').grid(column='0', padx='10', row='0')
        self.buttonPreVaksin = Button(self.button_frame,
font='{Segoe UI Semibold} 10 {}', relief='groove',
text='Prediksi Pre-Vaksin', width='20', command=self.PreVaksin)
        self.buttonPreVaksin.grid(column='0', row='1',
padx='10')
        self.buttonGenerateCSV = Button(self.button_frame,
font='{Segoe UI Semibold} 10 {}', relief='groove',
text='Generate CSV', width='20', command=self.GenerateAllData)
        self.buttonGenerateCSV.grid(column='1', row='0',
padx='10')
        self.buttonPascaVaksin = Button(self.button_frame,
font='{Segoe UI Semibold} 10 {}', relief='groove',
text='Prediksi Pasca Vaksin', width='20',
command=self.PascaVaksin)
        self.buttonPascaVaksin.grid(column='1', row='1',
padx='10', pady='10')
        self.buttonKembali = Button(self.button_frame,
font='{Segoe UI Semibold} 10 {}', relief='groove',
text='Kembali', width='8', command=self.onKembali)
        self.buttonKembali.grid(column='0', row='2',
columnspan='2', pady='10')

    def onKembali(self):
        root.destroy()
        os.system('python halaman_utama_admin.py')

    def GenerateAllData(self):
        self.GenerateCSVDataReal()
        self.GenerateCSVDataPreVaksin()
        self.GenerateCSVDataPascaVaksin()

    def GenerateGraph(self):
        np.set_printoptions(precision=3, suppress=True)
        cwd = os.getcwd()
        target_file = '\exported\Positif.csv'
```

```
        target_dir = cwd + target_file
        df = pd.read_csv(target_dir, parse_dates=True)
        alpha = 0.9
        kasus = df['Kasus']
        tanggal = pd.to_datetime(df['Tanggal'], format='%d-%b-
%Y')
        hasil = []
        jumlah_data = kasus.count()

        for i, x in enumerate(kasus):
            if i == 0:
                res = x
                hasil.append(x)
            else:
                tmp = alpha * x + (1 - alpha) * res
                hasil.append(tmp)
                res = tmp

        final = np.array(hasil)
        kasus_to_array = np.array(kasus)
        print(final)

        #hitung mean error
        forecast_error = np.delete(final, [-1])
        kasus_error = np.delete(kasus_to_array, [0])
        error = np.subtract(kasus_error, forecast_error)
        mean_error = np.nansum(error)

        #hitung mse
        sum_error = np.nansum(error)
        sum_total = np.nansum(kasus_to_array)
        sum_total_error = sum_error + sum_total
        mse = np.square(sum_total_error)/jumlah_data

        #hitung MAPE
        mape = np.nanmean(np.abs((kasus_to_array -
final)/kasus_to_array)) * 100
        mape = "{:.2f}".format(mape)
```

```
        #prediksi periode berikutnya
        next_period = int(final[-1])

        self.inputError.config(state='normal')
        self.inputMSE.config(state='normal')
        self.inputPrediksi.config(state='normal')
        self.inputAkurasi.config(state='normal')

        self.inputError.delete(0, END)
        self.inputMSE.delete(0, END)
        self.inputPrediksi.delete(0, END)
        self.inputAkurasi.delete(0, END)

        self.inputError.insert(END, mean_error)
        self.inputMSE.insert(END, mse)
        self.inputPrediksi.insert(END, next_period)
        self.inputAkurasi.insert(END, mape)

        self.inputError.config(state='readonly')
        self.inputMSE.config(state='readonly')
        self.inputPrediksi.config(state='readonly')
        self.inputAkurasi.config(state='readonly')

        plt.plot(tanggal, final, label='Ramalan')
        plt.plot(tanggal, kasus, label='Aktual')
        plt.legend()
        plt.show()

    def PreVaksin(self):
        np.set_printoptions(precision=3, suppress=True)
        cwd = os.getcwd()
        target_file = '\exported\DataPreVaksin.csv'
        target_dir = cwd + target_file
        df = pd.read_csv(target_dir, parse_dates=True)
        alpha = 0.9
        kasus = df['Kasus']
        tanggal = pd.to_datetime(df['Tanggal'], format='%d-%b-
%Y')
        hasil = []
```

```
        jumlah_data = kasus.count()

        for i, x in enumerate(kasus):
            if i == 0:
                res = x
                hasil.append(x)
            else:
                tmp = alpha * x + (1 - alpha) * res
                hasil.append(tmp)
                res = tmp

        final = np.array(hasil)
        kasus_to_array = np.array(kasus)
        print(final)

        # hitung mean error
        forecast_error = np.delete(final, [-1])
        kasus_error = np.delete(kasus_to_array, [0])
        error = np.subtract(kasus_error, forecast_error)
        mean_error = np.nansum(error)

        # hitung mse
        sum_error = np.nansum(error)
        sum_total = np.nansum(kasus_to_array)
        sum_total_error = sum_error + sum_total
        mse = np.square(sum_total_error) / jumlah_data

        # hitung MAPE
        mape = np.nanmean(np.abs((kasus_to_array - final) /
kasus_to_array)) * 100

        # prediksi periode berikutnya
        next_period = final[-1]

        self.inputError.config(state='normal')
        self.inputMSE.config(state='normal')
        self.inputPrediksi.config(state='normal')
        self.inputAkurasi.config(state='normal')
```

```
        self.inputError.delete(0, END)
        self.inputMSE.delete(0, END)
        self.inputPrediksi.delete(0, END)
        self.inputAkurasi.delete(0, END)

        self.inputError.insert(END, mean_error)
        self.inputMSE.insert(END, mse)
        self.inputPrediksi.insert(END, next_period)
        self.inputAkurasi.insert(END, mape)

        self.inputError.config(state='readonly')
        self.inputMSE.config(state='readonly')
        self.inputPrediksi.config(state='readonly')
        self.inputAkurasi.config(state='readonly')

        plt.plot(tanggal, final, label='Ramalan Pre Vaksin')
        plt.legend()
        plt.show()

    def PascaVaksin(self):
        np.set_printoptions(precision=3, suppress=True)
        cwd = os.getcwd()
        target_file = '\exported\DataPascaVaksin.csv'
        target_dir = cwd + target_file
        df = pd.read_csv(target_dir, parse_dates=True)
        alpha = 0.5
        kasus = df['Kasus']
        tanggal = pd.to_datetime(df['Tanggal'], format='%d-%b-
%Y')
        hasil = []
        jumlah_data = kasus.count()

        for i, x in enumerate(kasus):
            if i == 0:
                res = x
                hasil.append(x)
            else:
                tmp = alpha * x + (1 - alpha) * res
                hasil.append(tmp)
```

```
                res = tmp

        final = np.array(hasil)
        kasus_to_array = np.array(kasus)

        # hitung mean error
        forecast_error = np.delete(final, [-1])
        kasus_error = np.delete(kasus_to_array, [0])
        error = np.subtract(kasus_error, forecast_error)
        mean_error = np.nansum(error)

        # hitung mse
        sum_error = np.nansum(error)
        sum_total = np.nansum(kasus_to_array)
        sum_total_error = sum_error + sum_total
        mse = np.square(sum_total_error) / jumlah_data

        # hitung MAPE
        mape = np.nanmean(np.abs((kasus_to_array - final) /
kasus_to_array)) * 100

        # prediksi periode berikutnya
        next_period = final[-1]

        self.inputError.config(state='normal')
        self.inputMSE.config(state='normal')
        self.inputPrediksi.config(state='normal')
        self.inputAkurasi.config(state='normal')

        self.inputError.delete(0, END)
        self.inputMSE.delete(0, END)
        self.inputPrediksi.delete(0, END)
        self.inputAkurasi.delete(0, END)

        self.inputError.insert(END, mean_error)
        self.inputMSE.insert(END, mse)
        self.inputPrediksi.insert(END, next_period)
        self.inputAkurasi.insert(END, mape)
```

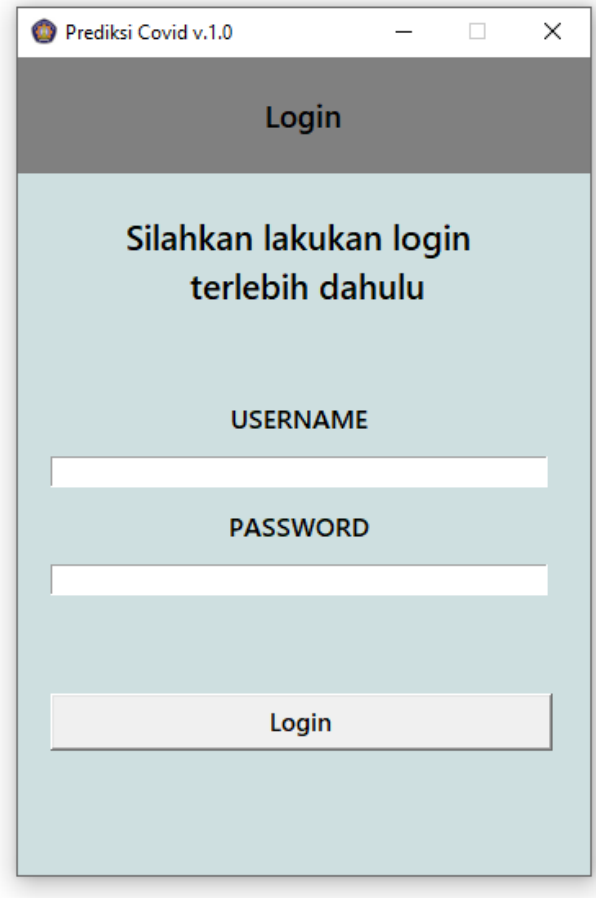
```
        self.inputError.config(state='readonly')
        self.inputMSE.config(state='readonly')
        self.inputPrediksi.config(state='readonly')
        self.inputAkurasi.config(state='readonly')

        plt.plot(tanggal, final, label='Ramalan Pre Vaksin')
        plt.legend()
        plt.show()

Peramalan(root)
root.mainloop()
```

5.6 Implementasi Antarmuka

Tahap implementasi antarmuka merupakan tahap lanjutan dari rancangan desain antarmuka dari BAB IV. Hasil rancangan tersebut diubah dalam bentuk kode yang bisa menampilkan tampilan antarmuka yang sudah didesain sebelumnya. Gambar 5.6 hingga 5.11 adalah hasil dari implementasi antarmuka

5.6.1 Implementasi Halaman Login

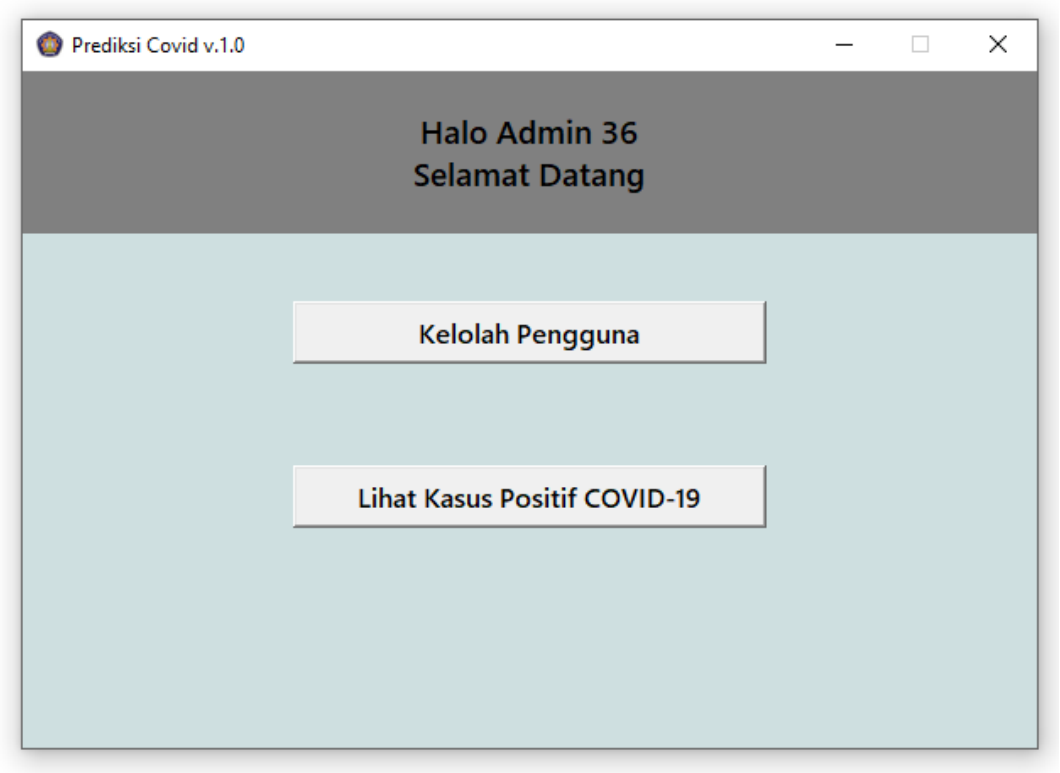
Gambar 5.6 menunjukkan implementasi halaman login. Halaman login merupakan halaman pertama yang akan dijumpai oleh pengguna aplikasi. Pengguna diwajibkan untuk memasukkan username dan password yang sudah didaftarkan.



Gambar 5.6 Implementasi Antarmuka Login

5.6.2 Implementasi Halaman Utama Admin

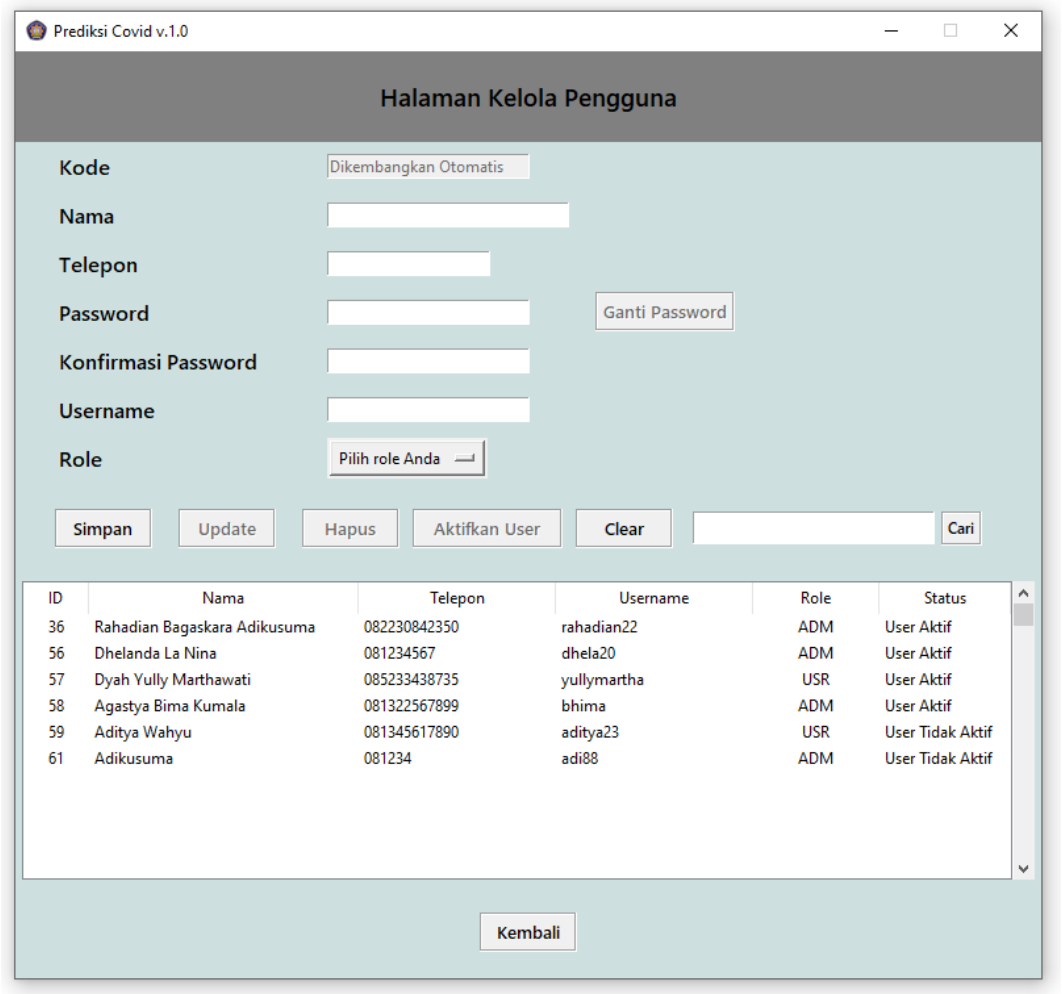
Gambar 5.7 menunjukkan hasil implementasi dari perancangan halaman utama. Halaman utama adalah halaman yang dijumpai oleh pengguna setelah melakukan login. Pada halaman utama tediri dari tiga menu utama yaitu Kelola Pengguna, Input Kasus Positif, dan Lakukan Prediksi Data.



Gambar 5.7 Implementasi Halaman Utama Admin

5.6.3 Implementasi Halaman Kelola Pengguna

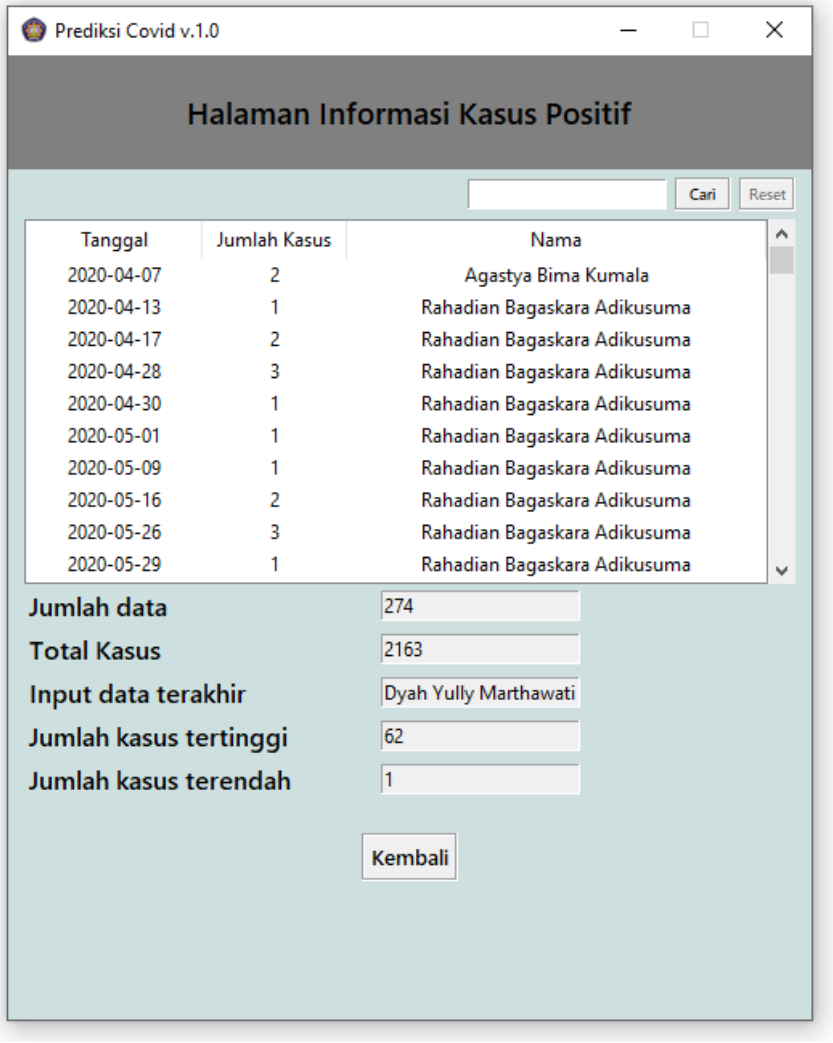
Halaman Kelola pengguna adalah halaman dimana pengguna dapat melakukan proses CRUD data admin. Pengguna yang sudah terdaftar maka akan dapat menggunakan aplikasi. Gambar 5.8 menunjukkan hasil implementasi halaman Kelola pengguna



Gambar 5.8 Implementasi Halaman Kelola Pengguna

5.6.4 Implementasi Halaman Lihat Kasus Positif

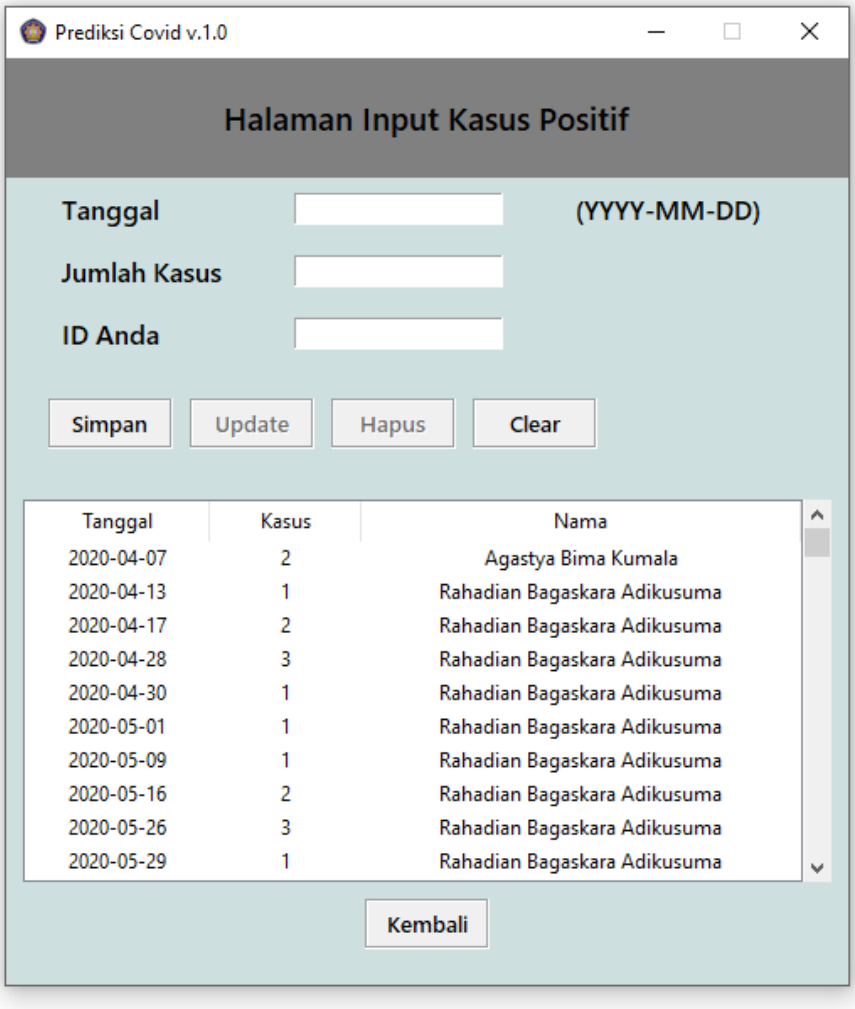
Halaman lihat kasus positif adalah halaman yang digunakan untuk menampilkan data kasus COVID-19 dengan lebih rinci. Perincian ini diberikan dengan informasi jumlah data, total kasus, pemasuk kasus terakhir, kasus terkecil, dan kasus terbesar. Gambar 5.9 adalah hasil dari implementasi desain antarmuka lihat data kasus positif



Gambar 5.9 Implementasi Lihat Kasus Positif

5.6.5 Implementasi Halaman Input Kasus Positif

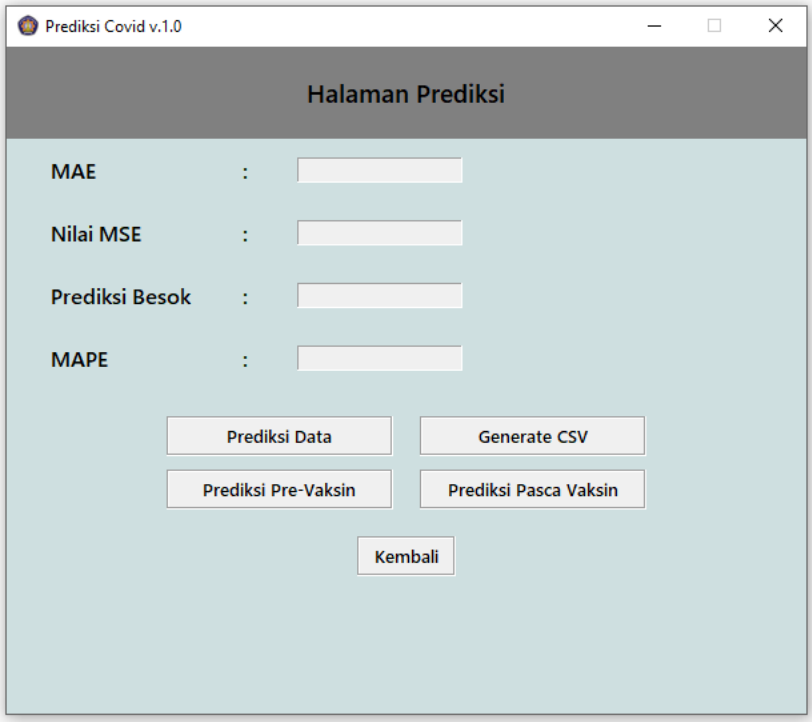
Halaman input kasus positif digunakan untuk memasukkan data kasus positif COVID-19. Data kasus positif kemudian disimpan kedalam *database*. Gambar 5.10 berikut ini merupakan implementasi dari desain antarmuka halaman input kasus positif



Gambar 5.10 Implementasi Halaman Input Kasus Positif

5.6.6 Implementasi Halaman Lakukan Prediksi

Tahap ini merupakan hasil implementasi dari tahap perancangan halaman lakukan prediksi. Pada halaman ini user dapat melakukan peramalan kasus positif, data pre dan pasca vaksin. Gambar 5.11 berikut ini adalah implementasi antarmuka halaman lakukan prediksi.



Gambar 5.11 Implementasi Halaman Prediksi

5.7 Pengujian Black Box

Pengujian dengan metode *black box* adalah tahap pengujian dimana akan dilakukan beberapa *test case* terhadap aplikasi. Uji coba dengan metode *black box* ini dilakukan dengan tujuan untuk mengetahui kelayakan fungsional aplikasi yang akan digunakan oleh pengguna sebelum diterjunkan ke instansi yang bersangkutan. Scenario yang dilakukan pengujian adalah antara lain fitur login, kelola pengguna, input kasus positif, dan prediksi data

5.7.1 Pengujian Fitur Login

Pada tahap ini akan dilakukan pengujian terhadap fitur login. Tabel 5.1 berikut ini menunjukkan hasil pengujian terhadap fitur login

Tabel 5.1 Test Scenario Login

ID	Aktor	Test Scenario

TS01	Admin atau user	Melakukan login dengan memasukkan username dan password yang benar
TS02	Admin atau user	Melakukan login dengan memasukkan username dan password yang salah
TS03	Admin atau user	Admin atau user tidak lagi aktif

5.7.2 Pengujian Halaman Utama Admin

Pada tahap ini dilakukan pengujian terhadap halaman utama *admin*. Hal ini dilakukan untuk mengetahui apakah seluruh fungsi pada halaman utama *admin* akan mengarah pada menu yang telah diatur. Tabel 5.2 menunjukkan hasil pengujian halaman utama

Tabel 5.2 Test Scenario Halaman Utama Admin

ID	Aktor	Test Scenario
TS04	Admin	Memilih kelola pengguna
TS05	Admin	Memilih melihat data kasus positif

5.7.3 Pengujian Halaman Utama User

Pada tahap ini dilakukan pengujian terhadap halaman utama *admin*. Hal ini dilakukan untuk mengetahui apakah seluruh fungsi pada halaman utama *admin* akan mengarah pada menu yang telah diatur. Tabel 5.3 menunjukkan hasil pengujian halaman utama

Tabel 5.3 Test Scenario Halaman Utama User

ID	Aktor	Test Scenario
TS06	User	Memilih Lihat Data Kasus Positif

TS07	User	Memilih Input Kasus Positif
TS08	User	Memilih Lakukan Prediksi Data

5.7.4 Pengujian Fitur Kelola Pengguna

Pada tahap ini dilakukan pengujian terhadap halaman kelola pengguna. Hal ini dilakukan untuk mengetahui apakah seluruh tombol pada halaman kelola pengguna akan bekerja sesuai dengan fungsinya. Tabel 5.4 menunjukkan hasil pengujian halaman kelola pengguna

Tabel 5.4 Test Scenario Kelola Pengguna

ID	Aktor	Test Scenario
TS09	Admin	Semua field kosong
TS10	Admin	Salah satu field kosong
TS11	Admin	Semua field terisi
TS12	Admin	Update data dengan salah satu field kosong
TS13	Admin	Update data dengan seluruh field terpenuhi
TS14	Admin	Memilih tombol clear
TS15	Admin	Non aktifkan pengguna
TS16	Admin	Setuju untuk menghapus data
TS17	Admin	Tidak setuju untuk mengahpus data

5.7.5 Pengujian Fitur Ganti *Password*

Pada tahap ini dilakukan pengujian terhadap halaman ganti *password*. Hal ini dilakukan untuk mengetahui apakah seluruh fungsi pada halaman ganti *password* telah bekerja sesuai dengan fungsinya. Tabel 5.5 menunjukkan hasil pengujian halaman ganti *password*

Tabel 5.5 Test Scenario Ganti Password

ID	Aktor	Test Scenario
TS18	Admin	Semua field terisi
TS19	Admin	Password tidak ditemukan
TS20	Admin	Konfirmasi password tidak sama

5.7.6 Pengujian Fitur Lihat Kasus Positif

Halaman kasus positif digunakan untuk menunjukkan informasi lebih rinci mengenai kasus positif COVID-19 di Kota Probolinggo. Pada tahap ini dilakukan pengujian untuk mengetahui apakah halaman informasi dapat bekerja sesuai dengan fungsinya. Tabel 5.6 menunjukkan bagaimana *test scenario* pada halaman lihat kasus positif

Tabel 5.6 Test Scenario Lihat Kasus Positif

ID	Aktor	Test Scenario
TS21	Admin atau user	Menampilkan informasi kasus positif COVID-19
TS22	Admin atau user	Fungsi pencarian data untuk mencari berdasarkan tanggal atau nama penginput
TS23	Admin atau user	Tombol reset dapat mengembalikan pencarian data

5.7.7 Pengujian Fitur Input Kasus Positif

Pada tahap ini dilakukan pengujian terhadap halaman input kasus positif. Hal ini dilakukan untuk mengetahui apakah seluruh fungsi pada halaman input kasus positif akan bekerja sesuai dengan fungsinya. Tabel 5.7 menunjukkan hasil pengujian halaman input kasus positif

Tabel 5.7 Test Scenario Input Kasus Positif

ID	Aktor	Test Scenario
TS24	User	Membiarkan field kosong
TS25	User	Salah satu field kosong
TS26	User	Format tanggal yang dimasukkan tidak sesuai
TS27	User	Jumlah kasus yang dimasukkan bukan angka
TS28	User	Tanggal yang dimasukkan sudah ada

5.7.8 Pengujian Fitur Peramalan

Pada tahap ini dilakukan pengujian terhadap halaman prediksi. Hal ini dilakukan untuk mengetahui apakah seluruh fungsi pada halaman prediksi akan bekerja sesuai dengan fungsinya. Tabel 5.8 menunjukkan hasil pengujian halaman prediksi.

Tabel 5.8 Test Scenario Lakukan Prediksi Data

ID	Aktor	Test Scenario
TS29	User	Memilih tombol Generate Graph
TS30	User	Memilih tombol Prediksi Pre Vaksin
TS31	User	Memilih tombol Generate CSV
TS32	User	Memilih tombol Prediksi Pasca Vaksin
TS33	User	Klik tombol Generate Graph – tidak ada data
TS34	User	Klik tombol Prediksi Pre Vaksin – tidak ada data
TS35	User	Klik tombol Prediksi Pasca Vaksin – tidak ada data

TS36	User	Klik tombol Generate CSV – data CSV sudah di export