APLIKASI RAPOR SEDERHANA
LAPORAN AKHIR PROYEK PEMROGRAMAN DASAR

Disusun Oleh:

Fasha Anandhikta Yanuara (25031554040)

Parama Zaidan Harandi (25031554157)

Sabrina Auliya Arinditha (25031554096)

Dosen Pengampu:

Hasanuddin Al-Habib, S.Si., M.Si

PROGRAM STUDI SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI INDONESIA

TAHUN 2025

Daftar Isi

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Proses pengolahan nilai siswa di sekolah pada umumnya masih dilakukan secara manual, baik melalui pencatatan di buku maupun pengolahan menggunakan media sederhana. Cara tersebut berpotensi menimbulkan berbagai permasalahan, seperti kesalahan perhitungan nilai, duplikasi data, serta membutuhkan waktu yang relatif lama dalam proses rekapitulasi nilai siswa.

Seiring dengan perkembangan teknologi informasi, pemanfaatan aplikasi digital dalam bidang pendidikan menjadi solusi yang tepat untuk meningkatkan efisiensi dan akurasi pengolahan data. Guru membutuhkan sebuah sistem yang mampu membantu dalam menyimpan data siswa, mengelola nilai mata pelajaran, serta melakukan perhitungan nilai secara otomatis sehingga proses penilaian menjadi lebih cepat dan terstruktur.

Berdasarkan permasalahan tersebut, dibuatlah Aplikasi Rapor Sederhana Berbasis Python yang berfungsi untuk mengelola data nilai siswa, menghitung rata-rata nilai, menentukan status kelulusan secara otomatis, serta menampilkan dan mengekspor hasil rapor dalam bentuk file PDF. Aplikasi ini diharapkan dapat membantu guru dalam proses administrasi penilaian serta meminimalkan kesalahan yang dapat terjadi pada pengolahan nilai secara manual.

1.2 Rumusan Masalah

Pengolahan data rapor siswa yang masih dilakukan secara manual sering menimbulkan permasalahan seperti kesalahan input nilai, kesulitan dalam

pengelolaan data, serta kurang efisiennya proses penyajian rapor. Selain itu, belum adanya aplikasi sederhana yang mampu mengelola data siswa, nilai mata pelajaran, dan menghasilkan rapor secara otomatis menjadi kendala dalam administrasi penilaian. Oleh karena itu, diperlukan sebuah aplikasi rapor digital yang dapat membantu pengguna dalam mengelola data nilai secara terstruktur, melakukan perhitungan rapor, serta menampilkan dan mengekspor hasil rapor secara praktis dan efisien.

1.3 Tujuan

1.3.1 Menyediakan fitur pengelolaan nilai siswa yang lebih terstruktur.
1.3.2 Mengotomatisasi perhitungan nilai rata-rata dan status kelulusan.
1.3.3 Memudahkan pencarian siswa atau mata pelajaran tertentu.
1.3.4 Mengotomatisasi pembuatan rapor siswa dalam format PDF.

BAB 2
ANALISIS DAN PERANCANGAN

2.1 Analisis Kebutuhan Aplikasi

Dalam pembuatan Aplikasi Rapor Digital Berbasis Python, dilakukan analisis kebutuhan untuk memahami alasan aplikasi ini diperlukan serta fungsi yang harus dimiliki agar dapat membantu proses pengolahan nilai siswa secara optimal. Analisis kebutuhan aplikasi ini dibagi menjadi dua bagian, yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

2.1.1 Kebutuhan Fungsional Aplikasi

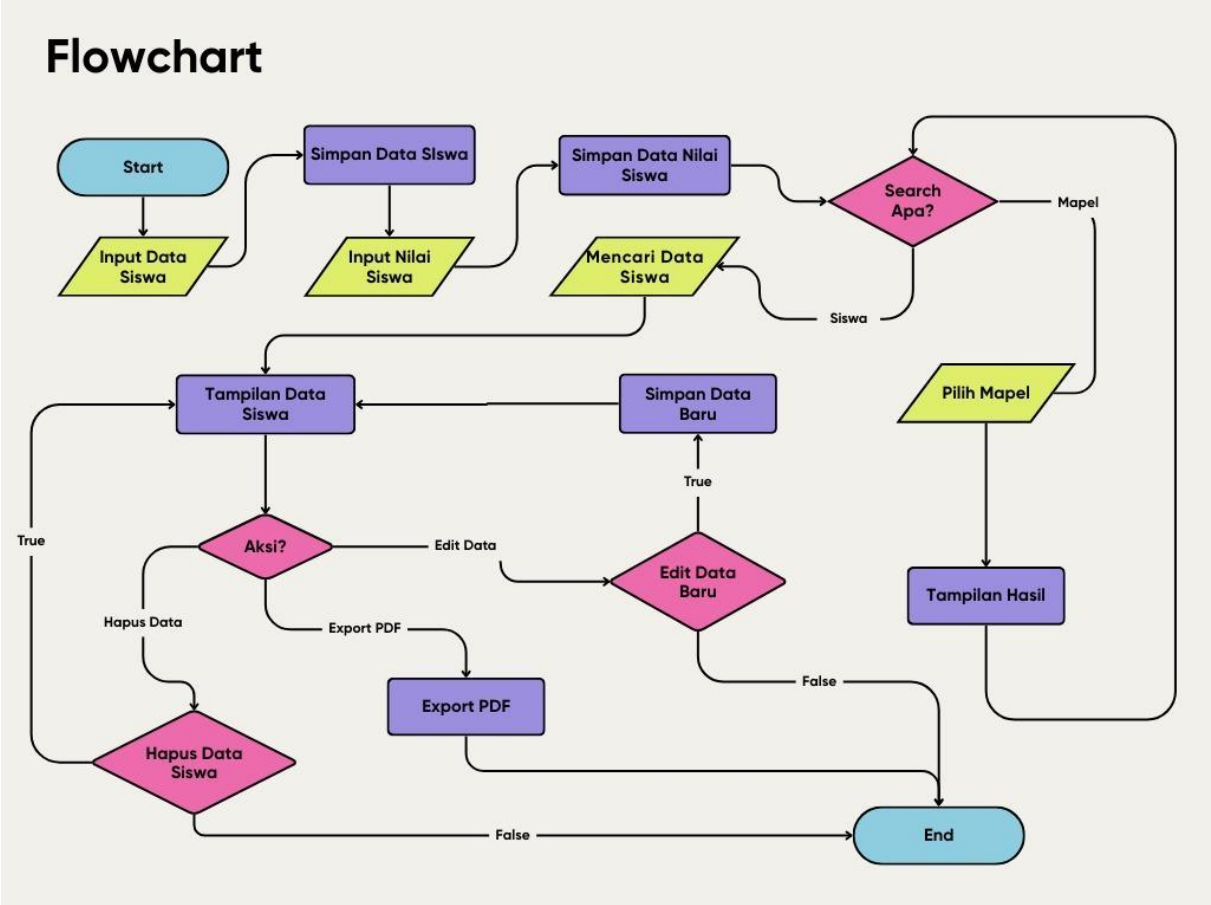
- Aplikasi mampu menyimpan data siswa secara terstruktur, seperti NISN, nama, dan nilai siswa.
- Aplikasi menyediakan fitur input, edit, dan hapus nilai data siswa.
- Aplikasi dapat melakukan perhitungan nilai secara otomatis, seperti perhitungan rata-rata nilai dan predikat.
- Aplikasi mampu menentukan status kelulusan siswa berdasarkan kriteria nilai yang ditetapkan.
- Aplikasi menyediakan fitur ekspor rapor ke dalam format PDF sebagai arsip digital.

2.1.2 Kebutuhan Non-Fungsional Aplikasi

- Aplikasi memiliki tampilan antarmuka yang sederhana dan mudah dipahami oleh pengguna.
- Aplikasi mampu menyimpan data secara lokal dengan aman.
- Aplikasi dapat digunakan secara offline tanpa memerlukan koneksi internet.

2.2 Diagram Alur (Flowchart)

Diagram alur (flowchart) digunakan untuk menggambarkan urutan proses kerja aplikasi secara keseluruhan, mulai dari aplikasi dijalankan hingga pengguna mengelola dan menampilkan data rapor. Flowchart ini bertujuan untuk memastikan setiap proses berjalan secara logis dan sistematis, sehingga alur kerja aplikasi mudah dipahami dari awal hingga akhir.



Gambar 1

2.3 Sketsa Desain Antarmuka

2.3.1 Dashboard Page

Komponen:

- Tittle: “Dashboard”
- Cards:
  - Total siswa
  - Presentase lulus
  - Jumlah lulus
- Detail Box:
  - NISN
  - Nama
  - Kelas
  - Rata-rata

- Status Kelulusan

2.3.2 Tambah Siswa

Komponen:

- Tittle: “Input Data Siswa”
- User Input:
  - NISN
  - Nama lengkap
  - Kelas
- Buttons:
  - Tambah
  - Clear
- Label:
  - NISN
  - Nama lengkap
  - Kelas

2.3.3 Input Nilai

Komponen:

- Tittle: “Input Nilai Siswa”
- User Input:
  - Input box
- Buttons:
  - Cari
  - Simpan Nilai
  - Clear
- Label:
  - Matematika
  - Bahasa Indonesia
  - Bahasa Inggris
  - IPA
  - IPS

2.3.4 Search Siswa

Komponen:

- Tittle: “Search Siswa (NISN atau nama)”
- User Input:
  - Input box
- Buttons:
  - Cari
  - Export PDF
  - Edit Data
  - Hapus Data
- Detail Box:
  - NISN
  - Nama
  - Kelas
  - Nilai Mata Pelajaran
  - Rata-rata
  - Predikat
  - Status Kelulusan

2.3.5 Search Mapel

Komponen:

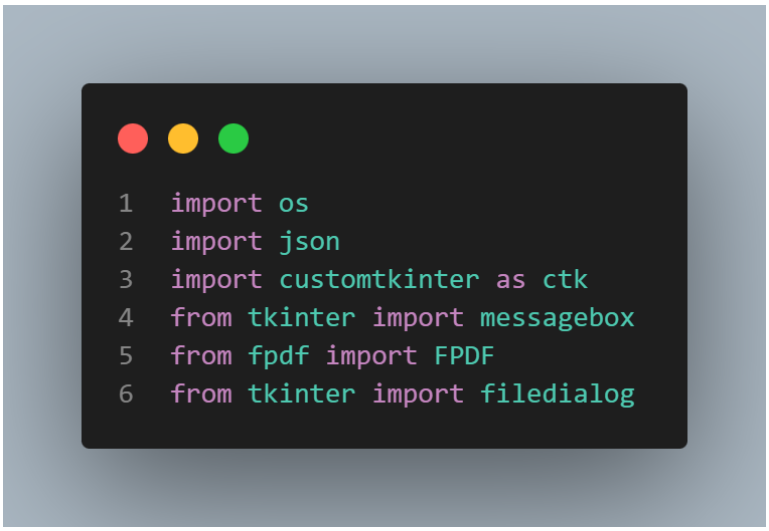
- Tittle: “Search Mapel”
- Buttons:
  - Mata Pelajaran
- Detail Box:
  - NISN
  - Nama
  - Kelas
  - Nilai Mapel

BAB 3
IMPLEMENTASI

3.1 Penjelasan Kode Program

Aplikasi ini dikembangkan menggunakan beberapa library dan fitur yang berperan dalam mendukung fungsi aplikasi dan kenyamanan pengguna.

Berikut adalah penjelasan rinci tentang bagian-bagian penting dari kode tersebut:



```
1  import os
2  import json
3  import customtkinter as ctk
4  from tkinter import messagebox
5  from fpdf import FPDF
6  from tkinter import filedialog
```

Gambar 2

Import library:

- os: digunakan untuk berinteraksi dengan sistem operasi, seperti menangani jalur file.
- json: digunakan untuk membaca dan menyimpan data siswa dalam bentuk file JSON.
- customtkinter: digunakan untuk membangun antarmuka grafis (GUI) aplikasi.
- tkinter.messagebox: digunakan untuk menampilkan pesan notifikasi kepada pengguna.
- fpdf: digunakan untuk membuat dan mengekspor rapor siswa dalam bentuk file PDF.
- tkinter.filedialog: digunakan untuk menampilkan dialog pemilihan lokasi penyimpanan file

```
# KONFIGURASI GLOBAL
COLOR_LULUS = "#22c55e"
COLOR_TIDAK_LULUS = "#ef4444"

DATA_FILE = "data_siswa.json"
SUBJECTS = ["Matematika", "Bahasa Indonesia", "Bahasa Inggris", "IPA", "IPS"]
```

Gambar 3

Konfigurasi Global:

- COLOR_LULUS: digunakan untuk menentukan warna tampilan status siswa yang dinyatakan lulus
- COLOR_TIDAK_LULUS: digunakan untuk menentukan warna tampilan status siswa yang tidak lulus
- DATA_FILE: digunakan untuk menentukan nama file yang berfungsi sebagai media penyimpanan data siswa dan nilai
- SUBJECTS: berisi daftar mata pelajaran yang digunakan dalam aplikasi
-

```
def load_data():
    if not os.path.exists(DATA_FILE):
        return {}
    try:
        with open(DATA_FILE, "r", encoding="utf-8") as f:
            return json.load(f)
    except Exception:
        return {}

def save_data(data):
    with open(DATA_FILE, "w", encoding="utf-8") as f:
        json.dump(data, f, indent=4, ensure_ascii=False)
```

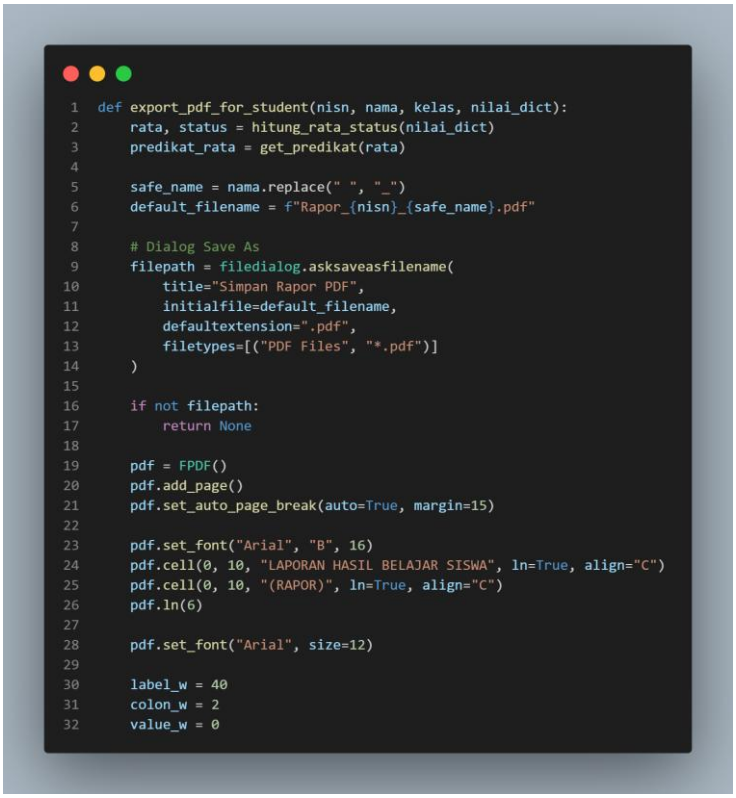
Gambar 4

Fungsi load_data() dan save_data() digunakan untuk mengelola penyimpanan data siswa dalam aplikasi rapor digital. Fungsi load_data() berperan dalam memuat data dari file JSON dengan terlebih dahulu memeriksa keberadaan file, sehingga aplikasi tetap dapat berjalan meskipun file data belum tersedia atau terjadi kesalahan saat proses pembacaan. Sementara itu, fungsi save_data(data) digunakan untuk menyimpan atau memperbarui data siswa ke dalam file JSON dengan format yang rapi dan mudah dibaca. Penggunaan kedua fungsi ini memungkinkan data siswa tersimpan secara permanen dan mendukung proses pengolahan data yang efisien serta aman.

```python
def hitung_rata_status(nilai_dict):
    total = sum(nilai_dict.values()) if nilai_dict else 0
    rata = total / len(nilai_dict) if nilai_dict else 0
    status = "LULUS" if rata >= 75 else "TIDAK LULUS"
    return rata, status

def get_predikat(nilai):
    if nilai >= 90:
        return "A"
    elif nilai >= 80:
        return "B"
    elif nilai >= 70:
        return "C"
    elif nilai >= 60:
        return "D"
    else:
        return "E"
```

Gambar 5

Fungsi hitung_rata_status(nilai_dict) digunakan untuk menghitung nilai rata-rata siswa berdasarkan kumpulan nilai mata pelajaran yang tersimpan dalam bentuk dictionary. Fungsi ini menjumlahkan seluruh nilai, kemudian membaginya dengan jumlah mata pelajaran untuk memperoleh nilai rata-rata. Berdasarkan hasil perhitungan tersebut, sistem secara otomatis menentukan status kelulusan siswa, yaitu **“**LULUS**”** apabila nilai rata-rata mencapai atau melebihi batas yang ditentukan, dan “TIDAK LULUS” apabila sebaliknya. Selain itu, fungsi get_predikat(nilai) digunakan untuk menentukan predikat nilai siswa berdasarkan rentang nilai tertentu, mulai dari predikat A hingga E. Kedua fungsi ini berperan penting dalam proses evaluasi hasil belajar siswa secara otomatis dan konsisten.



```python
def export_pdf_for_student(nisn, nama, kelas, nilai_dict):
    rata, status = hitung_rata_status(nilai_dict)
    predikat_rata = get_predikat(rata)

    safe_name = nama.replace(" ", "_")
    default_filename = f"Rapor_{nisn}_{safe_name}.pdf"

    # Dialog Save As
    filepath = filedialog.asksaveasfilename(
        title="Simpan Rapor PDF",
        initialfile=default_filename,
        defaultextension=".pdf",
        filetypes=[("PDF Files", "*.pdf")]
    )

    if not filepath:
        return None

    pdf = FPDF()
    pdf.add_page()
    pdf.set_auto_page_break(auto=True, margin=15)

    pdf.set_font("Arial", "B", 16)
    pdf.cell(0, 10, "LAPORAN HASIL BELAJAR SISWA", ln=True, align="C")
    pdf.cell(0, 10, "(RAPOR)", ln=True, align="C")
    pdf.ln(6)

    pdf.set_font("Arial", size=12)

    label_w = 40
    colon_w = 2
    value_w = 0
```

Gambar 6

```python
def row(label, value):
    pdf.cell(label_w, 8, label)
    pdf.cell(colon_w, 8, ":")
    pdf.cell(value_w, 8, str(value), ln=True)

row("Nama Peserta Didik", nama)
row("NISN", nisn)
row("Kelas", kelas)

pdf.ln(6)
col_no = 15
col_mapel = 85
col_nilai = 30
col_pred = 30

table_width = col_no + col_mapel + col_nilai + col_pred
page_width = pdf.w - 2 * pdf.l_margin
start_x = (page_width - table_width) / 2 + pdf.l_margin

pdf.set_font("Arial", "B", 12)
pdf.set_x(start_x)
pdf.cell(col_no, 8, "No", border=1, align="C")
pdf.cell(col_mapel, 8, "Mata Pelajaran", border=1, align="C")
pdf.cell(col_nilai, 8, "Nilai", border=1, align="C")
pdf.cell(col_pred, 8, "Predikat", border=1, align="C", ln=True)

pdf.set_font("Arial", size=12)
for i, (mapel, n) in enumerate(nilai_dict.items(), start=1):
    pred = get_predikat(n)
    pdf.set_x(start_x)
    pdf.cell(col_no, 8, str(i), border=1, align="C")
    pdf.cell(col_mapel, 8, mapel, border=1)
    pdf.cell(col_nilai, 8, str(n), border=1, align="C")
    pdf.cell(col_pred, 8, pred, border=1, align="C", ln=True)


pdf.ln(6)
pdf.set_font("Arial", size=12)
label_w = 40
colon_w = 2
value_w = 0
def row(label, value):
    pdf.cell(label_w, 8, label)
    pdf.cell(colon_w, 8, ":")
    pdf.cell(value_w, 8, str(value), ln=True)

row("Nilai Rata-rata", f"{rata:.2f}")
row("Predikat Rata-rata", predikat_rata)
row("Status Kelulusan", status)

pdf.output(filepath)
return filepath
```

Gambar 7

Fungsi export_pdf_for_student() digunakan untuk mengekspor data rapor siswa ke dalam bentuk file PDF. Fungsi ini diawali dengan perhitungan nilai rata-rata dan penentuan status kelulusan serta predikat rata-rata berdasarkan nilai yang dimiliki siswa. Selanjutnya, sistem menampilkan dialog penyimpanan file agar pengguna dapat menentukan lokasi dan nama file PDF yang akan dibuat. Setelah itu, aplikasi membentuk dokumen PDF yang berisi judul rapor, identitas siswa, tabel nilai mata pelajaran beserta predikatnya, serta ringkasan nilai rata-rata dan status kelulusan. Dengan adanya fungsi ini, hasil rapor siswa dapat disimpan secara digital dalam format PDF yang rapi, mudah dibaca, dan siap digunakan sebagai arsip maupun dokumen cetak.

```python
# GUI App
ctk.set_appearance_mode("Light")
ctk.set_default_color_theme("blue")

class RaporApp(ctk.CTk):
    def __init__(self):
        super().__init__()
        self.title("Aplikasi Rapor Sederhana")
        self.geometry("980x640")
        self.minsize(860, 540)

        self.data = load_data()

        # Layout: sidebar + content frame
        self.grid_rowconfigure(0, weight=1)
        self.grid_columnconfigure(1, weight=1)

        self.sidebar = ctk.CTkFrame(self, width=220)
        self.sidebar.grid(row=0, column=0, sticky="nswe", padx=10, pady=10)
        self.sidebar.grid_rowconfigure(8, weight=1)

        self.content = ctk.CTkFrame(self)
        self.content.grid(row=0, column=1, sticky="nswe", padx=(0, 10), pady=10)
        self.content.grid_rowconfigure(0, weight=1)
        self.content.grid_columnconfigure(0, weight=1)


        # Sidebar buttons
        self.btn_dashboard = ctk.CTkButton(self.sidebar, text="Dashboard", command=self.show_dashboard)
        self.btn_dashboard.grid(row=0, column=0, padx=12, pady=8, sticky="we")

        self.btn_siswa = ctk.CTkButton(self.sidebar, text="Tambah Siswa", command=self.show_siswa)
        self.btn_siswa.grid(row=1, column=0, padx=12, pady=8, sticky="we")

        self.btn_nilai = ctk.CTkButton(self.sidebar, text="Input Nilai", command=self.show_nilai)
        self.btn_nilai.grid(row=2, column=0, padx=12, pady=8, sticky="we")

        self.btn_search = ctk.CTkButton(self.sidebar, text="Search Siswa", command=self.show_search)
        self.btn_search.grid(row=3, column=0, padx=12, pady=8, sticky="we")

        self.btn_mapel = ctk.CTkButton(self.sidebar, text="Search Mapel", command=self.show_mapel)
        self.btn_mapel.grid(row=4, column=0, padx=12, pady=8, sticky="we")

        # Simpan semua tombol sidebar
        self.sidebar_buttons = [
            self.btn_dashboard,
            self.btn_siswa,
            self.btn_nilai,
            self.btn_search,
            self.btn_mapel
        ]

        self.COLOR_NORMAL = self.btn_dashboard.cget("fg_color")
        self.COLOR_ACTIVE = self.btn_dashboard.cget("hover_color")

        # Create pages
        self.pages = {}
        for Page in (DashboardPage, SiswaPage, NilaiPage, SearchPage, MapelPage):
            page = Page(self.content, self)
            self.pages[Page.__name__] = page
            page.grid(row=0, column=0, sticky="nsew")

        self.show_dashboard()
```

Gambar 8

Konfigurasi GUI:

- ctk.set_appearance_mode(“Light”): Mengatur mode tampilan aplikasi menjadi mode terang.
- ctk.set_default_color_theme(“blue”): Mengatur tema warna utama aplikasi menjadi biru agar tampilan konsisten.

Class RaporApp:

def __init__

- Instalasi window:
  - self.title(): menentukan judul aplikasi yang ditampilkan pada jendela.
  - self.geometry(): mengatur ukuran awal jendela aplikasi.
  - self.minsize(): menentukan ukuran minimum jendela agar tampilan tetap rapi.
- Inisialisasi Data
  - self.data = load_data(): memuat data siswa dari file JSON saat aplikasi pertama kali dijalankan sehingga data langsung tersedia.
- Layout Structure
  - Aplikasi menggunakan sistem layout grid yang dibagi menjadi dua bagian utama, yaitu sidebar dan area konten.
  - Sidebar digunakan sebagai menu navigasi, sedangkan area konten digunakan untuk menampilkan halaman aplikasi.
- Sidebar
  - Sidebar berisi tombol-tombol navigasi seperti Dashboard, Tambah Siswa, Input Nilai, Search Siswa, dan Search Mapel.
  - Setiap tombol terhubung dengan fungsi tertentu untuk menampilkan halaman yang sesuai.
- Active Button
  - Warna tombol normal digunakan untuk menandai menu tidak aktif.
  - Warna hover digunakan sebagai indikator menu yang sedang aktif.
  - Fitur ini membantu pengguna mengetahui halaman yang sedang digunakan.
- Create Page
  - Semua halaman aplikasi dibuat satu kali di awal program.
  - Warna hover digunakan sebagai indikator menu yang sedang aktif.
  - Fitur ini membantu pengguna mengetahui halaman yang sedang digunakan.

```
def show_page(self, page_name):
    page = self.pages.get(page_name)
    if page:
        try:
            page.update_contents()
        except Exception:
            pass
        page.tkraise()

def set_active_sidebar(self, active_btn):
    for btn in self.sidebar_buttons:
        btn.configure(fg_color=self.COLOR_NORMAL)
    active_btn.configure(fg_color=self.COLOR_ACTIVE)

def show_dashboard(self):
    self.show_page("DashboardPage")
    self.set_active_sidebar(self.btn_dashboard)

def show_siswa(self):
    self.show_page("SiswaPage")
    self.set_active_sidebar(self.btn_siswa)

def show_nilai(self):
    self.show_page("NilaiPage")
    self.set_active_sidebar(self.btn_nilai)

def show_search(self):
    self.show_page("SearchPage")
    self.set_active_sidebar(self.btn_search)

def show_mapel(self):
    self.show_page("MapelPage")
    self.set_active_sidebar(self.btn_mapel)
```

Gambar 9

def show_page:

- Fungsi ini digunakan untuk menampilkan halaman tertentu berdasarkan nama halaman (page_name).
- Halaman diambil dari dictionary self.pages yang menyimpan seluruh halaman aplikasi.
- Jika halaman memiliki fungsi update_contents(), fungsi tersebut akan dijalankan untuk memperbarui isi halaman sebelum ditampilkan.
- Metode tkraise() digunakan untuk menampilkan halaman terpilih ke bagian paling depan.

def set_active_sidebar:

- Fungsi ini digunakan untuk mengatur tampilan tombol sidebar yang sedang aktif.
- Semua tombol sidebar terlebih dahulu dikembalikan ke warna normal.
- Tombol yang dipilih kemudian diubah warnanya menjadi warna aktif sebagai penanda halaman yang sedang digunakan.

def show_(nama page):

- Fungsi ini digunakan untuk menampilkan masing-masing halaman
- Fungsi memanggil show_page() dengan parameter halaman.
- Tombol pada sidebar ditandai sebagai tombol aktif.

```python
class DashboardPage(ctk.CTkFrame):
    def __init__(self, parent, app):
        super().__init__(parent)
        self.app = app

        self.grid_rowconfigure(2, weight=1)
        self.grid_columnconfigure(0, weight=1)

        # TITLE
        self.label_title = ctk.CTkLabel(
            self,
            text="Dashboard",
            font=ctk.CTkFont(size=20, weight="bold")
        )
        self.label_title.pack(pady=(10, 4))

        self.label_sub = ctk.CTkLabel(
            self,
            text="Ringkasan data akademik siswa",
            font=ctk.CTkFont(size=13),
            text_color="gray"
        )
        self.label_sub.pack(pady=(0, 10))

        # CARD BACKGROUND
        self.card_bg = ctk.CTkFrame(
            self,
            fg_color=("gray92", "gray14"),
            corner_radius=16
        )
        self.card_bg.pack(padx=14, pady=8, fill="x")

        # CARD CONTAINER
        self.card_frame = ctk.CTkFrame(self.card_bg, fg_color="transparent")
        self.card_frame.pack(padx=10, pady=10, fill="x")
        self.card_frame.grid_columnconfigure((0, 1, 2), weight=1)

        self.card_total = self._create_card(
            self.card_frame, "Total Siswa", "0"
        )
        self.card_total.grid(row=0, column=0, padx=(0, 8), sticky="we")

        self.card_avg = self._create_card(
            self.card_frame, "Persentase Lulus", "0%"
        )
        self.card_avg.grid(row=0, column=1, padx=8, sticky="we")

        self.card_lulus = self._create_card(
            self.card_frame, "Jumlah Lulus", "0"
        )
        self.card_lulus.grid(row=0, column=2, padx=(8, 0), sticky="we")

        # DETAIL BOX
        self.info_box = ctk.CTkTextbox(self, state="disabled")
        self.info_box.pack(padx=10, pady=10, fill="both", expand=True)

        self.info_box.tag_config("lulus", foreground=COLOR_LULUS)
        self.info_box.tag_config("tidak_lulus", foreground=COLOR_TIDAK_LULUS)
```

Gambar 10

Class DashboardPage

def __init__:

- Tittle:
  - Label “Dashboard” ditampilkan sebagai judul utama dengan ukuran font lebih besar dan tebal.

- Subjudul “Ringkasan data akademik siswa” digunakan untuk menjelaskan fungsi halaman dashboard secara singkat.

- Card Container:
  - Card digunakan untuk menampilkan informasi ringkas dalam bentuk visual.
  - Terdapat tiga card utama yaitu Total Siswa, Persentase Lulus, dan Jumlah Lulus.
  - Card diletakkan dalam satu container dengan latar belakang dan sudut membulat agar tampilan lebih modern dan rapi.
- Detail Box:
  - `CTkTextbox` digunakan untuk menampilkan detail data seluruh siswa.
  - Textbox dibuat dalam keadaan nonaktif (`disabled`) agar tidak dapat diedit oleh pengguna.
  - Warna teks dibedakan menggunakan tag. Warna hijau untuk siswa yang lulus, sedangkan warna merah untuk siswa yang tidak lulus untuk memudahkan pengguna dalam membaca status kelulusan siswa.
  -

```python
def _create_card(self, parent, title, value):
    frame = ctk.CTkFrame(
        parent,
        height=90,
        fg_color=("white", "gray10"),
        border_width=1,
        border_color=("gray70", "gray30"),
        corner_radius=12
    )
    frame.pack_propagate(False)

    lbl_title = ctk.CTkLabel(
        frame,
        text=title,
        font=ctk.CTkFont(size=13)
    )
    lbl_title.pack(pady=(10, 0))

    lbl_value = ctk.CTkLabel(
        frame,
        text=value,
        font=ctk.CTkFont(size=22, weight="bold")
    )
    lbl_value.pack(pady=(2, 6))

    frame.value_label = lbl_value
    return frame
```

Gambar 11

def _create_card:

- Fungsi _create_card digunakan untuk membuat komponen card pada halaman Dashboard yang berfungsi menampilkan informasi ringkas seperti total siswa, persentase kelulusan, dan jumlah siswa lulus.
- Fungsi ini menerima parameter parent sebagai wadah peletakan card, title sebagai judul informasi, dan value sebagai nilai yang akan ditampilkan.
- Card dibuat menggunakan CTkFrame dengan tinggi tetap, warna latar tertentu, garis tepi, dan sudut membulat untuk menghasilkan tampilan yang rapi dan modern.
- pack_propagate(False) digunakan agar ukuran card tidak berubah mengikuti isi di dalamnya.
- Di dalam card terdapat label judul dengan ukuran font lebih kecil sebagai keterangan data dan label nilai dengan ukuran font lebih besar serta tebal agar mudah dibaca.
- Label nilai disimpan sebagai atribut value_label pada frame sehingga data pada card dapat diperbarui secara dinamis tanpa membuat ulang komponen.
- Fungsi ini mengembalikan frame card yang siap digunakan pada halaman Dashboard.

```
def update_contents(self):
    data = self.app.data
    total = len(data)

    count_nilai = 0
    lulus = 0

    for info in data.values():
        nilai = info.get("nilai", {})
        if nilai:
            count_nilai += 1
            _, status = hitung_rata_status(nilai)
            if status == "LULUS":
                lulus += 1

    persen_lulus = (lulus / count_nilai * 100) if count_nilai else 0

    # UPDATE CARD
    self.card_total.value_label.configure(text=str(total))
    self.card_avg.value_label.configure(text=f"{persen_lulus:.0f}%")
    self.card_lulus.value_label.configure(text=str(lulus))

    # UPDATE DETAIL BOX
    self.info_box.configure(state="normal")
    self.info_box.delete("1.0", "end")

    if not data:
        self.info_box.insert("end", "Belum ada data siswa.\n")
    else:
        for idx, (nisn, info) in enumerate(sorted(data.items()), start=1):
            nama = info.get("nama", "-")
            kelas = info.get("kelas", "-")
            nilai = info.get("nilai")

            self.info_box.insert(
                "end", f"{idx}. {nisn} — {nama} — {kelas}\n"
            )

            if not nilai:
                self.info_box.insert(
                    "end", "     Nilai belum diinput\n\n"
                )
            else:
                rata, status = hitung_rata_status(nilai)
                self.info_box.insert(
                    "end", f"     Rata-rata : {rata:.2f} | Status: "
                )

                if status == "LULUS":
                    self.info_box.insert(
                        "end", "LULUS\n\n", "lulus"
                    )
                else:
                    self.info_box.insert(
                        "end", "TIDAK LULUS\n\n", "tidak_lulus"
                    )

    self.info_box.configure(state="disabled")
```

Gambar 12

def update_contents:

- Fungsi update_contents berfungsi untuk memperbarui tampilan Dashboard berdasarkan data siswa yang tersimpan dalam aplikasi.
- Fungsi ini menghitung jumlah total siswa, jumlah siswa yang telah memiliki nilai, serta jumlah siswa dengan status kelulusan LULUS.
- Persentase kelulusan dihitung dari perbandingan jumlah siswa lulus terhadap jumlah siswa yang memiliki nilai.
- Hasil perhitungan ditampilkan pada komponen card Dashboard, yaitu total siswa, persentase kelulusan, dan jumlah siswa lulus.
- Selain itu, fungsi ini menampilkan ringkasan detail data siswa pada info_box, termasuk nilai rata-rata dan status kelulusan.

- Tampilan status kelulusan dibedakan menggunakan warna untuk memudahkan interpretasi informasi.

- Setelah pembaruan selesai, komponen info_box dikembalikan ke kondisi non-editable.

```
class SiswaPage(ctk.CTkFrame):
    def __init__(self, parent, app):
        super().__init__(parent)
        self.app = app

        title = ctk.CTkLabel(self, text="Input Data Siswa", font=ctk.CTkFont(size=18, weight="bold"))
        title.pack(pady=(10, 8))

        card = ctk.CTkFrame(
            self,
            corner_radius=16,
            fg_color=("gray92", "gray14")
        )
        card.pack(padx=40, pady=20)

        form_frame = ctk.CTkFrame(card, fg_color="transparent")
        form_frame.pack(padx=30, pady=20)
        form_frame.grid_columnconfigure(1, minsize=280)


        lbl_nisn = ctk.CTkLabel(form_frame, text="NISN", anchor="w")
        lbl_nisn.grid(row=0, column=0, padx=6, pady=6, sticky="w")
        self.entry_nisn = ctk.CTkEntry(form_frame, placeholder_text="1234567890")
        self.entry_nisn.grid(row=0, column=1, padx=6, pady=6)

        lbl_name = ctk.CTkLabel(form_frame, text="Nama lengkap", anchor="w")
        lbl_name.grid(row=1, column=0, padx=6, pady=6, sticky="w")
        self.entry_name = ctk.CTkEntry(form_frame, placeholder_text="Nama lengkap")
        self.entry_name.grid(row=1, column=1, padx=6, pady=6)

        lbl_kelas = ctk.CTkLabel(form_frame, text="Kelas", anchor="w")
        lbl_kelas.grid(row=2, column=0, padx=6, pady=6, sticky="w")
        self.entry_kelas = ctk.CTkEntry(form_frame, placeholder_text="Kelas")
        self.entry_kelas.grid(row=2, column=1, padx=6, pady=6)

        btn_frame = ctk.CTkFrame(card, fg_color="transparent")
        btn_frame.pack(pady=(0, 20))
        btn_add = ctk.CTkButton(
            btn_frame,
            text="Tambah",
            width=120,
            command=self.add_siswa
        )
        btn_add.pack(side="left", padx=8)

        btn_clear = ctk.CTkButton(
            btn_frame,
            text="Clear",
            width=120,
            fg_color="gray",
            command=self.clear_form
        )
        btn_clear.pack(side="left", padx=8)
```

Gambar 13

Class SiswaPage

def __init__:

- Title:
  - Label “Input Data Siswa” ditampilkan sebagai judul utama halaman dengan ukuran font lebih besar dan tebal.
  - Judul berfungsi sebagai penanda bahwa halaman ini digunakan untuk pengisian data siswa.

- Card Container:
  - Card digunakan sebagai wadah utama form input agar tampilan lebih terstruktur dan rapi.
  - Card memiliki sudut membulat dan warna latar yang menyesuaikan mode terang dan gelap
- Form Input:
  - Form input disusun dalam sebuah frame transparan untuk menjaga konsistensi tampilan.
  - Field NISN disediakan untuk memasukkan Nomor Induk Siswa Nasional.
  - Field Nama Lengkap digunakan untuk memasukkan identitas siswa.
  - Field kelas digunakan untuk memasukkan kelas siswa.
  - Setiap field dilengkapi dengan label untuk memudahkan pengguna dalam pengisian data.
- Button Control:
  - Tombol Tambah digunakan untuk menyimpan data siswa ke dalam sistem dengan memanggil fungsi add_siswa.
  - Tombol Clear digunakan untuk mengosongkan seluruh field input dengan memanggil fungsi clear_form

```
def update_contents(self):
    pass

def clear_form(self):
    self.entry_nisn.delete(0, "end")
    self.entry_name.delete(0, "end")
    self.entry_kelas.delete(0, "end")

def add_siswa(self):
    nisn = self.entry_nisn.get().strip()
    nama = self.entry_name.get().strip()
    kelas = self.entry_kelas.get().strip()

    # =VALIDASI INPUT=
    # NISN wajib, angka, dan 10 digit
    if not nisn:
        messagebox.showerror("Error", "NISN wajib diisi.")
        return
    if not nisn.isdigit():
        messagebox.showerror("Error", "NISN harus berupa angka.")
        return

    # Nama wajib dan tidak boleh angka
    if not nama:
        messagebox.showerror("Error", "Nama siswa wajib diisi.")
        return
    if any(char.isdigit() for char in nama):
        messagebox.showerror("Error", "Nama tidak boleh mengandung angka.")
        return
    nama = nama.title()

    # Kelas wajib diisi
    if not kelas:
        messagebox.showerror("Error", "Kelas wajib diisi.")
        return
    kelas = kelas.upper()

    # CEK DUPLIKAT NISN
    if nisn in self.app.data:
        messagebox.showerror("Error", f"NISN '{nisn}' sudah terdaftar (Nama: {self.app.data[nisn].get('nama')}).")
        return

    # SIMPAN DATA
    self.app.data[nisn] = {"nama": nama,"kelas": kelas}
    save_data(self.app.data)

    # Update halaman lain
    self.app.refresh_all()

    messagebox.showinfo(
        "Sukses",
        f"Data siswa '{nama}' (NISN: {nisn}) berhasil disimpan."
    )
    self.clear_form()

```

Gambar 14

def update_contents:

- Metode disediakan sebagai placeholder untuk pembaruan konten halaman di masa pengembangan selanjutnya.

def clear_form:

- Metode clear_form berfungsi untuk mengosongkan seluruh field input, yaitu NISN, nama siswa, dan kelas.

def add_siswa:

- Metode add_siswa digunakan untuk menambahkan data siswa ke dalam sistem.
- Proses diawali dengan pengambilan data dari field input.
- Sistem melakukan validasi terhadap data yang dimasukkan, meliputi:
- NISN wajib diisi dan harus berupa angka.
- Nama siswa wajib diisi dan tidak boleh mengandung angka.
- Kelas wajib diisi.
- Sistem memeriksa duplikasi NISN untuk mencegah data ganda.
- Data siswa yang valid disimpan ke dalam struktur data aplikasi dan disimpan secara permanen.
- Setelah penyimpanan, sistem memperbarui seluruh halaman aplikasi.

- Sistem menampilkan notifikasi keberhasilan dan mengosongkan kembali form input.

```
class NilaiPage(ctk.CTkFrame):
    def __init__(self, parent, app):
        super().__init__(parent)
        self.app = app

        title = ctk.CTkLabel(self, text="Input Nilai Siswa",
                             font=ctk.CTkFont(size=18, weight="bold"))
        title.pack(pady=(10, 8))

        # Bagian Search
        self.search_entry = ctk.CTkEntry(self, placeholder_text="Cari NISN / Nama...")
        self.search_entry.pack(padx=10, pady=(8, 4), fill="x")

        self.btn_search = ctk.CTkButton(self, text="Cari", command=self.search_siswa)
        self.btn_search.pack(padx=10, pady=4)

        self.search_result = ctk.CTkLabel(self, text="", font=ctk.CTkFont(size=14))
        self.search_result.pack(pady=4)

        # Frame Form Nilai (disembunyikan dulu)
        self.form_frame = ctk.CTkFrame(self)
        self.form_frame.pack_forget()

        self.subject_vars = {}

        # Buat form nilai di dalam self.form_frame
        for sub in SUBJECTS:
            row = ctk.CTkFrame(self.form_frame)
            row.pack(fill="x", padx=10, pady=4)

            lbl = ctk.CTkLabel(row, text=sub, width=140, anchor="w")
            lbl.pack(side="left")

            ent = ctk.CTkEntry(row, placeholder_text="0 - 100")
            ent.pack(side="left", fill="x", expand=True)

            self.subject_vars[sub] = ent

        # tombol submit + clear
        btn_row = ctk.CTkFrame(self.form_frame)
        btn_row.pack(pady=8)

        self.btn_submit = ctk.CTkButton(btn_row, text="Simpan Nilai", command=self.add_nilai)
        self.btn_submit.pack(side="left", padx=5)

        self.btn_clear = ctk.CTkButton(btn_row, text="Clear", command=self.clear_form)
        self.btn_clear.pack(side="left", padx=5)

        self.selected_nisn = None
        self.selected_name = None
```

Gambar 15

Class NilaiPage:

def __init__:

- Title:
  - Label Input Nilai Siswa ditampilkan sebagai judul halaman dengan ukuran font lebih besar dan tebal.
- Search Section:
  - Field pencarian disediakan untuk mencari siswa berdasarkan NISN atau nama.
  - Tombol Cari digunakan untuk menampilkan data siswa yang dicari.
  - Hasil pencarian ditampilkan dalam bentuk label informasi.
- Form Nilai:
  - Form input nilai ditempatkan dalam sebuah frame yang disembunyikan secara default.

    - Field nilai dibuat secara dinamis berdasarkan daftar mata pelajaran (SUBJECTS).
    - Setiap mata pelajaran memiliki satu field input nilai dengan rentang 0–100.
- Button Control:
    - Tombol Simpan Nilai digunakan untuk menyimpan data nilai siswa.
    - Tombol Clear digunakan untuk mengosongkan seluruh field nilai.
- Data Seleksi:
    - Variabel (selected_nisn) dan (selected_name) digunakan untuk menyimpan identitas siswa yang sedang dipilih

```python
def search_siswa(self):
    key = self.search_entry.get().strip()

    if not key:
        messagebox.showwarning("Kosong", "Masukkan NISN atau Nama.")
        return

    found = None

    # Cari berdasarkan NISN atau nama
    for nisn, info in self.app.data.items():
        if key == nisn or key.lower() in info["nama"].lower():
            found = (nisn, info)
            break

    if not found:
        self.search_result.configure(text="Siswa tidak ditemukan.")
        self.form_frame.pack_forget()
        return

    # Data ditemukan
    self.selected_nisn = found[0]
    self.selected_name = found[1]["nama"]

    self.search_result.configure(
        text=f"✓ Ditemukan: {self.selected_name} (NISN {self.selected_nisn})"
    )

    # tampilkan form nilai
    self.form_frame.pack(padx=10, pady=10, fill="x")

# Clear Form
def clear_form(self):
    for ent in self.subject_vars.values():
        ent.delete(0, "end")

# Tambah Nilai
def add_nilai(self):
    if not self.selected_nisn:
        messagebox.showwarning("Pilih siswa", "Cari dan pilih siswa dahulu.")
        return

    nilai_dict = {}

    # validasi tiap mapel (nilai kosong diabaikan)
    try:
        for sub, ent in self.subject_vars.items():
            v = ent.get().strip()
            if v == "":
                continue  # kosong = tidak dimasukkan
            n = int(v)
            if not (0 <= n <= 100):
                raise ValueError(f"Nilai {sub} harus 0-100.")
            nilai_dict[sub] = n
    except ValueError as e:
        messagebox.showerror("Kesalahan nilai", str(e))
        return

    # Simpan nilai ke data siswa
    if "nilai" not in self.app.data[self.selected_nisn]:
        self.app.data[self.selected_nisn]["nilai"] = {}

    # update nilai yang ada (tidak menimpa mapel lain yang sudah ada)
    self.app.data[self.selected_nisn]["nilai"].update(nilai_dict)
    save_data(self.app.data)

    self.app.refresh_all()

    messagebox.showinfo(
        "Sukses",
        f"Nilai untuk {self.selected_name} (NISN {self.selected_nisn}) berhasil disimpan!"
    )

    self.clear_form()
```

Gambar 16

def search_siswa:

- Metode search_siswa berfungsi untuk mencari data siswa berdasarkan NISN atau nama.
- Sistem melakukan validasi agar kolom pencarian tidak kosong.

- Pencarian dilakukan dengan mencocokkan input pengguna terhadap NISN atau nama siswa.
- Jika data tidak ditemukan, sistem menampilkan pesan dan menyembunyikan form input nilai.

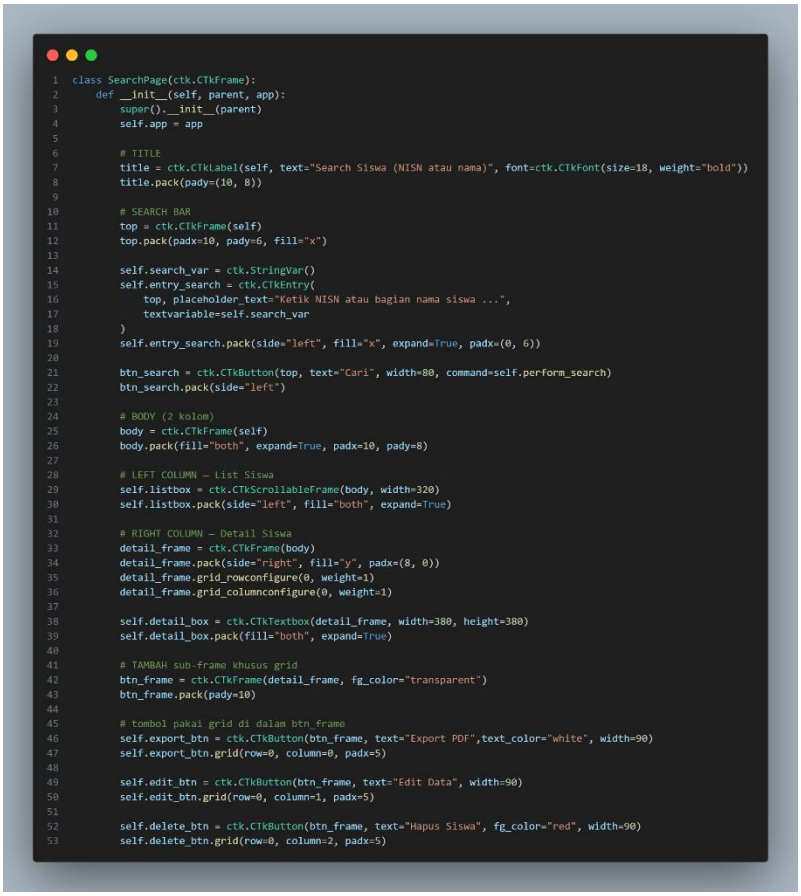
- Jika data ditemukan, identitas siswa disimpan dan form input nilai ditampilkan.

def clear_form:

- Metode clear_form digunakan untuk mengosongkan seluruh field input nilai mata pelajaran.

def add_nilai:

- Metode add_nilai berfungsi untuk menyimpan nilai siswa ke dalam sistem.
- Sistem memastikan siswa telah dipilih sebelum proses penyimpanan.
- Nilai divalidasi agar berada pada rentang 0–100, sedangkan field kosong diabaikan.
- Nilai yang valid disimpan dan diperbarui tanpa menghapus nilai mata pelajaran lain.
- Setelah penyimpanan, sistem memperbarui seluruh halaman aplikasi dan menampilkan notifikasi keberhasilan.
- Form input nilai dikosongkan kembali setelah proses selesai.



```python
class SearchPage(ctk.CTkFrame):
    def __init__(self, parent, app):
        super().__init__(parent)
        self.app = app

        # TITLE
        title = ctk.CTkLabel(self, text="Search Siswa (NISN atau nama)", font=ctk.CTkFont(size=18, weight="bold"))
        title.pack(pady=(10, 8))

        # SEARCH BAR
        top = ctk.CTkFrame(self)
        top.pack(padx=10, pady=6, fill="x")

        self.search_var = ctk.StringVar()
        self.entry_search = ctk.CTkEntry(
            top, placeholder_text="Ketik NISN atau bagian nama siswa ...",
            textvariable=self.search_var
        )
        self.entry_search.pack(side="left", fill="x", expand=True, padx=(0, 6))

        btn_search = ctk.CTkButton(top, text="Cari", width=80, command=self.perform_search)
        btn_search.pack(side="left")

        # BODY (2 kolom)
        body = ctk.CTkFrame(self)
        body.pack(fill="both", expand=True, padx=10, pady=8)

        # LEFT COLUMN — List Siswa
        self.listbox = ctk.CTkScrollableFrame(body, width=320)
        self.listbox.pack(side="left", fill="both", expand=True)

        # RIGHT COLUMN — Detail Siswa
        detail_frame = ctk.CTkFrame(body)
        detail_frame.pack(side="right", fill="y", padx=(8, 0))
        detail_frame.grid_rowconfigure(0, weight=1)
        detail_frame.grid_columnconfigure(0, weight=1)

        self.detail_box = ctk.CTkTextbox(detail_frame, width=380, height=380)
        self.detail_box.pack(fill="both", expand=True)

        # TAMBAH sub-frame khusus grid
        btn_frame = ctk.CTkFrame(detail_frame, fg_color="transparent")
        btn_frame.pack(pady=10)

        # tombol pakai grid di dalam btn_frame
        self.export_btn = ctk.CTkButton(btn_frame, text="Export PDF",text_color="white", width=90)
        self.export_btn.grid(row=0, column=0, padx=5)

        self.edit_btn = ctk.CTkButton(btn_frame, text="Edit Data", width=90)
        self.edit_btn.grid(row=0, column=1, padx=5)

        self.delete_btn = ctk.CTkButton(btn_frame, text="Hapus Siswa", fg_color="red", width=90)
        self.delete_btn.grid(row=0, column=2, padx=5)
```

Gambar 17

class SearchPage:

def __init__:

- Title:
  - Label “Search Siswa (NISN atau Nama)” ditampilkan sebagai judul halaman dengan font lebih besar dan tebal.
- Search Bar:
  - Kolom pencarian disediakan untuk mencari data siswa berdasarkan NISN atau bagian nama.
  - Tombol Cari digunakan untuk menjalankan proses pencarian data siswa.
- Body Layout:
  - Tampilan utama dibagi menjadi dua kolom untuk meningkatkan keterbacaan dan navigasi data.
- List Siswa:
  - Kolom kiri menampilkan daftar siswa dalam bentuk scrollable frame.

- Detail Siswa:
    - Kolom kanan digunakan untuk menampilkan detail data siswa secara lengkap dalam CTkTextbox.
- Button Control:
    - Tombol Export PDF digunakan untuk mencetak rapor siswa ke dalam format PDF.
    - Tombol Edit Data digunakan untuk memperbarui data siswa.
    - Tombol Hapus Siswa digunakan untuk menghapus data siswa dari sistem.

```
def update_contents(self):
    self.perform_search()

# SEARCH FUNCTION
def perform_search(self):
    key = self.search_var.get().strip().lower()

    for widget in self.listbox.winfo_children():
        widget.destroy()

    found_any = False

    for nisn, info in sorted(self.app.data.items()):
        nama = info.get("nama", "")
        kelas = info.get("kelas", "")
        if key == "" or key in nisn.lower() or key in nama.lower():
            label = f"{nisn} – {nama} – {kelas}"
            btn = ctk.CTkButton(
                self.listbox, text=label, anchor="w",
                command=lambda n=nisn: self.show_detail(n)
            )
            btn.pack(fill="x", padx=6, pady=3)
            found_any = True

    if not found_any:
        lbl = ctk.CTkLabel(self.listbox, text="Tidak ada hasil.", fg_color="transparent")
        lbl.pack(padx=6, pady=6)

    self.clear_detail()

# CLEAR DETAIL
def clear_detail(self):
    self.detail_box.configure(state="normal")
    self.detail_box.delete("1.0", "end")
    self.detail_box.configure(state="disabled")

    self.export_btn.configure(state="disabled", command=None)
    self.edit_btn.configure(state="disabled", command=None)
    self.delete_btn.configure(state="disabled", command=None)
```

Gambar 18

def update_contents:

- Metode update_contents berfungsi untuk memperbarui tampilan halaman pencarian dengan menjalankan kembali proses pencarian data siswa.

def perform_search:

- Metode perform_search digunakan untuk menampilkan daftar siswa berdasarkan kata kunci pencarian.
- Sistem membersihkan daftar hasil pencarian sebelum menampilkan data terbaru.
- Pencarian dilakukan dengan mencocokkan kata kunci terhadap NISN atau nama siswa.

- Data siswa yang sesuai ditampilkan dalam bentuk daftar interaktif.
- Jika tidak ditemukan data yang sesuai, sistem menampilkan pesan bahwa hasil pencarian tidak tersedia.
- Setelah proses pencarian, detail siswa dikosongkan kembali.

def clear_detail:

- Metode clear_detail digunakan untuk mengosongkan tampilan detail siswa.
- Kotak detail dikembalikan ke kondisi non-editable.
- Seluruh tombol aksi dinonaktifkan hingga pengguna memilih data siswa.

```
def show_detail(self, nisn):
    info = self.app.data.get(nisn, {})
    nama = info.get("nama", "-")
    kelas = info.get("kelas", "-")
    nilai = info.get("nilai", {})

    # Hitung rata dan status
    rata, status = hitung_rata_status(nilai)

    # Siapkan textbox
    self.detail_box.configure(state="normal")
    self.detail_box.delete("1.0", "end")
    self.detail_box.configure(font=("Consolas", 13))

    # Fungsi baris sejajar
    def row(label, value, label_w=18):
        return f"{label:<{label_w}} : {value}\n"

    # IDENTITAS SISWA
    self.detail_box.insert("end", row("Nama Peserta Didik", nama))
    self.detail_box.insert("end", row("NISN", nisn))
    self.detail_box.insert("end", row("Kelas", kelas))
    self.detail_box.insert("end", "\n")

    # TABEL NILAI
    col1 = 17   # Mata Pelajaran
    col2 = 7    # Nilai
    col3 = 9    # Predikat

    self.detail_box.insert(
        "end",
        f"{'Mata Pelajaran':<{col1}} {'Nilai':^{col2}} {'Predikat':^{col3}}\n"
    )
    self.detail_box.insert("end", "-" * (col1 + col2 + col3 + 2) + "\n")

    for m, v in nilai.items():
        pred = get_predikat(v)
        self.detail_box.insert(
        "end",
        f"{m:<{col1}} {str(v):^{col2}} {pred:^{col3}}\n"
    )

    # RINGKASAN NILAI
    self.detail_box.tag_config("lulus", foreground=COLOR_LULUS)
    self.detail_box.tag_config("tidak_lulus", foreground=COLOR_TIDAK_LULUS)

    self.detail_box.insert("end", "\n")
    self.detail_box.insert("end", row("Nilai Rata-rata", f"{rata:.2f}"))
    self.detail_box.insert("end", row("Predikat Rata-rata", get_predikat(rata)))
    self.detail_box.insert("end", f"{'Status Kelulusan':<18} : ")

    if status == "LULUS":
        self.detail_box.insert("end", "LULUS\n", "lulus")
    else:
        self.detail_box.insert("end", "TIDAK LULUS\n", "tidak_lulus")

    self.detail_box.configure(state="disabled")

    # Aktifkan tombol export
    self.export_btn.configure(
        state="normal",
        command=lambda: self.export_pdf(nisn, nama, kelas, nilai)
    )

    # aktifkan tombol edit
    self.edit_btn.configure(
        state="normal",
        command=lambda: self.open_edit_popup(nisn)
    )

    # aktifkan tombol delete
    self.delete_btn.configure(
        state="normal",
        command=lambda: self.delete_student(nisn)
    )

#konfirmasi dan hapus siswa
def delete_student(self, nisn):
    confirm = messagebox.askyesno("Konfirmasi", f"Yakin ingin menghapus siswa NISN '{nisn}'?")
    if not confirm:
        return

    # hapus dari data
    if nisn in self.app.data:
        del self.app.data[nisn]
        save_data(self.app.data)
    messagebox.showinfo("Sukses", f"Siswa dengan NISN '{nisn}' berhasil dihapus.")

    # refresh halaman & other pages
    self.app.refresh_all()
```

Gambar 19

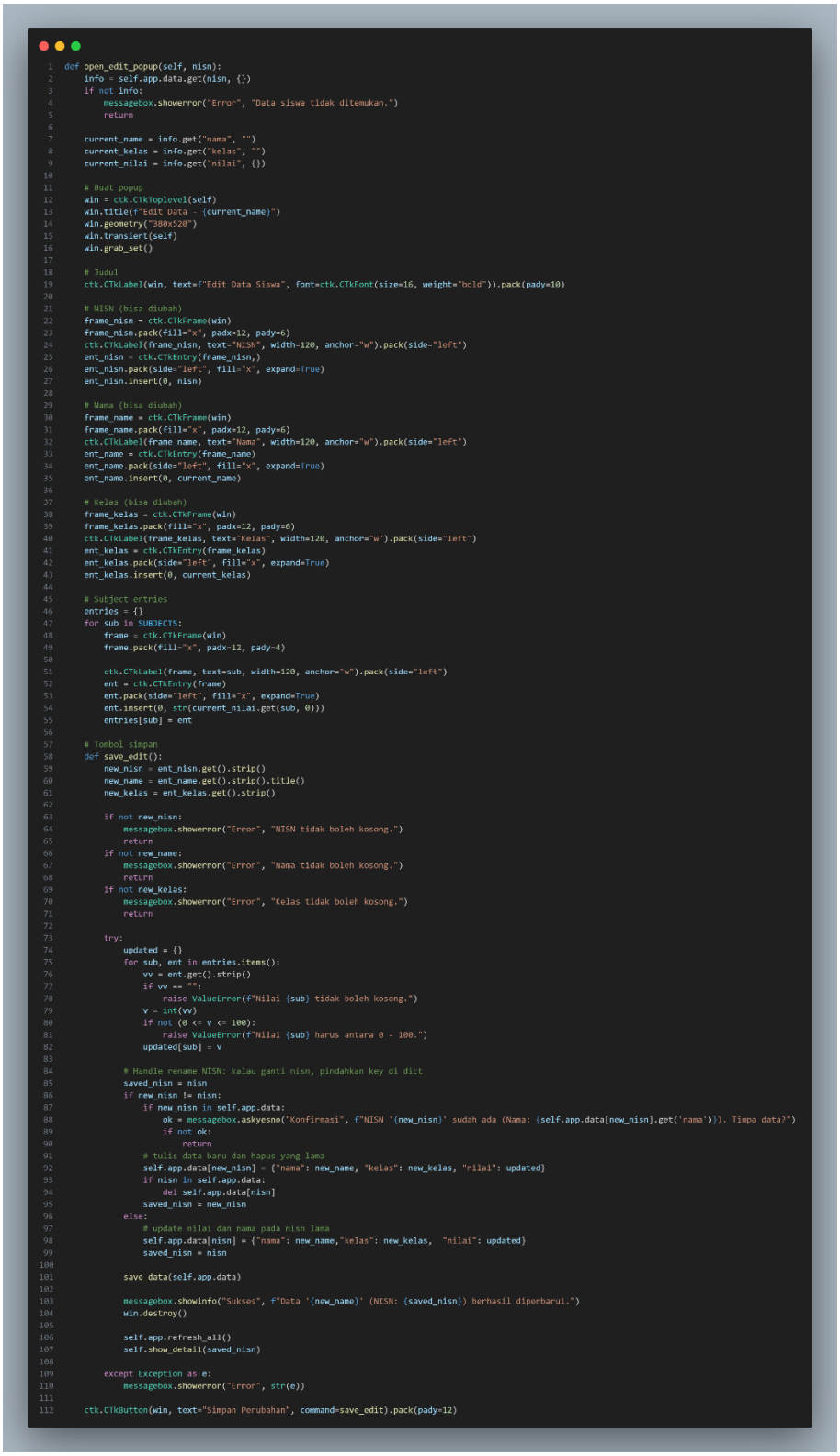
def show_detail:

- Metode show_detail berfungsi untuk menampilkan detail lengkap data siswa berdasarkan NISN yang dipilih.
- Sistem menampilkan identitas siswa yang meliputi nama, NISN, dan kelas.

- Data nilai ditampilkan dalam bentuk tabel yang memuat mata pelajaran, nilai, dan predikat.
- Sistem menghitung nilai rata-rata dan menentukan status kelulusan secara otomatis.
- Status kelulusan ditampilkan dengan perbedaan warna untuk memudahkan interpretasi informasi.
- Setelah data ditampilkan, tombol Export PDF, Edit Data, dan Hapus Siswa diaktifkan.

def delete_student:

- Metode delete_student digunakan untuk menghapus data siswa dari sistem.
- Sistem menampilkan konfirmasi penghapusan sebelum proses dilakukan.
- Data siswa yang dihapus akan disimpan secara permanen.
- Setelah penghapusan, sistem memperbarui seluruh halaman aplikasi dan menampilkan notifikasi keberhasilan.



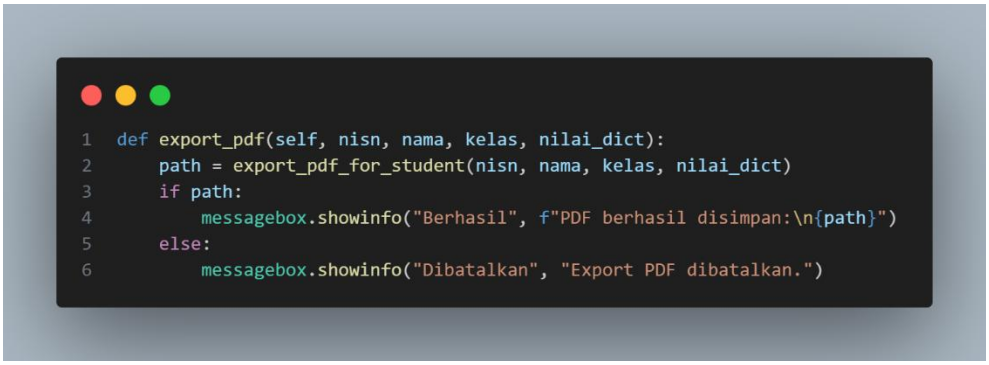
```python
def open_edit_popup(self, nisn):
    info = self.app.data.get(nisn, {})
    if not info:
        messagebox.showerror("Error", "Data siswa tidak ditemukan.")
        return

    current_name = info.get("nama", "")
    current_kelas = info.get("kelas", "")
    current_nilai = info.get("nilai", {})

    # Buat popup
    win = ctk.CTkToplevel(self)
    win.title(f"Edit Data - {current_name}")
    win.geometry("380x520")
    win.transient(self)
    win.grab_set()

    # Judul
    ctk.CTkLabel(win, text=f"Edit Data Siswa", font=ctk.CTkFont(size=16, weight="bold")).pack(pady=10)

    # NISN (bisa diubah)
    frame_nisn = ctk.CTkFrame(win)
    frame_nisn.pack(fill="x", padx=12, pady=6)
    ctk.CTkLabel(frame_nisn, text="NISN", width=120, anchor="w").pack(side="left")
    ent_nisn = ctk.CTkEntry(frame_nisn,)
    ent_nisn.pack(side="left", fill="x", expand=True)
    ent_nisn.insert(0, nisn)

    # Nama (bisa diubah)
    frame_name = ctk.CTkFrame(win)
    frame_name.pack(fill="x", padx=12, pady=6)
    ctk.CTkLabel(frame_name, text="Nama", width=120, anchor="w").pack(side="left")
    ent_name = ctk.CTkEntry(frame_name)
    ent_name.pack(side="left", fill="x", expand=True)
    ent_name.insert(0, current_name)

    # Kelas (bisa diubah)
    frame_kelas = ctk.CTkFrame(win)
    frame_kelas.pack(fill="x", padx=12, pady=6)
    ctk.CTkLabel(frame_kelas, text="Kelas", width=120, anchor="w").pack(side="left")
    ent_kelas = ctk.CTkEntry(frame_kelas)
    ent_kelas.pack(side="left", fill="x", expand=True)
    ent_kelas.insert(0, current_kelas)

    # Subject entries
    entries = {}
    for sub in SUBJECTS:
        frame = ctk.CTkFrame(win)
        frame.pack(fill="x", padx=12, pady=4)

        ctk.CTkLabel(frame, text=sub, width=120, anchor="w").pack(side="left")
        ent = ctk.CTkEntry(frame)
        ent.pack(side="left", fill="x", expand=True)
        ent.insert(0, str(current_nilai.get(sub, 0)))
        entries[sub] = ent

    # Tombol simpan
    def save_edit():
        new_nisn = ent_nisn.get().strip()
        new_name = ent_name.get().strip().title()
        new_kelas = ent_kelas.get().strip()

        if not new_nisn:
            messagebox.showerror("Error", "NISN tidak boleh kosong.")
            return
        if not new_name:
            messagebox.showerror("Error", "Nama tidak boleh kosong.")
            return
        if not new_kelas:
            messagebox.showerror("Error", "Kelas tidak boleh kosong.")
            return

        try:
            updated = {}
            for sub, ent in entries.items():
                vv = ent.get().strip()
                if vv == "":
                    raise ValueError(f"Nilai {sub} tidak boleh kosong.")
                v = int(vv)
                if not (0 <= v <= 100):
                    raise ValueError(f"Nilai {sub} harus antara 0 - 100.")
                updated[sub] = v

            # Handle rename NISN: kalau ganti nisn, pindahkan key di dict
            saved_nisn = nisn
            if new_nisn != nisn:
                if new_nisn in self.app.data:
                    ok = messagebox.askyesno("Konfirmasi", f"NISN '{new_nisn}' sudah ada (Nama: {self.app.data[new_nisn].get('nama')}). Timpa data?")
                    if not ok:
                        return
                # tulis data baru dan hapus yang lama
                self.app.data[new_nisn] = {"nama": new_name, "kelas": new_kelas, "nilai": updated}
                if nisn in self.app.data:
                    del self.app.data[nisn]
                saved_nisn = new_nisn
            else:
                # update nilai dan nama pada nisn lama
                self.app.data[nisn] = {"nama": new_name,"kelas": new_kelas,  "nilai": updated}
                saved_nisn = nisn

            save_data(self.app.data)

            messagebox.showinfo("Sukses", f"Data '{new_name}' (NISN: {saved_nisn}) berhasil diperbarui.")
            win.destroy()

            self.app.refresh_all()
            self.show_detail(saved_nisn)

        except Exception as e:
            messagebox.showerror("Error", str(e))

    ctk.CTkButton(win, text="Simpan Perubahan", command=save_edit).pack(pady=12)
```

Gambar 20

def open_edit_popup:

- Metode open_edit_popup berfungsi untuk menampilkan jendela pop-up yang digunakan dalam proses pengeditan data siswa.

- Sistem mengambil data siswa berdasarkan NISN dan menampilkan informasi awal berupa NISN, nama, kelas, serta nilai setiap mata pelajaran.
- Pop-up disajikan dalam bentuk form input yang memungkinkan pengguna memperbarui seluruh data siswa.
- Sistem melakukan validasi terhadap seluruh data yang dimasukkan, termasuk identitas siswa dan nilai mata pelajaran.
- Perubahan NISN ditangani dengan memindahkan data ke kunci baru tanpa kehilangan informasi nilai.
- Data yang telah diperbarui disimpan secara permanen dan ditampilkan notifikasi keberhasilan.
- Setelah proses selesai, tampilan aplikasi diperbarui dan detail siswa ditampilkan kembali.



```
def export_pdf(self, nisn, nama, kelas, nilai_dict):
    path = export_pdf_for_student(nisn, nama, kelas, nilai_dict)
    if path:
        messagebox.showinfo("Berhasil", f"PDF berhasil disimpan:\n{path}")
    else:
        messagebox.showinfo("Dibatalkan", "Export PDF dibatalkan.")
```

Gambar 21

def export_pdf:

- Metode export_pdf berfungsi untuk mengekspor rapor siswa ke dalam format PDF.
- Sistem memanggil fungsi export_pdf_for_student dengan parameter identitas siswa dan data nilai.
- Jika proses ekspor berhasil, sistem menampilkan notifikasi lokasi penyimpanan file PDF.
- Jika proses dibatalkan oleh pengguna, sistem menampilkan pesan bahwa ekspor PDF dibatalkan.

```python
class MapelPage(ctk.CTkFrame):
    def __init__(self, parent, app):
        super().__init__(parent)
        self.app = app

        title = ctk.CTkLabel(self, text="Search Mapel", font=ctk.CTkFont(size=18, weight="bold"))
        title.pack(pady=(10, 8))

        top = ctk.CTkFrame(self)
        top.pack(padx=10, pady=6, fill="x")

        self.mapel_var = ctk.StringVar(value=SUBJECTS[0])
        self.mapel_menu = ctk.CTkOptionMenu(
            top,
            values=SUBJECTS,
            variable=self.mapel_var,
            command=lambda _: self.show_mapel_list()  # refresh otomatis
        )
        self.mapel_menu.pack(side="left", padx=(0, 8))

        self.result_box = ctk.CTkTextbox(self, state="disabled")
        self.result_box.pack(fill="both", expand=True, padx=10, pady=8)
```

Gambar 22

Class MapelPage:

def __init__:

- Kelas MapelPage digunakan untuk menampilkan data nilai siswa berdasarkan mata pelajaran.
- Label “Search Mapel” ditampilkan sebagai judul halaman dengan font lebih besar dan tebal.
- Menu pilihan (OptionMenu) disediakan untuk memilih mata pelajaran dari daftar SUBJECTS.
- Perubahan pilihan mata pelajaran akan secara otomatis memperbarui tampilan data.
- Hasil pencarian ditampilkan dalam CTkTextbox yang dibuat dalam kondisi nonaktif untuk mencegah pengeditan oleh pengguna.

```
def update_contents(self):
    self.show_mapel_list()

def show_mapel_list(self):
    mapel = self.mapel_var.get()
    self.result_box.configure(state="normal")
    self.result_box.delete("1.0", "end")
    found = False
    for nisn, info in sorted(self.app.data.items()):
        nilai = info.get("nilai", {})
        if mapel in nilai:
            nama = info.get("nama", "-")
            kelas = info.get("kelas", "-")
            nilai_mapel = nilai[mapel]

            self.result_box.insert("end", f"{nisn} – {nama} – {kelas}\n")
            self.result_box.insert("end", f"Nilai {mapel}: {nilai_mapel}\n\n")
            found = True
    if not found:
        self.result_box.insert("end", "Belum ada data untuk mapel ini.")
    self.result_box.configure(state="disabled")
```

Gambar 23

def update_contents

- Metode update_contents berfungsi untuk memperbarui tampilan halaman dengan menampilkan data nilai berdasarkan mata pelajaran yang dipilih.
- def show_mapel_list
- Metode show_mapel_list digunakan untuk menampilkan daftar siswa yang memiliki nilai pada mata pelajaran tertentu.
- Sistem menampilkan informasi siswa berupa NISN, nama, kelas, dan nilai mata pelajaran.
- Jika belum terdapat data nilai untuk mata pelajaran yang dipilih, sistem menampilkan pesan pemberitahuan.
- Seluruh hasil ditampilkan dalam CTkTextbox yang bersifat non-editable.

```
# main program
if __name__ == "__main__":
    app = RaporApp()
    app.mainloop()
```

Gambar 24

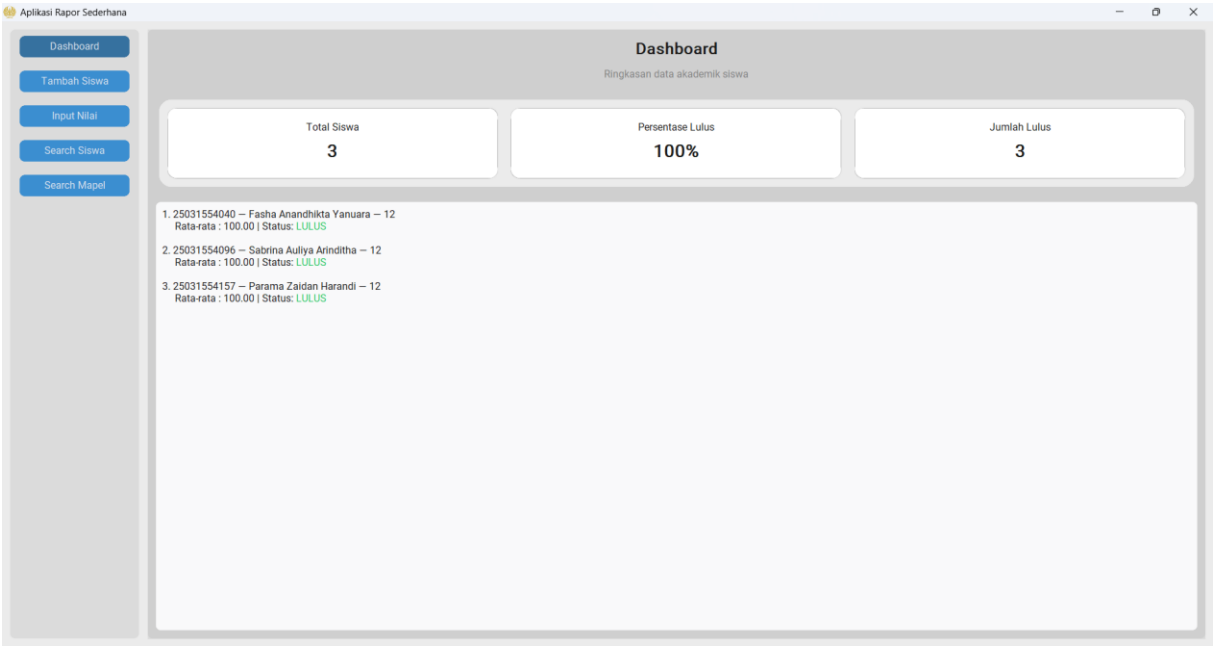
Main Program

- Bagian ini merupakan titik awal eksekusi aplikasi.
- Kondisi if __name__ == "__main__": digunakan untuk memastikan bahwa program dijalankan secara langsung, bukan sebagai modul impor.
- Objek RaporApp dibuat sebagai aplikasi utama.
- Metode mainloop() dijalankan untuk memulai dan mempertahankan antarmuka grafis aplikasi.

3.2 Screenshot Aplikasi

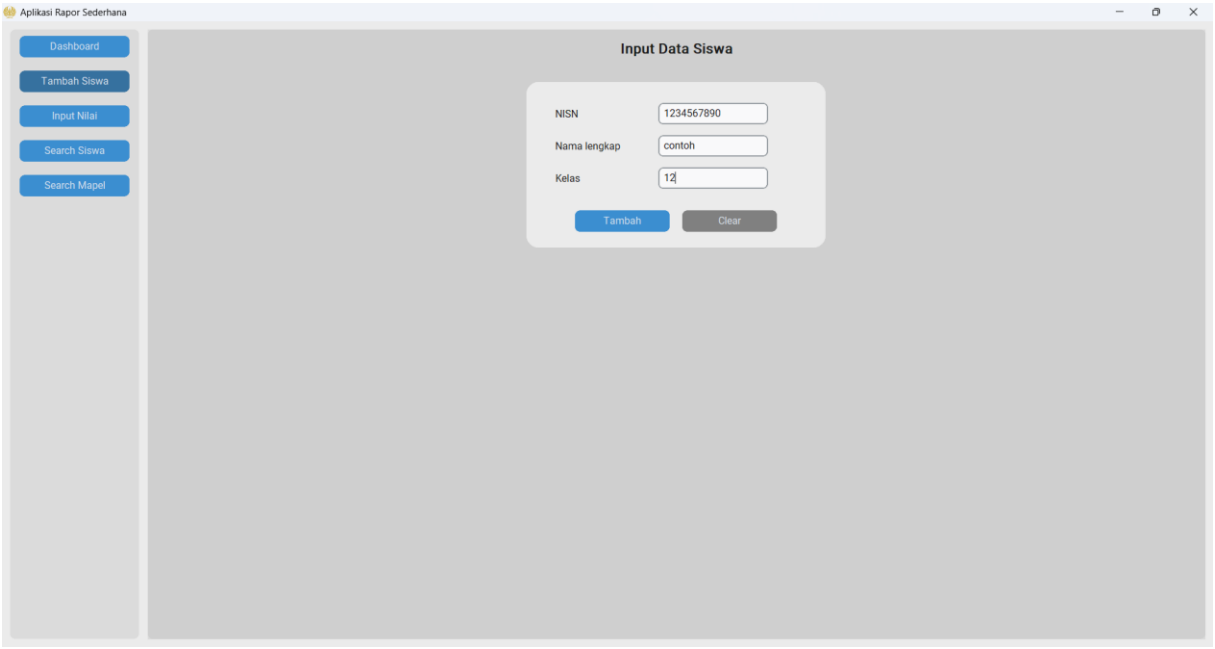
Untuk mendukung pemahaman terhadap implementasi program, diperlukan visualisasi berupa tampilan antarmuka aplikasi. Tangkapan layar pada setiap halaman aplikasi disertakan sebagai dokumentasi untuk menunjukkan hasil eksekusi kode program serta mempermudah interpretasi fungsi dan alur kerja aplikasi.

3.2.1 Tampilan Dashboard



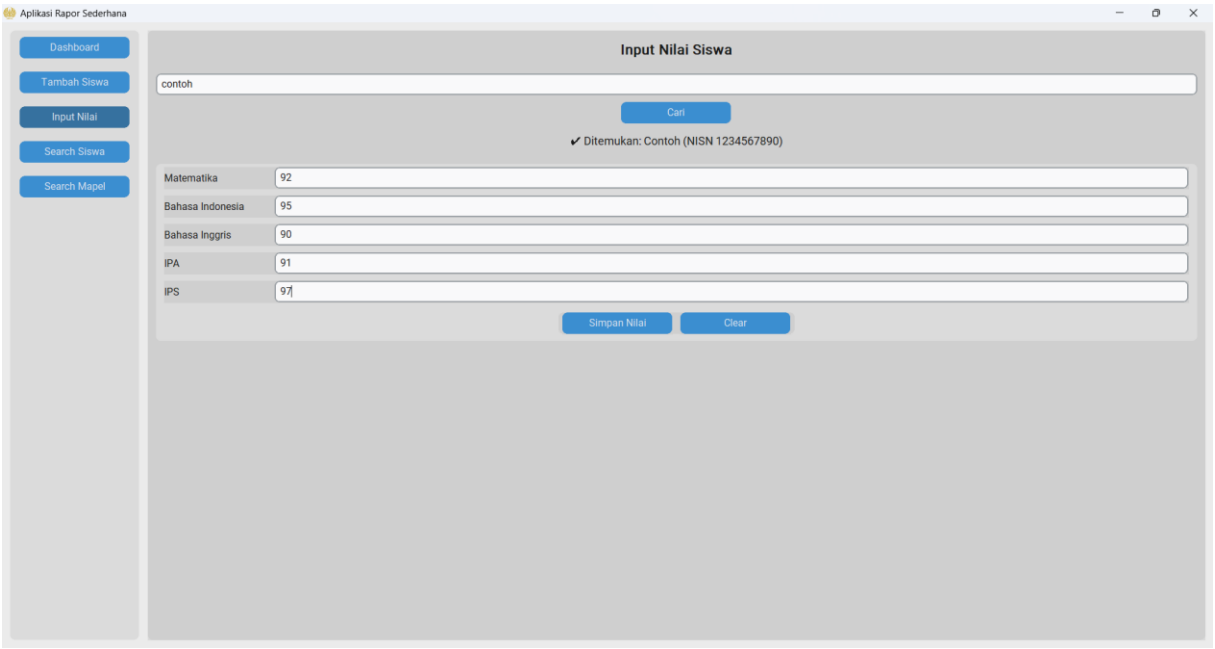
Gambar 25

3.2.2 Tampilan Input Data Siswa



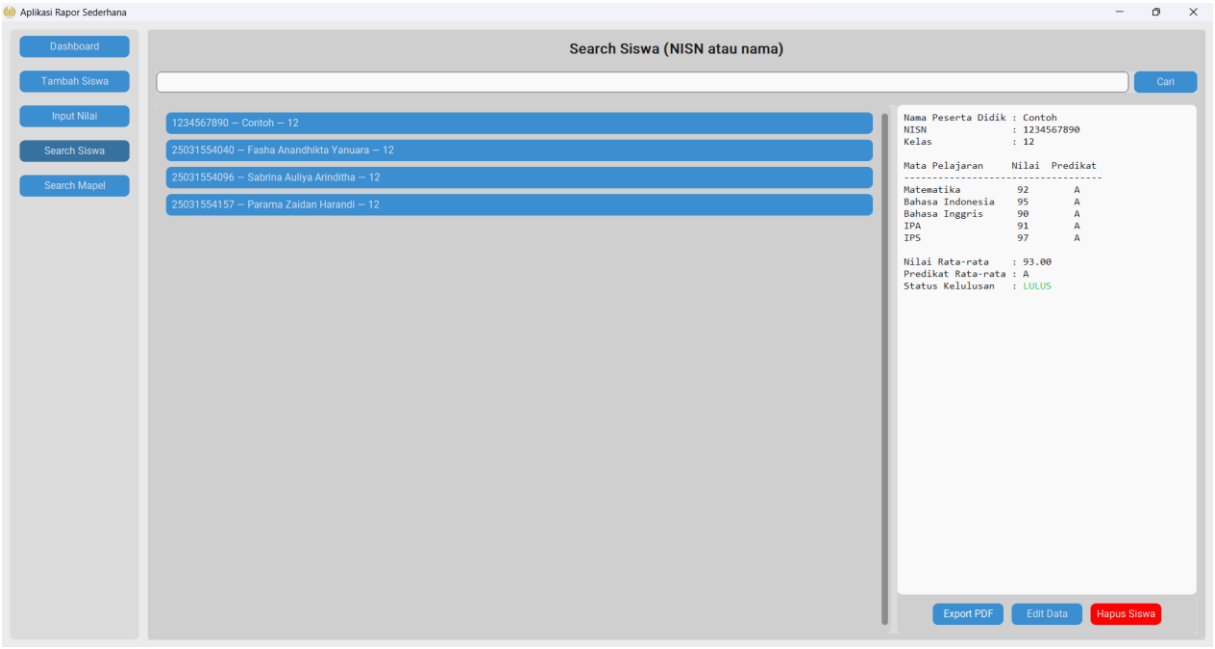
Gambar26

3.2.3 Tampilan Input Nilai Siswa



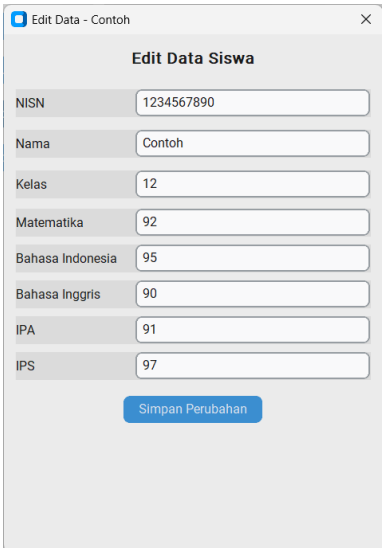
Gambar27

3.2.4 Tampilan Pencarian Data Siswa



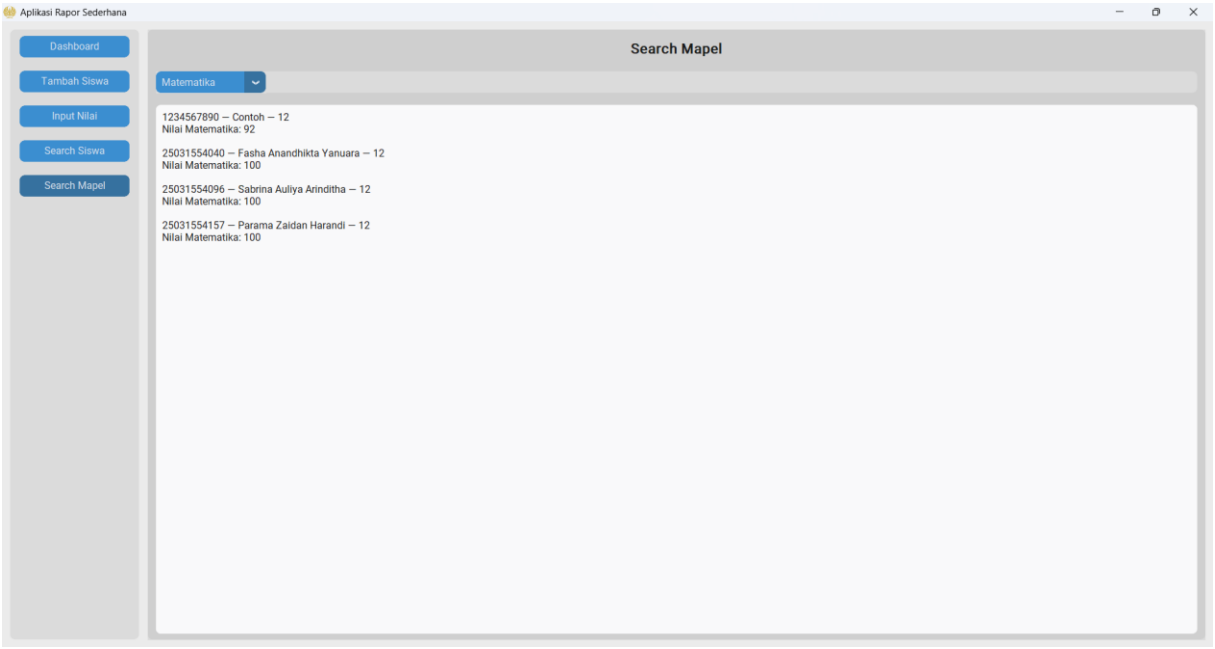
Gambar 28

3.2.5 Tampilan Mengedit Data Siswa



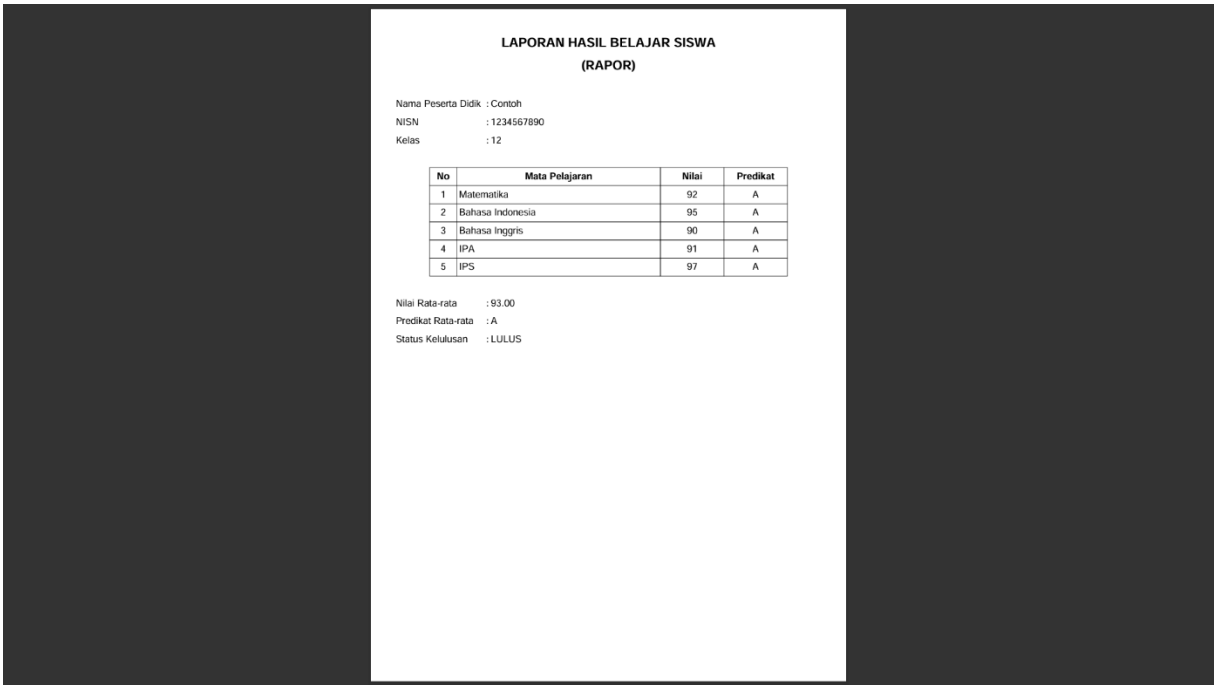
Gambar 29

3.2.6 Tampilan Pencarian Nilai Berdasarkan Mata Pelajaran



Gambar 30

3.2.7 Tampilan Ekspor Rapor ke PDF



LAPORAN HASIL BELAJAR SISWA
(RAPOR)

Nama Peserta Didik : Contoh
NISN : 1234567890
Kelas : 12

No	Mata Pelajaran	Nilai	Predikat
1	Matematika	92	A
2	Bahasa Indonesia	95	A
3	Bahasa Inggris	90	A
4	IPA	91	A
5	IPS	97	A

Nilai Rata-rata : 93.00
Predikat Rata-rata : A
Status Kelulusan : LULUS

Gambar31

BAB 4
LAMPIRAN

Folder project terlampir pada link google drive berikut:
https://drive.google.com/drive/folders/1f2mMcgvPxsWCbW5TGrxCIyh7UVVZPSKm?usp=drive_link

DAFTAR PUSTAKA

Irsan, M., Santos, F. T., & Husain, A. (2024, Juni 28). Implementasi Aplikasi Pandas (Phyton) Dalam Mengelola Data Excel Sebagai Media Persiapan Pelaporan Nilai Raport Siswa. *Jurnal Pengabdian Masyarakat Bangsa*.

Mulyana, D. I., Rowis, M. A., Iskandar, D., & Jaya, A. S. (2023). Implementasi E-Raport Berbasis Chatbot Studi Kasus di SMA Dipoenogoro 2 Jakarta. *Jurnal Pengambdian Nasional Indonesia*.

S, D., N, D., V, S., K, G., & K, R. (2023). Student Management System Using Tkinter. *ijsart*.

Student Results Management System Using Tkinter. (2025, July 23). Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/python/student-results-management-system-using-tkinter