Laporan Praktikum
Pemrograman Berorientasi Objek

Disusun Oleh:

Nama: Rafi Naufal Syah

NPM: 5230411241

Kelas: E

Sarjana Informatika

Universitas Teknologi Yogyakarta

1. Pendahuluan

Laporan praktikum ini adalah untuk mengimplementasikan aplikasi manajemen data setor hafalan Quran dengan menggunakan GUI (Graphical User Interface) menggunakan Bahasa pemrograman Python dan Library Tkinter.

2. Deskripsi Program

Program setoran hafalan Quran ini dibuat dengan menggunakan Bahasa pemrograman Python. Pada program ini terdapat berbagai fitur seperti:

a. Penambahan Data: Pengguna dapat memasukkan data setoran, seperti: nama, surat, ayat, status (lulus atau tidak lulus).
b. Pencarian Nama: Pengguna dapat menampilkan data yang cocok dengan nama yang dimasukkan di kolom pencarian.
c. Penghapusan Data: Pengguna dapat menghapus data berdasarkan nama yang sudah dicari.
d. Refresh Data: Setelah pengguna menggunakan fitur Pencarian Nama, pengguna dapat menampilkan kembali seluruh data dan membersihkan kolom pencarian.

3. Kode Program

```
4. import tkinter as tk
5. from tkinter import ttk, messagebox
6. 
7. # Simpan data hafalan dalam list
8. data_hafalan = []
9. 
10.# GUI Utama
11.class QuranApp:
12.    def __init__(self, root):
13.        self.root = root
14.        self.root.title("Setor Hafalan Quran")
15.        self.root.geometry("700x500")
16.
17.        # Frame Input
18.        self.frame_input = tk.Frame(self.root)
19.        self.frame_input.pack(pady=10)
20.
21.        tk.Label(self.frame_input, text="Nama:").grid(row=0, column=0,
   padx=5, pady=5)
22.        self.entry_nama = tk.Entry(self.frame_input)
23.        self.entry_nama.grid(row=0, column=1, padx=5, pady=5)
24.
```

```
25.         tk.Label(self.frame_input, text="Surat:").grid(row=1, column=0,
    padx=5, pady=5)
26.         self.entry_surat = tk.Entry(self.frame_input)
27.         self.entry_surat.grid(row=1, column=1, padx=5, pady=5)
28.
29.         tk.Label(self.frame_input, text="Ayat:").grid(row=2, column=0,
    padx=5, pady=5)
30.         self.entry_ayat = tk.Entry(self.frame_input)
31.         self.entry_ayat.grid(row=2, column=1, padx=5, pady=5)
32.
33.         tk.Label(self.frame_input, text="Status:").grid(row=3, column=0,
    padx=5, pady=5)
34.         self.combobox_status = ttk.Combobox(self.frame_input,
    values=["Lulus", "Belum Lulus"], state="readonly")
35.         self.combobox_status.grid(row=3, column=1, padx=5, pady=5)
36.         self.combobox_status.current(0)
37.
38.         tk.Button(self.frame_input, text="Tambah",
    command=self.tambah_data).grid(row=4, column=0, columnspan=2, pady=10)
39.
40.         # Frame Pencarian dan Hapus
41.         self.frame_cari_hapus = tk.Frame(self.root)
42.         self.frame_cari_hapus.pack(pady=10)
43.
44.         tk.Label(self.frame_cari_hapus, text="Cari Nama:").grid(row=0,
    column=0, padx=5, pady=5)
45.         self.entry_cari_hapus = tk.Entry(self.frame_cari_hapus)
46.         self.entry_cari_hapus.grid(row=0, column=1, padx=5, pady=5)
47.
48.         tk.Button(self.frame_cari_hapus, text="Cari Data",
    command=self.cari_data).grid(row=0, column=2, padx=5, pady=5)
49.         tk.Button(self.frame_cari_hapus, text="Hapus Data",
    command=self.hapus_data).grid(row=0, column=3, padx=5, pady=5)
50.         tk.Button(self.frame_cari_hapus, text="Refresh",
    command=self.refresh_data).grid(row=0, column=4, padx=5, pady=5)
51.
52.         # Tabel
53.         self.tree = ttk.Treeview(self.root, columns=("ID", "Nama",
    "Surat", "Ayat", "Status"), show="headings")
54.         self.tree.heading("ID", text="ID")
55.         self.tree.heading("Nama", text="Nama")
56.         self.tree.heading("Surat", text="Surat")
57.         self.tree.heading("Ayat", text="Ayat")
58.         self.tree.heading("Status", text="Status")
59.         self.tree.column("ID", width=30)
```

```
60.         self.tree.pack(pady=10)
61.
62.         self.load_data()
63.
64.     # Tambah data ke list dan refresh tabel
65.     def tambah_data(self):
66.         nama = self.entry_nama.get()
67.         surat = self.entry_surat.get()
68.         ayat = self.entry_ayat.get()
69.         status = self.combobox_status.get()
70.
71.         if nama and surat and ayat:
72.             # Tambahkan data ke dalam list
73.             new_data = {
74.                 "id": len(data_hafalan) + 1,
75.                 "nama_penyetor": nama,
76.                 "surat": surat,
77.                 "ayat": ayat,
78.                 "status": status
79.             }
80.             data_hafalan.append(new_data)
81.             self.load_data()
82.             messagebox.showinfo("Berhasil", "Data berhasil ditambahkan!")
83.
84.             # Kosongkan input
85.             self.entry_nama.delete(0, tk.END)
86.             self.entry_surat.delete(0, tk.END)
87.             self.entry_ayat.delete(0, tk.END)
88.         else:
89.             messagebox.showerror("Error", "Semua kolom harus diisi!")
90.
91.     # Muat data dari list ke tabel
92.     def load_data(self, filtered_data=None):
93.         """Memuat data ke tabel, bisa seluruh data atau data yang
    disaring."""
94.         for row in self.tree.get_children():
95.             self.tree.delete(row)
96.
97.         data_to_display = filtered_data if filtered_data else data_hafalan
98.         for data in data_to_display:
99.             self.tree.insert("", tk.END, values=(data["id"],
    data["nama_penyetor"], data["surat"], data["ayat"], data["status"]))
100.
101.          # Cari data berdasarkan nama
102.          def cari_data(self):
```

```
103.             nama_cari = self.entry_cari_hapus.get()
104.             if not nama_cari:
105.                 messagebox.showerror("Error", "Masukkan nama yang ingin
    dicari!")
106.                 return
107.
108.             # Filter data berdasarkan nama
109.             filtered_data = [data for data in data_hafalan if
    nama_cari.lower() in data["nama_penyetor"].lower()]
110.             if filtered_data:
111.                 self.load_data(filtered_data)
112.             else:
113.                 self.load_data()
114.                 messagebox.showinfo("Hasil Pencarian", f"Tidak ditemukan
    data dengan nama '{nama_cari}'!")
115.
116.         # Hapus data berdasarkan nama
117.         def hapus_data(self):
118.             nama_hapus = self.entry_cari_hapus.get()
119.             if not nama_hapus:
120.                 messagebox.showerror("Error", "Masukkan nama yang ingin
    dihapus!")
121.                 return
122.
123.             # Hapus data berdasarkan nama
124.             global data_hafalan
125.             data_hafalan = [data for data in data_hafalan if
    nama_hapus.lower() not in data["nama_penyetor"].lower()]
126.
127.             # Perbarui ID setelah data dihapus
128.             for idx, data in enumerate(data_hafalan):
129.                 data["id"] = idx + 1
130.
131.             self.load_data()
132.             messagebox.showinfo("Berhasil", f"Semua data dengan nama
    '{nama_hapus}' berhasil dihapus!")
133.             self.entry_cari_hapus.delete(0, tk.END)
134.
135.         # Refresh data untuk menampilkan semua data
136.         def refresh_data(self):
137.             self.load_data()
138.             self.entry_cari_hapus.delete(0, tk.END)
139.
140.     # Main
141.     if __name__ == "__main__":
```

```
142.          root = tk.Tk()
143.          app = QuranApp(root)
144.          root.mainloop()
145.
```

4. Penjelasan

1. __init__()

Fungsi ini adalah kontruktor yang dipanggil saat objek “QuranApp” dibuat. Di dalamnya terdapat berbagai elemen GUI seperti frame input, table, tombol-tombol disusun.

2. Tambah_data()

Fungsi ini digunakan untuk mengambil input dari kolom Nama, Surat, Ayat, dan Status. Jika semua kolom diisi, program akan membuat dictionary baru yang berisi data tersebut dan menambahkannya ke dalam list “data_hafalan”.

3. load_data()

Fungsi ini menerima parameter “filtered_data” yang merupakan data yang sudah difilter (misalnya, hasil pencarian). Jika tidak ada parameter yang diberikan, maka fungsi ini menampilkan seluruh data yang ada dalam “data_hafalan”.

4. cari_data()

Fungsi ini mengambil nama yang dimasukkan pada input pencarian “entry_cari_hapus” dan mencari semua data yang mengandung nama tersebut.

5. hapus_data()

Fungsi ini mengambil nama yang dimasukkan pada input pencarian dan menghapus semua data dalam “data_hafalan” yang nama penyetornya sesuai dengan nama yang dicari.

6. refresh_data()

Fungsi ini memanggil “load_data()” tanpa parameter untuk menampilkan semua data yang ada dalam “data_hafalan”.

7. If __name__ == “__main__”:

Bagian ini memastikan bahwa aplikasi hanya berjalan jika file ini dijalankan langsung, bukan saat diimpor sebagai modul.

5. Penutup

Demikian laporan praktikum ini saya buat untuk memberikan gambaran tentang penerapan konsep-konsep dalam pemrograman Python, khususnya dalam pembuatan aplikasi manajemen data hafalan Quran menggunakan GUI dengan Tkinter. Dalam praktikum ini, saya

telah berhasil mengimplementasikan berbagai fitur seperti penambahan, pencarian, penghapusan, dan penyegaran data dengan antarmuka pengguna yang interaktif.