PROGRAM DATA KELAS COURSE CODING

Oleh

Anggota Kelompok

10123449-Ahmad Riski

10123465-Attala Arrafi

10123461-Dzulfikar Sadid

10123216-Naufal Azka

10123457-Rizki Eskhart

Untuk memenuhi salah satu tugas mata kuliah

Algoritma dan Struktur Data 1

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK DAN ILMU KOMPUTER
UNIVERSITAS KOMPUTER INDONESIA
BANDUNG
2024

DAFTAR ISI

DAFTAR GAMBAR

DESKRIPSI PROGRAM

1. Deskripsi Program Algoritma: Program Data Kelas Course Coding

Program yang kami buat merupakan program yang terinspirasi dari platform *course online* dimana user dapat mengikuti kelas-kelas yang tersedia, terutama kelas pemrograman.

Tujuan:

Program ini bertujuan untuk mengelola data kelas course coding, termasuk:

- Data Siswa: Menyimpan informasi tentang siswa yang mengikuti course coding, seperti nama, email, dan nomor telepon.
- Data Kelas: Menyimpan informasi tentang kelas coding yang ditawarkan, seperti nama kursus, deskripsi, dan mentor.
- Data Siswa Kelas: Menyimpan informasi tentang siswa kelas coding yang telah mendaftar kelas sebelumnya.

2. Algoritma:

Program ini akan menggunakan berbagai algoritma untuk mengelola data kelas course coding, termasuk:

- Pencarian: Algoritma pencarian akan digunakan untuk menemukan data siswa, kursus, dan kelas tertentu.
- Pengurutan: Algoritma pengurutan akan digunakan untuk mengurutkan data siswa, kursus, dan kelas, berdasarkan huruf.
- Penyisipan: Algoritma penyisipan akan digunakan untuk menambahkan data siswa, kursus, dan kelas baru ke array.
- Pembaruan: Algoritma pembaruan akan digunakan untuk memperbarui data siswa, kursus, dan kelas jika ada data yang baru masuk.

3. **Struktur Data:**

Program ini akan menggunakan satu struktur data untuk menyimpan data kelas course coding, yaitu:

- Array: Array akan digunakan untuk menyimpan data siswa, kursus, mentor,dan kelas.

4. **Bahasa Pemrograman:**

Program ini kami diimplementasikan dalam 2 bahasa pemrograman, yaitu Python dan Algoritma Pseudo-code.

5. **Manfaat:**

Program ini akan memberikan banyak manfaat, termasuk:

- Meningkatkan efisiensi pengelolaan data kelas course coding.
- Mempermudah akses dan pencarian data.
- Meningkatkan akurasi data.
- Mempermudah pelacakan perkembangan jumlah siswa.

6. **Kesimpulan:**

Program data kelas course coding adalah program yang penting untuk mengelola data kelas course coding secara efektif dan efisien. Program ini akan memberikan banyak manfaat bagi penyelenggara course coding dan siswa.

MENU PROGRAM

1. Program Course Coding

Program Course Coding menyediakan berbagai menu untuk membantu Anda belajar coding dengan mudah dan menyenangkan. Berikut adalah daftar menu beserta penjelasannya:

1. **Daftar Kelas**
   - Menampilkan daftar course coding yang tersedia, beserta deskripsinya.
   - Anda dapat memilih course yang sesuai dengan minat.

2. **Daftar User**

   Menampilkan informasi detail tentang user kelas, seperti:
   - Nama User
   - Email User
   - Nomor Hp User

3. **Daftar User Kelas**
   - Menampilkan informasi daftar user yang terdaftar di setiap kelas

4. **Pencarian**
   - Mencari data nama kelas dan nama user yang tersedia

5. **Pengurutan**
   - Mengurutkan data kelas dan nama sesuai abjad dari A-Z
   - Mengurutkan data kelas dan nama sesuai abjad dari Z-A

ALGORITMA

```
Algoritma Bootcamp Online
{I.S: User memilih beberapa menu yang ditampilkan oleh sistem}
{F.S: Menampilkan konten beberapa menu yang ditampilkan oleh sistem}

Kamus (Global):
kelas : array [1..100, 1..3] of string
user: array [1..100, 1..3] of string
user_kelas: array[1..100, 1..3] of string
is_ascending: boolean

Procedure bubbleSortAscending(lst: array[1..100 , 1..3] of string):
    Kamus:
    n, I ,j: integer
    temp: string

    Algoritma:
    Is_ascending <- true
    n <- lst.length
    for i <- 1 to n do
        for j <- 1 to n - i - 1 do
            if lst[j] > lst[j + 1] then
                temp <- lst[j]
                lst[j] <- lst[j + 1]
                lst[j + 1] <- temp
            endif
        endfor
    endfor
```

```
EndProcedure

Procedure bubbleSortDescending(lst: array[1..100 , 1..3] of string):
    Kamus:
    n, I ,j: integer
    temp: string

    Algoritma:
    Is_ascending <- true
    n <- lst.length
    for i <- 1 to n do
        for j <- 1 to n - i - 1 do
            if lst[j] < lst[j + 1] then
                temp <- lst[j]
                lst[j] <- lst[j + 1]
                lst[j + 1] <- temp
            endif
        endfor
    endfor
EndProcedure


Procedure sortAndRefreshKelas(ascending):
    Algoritma:
    if is_ascending then
        bubbleSortAscending(kelas)
    else :
        bubbleSortDescending(kelas)
    endif
    refreshKelas()
EndProcedure
```

```
Procedure sortAndRefreshUser(ascending):
    Algoritma:
    if is_ascending then
        bubbleSortAscending(user)
    else :
        bubbleSortDescending(user)
    endif
    refreshUser()
EndProcedure

Function binarySearchAscending(lst: array[1..100, 1..3] of string,
target:string)->int
    Kamus:
    left,right,mid : integer
    Algoritma:
    left <- 0
    right <- lst.length
    while left <= right do
        mid <- (left + right) div 2
        if lst[mid][0] < target then
            left <- mid + 1
        else if target < lst[mid][0] then
            right <-  mid - 1
        else
            return mid
        endif
    endwhile
    return -1
EndFunction
```

```
Function binarySearchDescending(lst: array[1..100, 1..3] of string,
target:string)->int
    Kamus:
    left,right,mid : integer
    Algoritma:
    left <- 0
    right <- lst.length
    while left <= right do
        mid <- (left + right) div 2
        if lst[mid][0] > target then
            left <- mid + 1
        else if target > lst[mid][0] then
            right <-  mid - 1
        else
            return mid
        endif
    endwhile
    return -1
EndFunction

Procedure searchKelas(query:string)
    Kamus:
    result : integer
    Algoritma:
    if is_acending then
        bubbleSortAscending (kelas)
        result <- binarySearchAscending(kelas, query)
    else
        bubbleSortDescending (kelas)
```

```
        result <- binarySearchDescending(kelas, query)
    endif
    if result >= 0 then
        output("Found, Kelas ditemukan pada posisi " + result)
    else
        output("Not Found, Kelas tidak ditemukan")
    endif
EndProcedure

Procedure searchUser(query:string)
    Kamus:
    result : Integer
    Algoritma:
    if is_acending then
        bubbleSortAscending(user)
        result <- binarySearchAscending(user, query)
    else
        bubbleSortDescending(user)
        result <- binarySearchDescending(user, query)
    endif
    if result >= 0 then
        output("Found, User ditemukan pada posisi " + result)
    else
        output("Not Found, User tidak ditemukan")
    endif
EndProcedure

Procedure refreshUser()
    Algoritma:
    output(user)
```

```
EndProcedure
Procedure refreshKelas()
    Algoritma:
    output(kelas)
EndProcedure

Procedure refreshUserKelas()
    Algoritma:
    output(user_kelas)
EndProcedure

Procedure showJoinClassForm(kelas : string)
    Kamus:
    nama, email, phone : string
    Algoritma:
    Input(nama)
    Input(email)
    Input(phone)
    user[user.length+1] <- [nama, email, kelas]
    user_kelas[user_kelas.length+1] <- [nama, kelas]
EndProcedure


Algoritma:
kelas[1] <- ["Data Science", "Jane Smith", 3]
kelas[2] <- ["Web Development", "Alex Johnson", 5]
kelas[3] <- ["Machine Learning", "Emily Brown", 2]
kelas[4] <- ["Android Development", "Michael Wilson", 3]
kelas[5] <- ["IOS Development", "Sarah Thompson", 4]
kelas[6] <- ["UI/UX Design", "David Miller", 3]
kelas[7] <- ["Backend Development", "Jessica Davis", 5]
```

```
kelas[8] <- ["React Native", "John Doe", 4]

user[1] <- ['Dzulfikar Sadid', 'dzulfikar@mail.com','08123456789']
user[2] <- ['Ahmad Riski', 'ahmad@mail.com', '08123456789']
user[3] <- ['Naufal Azka', 'naufal@mail.com', '08123456789']

user_kelas[1] <- ['Dzulfikar Sadid', 'Data Science']
user_kelas[2] <- ['Ahmad Riski', 'Web Development']
user_kelas[3] <- ['Naufal Azka', 'Machine Learning']
user_kelas[4] <- ['Dzulfikar Sadid', 'React Native']
user_kelas[5] <- ['Ahmad Riski', 'Backend Development']
user_kelas[6] <- ['Naufal Azka', 'Backend Development']

sortAndRefreshKelas(true)
sortAndRefreshUser(true)

output("Selamat Datang di Bootcamp Online")
output("Silahkan pilih menu yang tersedia")
output("1. Lihat Kelas")
output("2. Lihat User")
output("3. Cari Kelas")
output("4. Cari User")
output("5. Keluar")
input(menu)
if menu == 1 then
show_kelas()
elseif menu == 2 then
show_user()
elseif menu == 3 then
input(query)
```

```
search_kelas(query)
elseif menu == 4 then
input(query)
search_user(query)
else
output("Terima kasih telah menggunakan Bootcamp Online")
endif
```

PYTHON

```
import tkinter as tk
from tkinter import ttk, messagebox, Toplevel


# List kelas yang tersedia
# Nama kelas, nama mentor, durasi (bulan)
kelas = [
    ["Data Science", "Jane Smith", 3],
    ["Web Development", "Alex Johnson", 5],
    ["Machine Learning", "Emily Brown", 2],
    ["Android Development", "Michael Wilson", 3],
    ["IOS Development", "Sarah Thompson", 4],
    ["UI/UX Design", "David Miller", 3],
    ["Backend Development", "Jessica Davis", 5],
    ["React Native", "John Doe", 4],
]


# List user yang terdaftar
# Nama, email, nomor telepon
user = [
    ['Dzulfikar Sadid', 'dzulfikar@mail.com',
     '08123456789'],
    ['Ahmad Riski', 'ahmad@mail.com', '08123456789'],
    ['Naufal Azka', 'naufal@mail.com', '08123456789'],
]
```

```
# List user yang terdaftar di kelas
# Nama, kelas
user_kelas = [
    ['Dzulfikar Sadid', 'Data Science'],
    ['Ahmad Riski', 'Web Development'],
    ['Naufal Azka', 'Machine Learning'],
    ['Dzulfikar Sadid', 'React Native'],
    ['Ahmad Riski', 'Backend Development'],
    ['Naufal Azka', 'Backend Development'],
]

# List terurut ascending atau tidak
is_ascending = True

# Inisialisai GUI
root = tk.Tk()
root.title("Bootcamp Online")
root.geometry("1280x720")
notebook = ttk.Notebook(root)
notebook.pack(fill='both', expand=True)
kelas_page = ttk.Frame(notebook)
user_page = ttk.Frame(notebook)
user_kelas_page = ttk.Frame(notebook)


# Sorting
def bubble_sort_ascending(lst):
    global is_ascending
```

```
    is_ascending = True
    n = len(lst)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if lst[j] > lst[j + 1]:
                lst[j], lst[j + 1] = lst[j + 1], lst[j]



def bubble_sort_descending(lst):
    global is_ascending
    is_ascending = False
    n = len(lst)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if lst[j] < lst[j + 1]:
                lst[j], lst[j + 1] = lst[j + 1], lst[j]



def sort_and_refresh_kelas(ascending):
    if ascending:
        bubble_sort_ascending(kelas)
    else:
        bubble_sort_descending(kelas)
    refresh_kelas()



def sort_and_refresh_user(ascending):
    if ascending:
```

```
        bubble_sort_ascending(user)
    else:
        bubble_sort_descending(user)
    refresh_user()



# Searching
def binary_search_ascending(lst, target):
    left = 0
    right = len(lst) - 1
    while left <= right:
        mid = (left + right) // 2
        if lst[mid][0] < target:
            left = mid + 1
        elif target < lst[mid][0]:
            right = mid - 1
        else:
            return mid
    return -1



def binary_search_descending(lst, target):
    left = 0
    right = len(lst) - 1
    while left <= right:
        mid = (left + right) // 2
        if lst[mid][0] > target:
            left = mid + 1
```

```
        elif target > lst[mid][0]:
            right = mid - 1
        else:
            return mid
    return -1



def search_kelas(query):
    global is_ascending
    if is_ascending:
        bubble_sort_ascending(kelas)
        result = binary_search_ascending(kelas, query)
    else:
        bubble_sort_descending(kelas)
        result = binary_search_descending(kelas, query)
    if result >= 0:
        show_kelas_search(result + 1, kelas[result])
    else:
        messagebox.showinfo("Not Found",
                            "Kelas tidak ditemukan")



def search_user(query):
    global is_ascending
    if is_ascending:
        bubble_sort_ascending(user)
        result = binary_search_ascending(user, query)
    else:
```

```
        bubble_sort_descending(user)
        result = binary_search_descending(user, query)
    if result >= 0:
        show_user_search(result + 1, user[result])
    else:
        messagebox.showinfo("Not Found",
                            "User tidak ditemukan")



# Tampilan Halaman Kelas
def refresh_kelas():
    for widget in kelas_page.winfo_children():
        widget.destroy()

    label = tk.Label(kelas_page, text="Daftar Kelas",
                     font=("Arial", 18))
    label.pack(pady=10)

    search_frame = tk.Frame(kelas_page)
    search_frame.pack(pady=5)

    search_label = tk.Label(search_frame, text="Search:",
                            font=("Arial", 14))
    search_label.pack(side="left")

    search_entry = tk.Entry(search_frame,
                            font=("Arial", 14))
    search_entry.pack(side="left", padx=5)
```

```
search_button = tk.Button(search_frame, text="Search",
                          font=("Arial", 14),
                          relief="groove", padx=5,
                          pady=2, width=10,
                          borderwidth=2, bd=1,
                          bg="gray", fg="white",
                          highlightthickness=0,
                          highlightcolor="white",
                          command=lambda: search_kelas(
                              search_entry.get()))
search_button.pack(side="left", padx=5)


for i in range(len(kelas)):
    kelas_frame = tk.Frame(kelas_page)
    kelas_frame.pack()


    kelas_button = tk.Button(kelas_frame,
                             text="{}. {}".format(i + 1,
                                                  kelas[
                                                      i][
                                                      0]),
                             anchor="w",
                             font=("Arial", 14),
                             relief="groove", padx=10,
                             pady=5, width=20,
                             borderwidth=4, bd=1,
                             bg="white", fg="black",
```

```
                                highlightthickness=0,
                                highlightcolor="white",
                                command=lambda
                                    kelas_detail=kelas[
                                        i]: show_kelas_detail(
                                    kelas_detail))
    kelas_button.pack(side="left", pady=5)


    join_button = tk.Button(kelas_frame,
                            text="Join Class",
                            font=("Arial", 16),
                            relief="groove", padx=5,
                            pady=2, width=10,
                            borderwidth=2, bd=1,
                            bg="blue", fg="white",
                            highlightthickness=0,
                            highlightcolor="white",
                            command=lambda: show_join_class_form(
                                kelas[i]))
    join_button.pack(side="left", pady=5)


sort_asc_button = tk.Button(kelas_page,
                            text="Sort Ascending",
                            font=("Arial", 18),
                            relief="groove", padx=5,
                            pady=2, width=15,
                            borderwidth=2, bd=1,
                            bg="green", fg="white",
```

```
                                highlightthickness=0,
                                highlightcolor="white",
                                command=lambda: sort_and_refresh_kelas(
                                    True))
    sort_asc_button.pack(pady=5)

    sort_desc_button = tk.Button(kelas_page,
                                 text="Sort Descending",
                                 font=("Arial", 18),
                                 relief="groove", padx=5,
                                 pady=2, width=15,
                                 borderwidth=2, bd=1,
                                 bg="red", fg="white",
                                 highlightthickness=0,
                                 highlightcolor="white",
                                 command=lambda: sort_and_refresh_kelas(
                                     False))
    sort_desc_button.pack(pady=5)



# Tampilan Halaman User
def refresh_user():
    for widget in user_page.winfo_children():
        widget.destroy()
    label = tk.Label(user_page, text="Daftar User",
                     font=("Arial", 24))
    label.pack(pady=10)
```

```
    search_frame = tk.Frame(user_page)
    search_frame.pack(pady=5)


    search_label = tk.Label(search_frame, text="Search:",
                            font=("Arial", 14))
    search_label.pack(side="left")


    search_entry = tk.Entry(search_frame,
                            font=("Arial", 14))
    search_entry.pack(side="left", padx=5)


    search_button = tk.Button(search_frame, text="Search",
                              font=("Arial", 14),
                              relief="groove", padx=5,
                              pady=2, width=10,
                              borderwidth=2, bd=1,
                              bg="gray", fg="white",
                              highlightthickness=0,
                              highlightcolor="white",
                              command=lambda: search_user(
                                  search_entry.get()))


    search_button.pack(side="left", padx=5)
    user_table = ttk.Treeview(user_page)
    user_table["columns"] = ("Nama", "Email", "Nomor")
    user_table.column("#0", width=50)
    user_table.column("Nama", width=150)
    user_table.column("Email", width=200)
```

```
user_table.column("Nomor", width=150)
user_table.heading("#0", text="No.")
user_table.heading("Nama", text="Nama")
user_table.heading("Email", text="Email")
user_table.heading("Nomor", text="Nomor")
for i, user_detail in enumerate(user):
    user_table.insert(parent="", index="end", iid=i,
                      text=i + 1, values=(
        user_detail[0], user_detail[1], user_detail[2]))
user_table.pack(pady=10)


sort_asc_button = tk.Button(user_page,
                            text="Sort Ascending",
                            font=("Arial", 18),
                            relief="groove", padx=5,
                            pady=2, width=15,
                            borderwidth=2, bd=1,
                            bg="green", fg="white",
                            highlightthickness=0,
                            highlightcolor="white",
                            command=lambda: sort_and_refresh_user(
                                True))
sort_asc_button.pack(pady=5)


sort_desc_button = tk.Button(user_page,
                             text="Sort Descending",
                             font=("Arial", 18),
                             relief="groove", padx=5,
```

```
                                pady=2, width=15,
                                borderwidth=2, bd=1,
                                bg="red", fg="white",
                                highlightthickness=0,
                                highlightcolor="white",
                                command=lambda: sort_and_refresh_user(
                                    False))
    sort_desc_button.pack(pady=5)



# Tampilan Halaman User Kelas
def refresh_user_kelas():
    global kelas
    for widget in user_kelas_page.winfo_children():
        widget.destroy()

    label = tk.Label(user_kelas_page, text="Daftar Kelas",
                     font=("Arial", 18))
    label.pack(pady=10)

    for i in range(len(kelas)):
        kelas_frame = tk.Frame(user_kelas_page)
        kelas_frame.pack()

        kelas_button = tk.Button(kelas_frame,
                                 text="{}. {}".format(i + 1,
                                                      kelas[
                                                          i][
```

```
                                                0]),
                            anchor="w",
                            font=("Arial", 14),
                            relief="groove", padx=10,
                            pady=5, width=20,
                            borderwidth=4, bd=1,
                            bg="white", fg="black",
                            highlightthickness=0,
                            highlightcolor="white",
                            command=lambda
                                kelas_detail=kelas[
                                    i]: show_kelas_detail(
                                kelas_detail))
    kelas_button.pack(side="left", pady=5)


    join_button = tk.Button(kelas_frame,
                            text="Show Users",
                            font=("Arial", 16),
                            relief="groove", padx=5,
                            pady=2, width=10,
                            borderwidth=2, bd=1,
                            bg="yellow", fg="black",
                            highlightthickness=0,
                            highlightcolor="white",
                            command=lambda
                                kelas_new=kelas[i][
                                    0]: show_user_kelas(
                                kelas_new))
```

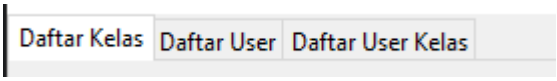
```
        join_button.pack(side="left", pady=5)



# Form untuk mengikuti kelas
def show_join_class_form(kelas):
    form_window = Toplevel(kelas_page)
    form_window.title(f"Join {kelas[0]}")

    # Create form labels
    name_label = tk.Label(form_window, text="Nama Anda:")
    name_label.grid(row=1, column=0, padx=10, pady=5,
                    sticky="w")
    name_entry = tk.Entry(form_window)
    name_entry.grid(row=1, column=1, padx=10, pady=5)

    email_label = tk.Label(form_window, text="Email Anda:")
    email_label.grid(row=2, column=0, padx=10, pady=5,
                     sticky="w")
    email_entry = tk.Entry(form_window)
    email_entry.grid(row=2, column=1, padx=10, pady=5)

    phone_label = tk.Label(form_window,
                           text="Nomor Telephone:")
    phone_label.grid(row=3, column=0, padx=10, pady=5,
                     sticky="w")
    phone_entry = tk.Entry(form_window)
    phone_entry.grid(row=3, column=1, padx=10, pady=5)
```

```
    submit_button = tk.Button(form_window, text="Submit",
                              command=lambda: submit_form(
                                  name_entry.get(),
                                  email_entry.get(),
                                  phone_entry.get(),
                                  kelas[0]))
    submit_button.grid(row=4, column=0, columnspan=2,
                       pady=10)



# Submit form
def submit_form(name, email, phone, kelas):
    user.append([name, email, phone])
    user_kelas.append([name, kelas])
    messagebox.showinfo("Berhasil",
                        "Selamat anda telah berhasil mendafar ke kelas ini!")
    sort_and_refresh_user(True)



# Details
def show_kelas_detail(kelas_detail):
    messagebox.showinfo("Kelas Detail",
                        f"Nama          Kelas:          {kelas_detail[0]}\nMentor:
{kelas_detail[1]}\nDurasi: {kelas_detail[2]} bulan")



def show_kelas_search(index, kelas_detail):
    messagebox.showinfo(
```

```
        f"Kelas Ditemukan di Posisi {index}",
        f"Nama   Kelas:   {kelas_detail[0]}\nMentor:   {kelas_detail[1]}\nDurasi:
{kelas_detail[2]} bulan")



def show_user_detail(user_detail):
    messagebox.showinfo("User Detail",
                        f"Nama: {user_detail[0]}\nEmail: {user_detail[1]}\nNomor:
{user_detail[2]}")



def show_user_search(index, user_detail):
    messagebox.showinfo(f"User Ditemukan di Posisi {index}",
                        f"Nama: {user_detail[0]}\nEmail: {user_detail[1]}\nNomor:
{user_detail[2]}")



def show_user_kelas(kelas):
    filtered_users = []
    for user in user_kelas:
        if user[1] == kelas:
            filtered_users.append(user[0])


    user_list = "\n".join(
        ["{}. {}".format(i + 1, user) for i, user in
         enumerate(filtered_users)])
    if len(filtered_users) == 0:
        messagebox.showinfo("Filtered Users",
                            "Tidak ada user yang terdaftar di kelas ini")
```

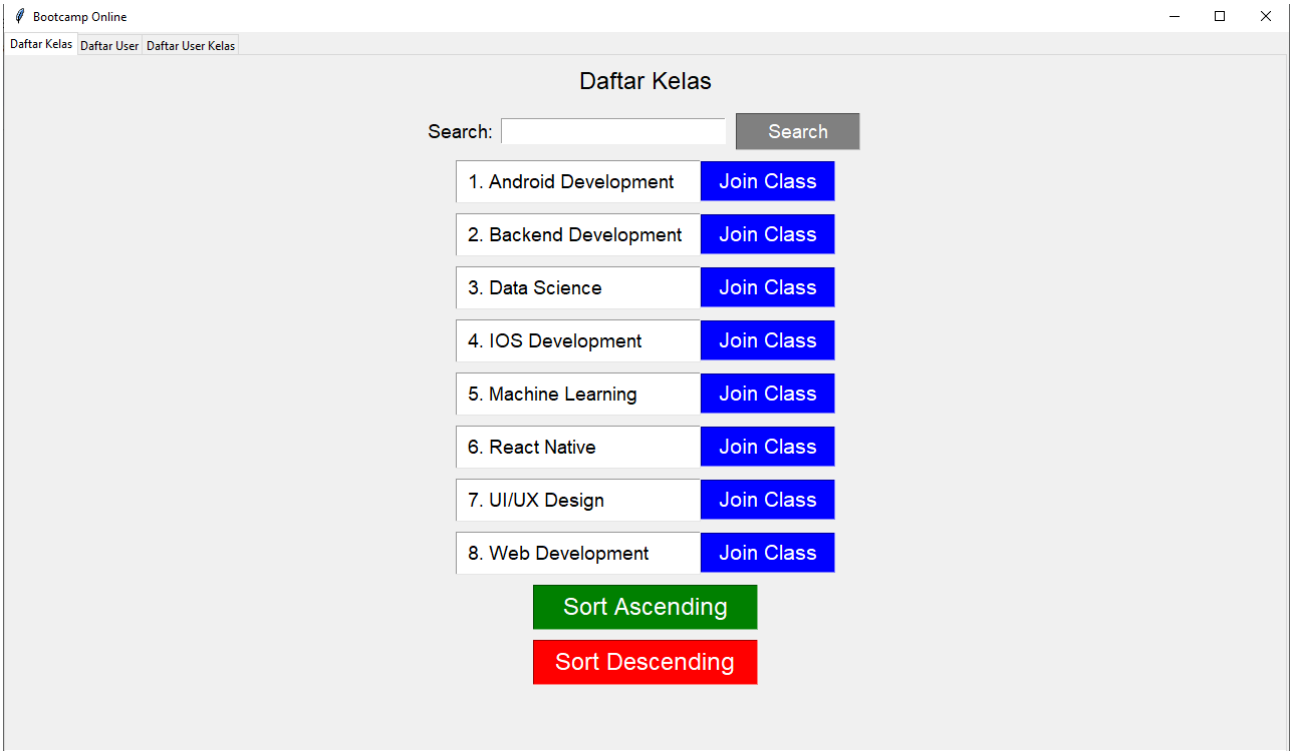
```
    else:
        messagebox.showinfo("Filtered Users", user_list)



# Fungsi tambahan untuk menampilkan halaman
def show_page(notebook, page):
    notebook.select(page)



def tampilkan_menu_gui():
    notebook.add(kelas_page, text='Daftar Kelas')
    notebook.add(user_page, text='Daftar User')
    notebook.add(user_kelas_page, text='Daftar User Kelas')
    refresh_kelas()
    refresh_user()
    refresh_user_kelas()
    root.mainloop()



# Memanggil fungsi / Bagian Utama Program
sort_and_refresh_kelas(True)
sort_and_refresh_user(True)
tampilkan_menu_gui()
```

SCREENSHOT PROGRAM



Gambar1. 1
Tampilan Bar Menu Pertama

Gambar menu yang muncul pertama saat program dijalankan, Ketika ditekan menu Daftar Kelas maka akan muncul **Gambar 1.2.**

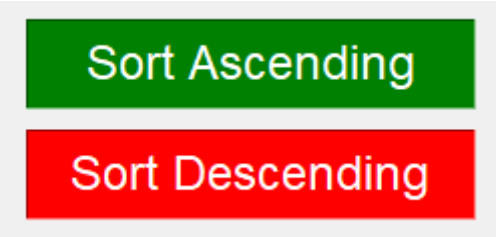


Gambar1. 2
Tampilan Isi Menu Pertama

Tampilan menu awal saat program dijalankan, terlihat ada menu Join Class untuk memasukkan data ke kelas.

Gambar1. 3
Tampilan Kolom Pencarian

Menu pencarian untuk memudahkan user dalam mencari data kelas di tampilan awal, user dapat memasukkan data nama kelas untuk menemukan kelas yang diinginkan.

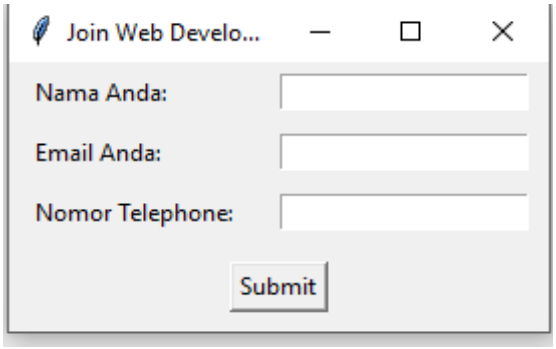


Gambar1. 4
Tampilan Tombol Pengurutan

Menu pengurutan untuk memudahkan user dalam mengurutkan data nama kelas, baik dari A-Z maupun dari Z-A.

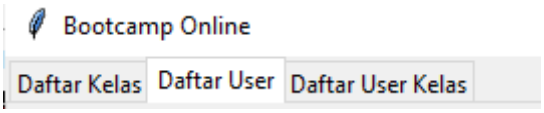
Gambar1. 5
Tampilan Data Kelas dan Tombol Join Class

Tampilan daftar nama kelas, jika user menekan tombol Join Class maka akan muncul tampilan seperti **Gambar 1.6.**



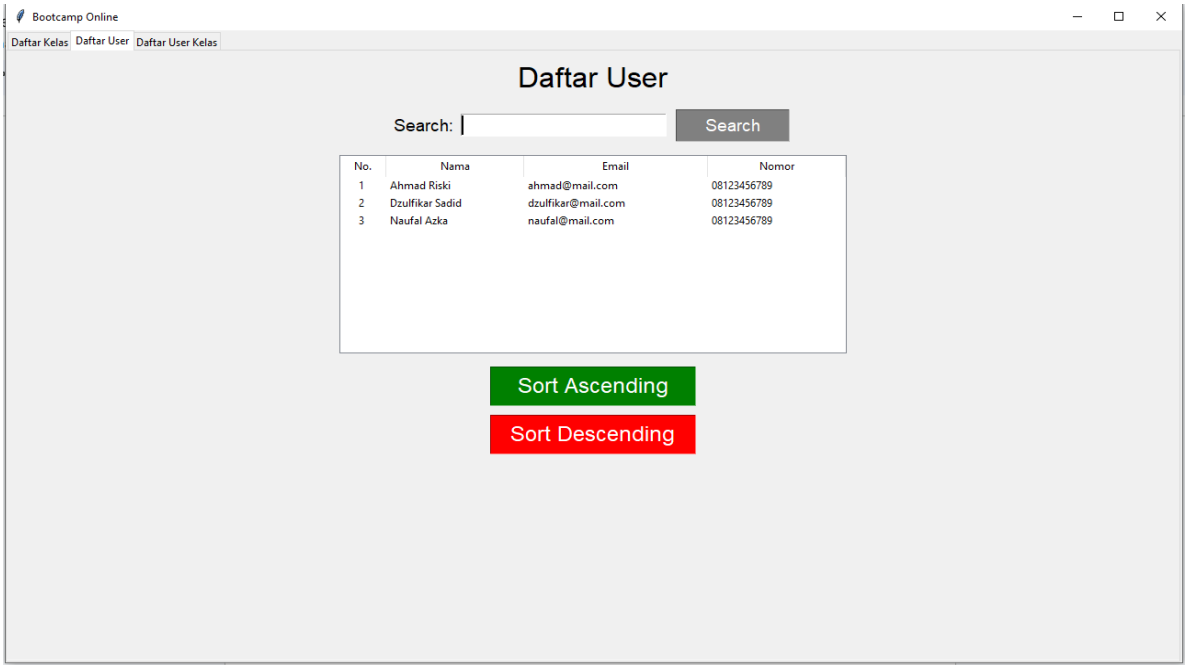
Gambar1. 6
Tampilan Form Join Class

Menu inputan untuk user, user dapat memasukkan data dirinya pada menu ini dan dapat menekan submit jika sudah selesai.



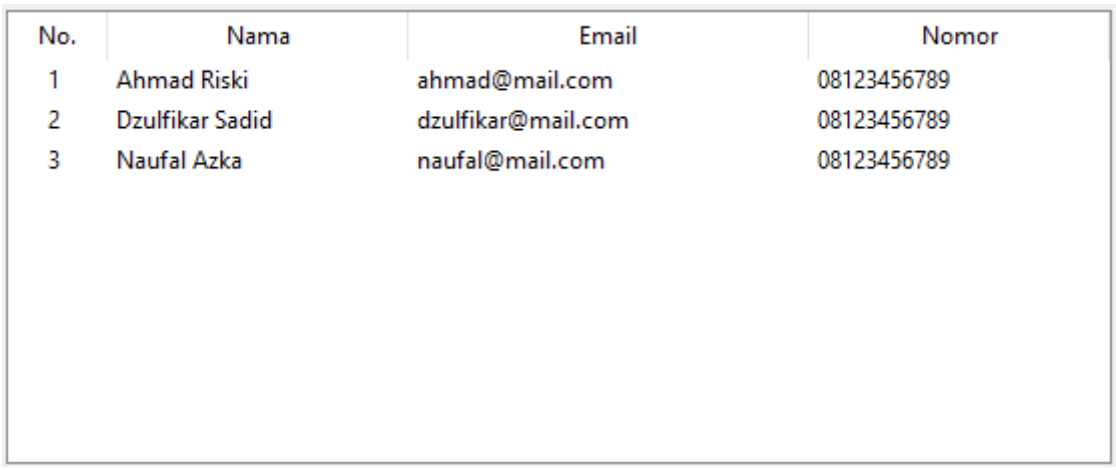
Gambar1. 7
Tampilan Bar Menu Kedua

Menu kedua, yaitu menu Daftar User, jika ditekan maka akan muncul tampilan seperti **Gambar 1.8.**



Gambar1. 8
Tampilan Isi Daftar User

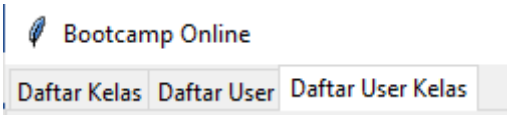
Tampilan menu Daftar User, disini pun terdapat fitur pencarian dan pengurutan untuk memudahkan user.



No.	Nama	Email	Nomor
1	Ahmad Riski	ahmad@mail.com	08123456789
2	Dzulfikar Sadid	dzulfikar@mail.com	08123456789
3	Naufal Azka	naufal@mail.com	08123456789

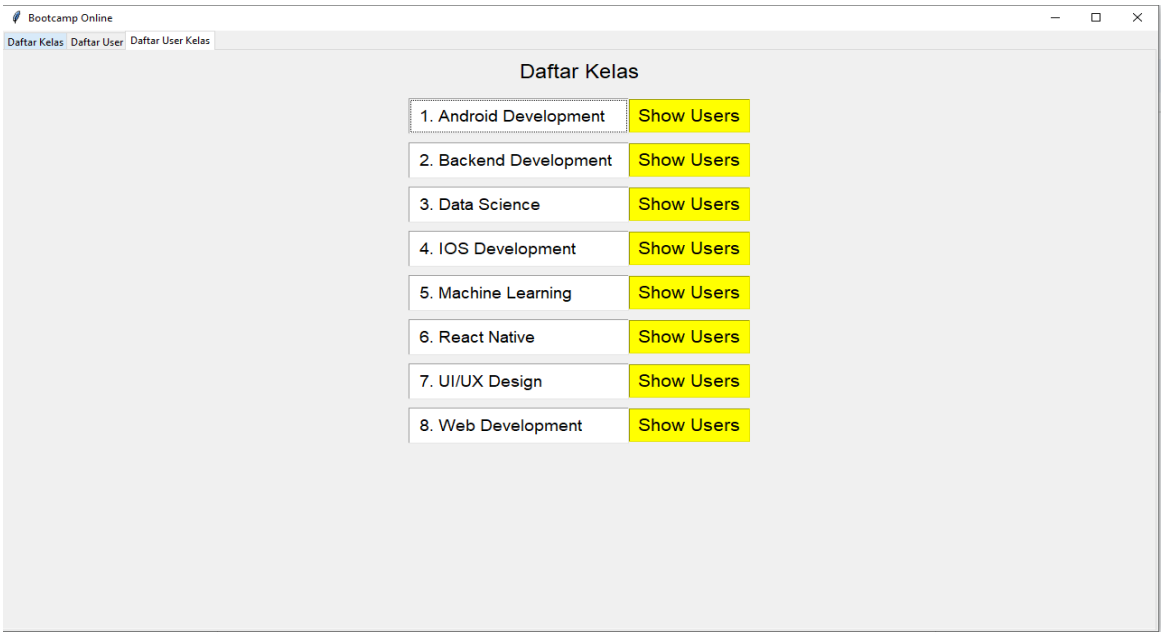
Gambar1. 9
Tampilan Tabel Daftar User

User dapat melihat list user yang terdaftar di kelas.



Gambar1. 10
Tampilan Menu Ketiga

Menu ketiga, yaitu Daftar User Kelas, jika ditekan, maka akan muncul tampilan seperti **Gambar 1.11.**

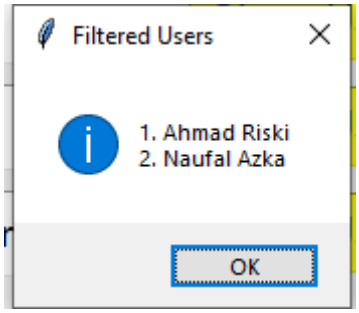


Gambar1. 11
Tampilan Isi Daftar User Yang Mengikuti Kelas

Tampilan menu Daftar User Kelas.

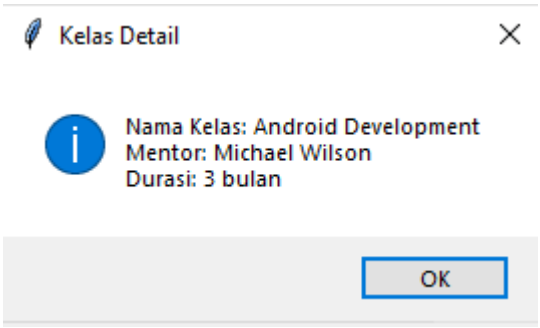
Gambar1. 12
Tampilan Detail User Kelas

Tampilan daftar kelas yang tersedia, user dapat melihat list nama yang terdaftar di dalam kelas, seperti pada **Gambar 1.13.**



Gambar1. 13
Tampilan List User Yang Mendaftar Kelas

Tampilan list nama yang terdaftar dikelas.



Gambar1. 14
Tampilan Detail Kelas

Tampilan Detail Kelas, jika user menekan nama kelas, baik di menu pertama, maupun menu ketiga.

KONTRIBUSI ANGGOTA KELOMPOK

1. Ahmad Riski = Riset ide, sistem dan membuat pseudo-code
2. Attala Arrafi = Membantu pseudo-code dan menyusun daftar isi dan daftar gambar
3. Dzulfikar Sadid = Mengubah pseudo-code menjadi python
4. Naufal Azka = Menyusun laporan dan melengkapi fitur
5. Rizki Eskhart = Menyusun laporan dan mengembangkan ide