LAPORAN TUGAS GUI
MANAJEMEN BOOKING LAPANGAN FUTSAL

PEMROGRAMAN BERBASIS OBJEK

5230411327

Muhammad Irfan Baihaqi

Kelas VIII

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS & TEKNOLOGI

UNIVERSITAS TEKNOLOGI YOGYAKARTA

YOGYAKARTA

2024

1. Pendahuluan

1.1.Latar Belakang

Dalam era digital yang terus berkembang, penggunaan teknologi untuk mendukung aktivitas bisnis menjadi kebutuhan yang tak terhindarkan. Salah satu contohnya adalah manajemen penyewaan lapangan futsal, yang semakin diminati oleh masyarakat untuk kegiatan olahraga maupun rekreasi. Namun, di banyak tempat, proses pengelolaan penyewaan ini masih dilakukan secara manual, seperti mencatat jadwal di buku atau lembar kerja sederhana.

Metode manual ini sering kali menimbulkan berbagai kendala, seperti jadwal yang tumpang tindih, kesalahan pencatatan, serta sulitnya melacak data pemesanan ketika dibutuhkan. Selain itu, tidak adanya validasi otomatis terhadap jadwal booking membuat risiko bentrok waktu menjadi tinggi. Hal ini tidak hanya merugikan pelanggan, tetapi juga dapat mengganggu operasional bisnis.

Oleh karena itu, dibutuhkan sebuah sistem yang mampu mengelola data pemesanan lapangan futsal secara lebih efisien, terstruktur, dan minim kesalahan. Sistem ini diharapkan dapat memberikan kemudahan bagi admin dalam mencatat, mengelola, dan mencari data, serta memastikan setiap transaksi berjalan lancar tanpa hambatan.

1.2.Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, terdapat beberapa permasalahan utama yang ingin diselesaikan:

a) Bagaimana cara admin untuk dapat dengan mudah memanajemen booking lapangan futsal agar terhidar dari jadwal yang bentrok ?
b) Bagaimana cara admin dalam mencari data daftar booking berdasarkan tanggal tertentu dan nama pembooking untuk memudahkan dalam mengelola jadwal?
c) Bagaimana cara menyimpan data daftar booking ke media penyimpanan yang lebih aman dan teorganisir?
d) Bagaimana cara agar admin selalu valid dalam menginputkan data booking baru?
e) Bagimana cara menghapus data booking bagi pelanggan yang membatalkan jadwal mereka, sehingga slot waktu dan lapangan tersebut dapat digunakan oleh pelanggan lain?

1.3.Solusi

Untuk mengatasi beragam tantangan yang sudah disebutkan pada bagian rumusan masalah, saya menawarkan sebuah aplikasi “Manajemen Booking Lapangan Futsal” . Aplikasi ini sudah terintegrasi dengan basis data MySQL yang tentunya adalah media penyimpanan yang jauh lebih baik dan lebih terogranisir ddaripada mencatatnya di buku atau lembar kerja sederhana. Aplikasi yang saya bangun juga sudah dilengkapi dengan *Graphical User Interface* yang sederhana dan mudah digunakan, bahkan untuk user awam sekalipun.

Untuk menyelesaikan masalah yang telah dirumuskan, sistem Manajemen Booking Lapangan Futsal dirancang dengan beberapa fitur utama sebagai solusi:

a) Form untuk pembooking baru yang dilengkapi dengan validasi.

   Sistem akan memastikan tidak ada inputan yang tidak valid , seperti salah saat menginputkan tanggal , misal menginputkan tanggal 32 atau bulan 14, atau menginputkan tanggal dan waktu booking yang kurang dari waktu saat ini. Selain itu , sistem juga memiliki validasi otomatis agar tidak terjadi waktu *booking* yang tumpang tindih. Hal ini dilakukan dengan memeriksa data di database sebelum menyimpan jadwal baru.

b) Fitur pencarian data booking pada tanggal tertentu dan nama pembooking.

   Admin dapat mencari data pemesanan berdasarkan tanggal tertentu, sehingga mempermudah pencatatan dan evaluasi jadwal harian. Selain itu, admin juga dapat melakukan pencarian data pemesanan booking oleh pelanggan / pembooking tertentu.

c) Integrasi dengan database MySQL

   Semua data, termasuk nama pembooking, nomor telepon, jadwal booking, dan detail lainnya, disimpan dalam database yang terorganisasi. Database ini juga mendukung pengelolaan data dalam jumlah besar dengan kecepatan akses yang optimal.

d) Antarmuka yang berbasis GUI dengan Tkinter

   Sistem dilengkapi dengan antarmuka berbasis GUI yang dirancang sederhana dan user-friendly. Admin dapat dengan mudah memasukkan data, melihat tabel pemesanan, atau menghapus data yang tidak lagi diperlukan.

e) Kemudahan Operasional

   Sistem ini menyediakan fitur tambahan, seperti tombol reset untuk menghapus semua inputan di form entry, konfirmasi transaksi sebelum data disimpan, dan scrolling untuk tabel yang menampilkan data dalam jumlah besar.

f) Delete Data Booking yang dibatalkan.

   Sistem dapat dengan mudah menghapus data / jadwal booking yang telah dibatalkan oleh pelanggan. Admin hanya perlu memilih data yang ingin dihapus pada tabel , kemudian menekan tombol hapus

Melalui solusi yang disediakan fitur-fitur pada aplikasi Manajemen Booking Lapangan Futsal diharapkan mampu meningkatkan efisiensi operasional, mengurangi kesalahan pencatatan, dan memberikan pengalaman yang lebih baik bagi pengguna.

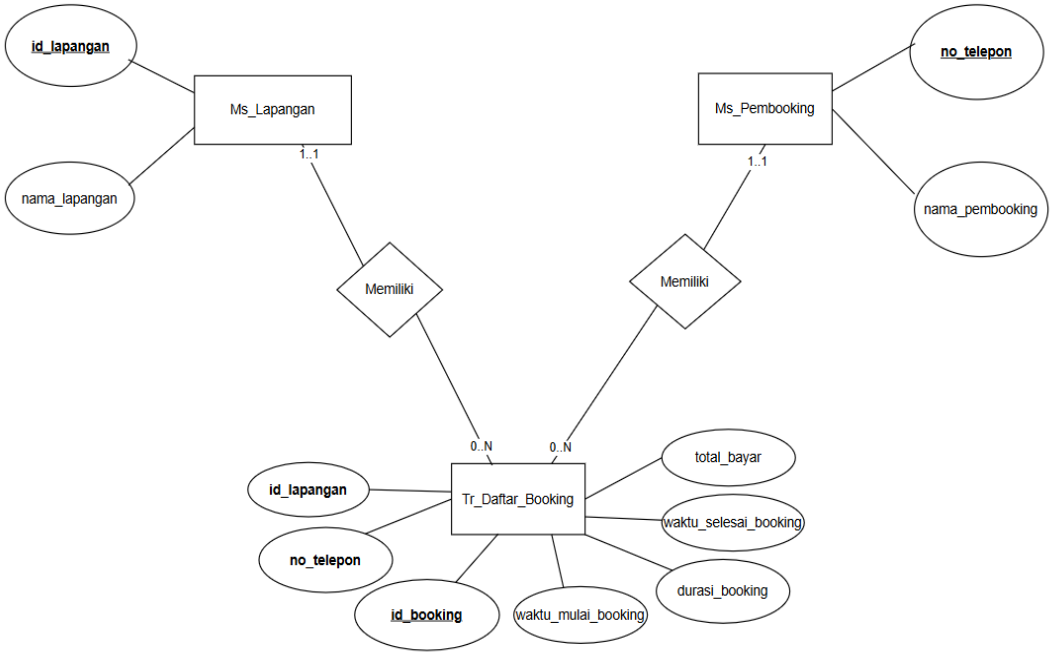
2. Rancangan Sistem

Sebelum memulai implementasi kode program, langkah awal yang saya lakukan adalah merancang sistem secara menyeluruh untuk memastikan setiap komponen berjalan sesuai tujuan. Perancangan ini mencakup aspek database dan alur kerja sistem, yang diwujudkan melalui diagram UML. Untuk menggambarkan

struktur dan hubungan data dalam sistem, saya menggunakan Entity Relationship Diagram (ERD), yang menjadi acuan dalam membangun dan mengintegrasikan database. Selain itu, untuk memvisualisasikan proses bisnis dan alur aktivitas utama dalam sistem, saya membuat Activity Diagram.

ERD membantu saya memahami dan mendefinsikan hubungan antar entitas yang ada, seperti data pemesan, data lapangan, dan data booking, sehingga pengelolaan informasi menjadi lebih terstruktur. Sementara itu, Activity Diagram memberikan gambaran yang jelas tentang bagaimana sistem akan beroperasi, mulai dari proses input data, validasi, hingga penyimpanan ke database. Dengan pendekatan ini, saya dapat memetakan setiap fungsi dan fitur sistem secara rinci sebelum beralih ke tahap implementasi program.

2.1. Entity Relational Diagram



Entitas Ms_Lapangan digunakan untuk Menyimpan data terkait lapangan futsal yang tersedia. Memiliki 2 atribut , yaitu atribut id_lapangan sebagai primary key dan nama_lapangan.

Entitas Ms_Pembooking digunakan untuk Menyimpan data pembooking atau pelanggan yang melakukan pemesanan lapangan. Memiliki 2 atribut, yaitu atribut no_telepon sebagai primary key dan nama_pembooking.

Entitas Tr_Daftar_Booking digunakan untuk Menyimpan data transaksi booking lapangan futsal. Memiliki beberapa atribut:

- id_booking (Primary Key): Identitas unik untuk setiap transaksi booking.
- id_lapangan (Foreign Key): Merujuk pada id_lapangan di tabel Ms_Lapangan untuk mengidentifikasi lapangan yang dipesan.
- no_telepon (Foreign Key): Merujuk pada no_telepon di tabel Ms_Pembooking untuk mengidentifikasi pembooking.
- waktu_mulai_booking: Waktu mulai dari pemesanan.

- durasi_booking: Durasi booking dalam satuan jam.
- waktu_selesai_booking: Waktu selesai dari pemesanan, dihitung berdasarkan --waktu_mulai_booking dan durasi_booking.
- total_bayar: Biaya yang harus dibayarkan berdasarkan durasi pemakaian lapangan.

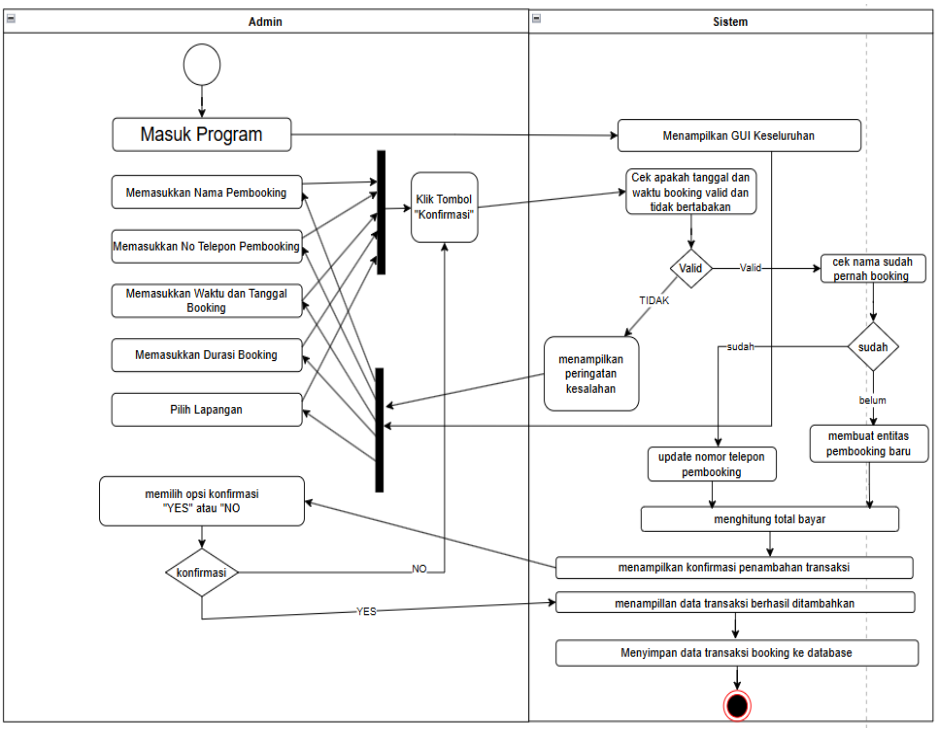
Hubungan Antar Entitas:

Hubungan yang terjadi di antara entitas Ms_Lapangan dan Tr_Daftar_Booking adalah hubungan “memiliki”. Ms_Lapangan memiliki kardinalitas 0..N terhadap Tr_Daftar_Booking , yang artinya 1 lapangan bisa memiliki minimal 0 transaksi dan maksimal banyak transaksi. Tr_Daftar_Booking memiliki kardinalitas 1..1 terhadap Ms_Lapangan , yang artinya 1 transaksi hanya bisa memiliki 1 lapangan. Relasi ini direpresentasikan dengan Foreign Key id_lapangan di tabel Tr_Daftar_Booking.

Hubungan yang terjadi di antara Ms_Pembooking dengan Tr_Daftar_Booking adalah hubungan “Memiliki”. Ms_Pembooking memiliki kardinalitas 0..N terhadap Tr_Daftar_Booking, yang artinya 1 pembooking minimal memiliki 0 transaksi dan maksimal banyak transaksi. Alasan pembooking bisa memiliki minimal 0 transaksi adalah karena ketika admin atau user menghapus data transaksi atau membatalkan booking dari pembooking , hanya data transaksi saja yang dihapus, data pada tabel pembooking tidak ikut dihapus. Maka dari itu, adalah mungkin jika pembooking memiliki 0 transaksi. Kemudian Tr_Daftar_Booking memiliki kardinalitas 1..1 terhadap entitas Ms_Pembooking, yang artinya 1 transaksi booking pasti memiliki 1 pembooking.

2.2. Activity Diagram

2.2.1 Menambahkan Data Transaksi Booking Baru



Saat admin ingin menambahkan transaksi booking baru , tentu akan membuka program terlebih dahulu. Kemudian program akan menampilkan GUI untuk melakukan proses yang ingin dilakukan. Admin bisa menambah data transaksi baru dengan mengisi beberapa entry , yaitu Nama Pembooking, Nomor telepon pembooking, waktu dan tanggal nooking, durasi booking dan memilih lapangan yang ingin dibooking pelanggan.

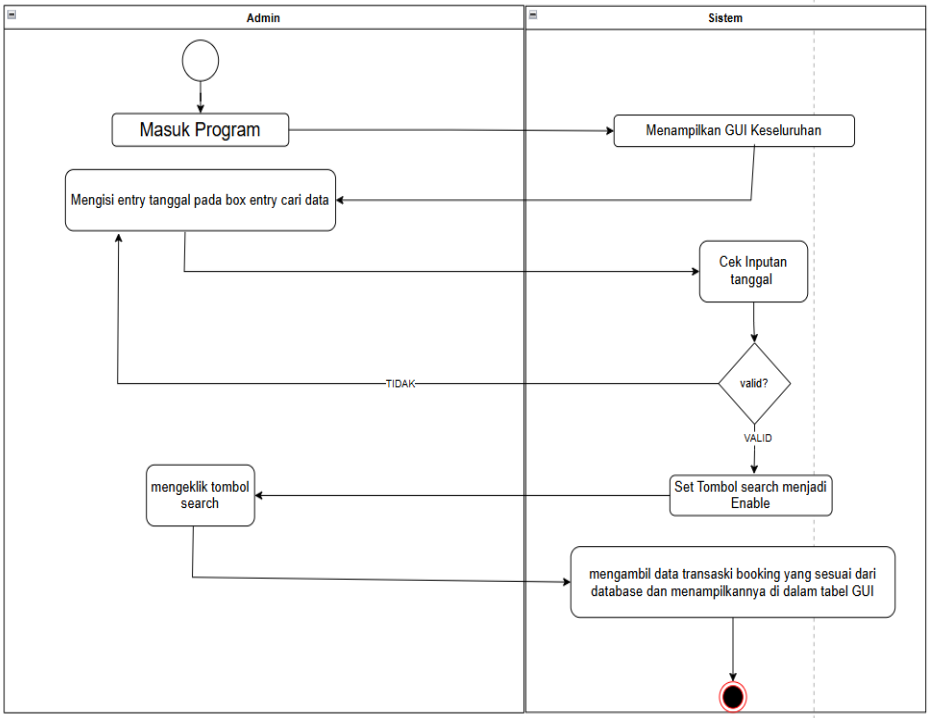
Setelah memasukkan semua entry, sistem akan mengecek apakah tanggal , waktu dan lapangan valid sesuai format tanggal dan apakah sudah sudah dipesan / dibooking pembooking sebelumnya. Jika sudah dibooking atau data entry tanggal dan waktu booking tidak valid, maka akan muncul peringatan. Maksud dari data tanggal dan waktu yang tidak valid adalah yang memenuhi kondisi yaitu diantaranya memasukkan tanggal dan waktu kurang dari waktu sekarang atau memasukkkan tanggal dan waktu yang tidak sesuai dengan format tanggal yang sebenarnya (misal 2024-13-32, tidak ada bulan 13 dan tanggal 32 dalam kalender nyata). Admin dapat mengisi ulang entry yang salah. Akan tetapi , jika Tanggal dan waktu valid dan tidak bertabrakan dengan jadwal booking pembooking sebelumnya , maka sistem akan melakukan pengecekan nama pembooking.

Jika nama pembooking sudah terdaftar , maka akan mengudate nomor telepon pembooking (agar nomor telepon pembooking selalu terbarukan) . Jika nama pembooking belum terdaftar , maka sistem akan membuat entitas baru di tabel Ms_Pembooking.

Baik sudah atau belum nama pembooking terdaftar di database , program akan melanjutkan proses selanjutnya , yaitu menghitung total

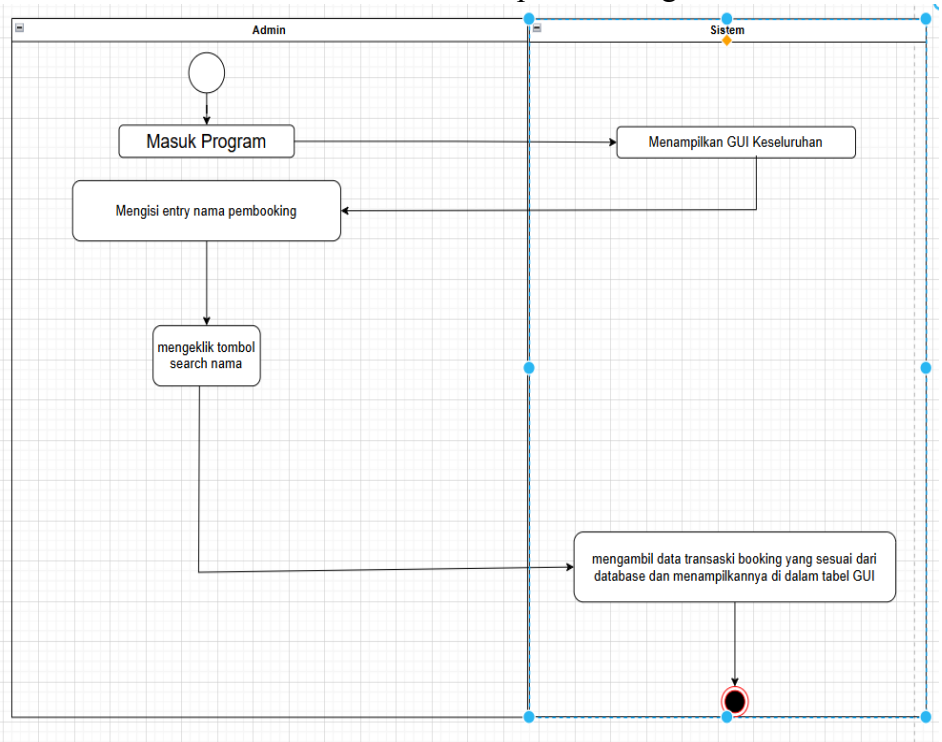
bayar. Setelah itu , program akan menampilkan pilihan konfimasi kepada admin. Admin akan memutuskan apakah akan menambahkan transaksi atau tidak. Jika tidak maka admin dapat mengecek sekali lagi data entrynya, lalu menekan tombol “Konfirmasi” Kembali. Jika admin menekan opsi “YA” , maka program akan menampilkan pemberitahuan bahwa daftar transaksi berhasil ditambahkan dan data transaksi booking akan ditambahkan ke database khususnya pada tabel Tr_Daftar_booking.

2.2.2 Mencari Data Transaksi Pada tanggal tertentu



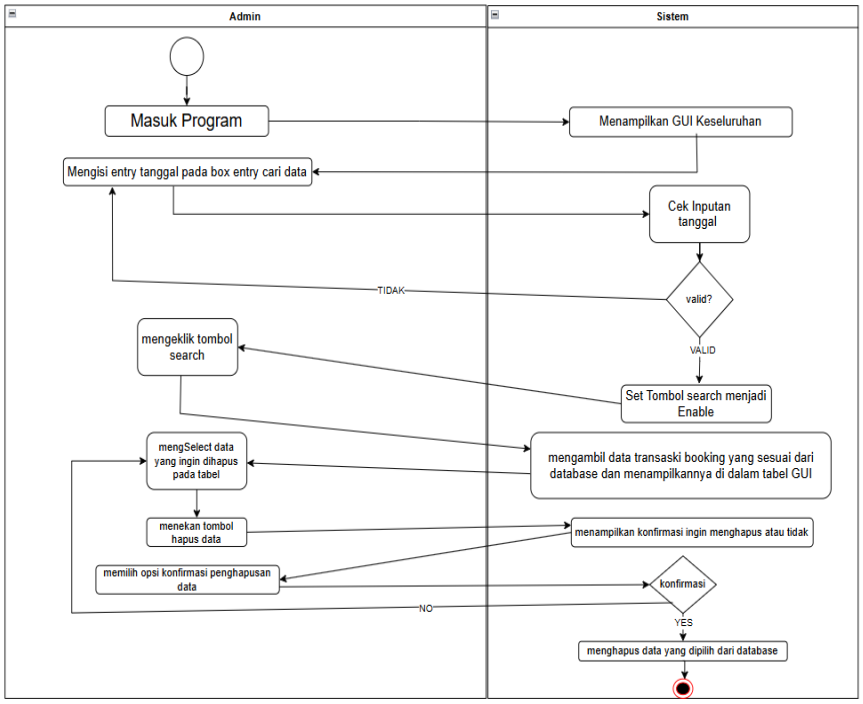
Saat admin ingin mencari data transaksi booking pada tanggal tertentu, tentu harus membuka program terlebih dahulu. Kemudian program akan menampilkan GUI untuk melakukan proses yang ingin dilakukan. Selanjutnya , untuk mencari data transaksi booking pada tanggal tertentu, admin dapat mengisikan box entry tanggal. Jika format tanggal yang diisikan sudah sesuai , maka tombol search akan menjadi enable. Admin dapat menekan tombol tersebut untuk mencari data transaksi booking. Sistem akan mencari data transaksi booking yang sesuai , mengambilnya dari database dan menampilkannya ke dalam tabel.

2.2.3 Mencari data transaksi berdasarkan nama pembooking.



Untuk mencari data transaksi booking pembooking tertentu , setelah menjalankan program , admin / user hanya perlu mengisi box entry cari nama, lalu menekan tombol cari nama. Maka sistem akan otomatis mencari data transaksi yang sesuai dengan nama tersebut, kemudian menampilkannya ke dalam tabel.

2.2.4 Menghapus data transaski booking



Saat admin ingin menghapus salah satu transaksi booking, tentu akan membuka program terlebih dahulu. Kemudian program akan menampilkan GUI untuk melakukan proses yang ingin dilakukan. Proses selanjutnya sama seperti saat mencari data transaksi di tanggal tertentu. Setelah data tampil di dalam tabel, admin dapat mengselect data yang ingin dihapus. Kemudian menekan tombol hapus data. Program akan menampilkan messagebox untuk konfirmasi. Jika admin memiih untuk batal menghapus data , admin dapat menekan "NO". Jika Admin sudah yakin dan ingin menghapus data yang diselect, dapat menekan "YA", program atau sistem kemudian akan melakukan penghapusan data yang dipilih dari database, khususnya pada tabel Tr_Daftar_Booking.

3. Penjelasan kode program
   Kode program secara keseluruhan sudah dilampirkan di folder github ini, yaitu file / modul M8_5230411327_MuhammadIrfanBaihaqi.py .

   penjelasan kode program :

```
import mysql.connector
from mysql.connector import Error

def koneksi():
    try:
        # Koneksi ke database
        connection = mysql.connector.connect(
            host='localhost',
            database='futsal',
            user='root',
            password=''
        )
        return connection

    except Error as e:
        print("Error saat mencoba menghubungkan ke MySQL", e)
```

   Kode di atas adalah fungsi Python bernama koneksi() yang digunakan untuk membuat koneksi ke database MySQL menggunakan library mysql.connector. Fungsi ini mencoba menghubungkan aplikasi ke server MySQL dengan menentukan detail konfigurasi seperti host (localhost), database (futsal), user (root), dan password (kosong dalam contoh ini). Jika koneksi berhasil, fungsi akan mengembalikan objek koneksi (connection) yang memungkinkan interaksi dengan database. Jika terjadi kesalahan (misalnya, server tidak dapat diakses atau kredensial salah), kesalahan tersebut akan ditangkap oleh blok except dan pesan error akan dicetak

menggunakan variabel e. Kode ini adalah dasar untuk mengelola transaksi database dalam aplikasi Python.

```
class AppOrder:
   def __init__(self, root) -> None:
      self.root = root
      self.root.title('Manajemen Booking Lapangan Futsal')
      self.root.geometry('1000x800+300+10')
      self.root.configure(bg='gray')

      # Frame utama
      self.frame_top = Frame(self.root, bg='white', relief=RIDGE, bd=2)
      self.frame_top.pack(side=TOP, fill=X, padx=10, pady=5)

      self.frame_middle = Frame(self.root, bg='white', relief=RIDGE, bd=2)
      self.frame_middle.pack(fill=BOTH, expand=True, padx=10, pady=5)

      self.frame_bottom = Frame(self.root, bg='white', relief=RIDGE, bd=2)
      self.frame_bottom.pack(side=BOTTOM, fill=X, padx=10, pady=5)
```

Kelas **AppOrder** bertujuan untuk mengelola antarmuka aplikasi Manajemen Booking Lapangan Futsal. Dalam metode __init__, objek utama aplikasi (root) diatur dengan berbagai properti. Misalnya, judul jendela aplikasi diatur menggunakan **self.root.title()** dengan teks "Manajemen Booking Lapangan Futsal", ukuran jendela diatur menggunakan **self.root.geometry('1000x800+300+10')** (lebar 1000 piksel, tinggi 800 piksel, dengan posisi awal di layar 300 piksel ke kanan dan 10 piksel ke bawah), serta latar belakang jendela diatur menjadi abu-abu (gray).

Untuk tata letak antarmuka, kode ini membuat tiga Frame utama: **frame_top, frame_middle, dan frame_bottom**, yang masing-masing merupakan kontainer untuk elemen-elemen lain (seperti tombol, tabel, atau formulir input). Frame-frame ini memiliki gaya berupa latar belakang putih (**bg='white'**), efek batas (**relief RIDGE**), dan ketebalan batas (**bd=2**). Posisi setiap frame ditentukan dengan metode **pack().** Frame_top berada di atas dengan mode horizontal (**side=TOP**) dan meluas ke seluruh lebar jendela (fill=X), frame_middle berada di tengah dengan mode dinamis untuk mengisi ruang yang tersedia (**fill=BOTH, expand=True**), sedangkan frame_bottom diletakkan di bagian bawah aplikasi dengan cara mirip frame_top. Struktur ini memberikan fleksibilitas dalam menambahkan elemen antarmuka secara terorganisir di setiap frame.

```
self.create_search_section()

self.create_table_section()

self.create_form_section()
```

Fungsi tersebut dipanggil di dalam init class ini untuk membuat 3 bagian utama interface, yaitu bagian pencarian, tabel data, dan form booking.

```
def create_search_section(self):
    # cari berdasarkan tanggal
    Label(self.frame_top, text="Cari Tanggal (YYYY-MM-DD):", bg='white').grid(row=0, column=0, padx=10, pady=10, sticky=W)
    self.ent_search_date = Entry(self.frame_top, width=20)
    self.ent_search_date.grid(row=0, column=1, padx=10, pady=10)
    self.btn_search = Button(self.frame_top, text="Cari Data", command=self.search_booking)
    self.btn_search.grid(row=0, column=2, padx=10, pady=10)
    self.btn_search["state"] = DISABLED
    self.ent_search_date.bind("<KeyRelease>", self.validate_date)

    # cari berdasarkan nama pembooking
    Label(self.frame_top, text="Cari Nama Pembooking: ", bg='white').grid(row=0, column=3, padx=10, pady=10, sticky=W)
    self.ent_search_name = Entry(self.frame_top, width=20)
    self.ent_search_name.grid(row=0, column=4, padx=10, pady=10)
    self.btn_search2 = Button(self.frame_top, text="Cari Nama", command=self.search_booking2)
    self.btn_search2.grid(row=0, column=5, padx=10, pady=10)
```

Kode di atas adalah metode create_search_section yang bertujuan untuk membuat bagian antarmuka pencarian data dalam aplikasi Manajemen Booking Lapangan Futsal. Metode ini menambahkan elemen-elemen input pada frame_top untuk memfasilitasi pencarian berdasarkan tanggal dan nama pembooking. Untuk pencarian tanggal, dibuat sebuah **Label** yang menampilkan teks "Cari Tanggal (YYYY-MM-DD):" sebagai panduan pengguna, diikuti oleh **widget Entry (self.ent_search_date)** sebagai tempat input tanggal. Tombol **Button** (**self.btn_search**) dengan teks "Cari Data" berfungsi untuk memicu fungsi search_booking ketika diklik. Posisi elemen-elemen ini diatur menggunakan layout grid. Label berada di kolom 0, Entry di kolom 1, dan tombol di kolom 2, semuanya dalam baris yang sama, sehingga tertata secara horizontal.

Tombol "Cari Data" awalnya dinonaktifkan (**state=DISABLED**) hingga input tanggal yang valid diberikan. Proses validasi dilakukan dengan menghubungkan event **KeyRelease** pada input tanggal ke fungsi **validate_date,** yang otomatis memeriksa validitas setiap kali pengguna mengetik atau mengubah nilai.

Untuk pencarian berdasarkan nama pembooking, metode ini menambahkan **Label** kedua dengan teks **"Cari Nama Pembooking:"** pada kolom 3, diikuti oleh **widget Entry (self.ent_search_name)** di kolom 4 sebagai tempat pengguna memasukkan nama. **Tombol Button (self.btn_search2)** dengan teks "Cari Nama" ditempatkan di kolom 5, berfungsi untuk **memicu fungsi search_booking2 ketika ditekan**. Sama seperti elemen-elemen sebelumnya, semuanya diatur dalam satu baris menggunakan layout grid. Dengan dua jenis pencarian ini, antarmuka menjadi fleksibel untuk mempermudah pengguna mencari data transaksi berdasarkan tanggal atau nama pembooking, sehingga lebih efisien dalam pengelolaan data.

```
def create_table_section(self):
        # Tabel Data
        self.tree = ttk.Treeview(
            self.frame_middle,
            columns=("ID Booking", "Nama Pembooking" ,"No Telepon", "ID Lapangan",
"Waktu Mulai", "Durasi", "Waktu Selesai", "Total Bayar"),
            show="headings",
        )
        self.tree.heading("ID Booking", text="ID Booking")
        self.tree.heading("Nama Pembooking", text="Nama Pembooking")
        self.tree.heading("No Telepon", text="No Telepon")
        self.tree.heading("ID Lapangan", text="ID Lapangan")
        self.tree.heading("Waktu Mulai", text="Waktu Mulai")
        self.tree.heading("Durasi", text="Durasi")
        self.tree.heading("Waktu Selesai", text="Waktu Selesai")
        self.tree.heading("Total Bayar", text="Total Bayar")

        self.tree.column("ID Booking", anchor='center')
        self.tree.column("Nama Pembooking", anchor='center')
        self.tree.column("No Telepon", anchor='center')
        self.tree.column("ID Lapangan", anchor='center')
        self.tree.column("Waktu Mulai", anchor='center')
        self.tree.column("Durasi", anchor='center')
        self.tree.column("Waktu Selesai", anchor='center')
        self.tree.column("Total Bayar", anchor='center')
        self.tree.pack(fill=BOTH, expand=True, padx=10, pady=10)
            # Scrollbar Vertikal
        scroll_y = Scrollbar(self.frame_middle, orient=VERTICAL,
        command=self.tree.yview)
        self.tree.configure(yscrollcommand=scroll_y.set)
        scroll_y.pack(side=RIGHT, fill=Y)
```

```
# scrollbar horizontal
 scroll_x = Scrollbar(self.frame_middle, orient=HORIZONTAL, command=self.tree.xview)
 self.tree.configure(xscrollcommand=scroll_x.set)
 scroll_x.pack(side=BOTTOM, fill=X)


 # Tombol Hapus Data
 self.btn_delete = Button(self.frame_middle, text="Hapus Data", command=self.delete_booking)
 self.btn_delete.pack(side=BOTTOM, pady=10)
 self.btn_delete["state"] = DISABLED
 self.tree.bind("<<TreeviewSelect>>", self.activate_delete_button)
```

Method di atas membuat sebuah tabel dengan modul ttk dan widget Treeview, kemudian dimasukkan ke dalam atribut tree. Tabel tersebut ditempatkan di dalam frame middle (self.frame_middle) dan memiliki 8 kolom yang dapat dilihat pada kode berikut:

```
columns=("ID Booking", "Nama Pembooking" ,"No Telepon", "ID Lapangan", "Waktu Mulai", "Durasi", "Waktu Selesai", "Total Bayar")
```

Setiap kolom diberi nama header menggunakan fungsi heading, serta posisi teks dalam kolom diatur dengan atribut anchor='center'. Tabel ini ditempatkan di dalam frame_middle dengan tata letak yang otomatis menyesuaikan ukuran (expand dan fill).

Untuk mempermudah navigasi data yang panjang atau lebar, metode ini menambahkan scrollbar vertikal dan horizontal. Scrollbar vertikal dibuat dengan orientasi VERTICAL dan dihubungkan ke Treeview menggunakan fungsi command=self.tree.yview. Konfigurasi yscrollcommand=scroll_y.set memastikan scrollbar mengikuti gerakan vertikal tabel. Scrollbar horizontal ditambahkan dengan cara yang sama untuk mendukung navigasi data secara horizontal. Kedua scrollbar ini ditempatkan di sisi kanan dan bawah tabel dengan tata letak pack.

Di bagian bawah tabel, terdapat tombol "Hapus Data" yang berfungsi untuk menghapus data yang dipilih dalam tabel. Namun, tombol ini awalnya dinonaktifkan (state=DISABLED) untuk mencegah pengguna menghapus data secara tidak sengaja sebelum memilih item. Fungsi self.activate_delete_button dihubungkan dengan event "<<TreeviewSelect>>", yang terjadi ketika pengguna memilih baris dalam tabel. Ketika event ini terpicu, tombol "Hapus Data" akan diaktifkan sehingga pengguna dapat menghapus data yang dipilih. Pendekatan ini meningkatkan interaksi pengguna dengan antarmuka sekaligus menjaga keamanan data.

```
def create_form_section(self):
        # Input Data
        Label(self.frame_bottom, text="Nama Pembooking:", bg='white').grid(row=0, column=0, padx=10, pady=10, sticky=W)
        self.ent_nama = Entry(self.frame_bottom, width=25)
        self.ent_nama.grid(row=0, column=1, padx=10, pady=10, sticky=W)

        Label(self.frame_bottom, text="No Telepon:", bg='white').grid(row=1, column=0, padx=10, pady=10, sticky=W)
        self.ent_telepon = Entry(self.frame_bottom, width=25)
        self.ent_telepon.grid(row=1, column=1, padx=10, pady=10, sticky=W)

        Label(self.frame_bottom, text="Waktu Booking (YYYY-MM-DD hh):", bg='white').grid(row=2, column=0, padx=10, pady=10, sticky=W)
        self.ent_date = Entry(self.frame_bottom, width=15)
        self.ent_date.grid(row=2, column=1, padx=5, pady=10, sticky=W)
        self.ent_hour = Entry(self.frame_bottom, width=5)
        self.ent_hour.grid(row=2, column=2, padx=5, pady=10, sticky=W)

        Label(self.frame_bottom, text="Durasi Booking (jam):", bg='white').grid(row=3, column=0, padx=10, pady=10, sticky=W)
        self.ent_durasi = Entry(self.frame_bottom, width=10)
        self.ent_durasi.grid(row=3, column=1, padx=10, pady=10, sticky=W)
        Label(self.frame_bottom, text="Pilih Lapangan:", bg='white').grid(row=4, column=0, padx=10, pady=10, sticky=W)
        self.combo_lapangan = ttk.Combobox(self.frame_bottom, state="readonly", width=23)
        self.combo_lapangan.grid(row=4, column=1, padx=10, pady=10, sticky=W)
        self.load_lapangan()

        # Tombol Form
        Button(self.frame_bottom, text="Hapus Inputan", command=self.clear_form).grid(row=5, column=0, padx=10, pady=20)
        Button(self.frame_bottom, text="Konfirmasi", command=self.confirm_booking).grid(row=5, column=1, padx=10, pady=20)
```

Metode `create_form_section` digunakan untuk membuat form input data booking yang terletak di bagian bawah aplikasi. Form ini memungkinkan pengguna untuk memasukkan data seperti nama pembooking, nomor telepon, waktu booking, durasi booking, dan pilihan lapangan. Setiap komponen input diletakkan dengan menggunakan widget Label dan Entry, serta diatur posisinya dalam tata letak grid. Misalnya, input untuk nama pembooking menggunakan widget `Entry` dengan lebar 25 karakter dan ditempatkan di kolom 1 baris 0. Komponen ini berfungsi untuk menerima input teks dari pengguna.

Bagian waktu booking dirancang agar pengguna dapat memasukkan tanggal dan jam secara terpisah. Tanggal dimasukkan melalui widget ‘Entry’ selebar 15 karakter, sedangkan jam dimasukkan melalui widget ‘Entry’ selebar 5 karakter. Format inputnya adalah `YYYY-MM-DD hh` untuk memastikan data yang dimasukkan sesuai dengan format yang diharapkan oleh sistem. Durasi booking diatur melalui input angka dalam satuan jam, yang juga menggunakan widget `Entry` selebar 10 karakter. Input durasi ini akan digunakan untuk menghitung waktu selesai booking secara otomatis.

Pilihan lapangan dikelola menggunakan widget Combobox dari ‘ttk’. Combobox ini menampilkan daftar lapangan yang tersedia, yang datanya dimuat dari database menggunakan fungsi `load_lapangan`. Dengan pengaturan `state="readonly"`, pengguna hanya dapat memilih dari daftar yang disediakan, sehingga mengurangi kemungkinan kesalahan input. Penempatan combobox dilakukan di kolom 1 baris 4, dengan label deskriptif di kolom 0.

Bagian tombol menyediakan dua fitur utama, yaitu “Hapus Inputan” dan “Konfirmasi”. Tombol "Hapus Inputan" menghapus semua data yang sudah diisi pengguna melalui fungsi `clear_form`. Tombol "Konfirmasi" mengarahkan pengguna untuk menyimpan data booking ke database melalui fungsi ‘confirm_booking’. Tombol-tombol ini membantu mengelola alur input data dengan lebih terstruktur.

Metode ini dapat ditambahkan dengan fitur untuk menampilkan total biaya booking secara otomatis. Komponen tambahan ini dapat berupa label atau field khusus yang ditempatkan di frame yang sama, misalnya di bawah input durasi. Total biaya dapat dihitung berdasarkan durasi booking dan tarif lapangan, sehingga pengguna dapat langsung melihat estimasi biaya tanpa perlu menghitung secara manual. Fitur ini akan meningkatkan kenyamanan pengguna dalam menggunakan aplikasi.

```
def load_lapangan(self):
        try:
            conn = koneksi()
            cursor = conn.cursor()
```

```
            cursor.execute("SELECT nama_lapangan FROM Ms_Lapangan")
            rows = cursor.fetchall()
            self.combo_lapangan["values"] = [f"{row[0]}" for row in rows]
            cursor.close()
            conn.close()
        except Exception as e:
            messagebox.showerror("Error", f"Gagal memuat data lapangan: {e}")
```

Metode “load_lapangan” digunakan untuk memuat data nama lapangan dari tabel database Ms_Lapangan dan menampilkannya sebagai pilihan di widget Combobox. Prosesnya diawali dengan membuat koneksi ke database menggunakan fungsi koneksi(), lalu membuat objek cursor untuk menjalankan perintah SQL. Perintah cursor.execute("SELECT nama_lapangan FROM Ms_Lapangan") menginstruksikan database untuk mengambil semua data dari kolom nama_lapangan di tabel Ms_Lapangan. Hasilnya disimpan dalam variabel rows menggunakan metode fetchall(), yang mengambil semua baris hasil query sebagai daftar tuple. Setiap tuple mewakili satu baris dari tabel, dan nilai kolom nama_lapangan diakses dengan row[0] dalam list comprehension [f"{row[0]}" for row in rows]. Daftar hasil ini kemudian dimasukkan ke atribut values milik *self.combo_lapangan*, sehingga semua nama lapangan tampil sebagai opsi pada Combobox. Akhirnya, koneksi dan cursor ditutup untuk membebaskan sumber daya. Jika terjadi kesalahan selama proses, pesan error ditampilkan menggunakan kotak dialog *messagebox.showerror.*

```
def validate_date(self, event):
    date_text = self.ent_search_date.get()
    try:
        datetime.strptime(date_text, "%Y-%m-%d")
        self.btn_search["state"] = NORMAL
    except ValueError:
        self.btn_search["state"] = DISABLED
```

Fungsi validate_date ini digunakan untuk memvalidasi input tanggal yang dimasukkan ke dalam Entry bernama ent_search_date. Jadi, setiap kali pengguna mengetik sesuatu di kolom input tanggal, fungsi ini akan otomatis dipanggil (karena terhubung dengan event <KeyRelease>). Fungsi ini membaca teks yang sedang diketik melalui get() dan mencoba memeriksa apakah formatnya sesuai dengan pola YYYY-MM-DD menggunakan datetime.strptime. Jika formatnya benar, tombol "Cari Data" (btn_search) akan

diaktifkan (state = NORMAL), sehingga pengguna bisa menekan tombol tersebut. Namun, jika formatnya salah (misalnya, ada karakter yang tidak sesuai atau urutan tidak benar), fungsi akan menangkap kesalahan ValueError dan otomatis menonaktifkan tombol "Cari Data" (state = DISABLED). Dengan cara ini, aplikasi memastikan bahwa input tanggal selalu dalam format yang valid sebelum melanjutkan pencarian.

```
def search_booking(self):
    date_text = self.ent_search_date.get()
    try:
        conn = koneksi()
        cursor = conn.cursor()
        query = """
            SELECT tr.id_booking,
            mp.nama_pembooking,
            tr.no_telepon,
            tr.id_lapangan,
            tr.waktu_mulai_booking,
            tr.durasi_booking,
            tr.waktu_selesai_booking,
            tr.total_bayar
            FROM Tr_Daftar_booking AS tr JOIN Ms_pembooking AS mp ON tr.no_telepon = mp.no_telepon WHERE DATE(tr.waktu_mulai_booking) = %s
        """
        cursor.execute(query, (date_text,))
        rows = cursor.fetchall()

        for i in self.tree.get_children():
            self.tree.delete(i)

        for row in rows:
            self.tree.insert("", "end", values=row)
        cursor.close()
        conn.close()
```

```
except Exception as e:
    messagebox.showerror("Error", f"Gagal memuat data booking: {e}")
```

Fungsi search_booking digunakan untuk mencari data booking berdasarkan tanggal yang diinputkan oleh pengguna di kolom pencarian. Pertama, fungsi mengambil input dari kolom ent_search_date menggunakan get() untuk mendapatkan teks yang diketik pengguna. Setelah itu, fungsi mencoba membuka koneksi ke database melalui fungsi koneksi().

Query SQL yang ada pada kode tersebut digunakan untuk mendapatkan data booking dengan menggabungkan tabel Tr_Daftar_booking (yang menyimpan informasi transaksi booking) dan tabel Ms_pembooking (yang menyimpan data pembooking) berdasarkan kolom no_telepon sebagai relasi antara kedua tabel. Penggunaan JOIN memungkinkan pengambilan data nama pembooking dari tabel Ms_pembooking yang berhubungan dengan transaksi booking di tabel Tr_Daftar_booking. Bagian WHERE DATE(tr.waktu_mulai_booking) = %s berfungsi untuk memfilter data hanya pada tanggal tertentu yang sesuai dengan input pengguna. Fungsi DATE() digunakan untuk mengambil bagian tanggal dari kolom waktu_mulai_booking, sehingga query tetap bekerja meskipun ada data dengan waktu yang berbeda di hari yang sama. Parameter %s digunakan sebagai placeholder untuk mencegah SQL Injection, dan nilai tanggal input pengguna akan dimasukkan ke dalam query melalui eksekusi cursor.execute(query, (date_text,)). Output dari query ini mencakup data seperti ID booking, nama pembooking, nomor telepon, ID lapangan, waktu mulai, durasi, waktu selesai, dan total bayar, yang kemudian dapat ditampilkan di aplikasi.

Selanjutnya, data yang berhasil diambil dari database (cursor.fetchall()) akan diolah untuk ditampilkan di tabel (Treeview) dalam aplikasi. Sebelum menampilkan data baru, semua data lama di Treeview dihapus menggunakan self.tree.delete(i) pada setiap child item. Kemudian, data hasil query diinsert satu per satu ke Treeview menggunakan self.tree.insert. Jika semua operasi selesai, koneksi dan cursor ke database ditutup untuk mencegah kebocoran sumber daya. Namun, jika terjadi kesalahan selama proses (misalnya koneksi gagal atau query error), fungsi akan menampilkan pesan error menggunakan messagebox.showerror agar pengguna tahu bahwa ada masalah dalam memuat data booking.

Untuk metode `def search_booking2(self)` hampir semuanya sama dengan `def search_booking(self)`. Hanya saja pada `def search_booking2(self)` digunakan untuk mencari data transaksi booking berdasarkan nama pembooking. Hal ini dapat dilihat pada query sql berikut:

```
…. WHERE mp.nama_pembooking = %s
```

Placeholder nantinya akan diisi oleh nama pembooking yang sedang dicari user.

```
 def activate_delete_button(self, event):
      self.btn_delete["state"] = NORMAL
```

Kode di atas adalah sebuah metode yang digunakan untuk mengaktifkan tombol "Hapus Data" setiap kali pengguna memilih data di dalam tabel. Fungsi ini dipicu oleh sebuah event yang terhubung dengan pemilihan item pada widget Treeview pada atribut self.tree yang sudah didefinisikan sebelumnya. Saat sebuah item dipilih (melalui klik), event tersebut akan memanggil fungsi ini, dan di dalamnya, properti ‘state’ dari tombol ‘btn_delete’ diubah menjadi NORMAL, yang berarti tombol tersebut kini aktif dan bisa digunakan. Hal ini dapat mengurangi kesalahan pengguna saat akan menghapus data karena harus ada data yang dipilih untuk bisa menghapus data.

```
def delete_booking(self):
      selected_item = self.tree.selection()
      if not selected_item:
         messagebox.showwarning("Peringatan", "Pilih data terlebih dahulu!")
         return

      confirm = messagebox.askyesno("Konfirmasi", "Apakah Anda yakin ingin
menghapus data yang dipilih?")
      if confirm:
         item = self.tree.item(selected_item)
         id_booking = item["values"][0]

         try:
            conn = koneksi()
            cursor = conn.cursor()
            query = "DELETE FROM Tr_Daftar_booking WHERE id_booking = %s"
            cursor.execute(query, (id_booking,))
            conn.commit()
            cursor.close()
            conn.close()
            self.tree.delete(selected_item)
            messagebox.showinfo("Sukses", "Data berhasil dihapus!")
         except Exception as e:
            messagebox.showerror("Error", f"Gagal menghapus data: {e}")
```

Kode di atas adalah fungsi untuk menghapus data pemesanan yang dipilih oleh pengguna dari tabel Treeview. Pada langkah pertama, fungsi memeriksa apakah ada item yang dipilih dengan menggunakan self.tree.selection(). Fungsi ini mengembalikan ID item yang dipilih oleh pengguna. Jika tidak ada item yang dipilih (hasilnya kosong), maka akan muncul peringatan menggunakan messagebox.showwarning() yang menginformasikan pengguna untuk memilih data terlebih dahulu. Setelah itu, proses penghentian fungsi dilakukan dengan return agar tidak ada langkah-langkah lain yang dijalankan.

Jika ada item yang dipilih, fungsi akan menampilkan konfirmasi melalui messagebox.askyesno() untuk memastikan apakah pengguna benar-benar ingin menghapus data tersebut. Jika pengguna memilih "Yes", maka sistem akan mengambil id_booking dari item yang dipilih menggunakan self.tree.item(selected_item), yang mengakses nilai data berdasarkan ID pemesanan pada kolom pertama tabel. Nilai id_booking ini yang nantinya digunakan untuk menghapus data yang sesuai dari database.

Setelah mendapatkan id_booking, fungsi akan menghubungkan ke database menggunakan koneksi yang dibuat melalui koneksi() dan menyiapkan cursor untuk mengeksekusi query SQL DELETE. Query ini menghapus data yang memiliki id_booking yang sesuai dengan nilai yang telah dipilih. Setelah query berhasil dijalankan, perubahan akan disimpan dengan conn.commit(), dan koneksi serta cursor akan ditutup. Pada akhirnya, item yang dihapus akan dihilangkan dari tampilan tabel menggunakan self.tree.delete(selected_item), dan pesan sukses akan muncul melalui messagebox.showinfo(). Jika terjadi kesalahan saat proses penghapusan, maka error akan ditangani dan pesan kesalahan ditampilkan menggunakan messagebox.showerror().

```
def clear_form(self):
    self.ent_nama.delete(0, END)
    self.ent_telepon.delete(0, END)
    self.ent_date.delete(0, END)
    self.ent_hour.delete(0, END)
    self.ent_durasi.delete(0, END)
    self.combo_lapangan.set("")
```

metode clear_form di atas digunakan untuk menghapus inputan pada masing-masing entry, menggunakan salah satu metode entry yaitu .delete . (0, END) artinya akan menghapus semua inputan dari indeks ke 0 sampai END / terakhir. self.combo_lapangan.set("") digunakan untuk mengosongkan nilai inputan pada combobox.

Metode def confirm_booking(self) digunakan sebagai validasi input untuk form transaksi booking baru. Penjelasan pada metode ini akan saya bagi menjadi beberapa bagian :

a) Mengambil data dari input form / entry.

```
nama = self.ent_nama.get().strip()
telepon = self.ent_telepon.get().strip()
tanggal = self.ent_date.get().strip()
jam = self.ent_hour.get().strip()
durasi = self.ent_durasi.get().strip()
lapangan = self.combo_lapangan.get().strip()
```

Kode di atas digunakan untuk mengambil data dari form inputan yang telah diisi oleh pengguna. Di sini, variabel-variabel seperti nama, telepon, tanggal, jam, durasi, dan lapangan diambil dengan menggunakan metode get() pada masing-masing widget (seperti Entry untuk teks dan Combobox untuk pilihan lapangan). Fungsi strip() digunakan untuk menghapus spasi tambahan di awal dan akhir input, memastikan data yang diambil bersih dan valid.

b) Validasi untuk mencegah input kosong

```
if not all([nama, telepon, tanggal, jam, durasi, lapangan]):
    messagebox.showerror("Error", "Semua field harus diisi.")
    return
```

Kode di atas digunakan untuk memastikan seluruh isian atau *entry* benar-benar terisi. [nama, telepon, tanggal, jam, durasi, lapangan] adalah sebuah list yang berisi variabel-variabel yang harus diisi oleh pengguna. Variabel-variabel ini mewakili informasi yang harus diisi dalam form, seperti nama, nomor telepon, tanggal booking, jam booking, durasi booking, dan lapangan yang dipilih. Fungsi all() dalam Python memeriksa apakah semua elemen dalam list adalah benar (truthy). Jika semua elemen dalam list memiliki nilai yang tidak kosong atau tidak nol (non-zero), maka all() akan mengembalikan True. Sebaliknya, jika ada satu saja elemen yang kosong atau nol (falsy), maka all() akan mengembalikan False. Kemudian hasil dari all akan dinegasikan menggunakan not. Jadi pengkondisiannya berubah menjadi jika seluruh data yang terisi maka hasil dari all akan true , maka jika dinegasikan akan menjadi false , implikasinya tidak akan mengeksekusi kode di bawahnya. Sebaliknya jika ada data yang belum terisi, maka hasil dari all akan false, kemudian akan dinegasikan oleh not menjadi true. Oleh karena hasilnya true, implikasinya akan menjalankan kode di bawahnya yaitu memunculkan messagabox yang menandakan eror dan tidak akan menyimpan data ke database.

c) Validasi format entry tanggal dan jam

```
try:
    waktu_booking = datetime.strptime(f"{tanggal} {jam}:00:00", "%Y-%m-%d
    %H:%M:%S")
    if waktu_booking < datetime.now():
        raise ValueError("Waktu booking tidak boleh kurang dari waktu
        sekarang.")
except ValueError as e:
    messagebox.showerror("Error", f"Format waktu tidak valid: {e}")
    return
```

pertama akan membuat variabel waktu_booking berisi tanggal dan waktu. datetime.strptime(...): Fungsi ini digunakan untuk mengonversi string tanggal dan waktu dari inputan menjadi objek datetime yang nantinya akan diperiksa. if waktu_booking < datetime.now() akan memeriksa apakah waktu_booking kurang dari waktu sekarang ini, jika kurang maka akan raise error.

d) Validasi entry durasi booking

```
try:
        durasi = int(durasi)
        if durasi <= 0:
            raise ValueError("Durasi harus lebih besar dari 0.")
except ValueError:
        messagebox.showerror("Error", "Durasi harus berupa angka positif.")
        return
```

Potongan kode di atas bertujuan untuk memvalidasi input durasi booking yang dimasukkan oleh pengguna. Pertama, kode mencoba mengonversi nilai ‘durasi’ yang didapat dari form menjadi tipe data ‘integer’ menggunakan fungsi ‘int()’. Jika konversi berhasil, kode kemudian memeriksa apakah nilai durasi lebih besar dari 0. Jika durasi bernilai 0 atau negatif, maka kode akan melemparkan ‘ValueError’ dengan pesan error yang menunjukkan bahwa durasi harus lebih besar dari 0. Jika input yang dimasukkan bukan angka atau tidak bisa dikonversi menjadi integer, maka ‘ValueError’ akan ditangkap, dan program akan menampilkan pesan kesalahan melalui ‘messagebox.showerror()’, yang memberi tahu pengguna bahwa durasi harus berupa angka positif. Dengan adanya validasi ini, aplikasi memastikan bahwa durasi yang dimasukkan adalah angka yang benar dan sesuai dengan aturan.

e) Perhitungan waktu selesai booking dan jumlah bayar secara otomatis

```
waktu_selesai = waktu_booking + timedelta(hours=durasi) - timedelta(seconds=1)
total_bayar = durasi * 10000 #SATU JAM 10 RIBU
```

f) Validasi lapangan dan jadwal booking tersedia

```
try:
    conn = koneksi()
    cursor = conn.cursor()
    query = "SELECT id_lapangan FROM Ms_lapangan WHERE nama_lapangan = %s"
    cursor.execute(query,(lapangan,))
    data_lapangan = cursor.fetchone()
    id_lapangan = data_lapangan[0]

    query = """
        SELECT * FROM Tr_Daftar_booking
        WHERE id_lapangan = %s
        AND (
            (%s BETWEEN waktu_mulai_booking AND waktu_selesai_booking) OR
            (%s BETWEEN waktu_mulai_booking AND waktu_selesai_booking) OR
            (waktu_mulai_booking BETWEEN %s AND %s)
        )
    """
```

```
            cursor.execute(query, (id_lapangan, waktu_booking,
            waktu_selesai, waktu_booking, waktu_selesai))
            existing_bookings = cursor.fetchall()
            if existing_bookings:
                messagebox.showerror("Error", "Lapangan sudah dibooking
                pada rentang waktu yang dipilih.")
                cursor.close()
                conn.close()
                return
        except Exception as e:
            messagebox.showerror("Error", f"Gagal memvalidasi lapangan:
            {e}")
            return
```

Potongan kode di atas bertujuan untuk memvalidasi apakah lapangan yang dipilih oleh pengguna sudah dibooking pada waktu yang dipilih. Pertama, kode mencoba untuk mendapatkan ‘id_lapangan’ dari nama lapangan yang dipilih oleh pengguna di form, dengan melakukan query ke tabel ‘Ms_lapangan’. Nama lapangan yang dipilih (disimpan dalam variabel ‘lapangan’) digunakan sebagai parameter dalam query. Hasil dari query ini adalah ‘id_lapangan’ yang terkait dengan nama lapangan tersebut, yang kemudian disimpan dalam variabel ‘id_lapangan’. Ini memastikan bahwa sistem tahu lapangan mana yang dimaksud dalam langkah selanjutnya.

Selanjutnya, sistem akan memeriksa apakah ada booking yang sudah ada untuk lapangan tersebut pada rentang waktu yang dipilih. Kode ini melakukan query kedua ke tabel ‘Tr_Daftar_booking’, dengan memeriksa apakah waktu booking yang diinginkan (dimulai pada ‘waktu_booking’ dan berakhir pada ‘waktu_selesai’) bertabrakan dengan waktu booking yang sudah ada. Di sini, dilakukan pengecekan dengan 3 kondisi, yaitu apakah waktu booking yang baru berada di dalam rentang waktu booking yang sudah ada, atau apakah waktu booking yang sudah ada berada dalam rentang waktu booking yang baru. Jika ada booking yang bertabrakan, maka hasilnya adalah ‘existing_bookings’ yang berisi data booking yang sudah ada.

Jika ada booking yang bertabrakan (artinya ‘existing_bookings’ berisi data), maka sistem akan menampilkan pesan error menggunakan ‘messagebox.showerror()’ untuk memberitahu pengguna bahwa lapangan sudah dibooking pada rentang waktu yang dipilih. Jika tidak ada masalah, kode ini akan menutup koneksi ke database dan query, dan melanjutkan ke proses berikutnya. Jika terjadi kesalahan dalam proses query atau koneksi, maka akan muncul pesan error yang menampilkan pesan kesalahan yang sesuai,

yang membantu pengguna atau pengembang untuk memahami masalah yang terjadi.

g) Validasi pembooking di database

```
try:
    # JIKA NAMA PEMBOOKING SUDAH ADA DALAM DATABASE MAKA AKAN DIUPDATE NO_TELEPONNYA MENJADI NO TELEPON PEMBOOKING TERBARU
    cursor.execute("SELECT no_telepon FROM Ms_Pembooking WHERE nama_pembooking = %s", (nama,))
    pembooking_data = cursor.fetchone()
    if pembooking_data:
        cursor.execute("UPDATE Ms_Pembooking SET no_telepon = %s WHERE nama_pembooking = %s", (telepon,nama))

        cursor.execute("UPDATE Tr_Daftar_booking SET no_telepon = %s WHERE no_telepon = %s", (telepon, pembooking_data[0]))
        no_telepon = telepon
    else:
        # Tambahkan pembooking baru ke tabel ms_pembooking, jika tidak ada atau pembooking belum pernah membooking.

        cursor.execute("INSERT INTO Ms_Pembooking (nama_pembooking, no_telepon) VALUES (%s, %s)", (nama, telepon))
        no_telepon = telepon
    conn.commit()
except Exception as e:
    conn.rollback()
    messagebox.showerror("Error", f"Gagal menyimpan data pembooking: {e}")
    return
```

Potongan kode di atas berfungsi untuk memeriksa apakah pembooking dengan nama yang dimasukkan sudah ada di dalam database, dan jika ada, akan memperbarui nomor teleponnya. Pada langkah pertama, kode melakukan query untuk mencari nomor telepon pembooking berdasarkan nama yang diberikan. Jika nama pembooking sudah ada dalam database (hasil query tidak kosong), maka sistem akan memperbarui nomor telepon pembooking di tabel ‘Ms_Pembooking’ menggunakan query ‘UPDATE’. Selain itu, sistem juga memperbarui nomor telepon di tabel ‘Tr_Daftar_booking’, memastikan bahwa data nomor telepon yang baru juga diterapkan pada semua booking yang terkait dengan pembooking tersebut. Setelah itu, nomor telepon terbaru akan disalin ke variabel ‘no_telepon’.

Namun, jika nama pembooking belum terdaftar di dalam database (hasil query kosong), maka sistem akan menambahkan data

pembooking baru dengan nama dan nomor telepon yang diberikan menggunakan query ‘INSERT INTO’. Hal ini memastikan bahwa pembooking baru akan terdaftar dalam tabel ‘Ms_Pembooking’. Setelah perubahan atau penambahan data selesai, sistem melakukan ‘commit’ untuk menyimpan perubahan di database. Jika terjadi kesalahan selama proses ini, seperti kesalahan pada query atau masalah koneksi, maka transaksi akan dibatalkan menggunakan ‘rollback’, dan pesan kesalahan akan ditampilkan melalui ‘messagebox.showerror()’ untuk memberi tahu pengguna mengenai masalah yang terjadi.

h) Konfirmasi transaksi

```
confirm = messagebox.askyesno("Konfirmasi", f"Apakah Anda yakin ingin menambahkan transaksi booking? TOTAL BAYAR : {total_bayar}")
if not confirm:
        return
```

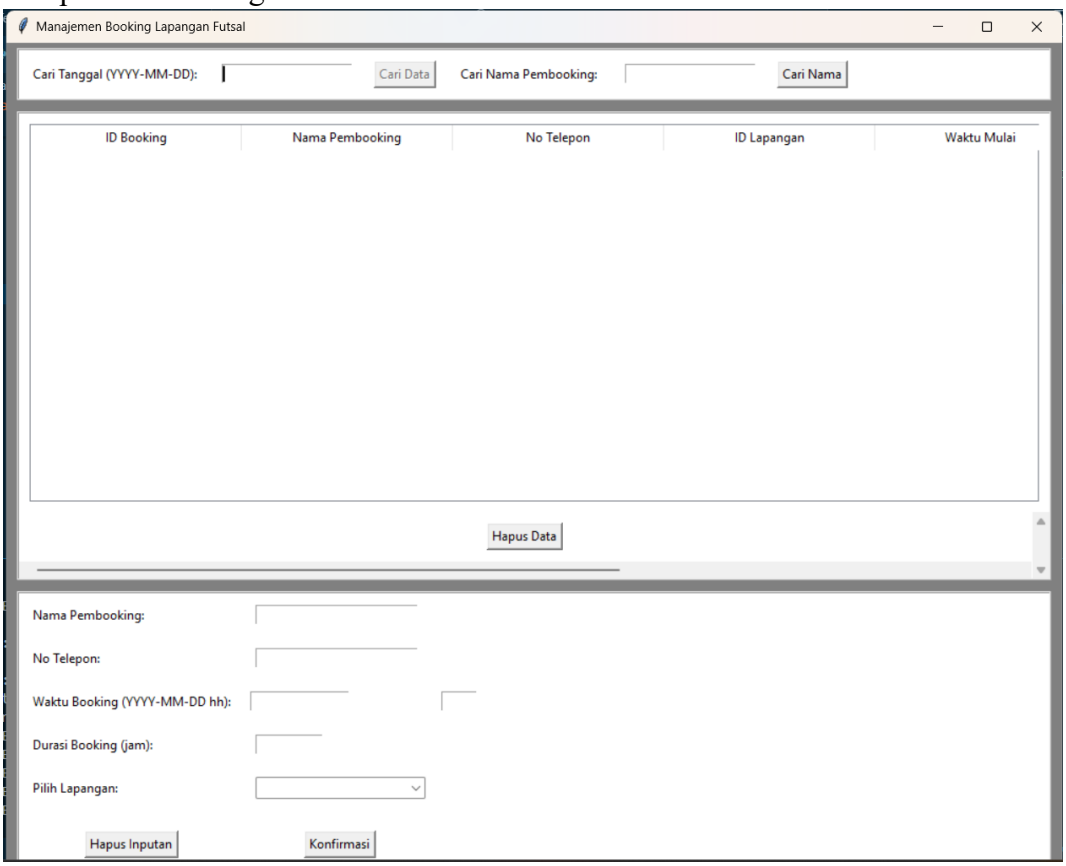
i) Menyimpan transaksi booking ke database

```
try:
        query = """
                INSERT INTO Tr_Daftar_booking (no_telepon, id_lapangan, waktu_mulai_booking, durasi_booking, waktu_selesai_booking, total_bayar)
                VALUES (%s, %s, %s, %s, %s, %s)
        """
        cursor.execute(query, (no_telepon, id_lapangan, waktu_booking, durasi, waktu_selesai, total_bayar))
        conn.commit()

        messagebox.showinfo("Sukses", "Transaksi berhasil ditambahkan!")
        self.clear_form()
 except Exception as e:
        conn.rollback()
        messagebox.showerror("Error", f"Gagal menyimpan transaksi booking: {e}")
  finally:
        cursor.close()
        conn.close()
```

4. Hasil Runinng Program

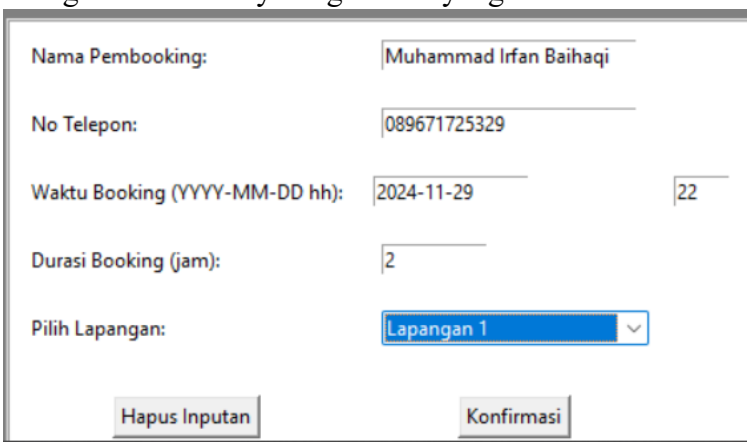
Tampilan Awal Program:



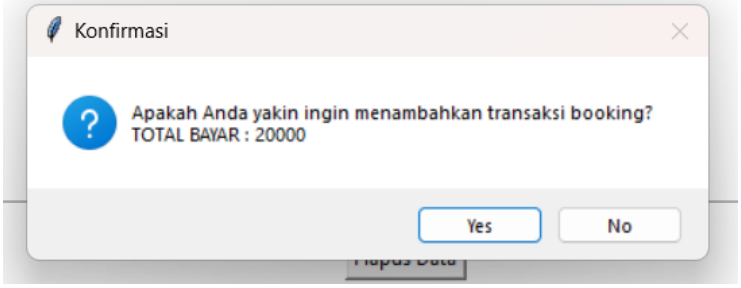
4.1 . Memasukkan Data Transaksi Baru

Langkah – Langkah :

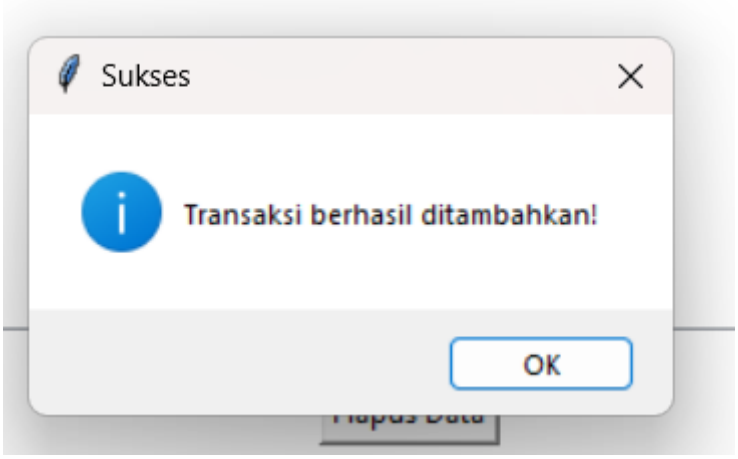
- Mengisi semua entry dengan data yang valid



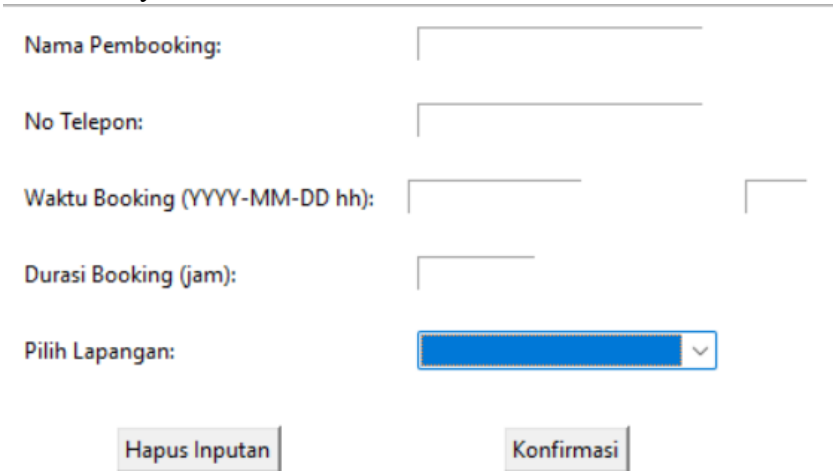
- Tekan Tombol Konfirmasi, Maka akan muncul messagebox askyesno untuk konfirmasi. Klik yes untuk melanjutkan



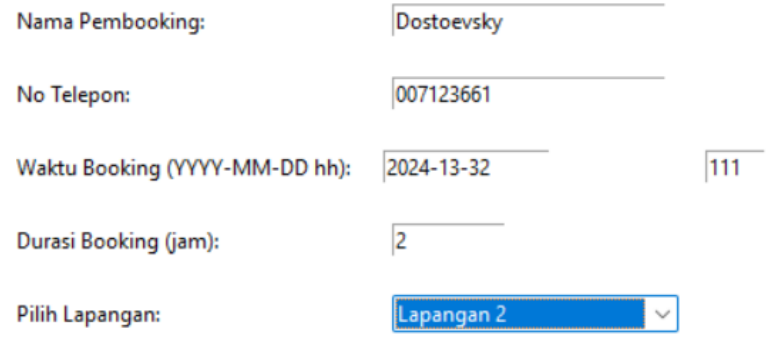
- Akan muncul messagebox pemberitahuan transaksi booking berhasil dilakukan



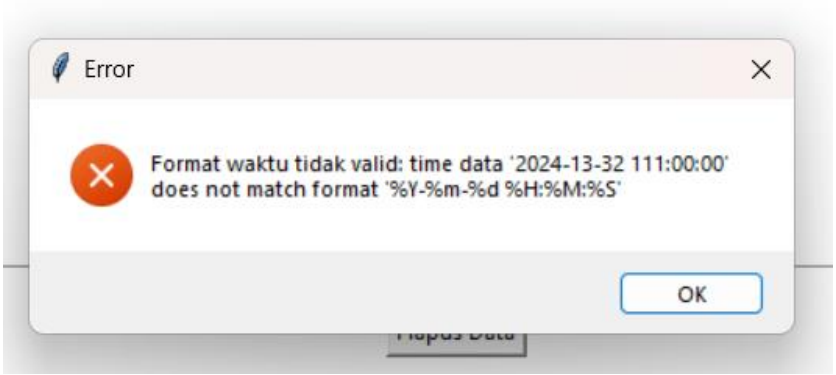
- Kotak entry akan otomatis dibersihkan



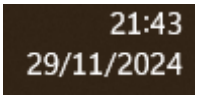
KASUS 1
WAKTU BOOKING TIDAK VALID

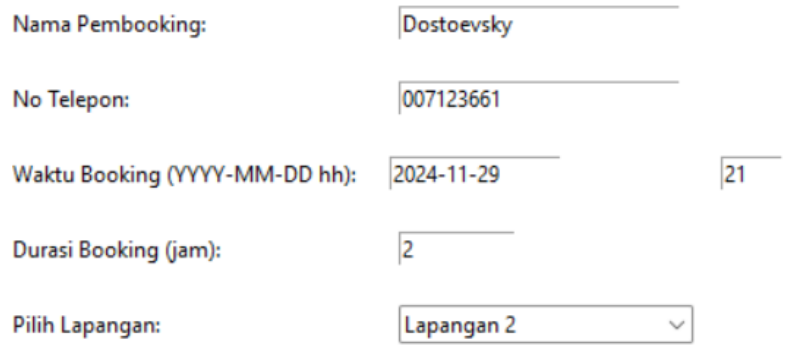


MAKA KETIKA MENEKAN TOMBOL KONFIRMASI SISTEM AKAN MENOLAK DAN MEMUNCULKAN MESSAGEBOX

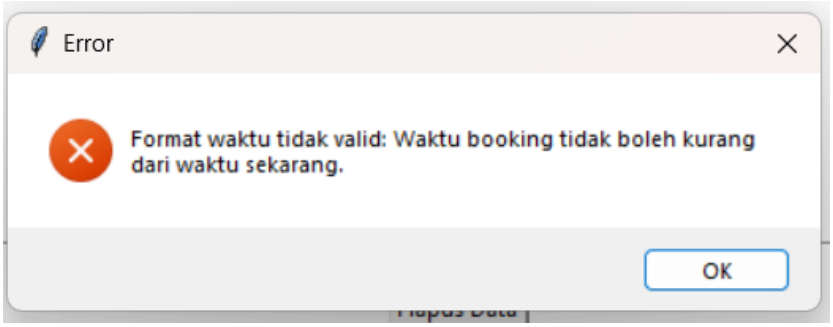


KASUS 2:
WAKTU BOOKING KURANG DARI WAKTU SAAT INI
Waktu saat ini:





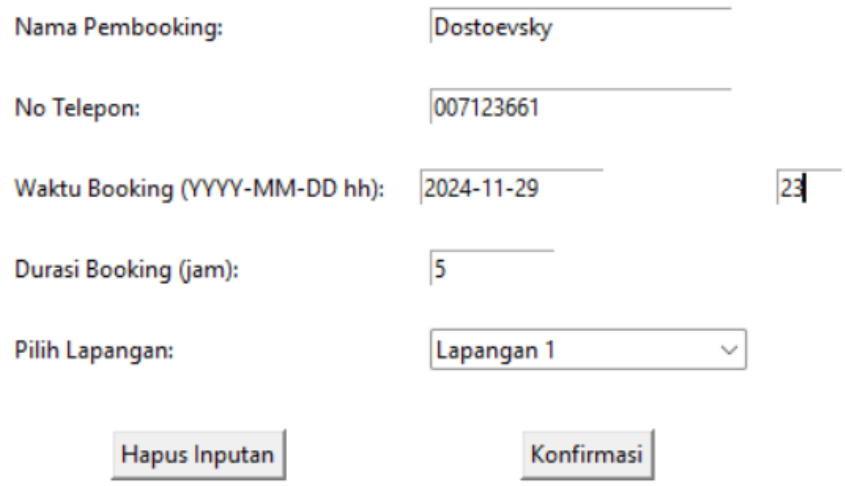
Dapat dilihat waktu (jam) kurang dari waktu saat ini
MAKA KETIKA MENEKAN TOMBOL KONFIRMASI SISTEM AKAN MENOLAK DAN MEMUNCULKAN MESSAGEBOX



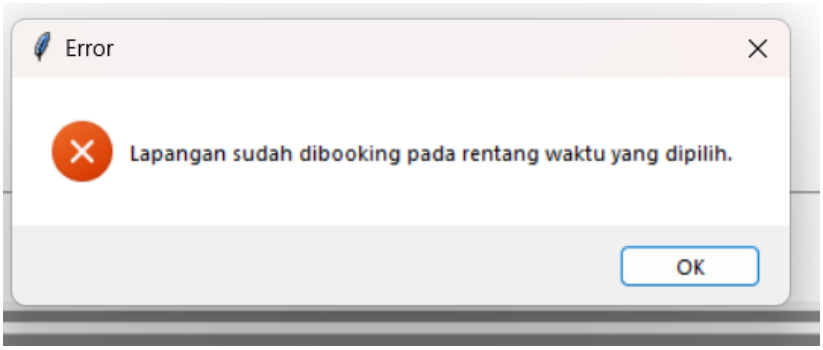
KASUS 3
JIKA WAKTU BOOKING DAN LAPANGAN SUDAH DIPESAN PEMBOOKING SEBELUMNYA

ID Lapangan	Waktu Mulai	Durasi	Waktu Selesai	To
1	2024-11-29 22:00:00	2	2024-11-29 23:59:59	

Dapat dilihat pada gambar bahwa lapangan 1 pada jam 10 malam sampai jam 12 sudah dipesan



Maka ketika ada pembooking baru yang membooking lapangan yang sama dan bertabrakan waktu dengan pembooking sebelumnya , maka sistem akan menolak ketika menekan tombol Konfirmasi.



4.2 . Mencari Data Transaksi Booking Berdasarkan Tanggal

Langkah – Langkah:

- Memasukkan tanggal yang valid di kotak entry search tanggal (harus valid agar tombol search bisa dipencet). Lalu tekan tombol Cari Data

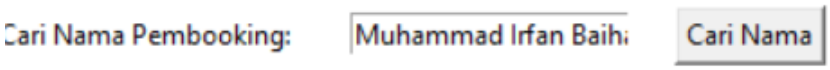
- Maka akan muncul data transaksi booking yang sesuai tanggal yang dimasukkan.

Booking	Nama Pembooking	No Telepon	ID Lapangan	Waktu Mulai	Durasi	Waktu Selesai	Total Bayar
19	muliani	123	2	2024-12-12 12:00:00	3	2024-12-12 14:59:59	30000
25	muliono	085228479111	3	2024-12-12 12:00:00	5	2024-12-12 16:59:59	50000

4.3 . Mencari Data Transaksi Baru Berdasarkan Nama

Langkah – Langkah :

- Masukkan data pada kotak entry search Nama Pembooking. Lalu tekan tombol Cari Nama.



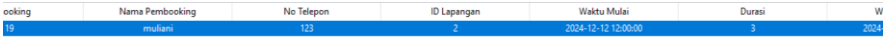
- Maka akan muncul data transaksi yang sesuai dengan nama yang dicari.

4.4 . Menghapus Data Transaksi

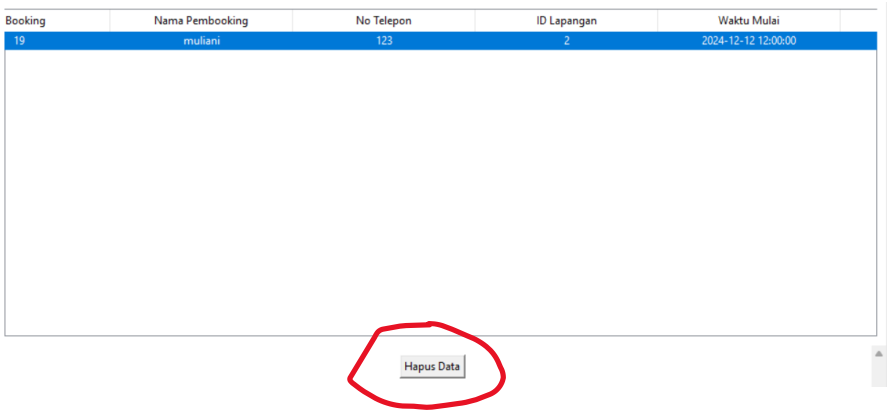
Langkah – Langkah :

- Cari data yang akan dihapus (bisa menggunakan search tanggal atau nama pembooking). Kemudian data akan tampil di tabel.

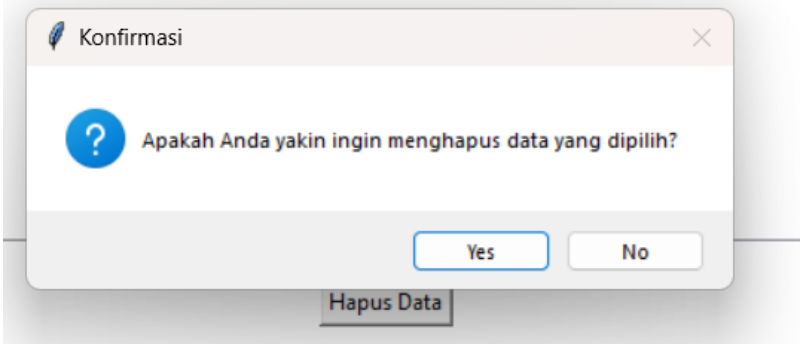
- Select salah satu data yang akan dihapus, dengan mengeklik data yang ada di tabel. Jika sudah berwarna biru, berarti sudah berhasil deselect.



- Klik tombol Hapus Data



- Maka program akan menampilkan messagebox askyesno. Jika sudah yakin untuk menghapus data yang dpilih, klik “YES”



- Akan muncul messagebox pemberitahuan bahwa data berhasil dihapus.