LAPORAN UAS
SISTEM BOOKING LAPANGAN

Mata Kuliah :

Pemrograman Berbasis Objek (PBO)

Oleh :

Nabil Fajar Ilham

24091397061

2024B

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2025

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Dalam upaya mengatasi inefisiensi dan kerentanan double-booking yang sering terjadi pada sistem reservasi lapangan olahraga konvensional, proyek "Aplikasi Booking Lapangan Olahraga" dikembangkan menggunakan Python dan Tkinter. Proyek ini bertujuan menyediakan solusi digital berbasis antarmuka grafis yang memungkinkan pengguna melihat ketersediaan slot lapangan futsal dan basket secara real-time dan akurat. Dengan mengimplementasikan Pemrograman Berorientasi Objek (OOP) melalui kelas-kelas spesifik (Futsal, Basket, Booking), aplikasi ini mampu menghitung biaya secara dinamis dan memvalidasi bentrokan jadwal secara otomatis. Aplikasi ini juga dilengkapi dengan mekanisme kontrol akses ganda (Mode USER untuk melihat jadwal dan Mode ADMIN untuk booking manual serta pembatalan), memastikan sistem yang transparan, terkelola, dan efisien bagi pengelola dan pelanggan.

1.2 Tujuan Proyek

- Tujuan Umum

  Tujuan umum proyek ini adalah untuk mendigitalisasi dan memodernisasi proses reservasi lapangan olahraga (futsal dan basket) yang selama ini bersifat manual. Tujuannya adalah menciptakan sebuah sistem yang menyediakan visibilitas jadwal real-time, meningkatkan akurasi booking, serta menyediakan platform yang terkelola dengan baik bagi pengelola dan transparan bagi pelanggan.

- Tujuan Spesifik
  - Mengimplementasikan struktur Pemrograman Berorientasi Objek (OOP) untuk model lapangan (Futsal, Basket) dan proses booking.
  - Mengembangkan antarmuka grafis (GUI) menggunakan Tkinter untuk visualisasi matriks ketersediaan per jam.
  - Memastikan validasi ketersediaan slot yang ketat untuk mencegah double-booking.
  - Menerapkan kontrol akses berbasis peran (USER dan ADMIN) dengan fungsionalitas yang berbeda.
  - Menghitung total biaya secara otomatis berdasarkan durasi dan karakteristik spesifik setiap jenis lapangan.

1.3 Ruang Lingkup Proyek

Fungsionalitas Inti :

- Manajemen Lapangan Berbasis OOP (Lapangan, Futsal, Basket).
- Visualisasi jadwal real-time slot waktu per jam (matriks).
- Validasi booking yang mencegah bentrokan jadwal (double-booking).
- Akses Berbasis Peran (USER dan ADMIN) dengan hak akses berbeda.
- Fungsi booking manual dan pembatalan reservasi (khusus ADMIN).
- Penghitungan biaya otomatis berdasarkan jenis lapangan, durasi, dan karakteristik.

Batasan Teknis & Non-Fungsional :

- Aplikasi dikembangkan sebagai Desktop Application menggunakan Python dan Tkinter.
- Penyimpanan data bersifat simulasi dalam memori (variabel global).
- Aplikasi tidak menggunakan database persisten eksternal.
- Lingkungan single-user tidak menangani akses multi-pengguna (concurrency).
- Proses transaksi/pembayaran hanya disimulasikan; tanpa integrasi gateway pembayaran nyata.

1.4 Metodologi Pengembangan

Metodologi yang digunakan dalam pengembangan aplikasi booking lapangan olahraga ini adalah Model Waterfall (Air Terjun) yang bersifat sekuensial dan sistematis. Proses pengembangan dibagi menjadi tahapan-tahapan yang jelas, di mana setiap tahapan harus diselesaikan sebelum beralih ke tahapan berikutnya.

A. Tahap Perencanaan (Planning) :

   Tahap ini berfokus pada penentuan cakupan dan kebutuhan fungsional aplikasi.

   - Identifikasi Kebutuhan: Menentukan fungsi utama aplikasi (booking, admin, perhitungan harga), peran pengguna (USER, ADMIN), dan batasan proyek.
   - Analisis Sistem: Menganalisis kebutuhan data (daftar lapangan, data booking), logika validasi slot, dan aturan bisnis (struktur harga lapangan futsal vs. basket).

- Perancangan Struktur OOP: Membuat desain kelas-kelas utama (Lapangan, Futsal, Basket, Booking) beserta atribut dan metode yang akan diimplementasikan.

B. Tahap Perancangan (Design) :

Tahap ini menerjemahkan kebutuhan ke dalam representasi yang dapat diprogram.

- Perancangan Arsitektur Aplikasi: Menentukan bagaimana komponen data (variabel global) dan GUI (Tkinter) akan saling berinteraksi.
- Perancangan Antarmuka (GUI Design): Merancang layout antarmuka pengguna menggunakan Tkinter, termasuk tata letak tombol mode, filter, dan matriks visualisasi slot waktu.
- Perancangan Logika Program: Membuat pseudocode atau flowchar*t* untuk fungsi kritis, seperti:
  - Logika hitung_biaya() dalam kelas turunan.
  - Logika cek_ketersediaan() dalam kelas Booking.
  - Logika login dan penentuan peran aktif.

C. Tahap Implementasi (Coding) :

Tahap ini adalah realisasi dari hasil perancangan menjadi kode program.

- Penyusunan Kode Dasar (OOP): Mengembangkan semua kelas (termasuk inheritance dan polymorphism) dan metode intinya dalam bahasa Python.
- Implementasi GUI: Menerjemahkan desain wireframe menjadi widget Tkinter (Frame, Button, Label, Entry) dan menghubungkannya dengan fungsi-fungsi utama.
- Integrasi Fungsionalitas: Menghubungkan logika booking (OOP) dengan event handler tombol-tombol pada GUI, termasuk pembaruan tampilan slot secara dinamis.

D. Tahap Pengujian (Testing) :

Tahap ini bertujuan untuk memastikan aplikasi berfungsi sesuai dengan kebutuhan dan bebas dari bug.

- Uji Unit: Menguji setiap unit kode (metode dan fungsi) secara terpisah, misalnya: menguji akurasi hitung_biaya() untuk Futsal Premium dan Basket Outdoor.
- Uji Integrasi: Memastikan bahwa komponen-komponen yang berbeda dapat bekerja sama (misalnya, tombol booking pada GUI berhasil memanggil metode simpan_booking() dan memperbarui tampilan).
- Uji Fungsionalitas: Menguji skenario end-to-end utama, seperti: login ADMIN, booking multi-jam, pembatalan booking, dan validasi kegagalan booking karena bentrok.

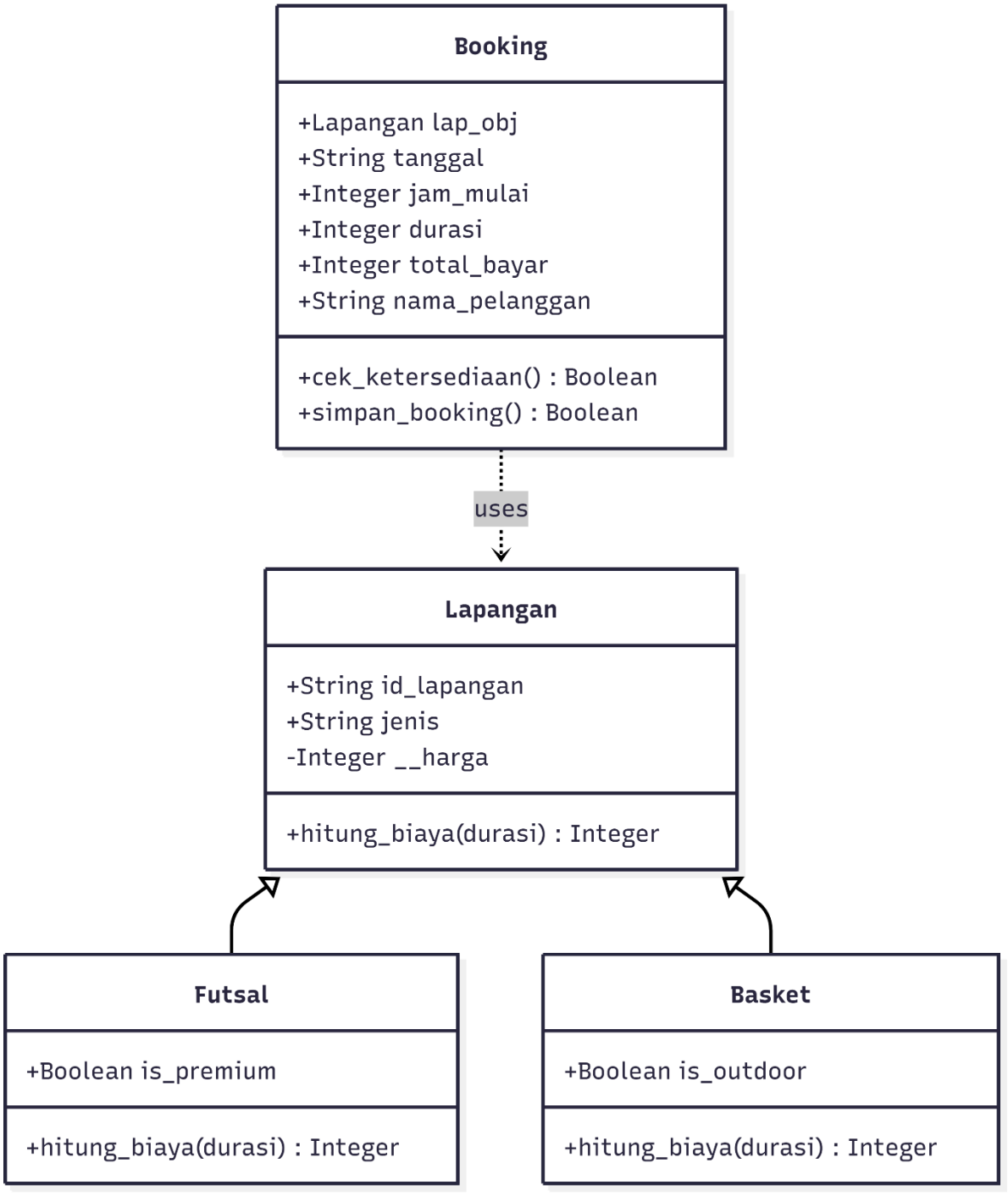
E. Tahap Dokumentasi :

Dokumentasi dilakukan sepanjang proses dan diselesaikan di akhir proyek.

- Dokumentasi Kode: Memberikan komentar yang jelas pada kode Python untuk menjelaskan logika dan alur kerja.
- Dokumentasi Teknis: Menyusun laporan proyek yang mencakup Latar Belakang, Tujuan, Ruang Lingkup, Metodologi, dan Hasil Pengujian.

BAB II PERANCANGAN APLIKASI

2.1 Diagram Class & Penjelasan



Penjelasan :

Arsitektur aplikasi booking lapangan ini dirancang menggunakan prinsip Pemrograman Berorientasi Objek (OOP), yang diilustrasikan melalui empat kelas utama yang saling berelasi. Sistem ini dibangun di atas fondasi Kelas Lapangan, yang berfungsi sebagai kelas

induk (parent class). Kelas ini menetapkan atribut esensial seperti id_lapangan, jenis, dan __harga dasar per jam, yang terakhir dienkapsulasi sebagai private. Kelas Lapangan juga memperkenalkan metode hitung_biaya(durasi) sebagai kontrak dasar untuk kalkulasi harga.

Selanjutnya, prinsip Pewarisan (Inheritance) diimplementasikan melalui Kelas Futsal dan Kelas Basket, yang merupakan turunan dari Lapangan. Kedua kelas turunan ini memanfaatkan Polymorphism dengan menimpa (override) metode hitung_biaya(durasi) milik induk. Kelas Futsal menambahkan atribut is_premium dan menyisipkan logika untuk mengenakan biaya tambahan (surcharge) per jam jika lapangan premium. Sementara itu, Kelas Basket memperkenalkan atribut is_outdoor dan memodifikasi kalkulasi untuk memberikan diskon atau pengurangan biaya tertentu per jam, merefleksikan dinamika harga berdasarkan kondisi lapangan.

Peran fungsional kritis dipegang oleh Kelas Booking. Kelas ini menunjukkan konsep Asosiasi karena ia memiliki referensi ke objek Lapangan melalui atribut lap_obj. Ketergantungan ini memungkinkan objek Booking mengakses harga polimorfis dan ID lapangan terkait. Tanggung jawab utama kelas Booking adalah memvalidasi integritas jadwal melalui metode cek_ketersediaan() untuk mencegah double-booking serta mencatat reservasi yang sah ke sistem melalui metode simpan_booking(). Secara keseluruhan, struktur ini menghasilkan kode yang modular, fleksibel dalam penyesuaian harga, dan efektif dalam pengelolaan jadwal.

2.2 Alur Aplikasi

Aplikasi booking lapangan ini memulai operasinya dengan memuat data simulasi global, kemudian langsung menampilkan antarmuka utama yang secara default berada dalam Mode USER. Pada tahap inisiasi ini, pengguna disajikan dengan matriks slot waktu lapangan Futsal (sebagai jenis default) pada tanggal hari ini. Pengguna dapat dengan mudah berpindah pandangan antara lapangan Futsal dan Basket menggunakan tombol filter yang tersedia, di mana setiap pergantian akan langsung memuat dan merefresh tampilan slot waktu yang relevan.

Interaksi utama terjadi pada pemilihan mode akses. Pengguna dapat memilih untuk tetap di Mode USER—di mana mereka hanya bisa melihat status ketersediaan (tersedia berwarna hijau, terisi berwarna abu-abu, dan tombol booking dinonaktifkan)—atau memilih tombol ADMIN. Jika memilih ADMIN, sistem akan meminta login sederhana.

Setelah login berhasil, tampilan beralih ke Mode ADMIN, di mana slot yang tersedia kini diberi label "Book Man." (kuning) dan dapat diklik, sementara slot yang terisi memungkinkan aksi pembatalan (Cancel).

Ketika seorang Administrator memilih slot yang tersedia dan menekan tombol "Book Man.", sistem memicu alur proses booking baru. Sebuah jendela pop-up muncul untuk mengumpulkan detail reservasi, yaitu nama pelanggan dan durasi sewa yang diinginkan. Setelah data diinput, sistem segera membuat objek Booking sementara dan menjalankan validasi ganda yang kritis: pertama, memanggil logika di dalam kelas Booking untuk memastikan waktu selesai tidak melebihi batas operasional (Jam 23:00); dan kedua, menjalankan metode cek_ketersediaan() untuk memastikan tidak ada bentrokan waktu dengan reservasi lain yang sudah tercatat. Jika validasi berhasil, sistem menghitung total biaya menggunakan metode polimorfik dari objek lapangan terkait, menyimpan detail booking ke data global, menampilkan pesan sukses, dan secara otomatis merefresh tampilan matriks slot untuk merefleksikan perubahan jadwal. Jika validasi gagal (terjadi bentrok atau jam invalid), sistem akan menampilkan pesan error tanpa menyimpan booking, memastikan integritas jadwal lapangan tetap terjaga.

2.3 Perancangan Penyimpanan Data

Perancangan penyimpanan data pada aplikasi ini didasarkan pada prinsip simulasi karena keterbatasan proyek yang tidak menggunakan database persisten, melainkan memanfaatkan variabel global dalam memori program. Penyimpanan dibagi menjadi tiga kategori utama: data master, data transaksional, dan data akses. Untuk data master, yaitu informasi mengenai lapangan, sistem menggunakan variabel global DAFTAR_LAPANGAN yang berjenis List dan menyimpan objek-objek kelas turunan (Futsal atau Basket). Penyimpanan sebagai objek ini sangat efisien karena setiap item membawa logika dan karakteristik harganya sendiri (misalnya, status premium atau outdoor), sesuai dengan implementasi OOP.

Selanjutnya, untuk data transaksional yang dinamis, digunakan variabel global LIST_BOOKING_SEMUA, yang bertugas mencatat semua reservasi yang berhasil disimpan. Variabel ini juga berjenis List, namun setiap elemen di dalamnya adalah Dictionary yang merepresentasikan satu catatan booking. Struktur Dictionary ini sengaja dirancang untuk mencakup detail penting seperti id_lapangan, tanggal, jam_mulai, dan jam_selesai. Kehadiran jam_selesai secara eksplisit sangat vital, karena memungkinkan

fungsi validasi (cek_ketersediaan) untuk melakukan iterasi loop yang akurat pada setiap jam di antara rentang waktu booking untuk mencegah adanya bentrokan jadwal.

Terakhir, data akses yang sederhana, yang terdiri dari username dan password ADMIN, disimpan sebagai string sederhana (hardcoded) dalam variabel global terpisah (ADMIN_USERNAME dan ADMIN_PASSWORD). Data ini, bersama dengan variabel status mode aplikasi (PERAN_AKTIF), hanya berfungsi untuk menyimulasikan perbedaan fungsionalitas dan hak akses antara Mode USER dan Mode ADMIN, menjadikannya pelengkap yang diperlukan untuk alur kerja aplikasi secara keseluruhan.

BAB III IMPLEMENTASI KONSEP OOP

3.1 Implementasi Encapsulation

Konsep Enkapsulasi diterapkan untuk melindungi data penting pada Lapangan, yaitu harga dasar per jam. Dalam sistem ini, enkapsulasi diimplementasikan pada class Lapangan, di mana atribut harga (__harga) tidak diakses langsung dari luar kelas. Perubahan atau perhitungan total biaya hanya dapat dilakukan melalui metode tertentu seperti hitung_biaya.

```
11  class Lapangan:
12      def __init__(self, id_lapangan, jenis, harga):
13          self.id_lapangan, self.jenis, self.__harga = id_lapangan, jenis, harga
14      # Harga/jam murni harga dasar
15      def hitung_biaya(self, durasi): return self.__harga * durasi
```

Pada kode di atas:

- Atribut __harga bersifat internal/dilindungi (menggunakan double underscore), sehingga tidak dapat dimodifikasi langsung dari luar kelas.
- Akses terhadap data tersebut (untuk perhitungan total) dilakukan melalui metode publik hitung_biaya(self, durasi).

3.2 Implementasi Inheritance

Konsep Inheritance diterapkan untuk membuat hierarki Lapangan yang spesifik (Futsal dan Basket) berdasarkan Lapangan umum. Ini memungkinkan penggunaan ulang properti dan metode dasar dari kelas induk.

```
17  class Futsal(Lapangan):
18      def __init__(self, id, harga, premium):
19          super().__init__(id, "Futsal", harga); self.is_premium = premium
20      # Menggunakan hitung_biaya dari class Lapangan (Tanpa tambahan/pengurangan)
21      def hitung_biaya(self, durasi):
22          return super().hitung_biaya(durasi)
23
24  class Basket(Lapangan):
25      def __init__(self, id, harga, outdoor):
26          super().__init__(id, "Basket", harga); self.is_outdoor = outdoor
27      # Menggunakan hitung_biaya dari class Lapangan (Tanpa tambahan/pengurangan)
28      def hitung_biaya(self, durasi):
29          return super().hitung_biaya(durasi)
```

Pada kode di atas:

- Futsal dan Basket mewarisi (inherit) dari Lapangan.

- Pemanggilan super().__init__(...) memastikan bahwa properti dasar seperti ID dan harga diinisialisasi oleh kelas induk (Lapangan), sehingga kode lebih efisien.

3.3 Implementasi Polymorphism

Konsep Polymorphism diterapkan melalui *Method Overriding* pada metode hitung_biaya. Hal ini memungkinkan kelas Booking memanggil satu metode yang sama, dan sistem secara otomatis menjalankan logika perhitungan yang benar, baik untuk objek Futsal maupun Basket.

```
17  class Futsal(Lapangan):
18      def __init__(self, id, harga, premium):
19          super().__init__(id, "Futsal", harga); self.is_premium = premium
20      # Menggunakan hitung_biaya dari class Lapangan (Tanpa tambahan/pengurangan)
21      def hitung_biaya(self, durasi):
22          return super().hitung_biaya(durasi)
23
24  class Basket(Lapangan):
25      def __init__(self, id, harga, outdoor):
26          super().__init__(id, "Basket", harga); self.is_outdoor = outdoor
27      # Menggunakan hitung_biaya dari class Lapangan (Tanpa tambahan/pengurangan)
28      def hitung_biaya(self, durasi):
29          return super().hitung_biaya(durasi)
30
31  class Booking:
32      def __init__(self, lap_obj, tgl, mulai, durasi, pelanggan):
33          self.lap_obj, self.tanggal = lap_obj, tgl
34          self.jam_mulai, self.durasi = mulai, durasi
35          self.jam_selesai = mulai + durasi
36          self.total_bayar = lap_obj.hitung_biaya(durasi)
37          self.nama_pelanggan = pelanggan
```

Pada kode di atas:

- Kedua subkelas meng-override hitung_biaya, menciptakan potensi untuk implementasi biaya yang unik.
- Di dalam class Booking, baris 36 (lap_obj.hitung_biaya(durasi)) adalah panggilan polimorfis. Kelas Booking tidak perlu tahu apakah lap_obj adalah Futsal atau Basket; ia hanya memanggil metode tersebut, dan implementasi yang benar akan dijalankan.

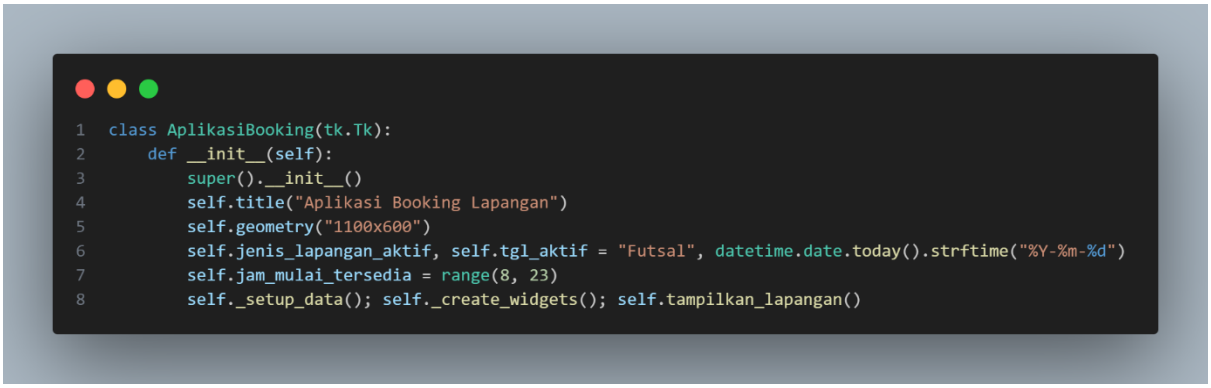
3.4 Abstraksi

Menyembunyikan detail implementasi yang kompleks, seperti proses validasi slot dan pembangunan GUI, dari pengguna kelas.

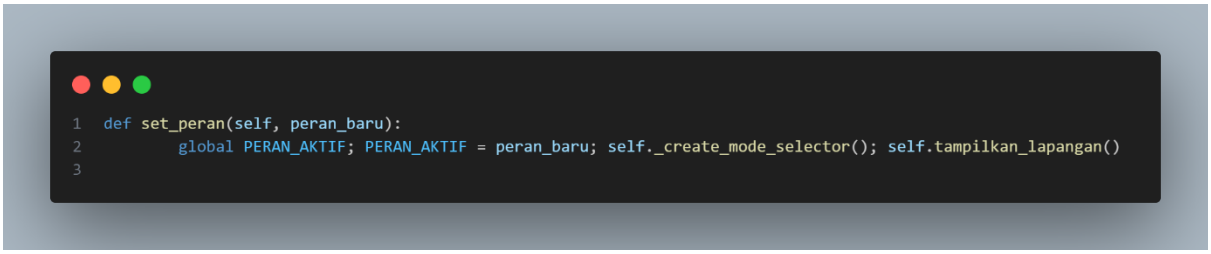
```
class Booking:
    def __init__(self, lap_obj, tgl, mulai, durasi, pelanggan):
        self.lap_obj, self.tanggal = lap_obj, tgl
        self.jam_mulai, self.durasi = mulai, durasi
        self.jam_selesai = mulai + durasi
        self.total_bayar = lap_obj.hitung_biaya(durasi)
        self.nama_pelanggan = pelanggan

    def cek_ketersediaan(self):
        if self.jam_selesai > 23: return False
        for j in range(self.jam_mulai, self.jam_selesai):
            for b in LIST_BOOKING_SEMUA:
                if (b["id_lapangan"] == self.lap_obj.id_lapangan and b["tanggal"] == self.tanggal):
                    if (j >= b["jam_mulai"] and j < b["jam_selesai"]): return False
        return True

    def simpan_booking(self):
        if self.cek_ketersediaan():
            LIST_BOOKING_SEMUA.append({"id_lapangan": self.lap_obj.id_lapangan, "tanggal": self.tanggal, "jam_mulai": self.jam_mulai, "jam_selesai": self.jam_selesai, "durasi": self.durasi, "total_bayar": self.total_bayar, "pelanggan": self.nama_pelanggan })
            return True
        return False
```



```
class AplikasiBooking(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Aplikasi Booking Lapangan")
        self.geometry("1100x600")
        self.jenis_lapangan_aktif, self.tgl_aktif = "Futsal", datetime.date.today().strftime("%Y-%m-%d")
        self.jam_mulai_tersedia = range(8, 23)
        self._setup_data(); self._create_widgets(); self.tampilkan_lapangan()
```



```
def set_peran(self, peran_baru):
        global PERAN_AKTIF; PERAN_AKTIF = peran_baru; self._create_mode_selector(); self.tampilkan_lapangan()

```

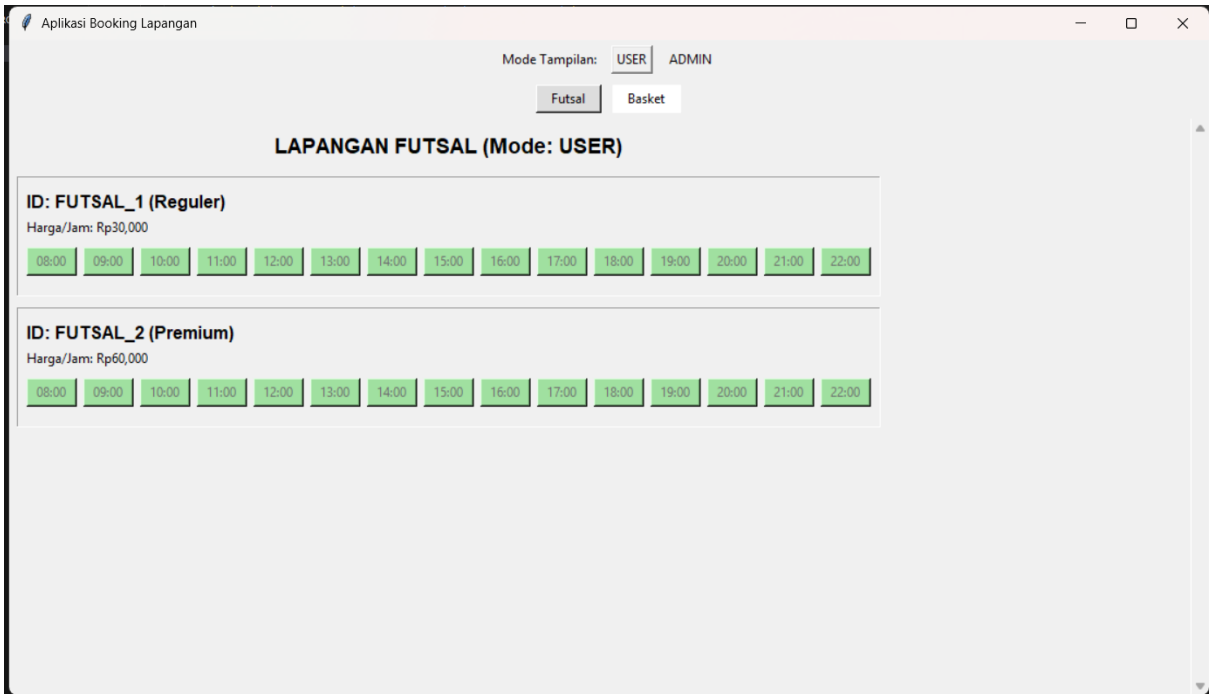
Penjelasan:

- Logika Booking : Pemrogram hanya perlu memanggil booking_baru.simpan_booking() dan mendapatkan True/False. Semua perulangan dan pengecekan bentrok jam diabstraksi di balik metode ini.
- Setup GUI : Panggilan ke _setup_data(), _create_widgets(), dan tampilkan_lapangan() memungkinkan programmer memahami alur inisialisasi aplikasi tanpa harus tahu detail loop atau widget Tkinter yang digunakan di dalamnya.

BAB IV HASIL & IMPLEMENTASI

4.1 Hasil Implementasi Aplikasi

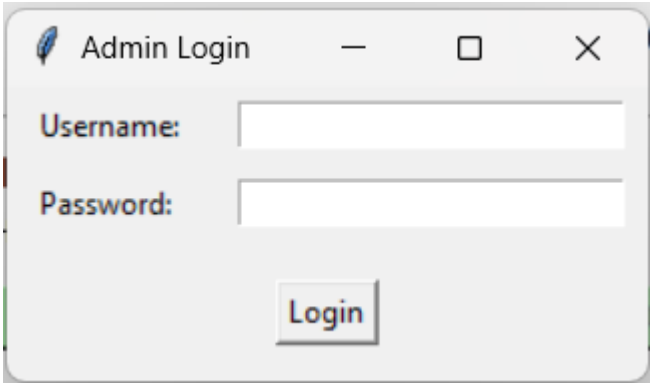
Implementasi aplikasi Booking Lapangan Berbasis Object-Oriented Programming (OOP) telah berhasil diselesaikan menggunakan bahasa pemrograman Python dengan pustaka Tkinter sebagai antarmuka grafis (GUI). Aplikasi berjalan sesuai dengan perencanaan dan mampu mengelola seluruh fungsionalitas utama, yang meliputi: menampilkan ketersediaan slot waktu lapangan Futsal dan Basket, memproses booking manual oleh administrator, pembatalan/penguncian slot, serta pemfilteran dan penampilan riwayat booking harian. Secara struktural, proyek ini sukses menerapkan pilar-pilar OOP termasuk Encapsulation untuk melindungi data harga dasar, Inheritance untuk membuat hierarki Lapangan spesifik, Polymorphism untuk perhitungan biaya, dan Abstraction untuk menyembunyikan detail validasi slot dan pembangunan widget Tkinter dalam sebuah aplikasi GUI yang fungsional. Seluruh fitur administrasi dan user view dapat diakses secara terstruktur setelah pengguna memilih mode peran yang sesuai (Admin atau User).



4.2 Pembahasan Fitur Aplikasi

4.2.1 Fitur Login Admin

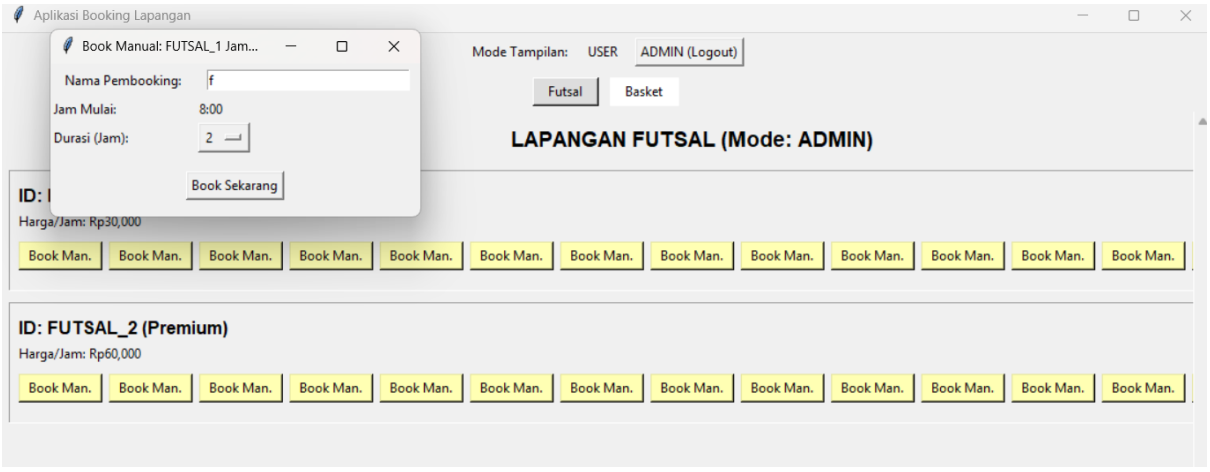
Fitur ini mengamankan akses ke mode administratif yang memiliki kontrol penuh atas booking dan manajemen data.

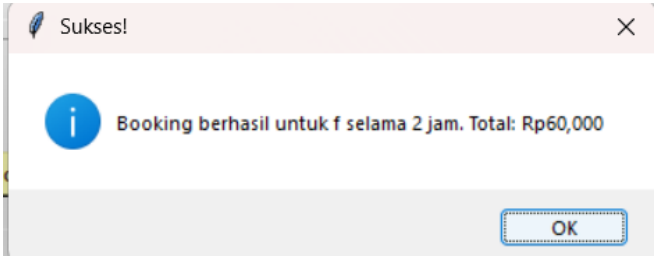


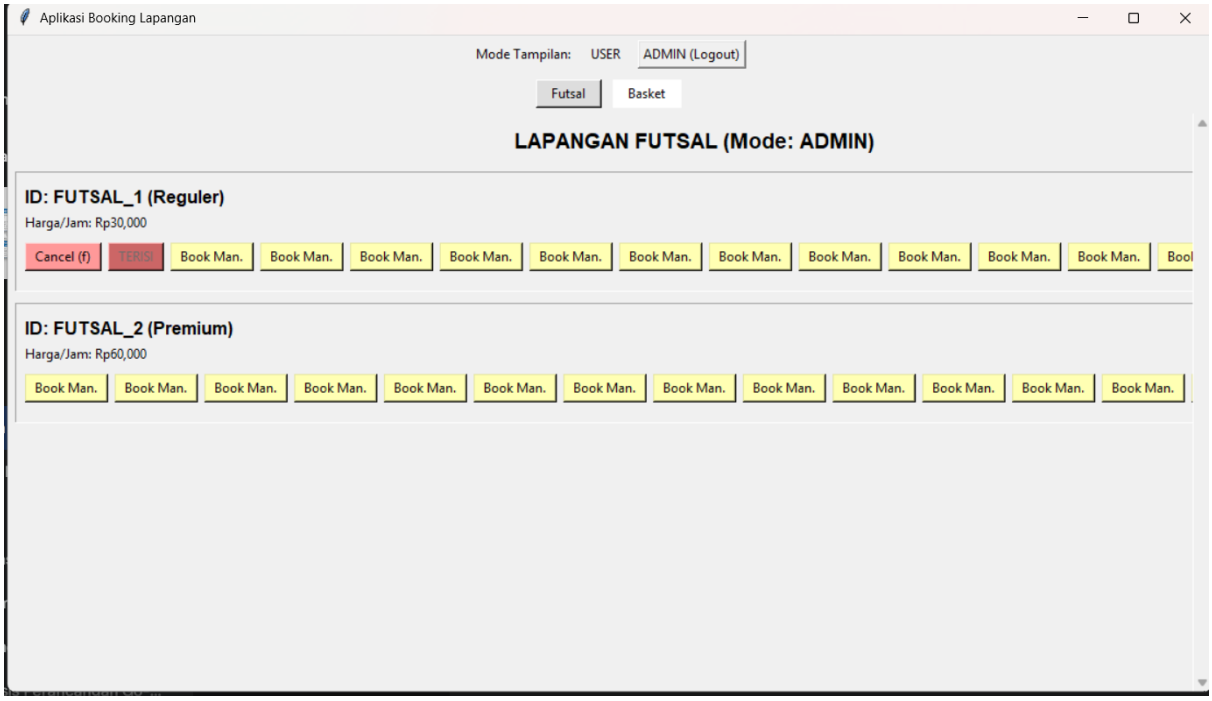
- Fungsi: Mengautentikasi pengguna sebagai Administrator untuk membuka fitur booking manual, pembatalan, dan tampilan data raw.
- Mekanisme: Menggunakan password statis (ADMIN_PASSWORD = "admin, 123") yang dicek saat pengguna mencoba beralih ke mode Admin. Jika otentikasi berhasil, variabel global PERAN_AKTIF diubah menjadi "ADMIN".

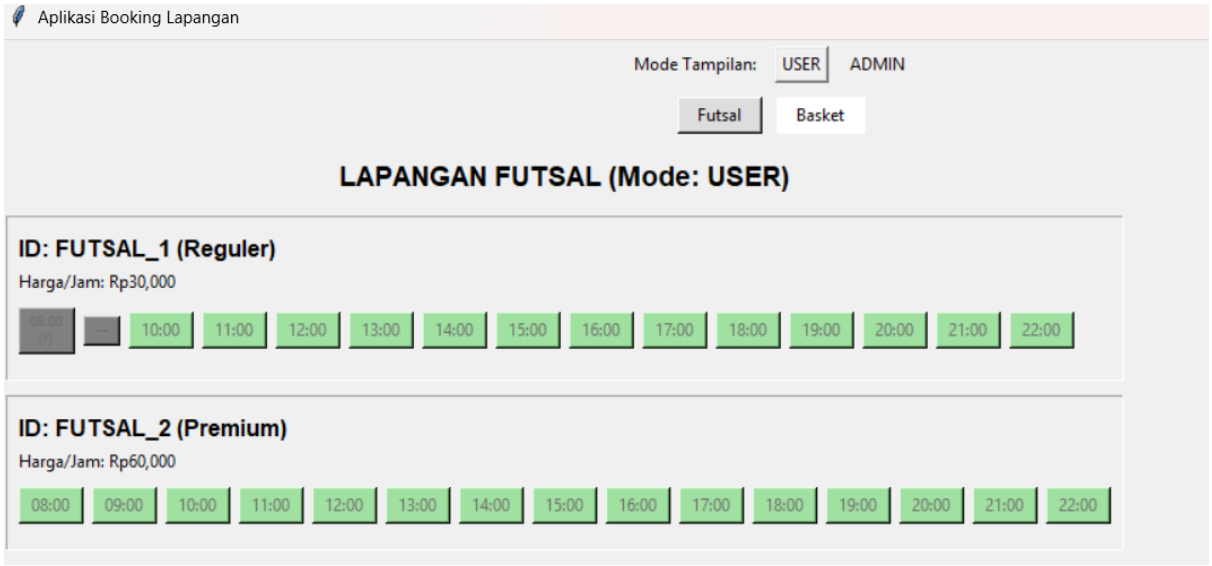
4.2.2 Fitur Booking

Fitur ini memungkinkan Admin untuk mencatatkan pemesanan pada slot waktu yang masih tersedia.





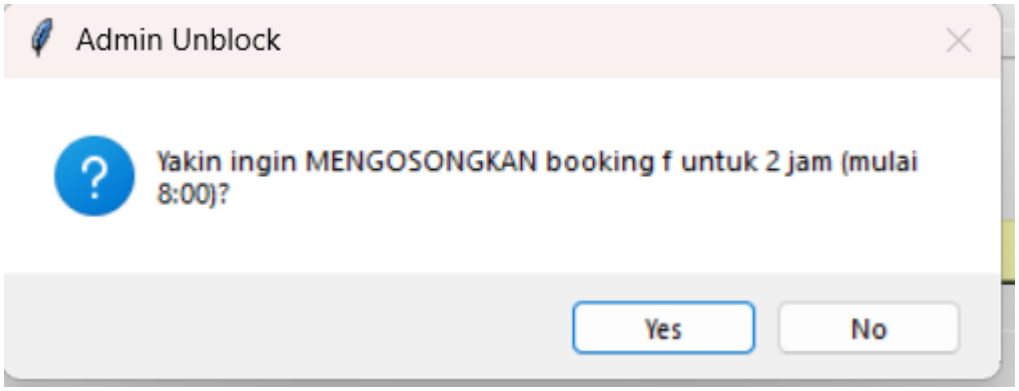


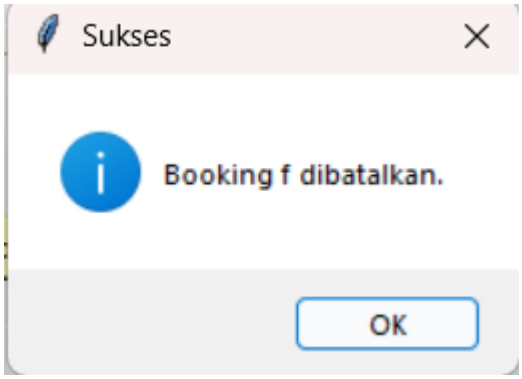


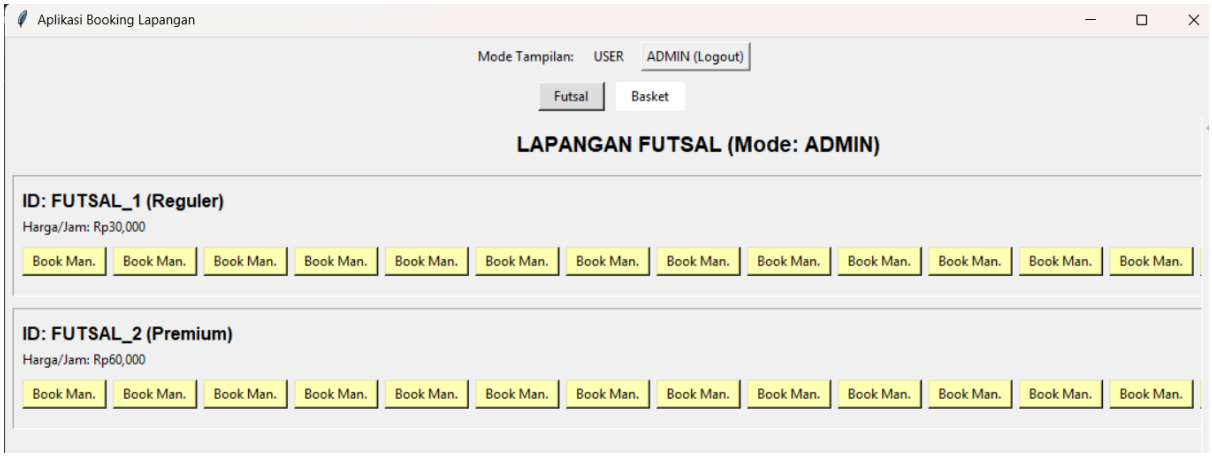
- Fungsi: Memungkinkan Admin untuk memasukkan detail booking (Nama Pelanggan, Durasi) pada slot yang belum terisi.
- Mekanisme OOP:
    - Data booking dienkapsulasi ke dalam objek Booking baru.
    - Objek ini memanggil lap_obj.hitung_biaya(durasi) (Polymorphism) untuk menentukan total bayar.
    - Objek memanggil simpan_booking() (Abstraction), yang secara internal menjalankan validasi ketersediaan.
    - Jika valid, data ditambahkan ke LIST_BOOKING_SEMUA.

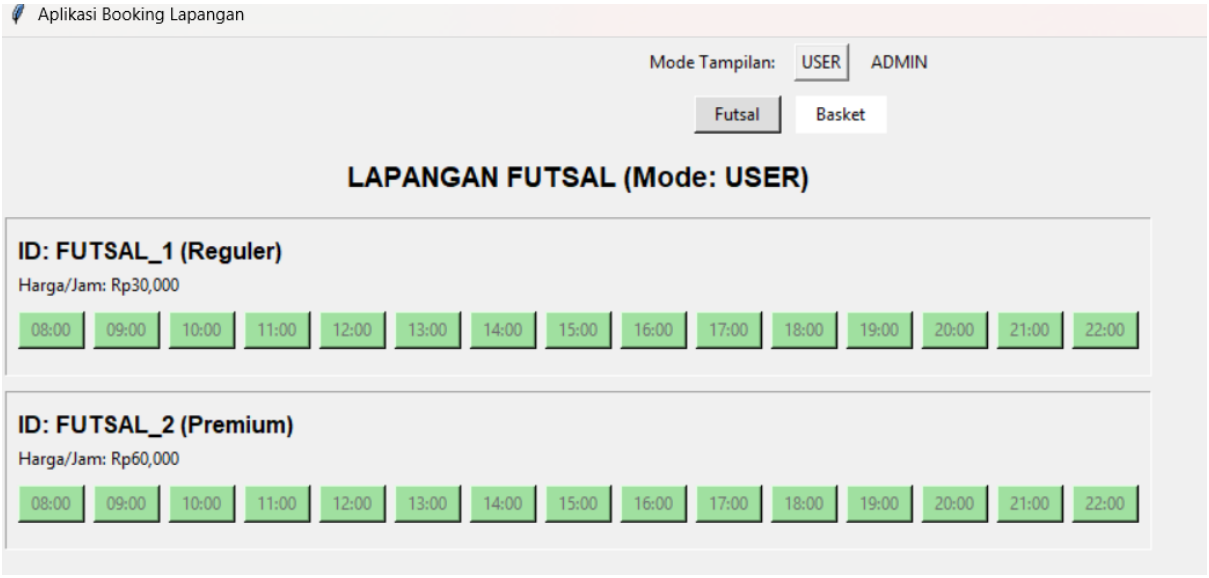
4.2.3 Fitur Cancel Booking

Fitur ini memberikan kemampuan kepada Admin untuk menghapus pemesanan yang sudah tercatat.





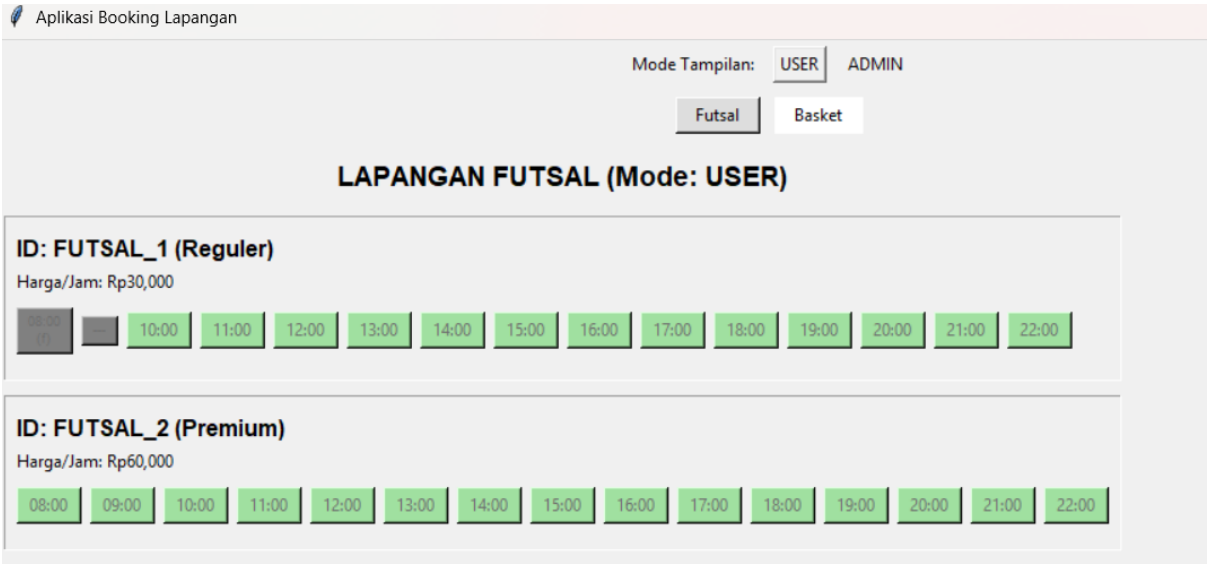




- Fungsi: Mengizinkan Admin untuk membatalkan booking yang sudah ada, melepaskan slot waktu tersebut sehingga tersedia kembali.
- Mekanisme: Ketika Admin mengklik slot yang sudah terisi, metode _handle_cancel dipanggil. Metode ini mengidentifikasi booking yang sesuai dalam LIST_BOOKING_SEMUA dan menghapusnya. Tampilan kemudian diperbarui (melalui tampilkan_lapangan) untuk menunjukkan slot tersebut sebagai tersedia kembali.

4.2.4 Fitur User bisa Melihat Ketersediaan Lapangan

Fitur ini adalah fungsi utama dari user view, di mana pengguna tanpa hak akses Admin dapat memeriksa status lapangan.



- Fungsi: Menampilkan status real-time dari setiap slot waktu (Tersedia atau Terisi) pada jenis lapangan yang dipilih.
- Mekanisme: Saat PERAN_AKTIF adalah "USER", metode tampilkan_lapangan akan me-render tombol slot jam:
    - Slot Tersedia ditampilkan dengan warna indikasi positif (misalnya, hijau).
    - Slot Terisi ditampilkan dengan label booking (misalnya, nama pelanggan) atau dinonaktifkan untuk mencegah interaksi.

4.3 Tantangan dan Kendala Pengembangan

Selama proses pengembangan aplikasi Booking Lapangan, terdapat beberapa tantangan yang dihadapi. Tantangan utama yang pertama adalah memastikan konsistensi dan integritas data booking, terutama dalam mencegah bentrok slot waktu. Hal ini diatasi dengan menerapkan metode cek_ketersediaan() dalam kelas Booking, yang secara sistematis memvalidasi setiap jam dalam durasi booking yang diajukan terhadap data booking yang sudah ada dalam LIST_BOOKING_SEMUA.

Selain itu, pengelolaan alur GUI menggunakan Tkinter juga menjadi tantangan tersendiri, terutama dalam mengatur pembaruan dan transisi tampilan antara mode User dan Admin tanpa menumpuk widget sebelumnya. Masalah ini diselesaikan dengan menerapkan metode penghapusan widget (widget.destroy()) sebelum menampilkan tampilan baru.

Tantangan lainnya adalah penerapan konsep OOP agar benar-benar terlihat nyata dan tidak hanya bersifat teori; dengan memisahkan tanggung jawab kelas dan menerapkan Inheritance serta Polymorphism pada perhitungan biaya dan Abstraction pada logika validasi, tantangan ini dapat diatasi dengan baik.

BAB V PENUTUP

5.1 Kesimpulan

Aplikasi Booking Lapangan Berbasis Object-Oriented Programming (OOP) yang dikembangkan menggunakan Python dan pustaka Tkinter telah berhasil diselesaikan dan diimplementasikan sesuai dengan tujuan proyek. Aplikasi ini mampu menyajikan fungsionalitas utama secara efektif, termasuk menampilkan ketersediaan slot waktu, memproses booking manual oleh Administrator, dan memfasilitasi pembatalan booking.

Keberhasilan utama proyek terletak pada penerapan prinsip-prinsip OOP secara konsisten, di mana Encapsulation melindungi data harga, Inheritance menciptakan hierarki Lapangan yang efisien, Polymorphism memberikan fleksibilitas pada perhitungan biaya, dan Abstraction menyederhanakan logika validasi ketersediaan slot.

Penerapan OOP ini telah menghasilkan kode yang modular, terstruktur, dan mudah dipelihara, memvalidasi bahwa arsitektur berorientasi objek sangat cocok untuk mengelola logika bisnis yang kompleks seperti manajemen booking dan validasi ketersediaan waktu.

5.2 Saran

Untuk pengembangan aplikasi Booking Lapangan di masa mendatang, disarankan beberapa perbaikan untuk meningkatkan kapabilitas dan skalabilitas sistem. Pertama, sistem penyimpanan data harus ditingkatkan dengan mengimplementasikan penyimpanan data permanen, baik menggunakan file JSON persisten atau database relasional seperti SQLite, untuk memastikan data booking tetap utuh meskipun aplikasi ditutup. Kedua, perlu dikembangkan fitur validasi waktu dan biaya yang lebih canggih, seperti perhitungan harga yang berbeda untuk weekend atau penerapan diskon khusus. Ketiga, disarankan untuk mengimplementasikan pola desain Model-View-Controller (MVC) untuk memisahkan logika back-end (kelas OOP) dari antarmuka front-end (Tkinter) agar pemeliharaan dan pengembangan UI menjadi lebih dinamis dan terorganisir.