LAPORAN

IMPLEMENTASI SORT, SEARCH DAN QUEUE DALAM APLIKASI

RESERVASI HOTEL

Dibuat Untuk Menyelesaikan Ujian Akhir Semester Mata Kuliah Struktur Data

Dosen Pengampu:

Ahmad Baihaqi, M. Kom

Disusun Oleh:

1. ICHSAN AZALA WALANADZA (2213020076)
2. YOGA FIRNANDA WICAKSONO (2213020128)
3. RACHEL ROBIN SUGIARTI (2213020088)
4. SHAFIRA AGUSTINA PUTRI (2213020131)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NUSANTARA PGRI KEDIRI

2024

DAFTAR ISI

BAB I

PENDAHULUAN

A. Latar Belakang

perkembangan industri perhotelan yang pesat dan perubahan perilaku konsumen telah mendorong kebutuhan akan solusi reservasi hotel yang efisien dan inovatif. Aplikasi reservasi hotel muncul sebagai respons terhadap tantangan dan keinginan konsumen untuk memperoleh aksesibilitas yang lebih baik, kemudahan dalam proses pemesanan, dan pengalaman yang lebih personal.

Pemanfaatan teknologi digital, aplikasi mobile, dan platform online tidak hanya meningkatkan efisiensi proses pemesanan, tetapi juga menciptakan lingkungan persaingan yang lebih ketat di antara hotel dan platform pemesanan online. Keamanan transaksi dan kepercayaan konsumen menjadi aspek kritis dalam keberhasilan aplikasi reservasi hotel, dan industri secara keseluruhan telah menanggapi kebutuhan tersebut dengan langkah-langkah perlindungan data yang lebih baik.

Dengan mengintegrasikan fitur-fitur seperti opsi pembatalan yang fleksibel, pembaruan ketersediaan secara real-time, dan penawaran khusus, aplikasi reservasi hotel tidak hanya memenuhi ekspektasi konsumen tetapi juga membantu hotel meningkatkan visibilitas mereka. Peningkatan dalam layanan melalui inovasi seperti program loyalitas dan integrasi dengan

teknologi pintar di dalam kamar menjadi faktor kunci dalam meningkatkan pengalaman pengguna.

Dengan demikian, aplikasi reservasi hotel bukan hanya alat praktis untuk pelancong, tetapi juga menjadi sarana bagi hotel untuk meningkatkan daya saing mereka dan memperluas cakupan pasar. Melalui solusi-solusi ini, aplikasi reservasi hotel mencerminkan transformasi positif dalam cara industri perhotelan berinteraksi dengan konsumen, memberikan manfaat yang nyata bagi konsumen

B. Deskripsi Aplikasi

Aplikasi reservasi hotel adalah platform digital yang menyederhanakan proses pencarian, pemesanan, dan pengelolaan akomodasi hotel. Pengguna dapat dengan cepat mencari ketersediaan kamar, melihat informasi lengkap tentang hotel, dan melakukan pemesanan real-time melalui antarmuka yang ramah pengguna. Fitur pembayaran aman, konfirmasi instan, dan manajemen pemesanan memberikan pengalaman yang efisien. Notifikasi, program loyalitas, dan ulasan pengguna meningkatkan interaksi positif. Aplikasi ini dirancang untuk memenuhi kebutuhan konsumen modern yang mengutamakan kemudahan, kecepatan, dan kenyamanan dalam merencanakan perjalanan mereka.

C. Kebutuhan Data dan Informasi

Aplikasi ini memiliki kebutuhan data yang mencakup informasi mengenai reservasi hotel, tamu beserta detailnya, serta riwayat transaksi pemesanan kamar. Adapun informasi yang akan dihasilkan melibatkan data tamu yang sedang melakukan pemesanan kamar, kamar yang sedang disewa, dan status check in dan check out.

Pentingnya data tamu pemesan kamar melibatkan informasi personal, seperti nama dan alamat yang diperlukan untuk mencatat data pribadi serta melacak aktivitas pemesanan. Detail transaksi, seperti no kamar, harga, dan jenis kamar, juga menjadi bagian integral dalam kebutuhan data aplikasi ini.

Riwayat transaksi pemesanan mencakup informasi waktu pemesanan, batas waktu pemesanan, dan status kamarapakah sudah dipakai apa belum. Dengan memantau riwayat pemesanan ini, aplikasi diharapkan mampu menghasilkan laporan yang akurat terkait pemesanan, transaksi pemesanan yang sedang berlangsung, dan informasi pemesanan untuk pengelolaan reservasi hotel yang lebih efektif.

Dengan implementasi fitur-fitur tersebut, aplikasi ini diharapkan mampu memberikan laporan yang komprehensif, memudahkan pengelola reservasi hotel dalam mengambil keputusan yang tepat, serta meningkatkan pengalaman pengguna dalam akses dan manipulasi data dengan cepat dan akurat.

D. Input dan Output

Aplikasi ini menerima input dalam bentuk data tamu pemesan hotel, informasi tamu, dan data transaksi pemesanan. Input ini dapat dengan mudah dimasukkan melalui formulir yang disediakan dalam antarmuka grafis pengguna (GUI) aplikasi. Pengguna, baik administrator reservasi hotel

maupun tamu, dapat mengisi formulir dengan data yang diperlukan, seperti informasi jenis kamar, tamu, atau transaksi pemesanan.

Selain menerima input, aplikasi juga memberikan output yang informatif melalui antarmuka grafis. Output ini mencakup tampilan data pemsan kamar, informasi kamar, dan riwayat transaksi kamar dalam bentuk tabel yang terstruktur. Dengan demikian, pengguna dapat dengan mudah melihat, memanipulasi, dan memverifikasi data-data yang relevan melalui tampilan yang disajikan dengan rapi dan jelas.

Pilihan untuk menggunakan formulir pada antarmuka grafis tidak hanya meningkatkan kemudahan pengguna dalam menginput data, tetapi juga memastikan bahwa output yang dihasilkan dapat diakses dan dimengerti dengan mudah. Ini menjadi langkah penting dalam memastikan keakuratan dan kejelasan data yang dikelola oleh aplikasi resevasi hotel.

E. Algoritma Searching, Sorting dan Penggunaan Queue

Dalam pengelolaan data aplikasi ini, diterapkan algoritma sorting dan searching guna meningkatkan efisiensi dalam pencarian dan pengurutan data. Algoritma sorting dapat diterapkan pada berbagai elemen, seperti pengurutan daftar yang diinginkan berdasarkan kolom yang diminta. Sebaliknya, algoritma searching digunakan untuk pencarian informasi spesifik, misalnya, mencari data tamu berdasarkan nama.

Salah satu penggunaan struktur data yang penting dalam aplikasi ini adalah penggunaan queue untuk mendukung fitur antrian cetak nota Struktur queue memungkinkan mencetak antrian dan mencetak nota pemesanan kamar, sehingga mereka dapat dengan mudah mengetahui detail pemesanan kamar yang dipesan pada Database.

Data pada aplikasi ini disimpan dan dikelola menggunakan MySQL sebagai sistem manajemen basis data (DBMS). MySQL dipilih sebagai solusi yang handal untuk menyimpan data tamu pemesan kamar informasi tamu, dan riwayat transaksi pemesanan. Keberadaan basis data memungkinkan penyimpanan yang terstruktur dan efisien, mempermudah pengelolaan serta pengambilan data secara cepat.

Struktur basis data aplikasi mencakup tabel-tabel utama seperti `tbl_admin` untuk data pegawai hotel, `tbl_kamar` untuk informasi kamar, `tbl_tamu` untuk data informasi tamu, dan `tbl_transaksi` untuk mencatat riwayat transaksi pemesanan. Setiap tabel memiliki hubungan yang terdefinisi dengan tabel lainnya, sehingga menjaga integritas data di dalam basis data. Misalnya, tabel `tbl_transaksi` dapat terkait dengan tabel `tbl_tamu` dan `tbl_admin` untuk mencatat data pegawai.

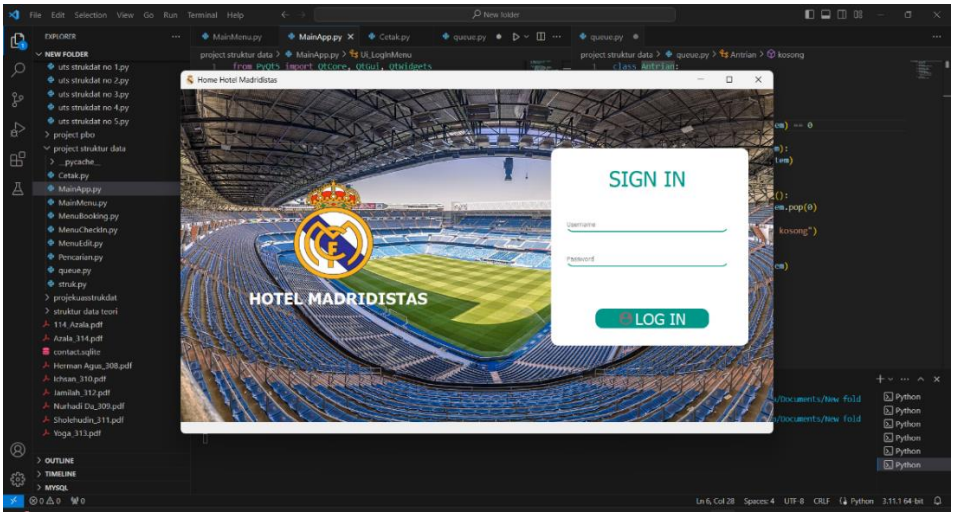
Penggunaan MySQL sebagai DBMS tidak hanya menjamin keamanan dan keakuratan data, tetapi juga memungkinkan aplikasi untuk mengelola volume data yang signifikan dengan efisien. Dengan demikian, basis data menjadi fondasi yang kokoh dalam mendukung operasional sehari-hari dan fungsionalitas aplikasi reservasi hotel.

BAB II

IMPLEMENTASI

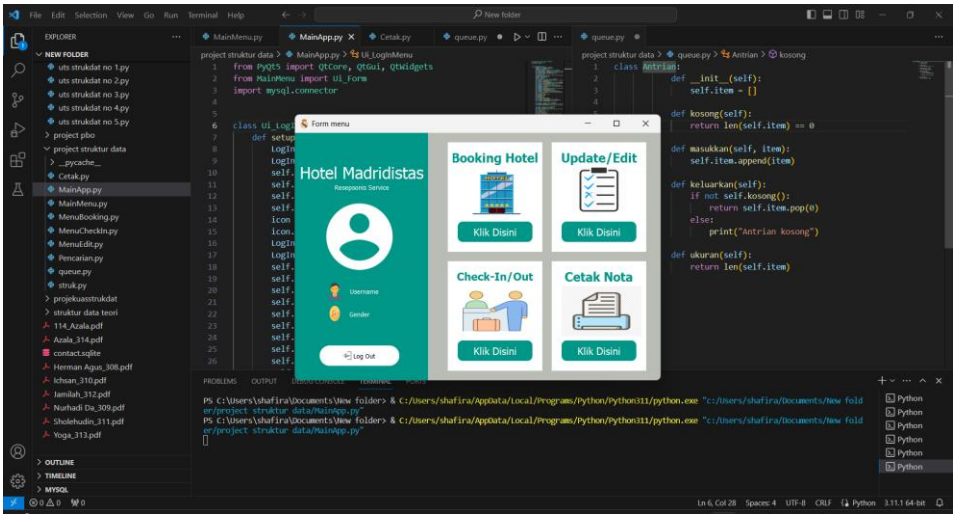
A. Alur Aplikasi

1. Form Login



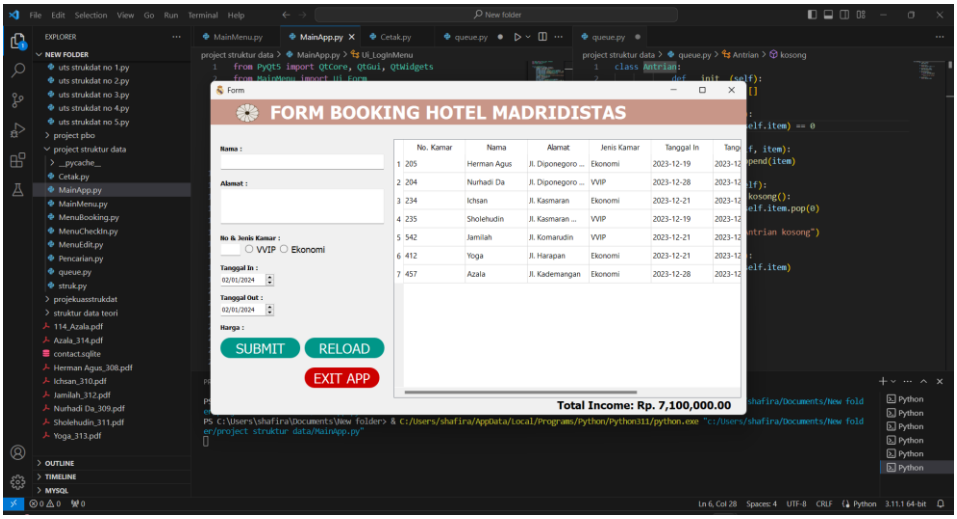
Login menggunakan username dan password yang sudah di masukkan dalam database. Jika username dan password sudah di isi klik login.

2. Form main menu



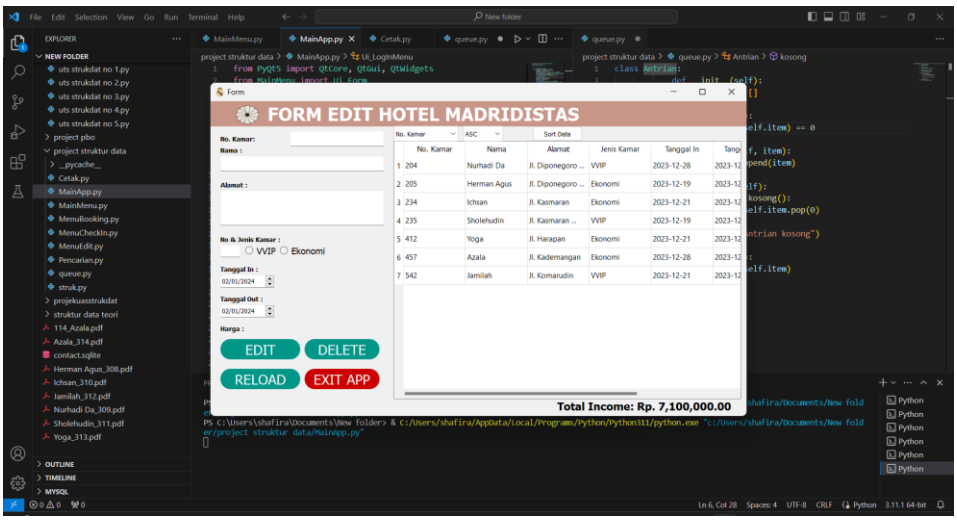
Jika login benar tampilan akan menuju ke menu utama yang menyediakan beberapa opsi yaitu untuk booking hotel, edit/update, check in/check out dan cetak nota

3. Menu booking



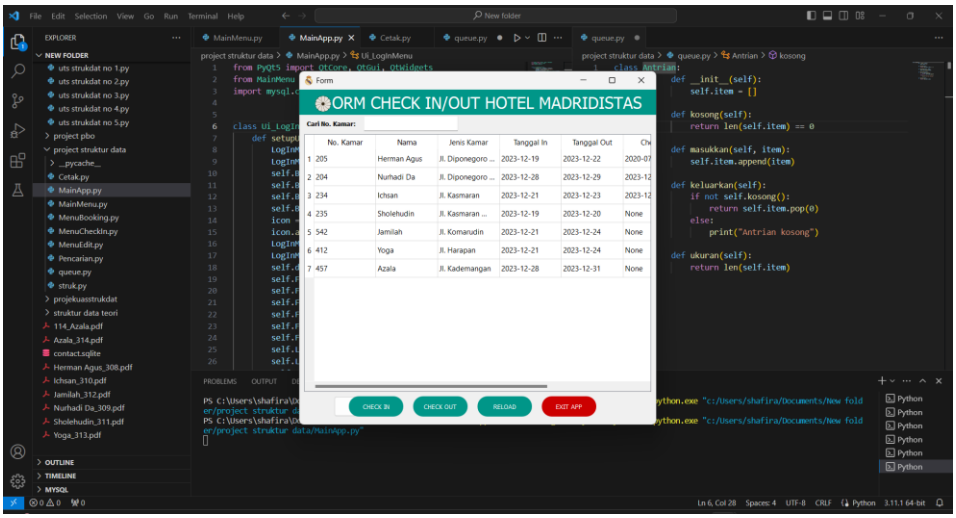
Tampilan di atas digunakan untuk booking/pesan kamar dengan cara meng input beberapa data yang dibutuhkan

4. Menu edit/update



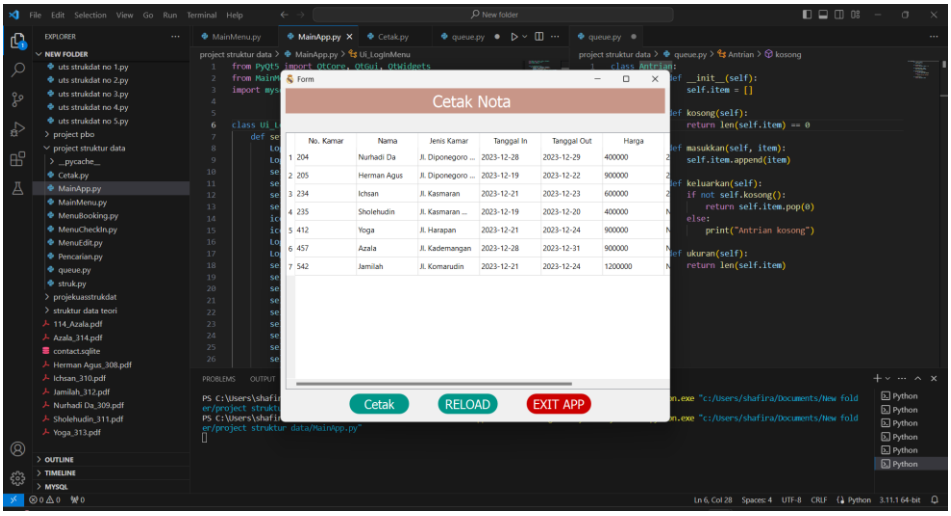
Dalam menu edit/update akan memberi opsi untuk mengupdate data atau pesanan kamar yang telah di pesan

5. Menu check in/out



Pada menu ini akan bekerja sebagai penanda bahwa pemesan kamar telah masuk(check in) kamar dan penanda penyewa kamar telah keluar(check out) kamar untuk mengahiri penyewaan kamar

6. Menu Cetak nota



Menu ini berfungsi sebagai cetak nota untuk transaksi yang telah dibuat, cetak nota akan di lakukan secara berurut dari atas karena menggunakan implementasi queue

B. Source Code

1. Implementasi search

```
def SearchData(self):
        search_text = self.LineSearch.text().lower()

        self.ReloadData()
        self.SortData()

        if not search_text:
            return

        left, right = 0, self.TableInform.rowCount() - 1

        while left <= right:
            mid = (left + right) // 2
            item = self.TableInform.item(mid, 0)

            if item and search_text in item.text().lower():
                for col in range(self.TableInform.columnCount()):
                    self.TableInform.item(mid,
col).setBackground(QtGui.QColor(255, 255, 0))
                return
            elif item and search_text < item.text().lower():
                right = mid - 1
            else:
                left = mid + 1
```

Penjelasan:

Fitur ini bertujuan untuk mencari data berdasarkan nomer kamar yang di input ke dalam line ya disediakan, untuk data yang ditemukan akan berubah manjadi warna kuning.

Fungsi **SearchData** ini terlihat digunakan untuk melakukan pencarian data dengan menggunakan metode binary search pada tabel (**self.TableInform**). Berikut adalah penjelasan langkah-langkahnya:

1. Ambil teks pencarian dari objek LineEdit (**self.LineSearch**) dan konversi ke huruf kecil.

2. Muat ulang data dan urutkan berdasarkan nomor ruangan dengan memanggil fungsi **ReloadData** dan **SortData**.

3. Jika teks pencarian kosong, maka tidak ada yang perlu dicari dan fungsi berakhir.

4. Lakukan pencarian biner pada data yang sudah diurutkan. Variabel **left** diatur ke indeks awal (0), dan **right** diatur ke indeks akhir (**self.TableInform.rowCount() - 1**).

5. Selama indeks **left** tidak melewati indeks **right**, lakukan langkah-langkah berikut: a. Hitung indeks tengah (**mid**) antara **left** dan **right**. b. Ambil item pada kolom pertama (indeks 0) pada baris **mid**. c. Jika item ada dan teks pencarian ditemukan dalam teks item (dengan membandingkan huruf kecil), maka warnai seluruh baris dengan warna kuning (**QColor(255, 255, 0)**) dan fungsi berakhir. d. Jika item ada dan teks pencarian lebih kecil dari teks item, maka ubah **right** menjadi **mid - 1**. e. Jika item ada dan teks pencarian lebih besar dari teks item, maka ubah **left** menjadi **mid + 1**.

Dengan cara ini, pencarian biner dilakukan pada data yang sudah diurutkan, dan jika teks pencarian ditemukan, baris yang sesuai ditandai dengan latar belakang warna kuning pada seluruh kolomnya.

2. Implementasi sorting

```
def BubbleSort(self, sort_column, sort_order):
        rows = self.TableInform.rowCount()
        cols = self.TableInform.columnCount()

        data = [[self.TableInform.item(row, col).text() for col
in range(cols)] for row in range(rows)]

        # Get the column index for sorting
        column_index = self.get_column_index(sort_column)

        # Bubble Sort Implementation
        for i in range(rows - 1):
            for j in range(0, rows - i - 1):
                if (sort_order == "ASC" and data[j][column_index]
> data[j + 1][column_index]) or \
                        (sort_order == "DESC" and
data[j][column_index] < data[j + 1][column_index]):
                    data[j], data[j + 1] = data[j + 1], data[j]

    # Update the table after sorting
        for row in range(rows):
            for col in range(cols):
                self.TableInform.setItem(row, col,
QtWidgets.QTableWidgetItem(str(data[row][col])))
```

Penjelasan:

Fitur ini bertujuan untuk menyorting data pada form yang ada berdasarkan no.kamar, nama, tanggal in dan tanggal out, pada fitur ini juga di sediakan pilihan ASCENDING dan DESCENDING.

Fungsi **BubbleSort** ini digunakan untuk mengurutkan data pada tabel (**self.TableInform**) berdasarkan kolom tertentu dengan menggunakan algoritma Bubble Sort. Berikut adalah penjelasan langkah-langkahnya:

1. Ambil jumlah baris (**rows**) dan kolom (**cols**) dari tabel.

2. Buat matriks **data** yang berisi seluruh teks dari setiap sel pada tabel. Setiap elemen **data** mewakili satu baris data, dan setiap elemen dalam satu baris mewakili teks dari sel pada kolom tersebut.
3. Dapatkan indeks kolom yang akan digunakan untuk pengurutan dengan memanggil fungsi **get_column_index** dan simpan dalam variabel **column_index**.
4. Lakukan implementasi algoritma Bubble Sort: a. Iterasi dari **i = 0** hingga **i = rows - 1**. b. Di dalam iterasi pertama, iterasi dari **j = 0** hingga **j = rows - i - 1**. c. Bandingkan elemen pada kolom yang ditentukan (**column_index**) pada baris **j** dan **j + 1**. Jika urutan pengurutan adalah ASC dan elemen **j** lebih besar dari elemen **j + 1**, atau urutan pengurutan adalah DESC dan elemen **j** lebih kecil dari elemen **j + 1**, maka tukar baris **j** dan **j + 1**.
5. Setelah proses Bubble Sort selesai, update tabel dengan menyusun kembali data yang sudah diurutkan: a. Iterasi melalui setiap baris dan kolom tabel. b. Setel item di tabel pada baris dan kolom yang sesuai dengan nilai yang sudah diurutkan dari matriks **data**.

Dengan menggunakan algoritma Bubble Sort, data pada tabel akan diurutkan berdasarkan kolom tertentu sesuai dengan urutan yang ditentukan (**ASC** untuk urutan menaik dan **DESC** untuk urutan menurun)

3. Implementasi Queue pada file member.py

S

```
class Queue:
        def __init__(self):
            self.items = []

        def is_empty(self):
            return len(self.items) == 0
```

```
        def enqueue(self, item):
            self.items.append(item)

        def dequeue(self):
            if not self.is_empty():
                return self.items.pop(0)
            else:
                print("Antrian kosong")

        def size(self):
            return len(self.items)

    def setupUi(self, Form):
        self.baris = self.Queue()

def ClickedCetak(self):
        db_config = {
            'user': 'root',
            'password': '',
            'host': 'localhost',
            'database': 'hotelst',
            'raise_on_warnings': True
         }

        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()

        query = (
    "SELECT "
        "Transaksi.id_transaksi, "
        "Tamu.nama_tamu, "
        "Transaksi.tanggal_in, "
        "Transaksi.tanggal_out, "
        "Kamar.no_kamar, "
        "Kamar.jenis_kamar, "
        "CASE "
            "WHEN Kamar.jenis_kamar = 'Ekonomi' THEN 300000 "
            "WHEN Kamar.jenis_kamar = 'VVIP' THEN 400000 "
            "ELSE 0 "
        "END AS harga, "
        "DATEDIFF(Transaksi.tanggal_out, Transaksi.tanggal_in) AS
num_days "
    "FROM Transaksi "
```

```
    "JOIN Tamu ON Transaksi.id_tamuFK = Tamu.id_tamu "
    "JOIN Kamar ON Transaksi.no_kamarFK = Kamar.no_kamar;"
)
        cursor.execute(query)

        rows = cursor.fetchall()


        for row in rows:
            self.baris.enqueue(row)
        while not self.baris.is_empty():
            baris_data = self.baris.dequeue()
            id_transaksi, nama_tamu, tanggal_in, tanggal_out,
no_kamar, jenis_kamar, harga, num_days = baris_data
            if jenis_kamar in ['VVIP', 'Ekonomi']:
                total_cost_float = 0.0  # Set a default value for
'VVIP' and 'Ekonomi'
            else:
                print(f"Row Harga: {harga}")
                try:
                    total_cost_float = float(harga)
                    print(f"Parsed Harga: {total_cost_float}")
                except ValueError:
                    print("Invalid total_cost value")
                    total_cost_float = 0.0
            # Create a PDF payment receipt
            self.create_payment_receipt(
                file_name=f"{nama_tamu}_{id_transaksi}.pdf",
                id_transaksi=id_transaksi,
                nama_tamu=nama_tamu,
                jenis_kamar=jenis_kamar,
                no_kamar=no_kamar,
                tanggal_in=tanggal_in,
                tanggal_out=tanggal_out,
                harga=harga,
                total_cost=num_days * harga,
                num_days=num_days
            )
            # Display a message
            self.show_message("Cetak Nota", f"Payment receipt
created: {nama_tamu}_{id_transaksi}.pdf")

        cursor.close()
```

```
        conn.close()
```

Penjelasan:

Pada fitur ini queue berfungsi sebagai pengatur alur atau antrian mencetak nota dengan cara bergantian dari atas ke bawah yang memiliki konsep sama dengan queue yaitu FIFO(First IN First Out).

Berikut adalah penjelasan singkat dari beberapa bagian utama dari kode ini:

1. **Kelas Queue:**
    - **__init__(self)**: Membuat objek antrian dengan list kosong sebagai basis penyimpanan elemen-elemen antrian.
    - **is_empty(self)**: Mengembalikan True jika antrian kosong, False sebaliknya.
    - **enqueue(self, item)**: Menambahkan elemen ke akhir antrian.
    - **dequeue(self)**: Menghapus dan mengembalikan elemen pertama dari antrian, jika antrian tidak kosong. Jika kosong, mencetak pesan "Antrian kosong".
    - **size(self)**: Mengembalikan jumlah elemen dalam antrian.
2. **Metode setupUi(self, Form):**
    - Membuat instance dari kelas **Queue** sebagai atribut **self.baris**.
3. **Metode ClickedCetak(self):**
    - Membuat koneksi ke database MySQL dan mengeksekusi query untuk mengambil data transaksi dari tabel yang terhubung.
    - Setiap baris hasil query dimasukkan ke dalam antrian **self.baris**.
    - Menggunakan loop **while** untuk mengambil dan memproses setiap elemen dalam antrian.

- Untuk setiap baris data, mengakses nilai-nilai yang diinginkan (seperti **id_transaksi**, **nama_tamu**, **harga**, dll.).
- Membuat nota pembayaran (dengan memanggil metode **create_payment_receipt**) dan menampilkan pesan.
- Menutup kursor dan koneksi ke database setelah selesai.

BAB III

PENUTUP

A. Kesimpulan

Melalui penerapan struktur data dan algoritma pada aplikasi reservasi ini, kami dapat menyimpulkan beberapa hal penting. Pertama, algoritma pencarian yang telah diimplementasikan pada data tamu dan transaksi telah memberikan kemudahan bagi pengguna untuk menemukan informasi dengan cepat. Meskipun algoritma pencarian linear saat ini sudah memadai, disarankan untuk mempertimbangkan algoritma pencarian yang lebih efisien, seperti binary search, terutama jika jumlah data semakin meningkat.

Selanjutnya, algoritma pengurutan pada data anggota dan data buku berdasarkan nama atau judul telah mempermudah pengguna dalam menavigasi dan mencari buku. Saat ini, algoritma sorting bubble yang diterapkan memberikan hasil yang memadai, tetapi eksplorasi dan evaluasi terhadap algoritma sorting lainnya, seperti quicksort atau mergesort, dapat meningkatkan efisiensi pengurutan pada skala data yang lebih besar.

Yang terakhir, implementasi struktur data queue untuk fitur mencetak nota telah memberikan nilai tambah yang signifikan pada aplikasi. Pengguna dapat menciptakan pengalaman pengguna yang lebih baik dan intuitif. Konsep queue ini juga dapat menjadi dasar yang kuat untuk pengembangan fitur masa depan yang memerlukan manajemen antrian data.

B. Saran

a. Optimalisasi Algoritma Pencarian dan Pengurutan

   Perlu dievaluasi secara mendalam mengenai pemilihan algoritma pencarian dan pengurutan pada data tamu dan transaksi. Meskipun implementasi saat ini sudah berjalan dengan baik, namun penggunaan algoritma yang lebih efisien, seperti QuickSort atau MergeSort, dapat meningkatkan kinerja aplikasi, terutama jika jumlah data dalam reservasi hotel terus bertambah. Evaluasi skenario penggunaan yang potensial dan penerapan algoritma yang sesuai dapat menjadi langkah proaktif untuk meningkatkan efisiensi.

b. Perluasan Penggunaan Struktur Data Queue

   Sangat positif bahwa struktur data stack digunakan untuk fitur antrian cetak nota pada aplikasi. Disarankan untuk mempertimbangkan perluasan penggunaan queue dalam fitur lain yang melibatkan manajemen tindakan pengguna. Misalnya, queue dapat digunakan untuk melacak riwayat perubahan atau pemulihan data, memberikan konsistensi dalam manajemen interaksi pengguna, dan meningkatkan keandalan fitur aplikasi.

c. Pertimbangkan Penggunaan Struktur Data Lanjutan

   Dengan pertumbuhan aplikasi, evaluasilah apakah penggunaan struktur data yang lebih lanjut, seperti tree atau graph, dapat memberikan manfaat tambahan. Terutama jika aplikasi dikembangkan dengan fitur-fitur yang melibatkan hubungan hierarkis antara entitas data. Evaluasi kebutuhan aplikasi secara proaktif dapat membantu dalam pemilihan dan penerapan struktur data yang paling sesuai untuk mendukung fungsionalitas masa depan.

d. Evaluasi dan Optimalkan Koneksi Database

Disarankan untuk secara rutin mengevaluasi dan mengoptimalkan koneksi ke database. Langkah-langkah seperti pemantauan kinerja database, penyesuaian indeks, dan optimasi query dapat meningkatkan efisiensi operasional. Seiring dengan pertumbuhan jumlah pengguna atau data, pertimbangkan praktik-praktik terbaik dalam manajemen database untuk memastikan kinerja aplikasi tetap optimal.

e. Peningkatan User Interface

Terus tingkatkan antarmuka pengguna untuk memberikan pengalaman yang lebih baik kepada pengguna. Secara berkala, tinjau desain antarmuka dengan fokus pada kejelasan, navigasi yang intuitif, dan responsivitas. Berdasarkan umpan balik pengguna, selalu pertimbangkan penyederhanaan desain agar pengguna dapat berinteraksi dengan aplikasi dengan mudah. Upaya ini dapat berkontribusi pada peningkatan daya tarik dan penerimaan pengguna terhadap aplikasi.