

**LAPORAN TUGAS AKHIR**  
**Pemrograman Berorientasi Objek**

**“Pemesanan Tiket Bioskop Online”**

*Dosen pengampu : M. Bahrul Subkhi M.Kom*



**Disusun oleh :**

- |                                |              |
|--------------------------------|--------------|
| 1. Balqis Salsabila Nurul Huda | (2213020217) |
| 2. Fadya Nur Ayni              | (2213020212) |
| 3. Deys Aishara Angelina       | (2213020233) |

PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNIK DAN ILMU KOMPUTER  
UNIVERSITAS NUSANTARA PGRI KEDIRI  
2023

## **KATA PENGANTAR**

Puji syukur kami panjatkan kehadiran Tuhan Yang Maha Esa yang telah melimpahkan rahmat, hidayah serta karunia-Nya kepada kami sehingga bisa menyelesaikan laporan tugas akhir pada mata kuliah Pemrograman Berorientasi Objek II yang berjudul “Pemesanan Tiket Bioskop Online”.

Laporan tugas akhir ini kami susun sesuai dengan ilmu dan pengalaman yang kami peroleh pada saat mengikuti perkuliahan Pemrograman Berorientasi Objek II sehingga memudahkan kami dalam pembuatan laporan tugas akhir ini. Untuk itu saya sampaikan terima kasih kepada Bapak Bahrul yang telah mengajar kami ketika waktu perkuliahan.

Terlepas dari segala hal tersebut, kami sadar sepenuhnya bahwa masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasanya. Oleh karena itu kami dengan lapang dada menerima segala saran dan kritik dari teman-teman atau pembaca agar kami bisa memperbaiki laporan tugas akhir ini dengan baik. Kami berharap semoga laporan tugas akhir Pemrograman Berorientasi Objek II ini berjudul “Pemesanan Tiket Bioskop Online” ini bisa memberikan manfaat maupun inspirasi kepada penulis dan pembaca khususnya.

Kediri, 25 Desember 2023

Penyusun

## DAFTAR ISI

KATA PENGANTAR.....	2
DAFTAR ISI.....	3
BAB I.....	4
PENDAHULUAN.....	4
1.1 Latar Belakang.....	4
1.2 Rumusan Masalah.....	4
1.3 Tujuan Penelitian.....	4
BAB II.....	5
ANALISA DAN PERMODELAN.....	5
2.1 Flowchart Sistem.....	5
2.2 Class Diagram.....	6
BAB III.....	9
SOURCE CODE DAN IMPLEMENTASI.....	9
3.1 Implementasi Hasil.....	9
3.2 Challenge Mengubah Tampilan.....	32
BAB IV.....	34
PENUTUP.....	34
4.1 Kesimpulan.....	34
4.2 Saran.....	34
DAFTAR PUSTAKA.....	35

## **BAB I PENDAHULUAN**

### **1.1 Latar Belakang**

Semakin banyaknya masyarakat yang mencari hiburan dengan menonton film di bioskop saat ini, maka pemesanan tiket film di bioskop menjadi masalah yang penting. Ada sebagian masyarakat yang rela mengantri dari pagi untuk mendapat tiket film di sebuah bioskop. Dengan pemesanan tiket bioskop yang semakin hari semakin banyak maka diperlukan sistem untuk dapat mempermudah masyarakat dalam memesan tiket film di bioskop. Saat ini pemesanan tiket bioskop sudah dapat dilakukan secara online dengan membuka halaman web atau aplikasi dan membeli secara langsung dengan datang ke bioskop. Pemesanan tiket bioskop dengan menggunakan web harus mencari web tersebut di internet. Sehingga jika ingin memesan harus mencari kata kunci untuk membuka web, dan sebagian orang mungkin tidak tahu mana web yang bisa memesan tiket tersebut. maka diciptakan aplikasi pemesanan tiket bioskop online untuk mempermudah customer mencari film dan jadwal tayang, tidak perlu mengantri seperti saat memesan secara langsung, jua tetap bisa memilih kursi. Tetapi pemesanan tiket bioskop dengan menggunakan web maupun aplikasi memiliki kelemahan.

### **1.2 Rumusan Masalah**

Berdasarkan latar belakang diatas dapat dirumuskan permasalahan – permasalahan yaitu:

1. Bagaimana cara agar pemesan tiket bioskop dapat memesan tiket dengan cara mudah dan cepat dimanapun pembeli berada?
2. Bagaimana cara agar pemesan tiket dengan bebas memilih posisi tempat duduk yang diinginkan pembeli?
3. Bagaimana mengetahui informasi mengenai film apa saja yang ditampilkan pada bioskop?

### **1.3 Tujuan Penelitian**

Berdasarkan rumusan yang telah diuraikan, maka saya dapat mengetahui tujuan penelitian yang diperlukan sebagai berikut:

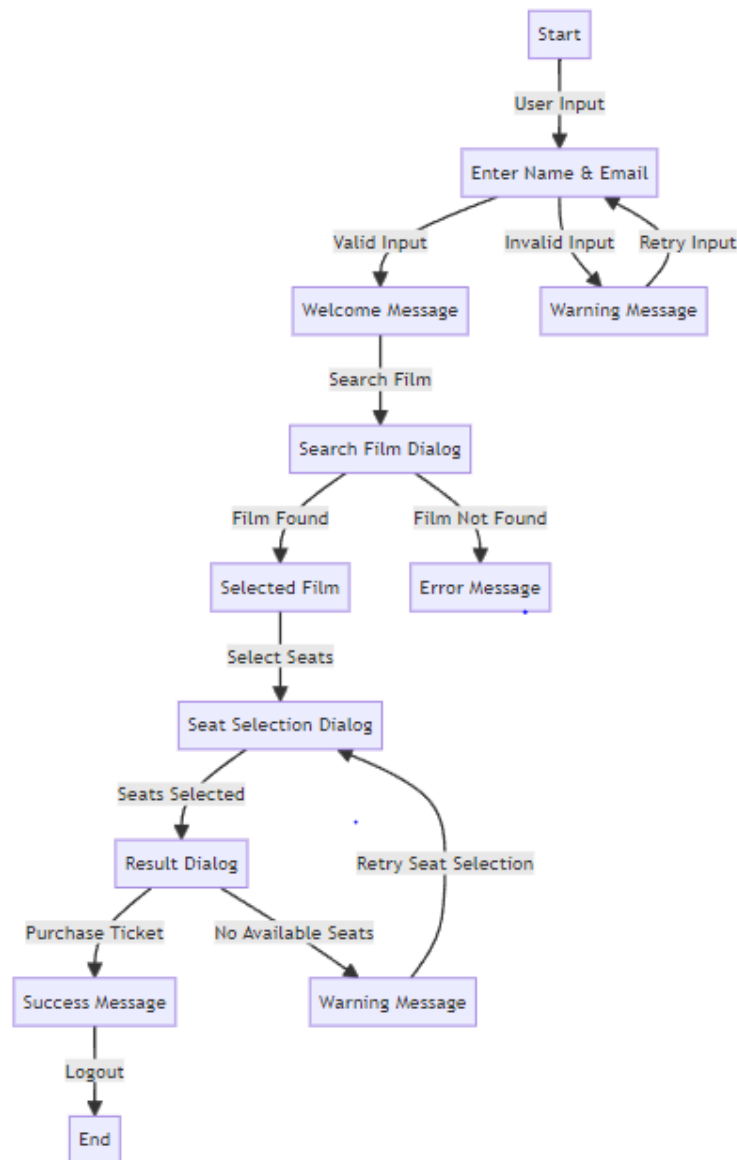
1. Membuat program untuk pemesanan tiket bioskop yang mudah.
2. Pemesan tiket dapat memilih secara langsung posisi tempat duduk yang pembeli inginkan dengan melihat denah tempat duduk pada aplikasi mobile untuk pemesanan tiket bioskop.
3. Aplikasi mobile untuk pemesanan tiket bioskop dapat melihat informasi mengenai bioskop, film-film yang sedang ditayangkan dan jadwal

## **BAB II**

### **ANALISA DAN PERMODELAN**

#### **2.1 Flowchart Sistem**

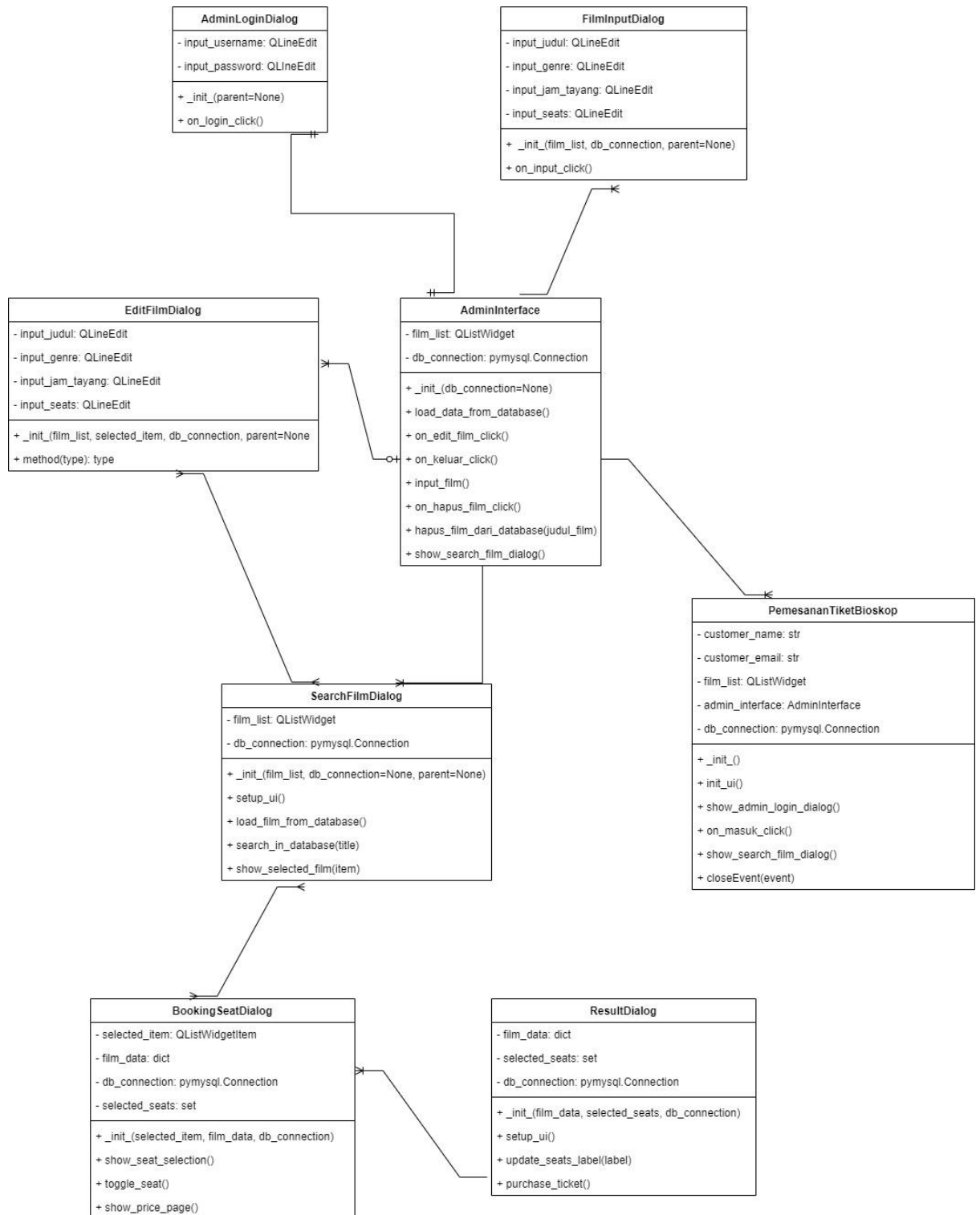
Flowchart sistem adalah diagram yang menunjukkan alur kerja atau proses secara keseluruhan dari suatu sistem. Flowchart sistem menggambarkan dengan urutan detail dari setiap prosedur yang ada pada sistem. Flowchart sistem terdiri dari lima jenis, yaitu flowchart dokumen, flowchart program, flowchart proses, flowchart skematik, dan flowchart sistem. Sistem flowchart sangat berguna untuk menghubungkan antara kebutuhan teknis dan non-teknis serta memberi gambaran ringkasan suatu program dari satu proses ke proses lainnya sehingga alur program menjadi mudah dipahami oleh semua orang.



Gambar 2.1 Flowchart Sistem pada program.

## 2.2 Class Diagram

Class diagram adalah jenis diagram struktur pada Unified Modeling Language (UML) yang menggambarkan kelas-kelas dalam sebuah sistem dan merangkum antara satu dengan yang lain, serta dimasukkan pula atribut dan operasi. Class diagram berfungsi untuk menjelaskan suatu model data untuk sebuah program, baik model data sederhana maupun kompleks. Class diagram juga dapat membantu proses dokumentasi sistem dan meringankan kinerja programmer dalam pembuatan sistem



Gambar 2.2 Flowchart Sistem pada program

Berikut penjelasannya

**AdminLoginDialog:**

Memiliki relasi dengan AdminInterface karena digunakan dalam method `show_admin_login_dialog()` pada kelas PemesananTiketBioskop.

**Penjelasan:**

Kelas ini digunakan untuk membuat dialog login admin. Jika login berhasil, AdminInterface akan ditampilkan.

FilmInputDialog:

Memiliki relasi dengan AdminInterface karena digunakan dalam method `input_film()` pada kelas AdminInterface.

Penjelasan:

Kelas ini bertanggung jawab untuk membuat dialog input film. Data film yang dimasukkan akan ditambahkan ke database dan ditampilkan dalam QListWidget pada AdminInterface.

EditFilmDialog:

Memiliki relasi dengan AdminInterface karena digunakan dalam method `on_edit_film_click()` pada kelas AdminInterface.

Penjelasan:

Kelas ini digunakan untuk membuat dialog pengeditan data film. Data yang diubah akan disimpan ke database dan diupdate pada QListWidget di AdminInterface.

AdminInterface:

Memiliki relasi dengan FilmInputDialog dan EditFilmDialog karena menggunakan keduanya untuk operasi input dan edit film.

Memiliki relasi dengan SearchFilmDialog karena menggunakan dialog tersebut untuk melakukan pencarian film.

Penjelasan:

Kelas ini berfungsi sebagai antarmuka admin. Menampilkan daftar film dalam QListWidget dan memberikan fungsionalitas untuk input, edit, hapus film, dan pencarian film.

SearchFilmDialog:

Memiliki relasi dengan AdminInterface karena digunakan dalam method `show_search_film_dialog()` pada kelas PemesananTiketBioskop.

Memiliki relasi dengan BookingSeatDialog karena digunakan untuk menampilkan data film saat melakukan pencarian.

Penjelasan:

Kelas ini digunakan untuk membuat dialog pencarian film. Menampilkan hasil pencarian dalam QListWidget dan memungkinkan pengguna untuk memilih film untuk pemesanan.

BookingSeatDialog:

Memiliki relasi dengan ResultDialog karena digunakan untuk menampilkan data harga dan melakukan pembelian tiket.

Memiliki relasi dengan SearchFilmDialog karena muncul saat pemilihan kursi.

Penjelasan:

Kelas ini bertanggung jawab untuk menampilkan pemilihan kursi dan mengarahkan pengguna ke halaman harga setelah pemilihan selesai.

ResultDialog:

Memiliki relasi dengan BookingSeatDialog karena dipanggil saat pembelian tiket berhasil.

Kelas ini menampilkan informasi pembelian tiket, termasuk judul film, jam tayang, kursi yang dipilih, dan harga. Melakukan update terhadap jumlah kursi yang tersedia dan menyimpan data pembelian tiket ke database.

PemesananTiketBioskop:

Memiliki relasi dengan AdminInterface karena menyimpan instance dari kelas tersebut.



Penjelasan:

Kelas utama aplikasi ini bertanggung jawab untuk membuat antarmuka pengguna untuk pemesanan tiket bioskop. Mencakup logika untuk menghubungkan dengan antarmuka admin dan dialog pencarian film.

Semua hubungan dan interaksi di atas membentuk alur kerja aplikasi pemesanan tiket bioskop Anda.

## **BAB III**

### **SOURCE CODE DAN IMPLEMENTASI**

#### **3.1 Source Code**

##### **3.1.1 Import Modul**

```
import sys
import pymysql
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QVBoxLayout, QPushButton, QLabel,
    QLineEdit, QDialog, QMessageBox, QListWidget, QListWidgetItem,
    QFormLayout, QWidget, QAction, QGridLayout
)
from datetime import datetime
```

##### **3.1.2 Class Admin Login Dialog**

```
class AdminLoginDialog(QDialog):
    def __init__(self, parent=None):
        super(AdminLoginDialog, self).__init__(parent)
        self.setWindowTitle('Admin Login')
        self.setGeometry(200, 200, 300, 150)

        layout = QVBoxLayout()

        label_username = QLabel('Username:')
        self.input_username = QLineEdit(self)
        layout.addWidget(label_username)
        layout.addWidget(self.input_username)

        label_password = QLabel('Password:')
        self.input_password = QLineEdit(self)
        self.input_password.setEchoMode(QLineEdit.Password)
        layout.addWidget(label_password)
        layout.addWidget(self.input_password)

        button_login = QPushButton('Login', self)
        button_login.clicked.connect(self.on_login_click)
```

```

        layout.addWidget(button_login)

    self.setLayout(layout)

    def on_login_click(self):
        username = self.input_username.text()
        password = self.input_password.text()

        if username == 'admin' and password == 'admin123':
            self.accept()
        else:
            QMessageBox.warning(self, 'Login Failed', 'Invalid username
or password.')

```

Tujuan : Kelas ini mewakili dialog untuk login admin.

Kegunaan dan Fungsi :

- Digunakan untuk mengautentikasi administrator dengan meminta untuk memasukkan username dan password.
- Menampilkan dialog sederhana dengan kolom input untuk username dan password.
- Menyediakan metode on\_login\_click yang dipicu ketika tombol login di klik, dan memverifikasi kredensial yang dimasukkan.

### 3.1.3 Class Film Input Dialog

```

class FilmInputDialog(QDialog):
    def __init__(self, film_list, db_connection, parent=None):
        super(FilmInputDialog, self).__init__(parent)
        self.film_list = film_list
        self.db_connection = db_connection

        self.setWindowTitle('Input Film')
        self.setGeometry(200, 200, 300, 150)

        layout = QFormLayout()

        self.input_judul = QLineEdit(self)
        layout.addRow('Judul Film:', self.input_judul)

        self.input_genre = QLineEdit(self)
        layout.addRow('Genre Film:', self.input_genre)

        self.input_jam_tayang = QLineEdit(self)
        layout.addRow('Jam Tayang:', self.input_jam_tayang)

```

```

self.input_seats = QLineEdit(self)
layout.addRow('Seat:', self.input_seats)

button_input = QPushButton('Input', self)
button_input.clicked.connect(self.on_input_click)
layout.addRow(button_input)

self.setLayout(layout)

def on_input_click(self):
    judul = self.input_judul.text()
    genre = self.input_genre.text()
    jam_tayang = self.input_jam_tayang.text()
    seats = self.input_seats.text()

    try:
        if 0 < int(seats) <= 40:
            with self.db_connection.cursor() as cursor:
                sql = "INSERT INTO film (judul, genre, jam_tayang,
seats) VALUES (%s, %s, %s, %s)"
                cursor.execute(sql, (judul, genre, jam_tayang,
seats))

                self.db_connection.commit()

                film_data = f'Judul: {judul}, Genre: {genre}, Jam
Tayang: {jam_tayang}, Seats: {seats}'
                item = QListWidgetItem(film_data)
                self.film_list.addItem(item)
                self.film_list.sortItems()

                QMessageBox.information(self, 'Input Film', 'Film
berhasil diinput.')
            else:
                QMessageBox.warning(self, 'Error', 'Jumlah seats tidak
valid. Harap masukkan nilai antara 1 dan 40.')

        except ValueError:
            QMessageBox.warning(self, 'Error', 'Masukkan nilai seats
yang valid.')

        except Exception as e:
            QMessageBox.warning(self, 'Error', f'Gagal menyimpan data
film: {str(e)}')

```

Tujuan : Mewakili dialog untuk memasukkan detail film.

Kegunaan dan Fungsi :

- Memungkinkan admin untuk memasukkan detail seperti judul film, genre, jam\_tayang (waktu pemutaran), dan jumlah kursi.
- Memvalidasi input dan memasukkan data film ke dalam database.
- Menampilkan peringatan jika terjadi kesalahan pada input atau interaksi database.

### 3.1.4 Class Edit Film Dialog

```
class EditFilmDialog(QDialog):
    def __init__(self, film_list, selected_item, db_connection,
parent=None):
        super(EditFilmDialog, self).__init__(parent)
        self.film_list = film_list
        self.selected_item = selected_item
        self.db_connection = db_connection

        self.setWindowTitle('Edit Film')
        self.setGeometry(200, 200, 300, 150)

        layout = QFormLayout()

        self.input_judul = QLineEdit(self)

self.input_judul.setText(selected_item.text().split(',')[0].split(':
')[1]) # Ambil judul film dari item terpilih
        layout.addRow('Judul Film:', self.input_judul)

        self.input_genre = QLineEdit(self)

self.input_genre.setText(selected_item.text().split(',')[1].split(':
')[1]) # Ambil genre film dari item terpilih
        layout.addRow('Genre Film:', self.input_genre)

        self.input_jam_tayang = QLineEdit(self)

self.input_jam_tayang.setText(selected_item.text().split(',')[2].split(
': ')[1]) # Ambil jam tayang film dari item terpilih
        layout.addRow('Jam Tayang:', self.input_jam_tayang)

        self.input_seats = QLineEdit(self)
```

```

self.input_seats.setText(selected_item.text().split(',')[3].split(':')
')[1]) # Ambil seats film dari item terpilih
    layout.addRow('Seat:', self.input_seats)

    button_save = QPushButton('Simpan', self)
    button_save.clicked.connect(self.on_save_click)
    layout.addRow(button_save)

    self.setLayout(layout)

def on_save_click(self):
    judul = self.input_judul.text()
    genre = self.input_genre.text()
    jam_tayang = self.input_jam_tayang.text()
    seats = self.input_seats.text()

    # Remove the existing item from the list widget
    row = self.film_list.row(self.selected_item)
    self.film_list.takeItem(row)

    # Add a new item with the updated film data
    film_data = f'Judul : {judul}, Genre : {genre}, Jam Tayang :
{jam_tayang}, Seats : {seats}'
    item = QListWidgetItem(film_data)
    self.film_list.insertItem(row, item)
    self.film_list.sortItems()

    # Update film data in the database
    try:
        with self.db_connection.cursor() as cursor:
            # SQL query to update data in the 'film' table (adjust
the table name if needed)
            sql_update_film = "UPDATE film SET judul=%s, genre=%s,
jam_tayang=%s, seats=%s WHERE judul=%s"
            film_values = (judul, genre, jam_tayang, seats,
self.selected_item.text().split(',')[0].split(': ')[1])
            cursor.execute(sql_update_film, film_values)

            self.db_connection.commit()
            QMessageBox.information(self, 'Success', 'Film data updated
successfully.')

```

```
except pymysql.Error as e:
    self.show_database_error(e)

self.accept()

def show_database_error(self, error):
    QMessageBox.warning(self, 'Database Error', f'Database error:
{error}')
```

Tujuan: Mewakili dialog untuk mengedit detail film.

Kegunaan dan Fungsi:

Mirip dengan FilmInputDialog, tetapi dirancang untuk mengedit detail film yang sudah ada. Mengambil detail film yang dipilih, mengizinkan admin untuk mengeditnya, dan memperbarui daftar dan database. Menampilkan peringatan jika terjadi kesalahan.

### 3.1.5 Class Admin Interface

```

class AdminInterface(QMainWindow):
    def __init__(self, db_connection=None):
        super(AdminInterface, self).__init__()

        self.setWindowTitle('Admin Interface')
        self.setGeometry(300, 300, 400, 200)

        menubar = self.menuBar()
        film_menu = menubar.addMenu('Film')

        input_film_action = QAction('Input Film', self)
        input_film_action.triggered.connect(self.input_film)
        film_menu.addAction(input_film_action)

        self.film_list = QListWidget(self)

        layout = QVBoxLayout()
        layout.addWidget(self.film_list)

        central_widget = QWidget()
        central_widget.setLayout(layout)
        self.setCentralWidget(central_widget)

        button_edit_film = QPushButton('Edit Film', self)
        button_edit_film.clicked.connect(self.on_edit_film_click)
        layout.addWidget(button_edit_film)

        button_keluar = QPushButton('Keluar', self)
        button_keluar.clicked.connect(self.on_keluar_click)
        layout.addWidget(button_keluar)

```

```

        button_hapus_film = QPushButton('Hapus Film', self)
        button_hapus_film.clicked.connect(self.on_hapus_film_click)
        layout.addWidget(button_hapus_film)

        self.db_connection = db_connection
        self.load_data_from_database()

    def load_data_from_database(self):
        try:
            with self.db_connection.cursor() as cursor:
                sql = "SELECT * FROM film"
                cursor.execute(sql)
                films = cursor.fetchall()

                for film in films:
                    item_text = f"Judul: {film['judul']}, Genre: {film['genre']}, Jam Tayang: {film['jam_tayang']}, Seat: {film['seats']}"
                    item = QListWidgetItem(item_text)
                    self.film_list.addItem(item)

        except Exception as e:
            QMessageBox.warning(self, 'Error', f'Failed to load film data from database: {str(e)}')

    def on_edit_film_click(self):
        selected_item = self.film_list.currentItem()
        if selected_item:
            edit_dialog = EditFilmDialog(self.film_list, selected_item, self.db_connection)
            edit_dialog.exec_()

```

```

def on_keluar_click(self):
    # Implementasi fungsi keluar ke layar sebelumnya di sini
    # Misalnya, bisa menggunakan self.close() atau self.hide()
    self.close() # Ini akan menutup jendela AdminInterfac

def input_film(self):
    film_input_dialog = FilmInputDialog(self.film_list, self.db_connection, self)
    film_input_dialog.exec_()
    # Save film data to file after input

def on_hapus_film_click(self):
    selected_item = self.film_list.currentItem()
    if selected_item:
        judul_film = selected_item.text().split(',')[0].split(': ')[1]
        self.hapus_film_dari_database(judul_film)
        self.film_list.takeItem(self.film_list.row(selected_item))

def hapus_film_dari_database(self, judul_film):
    try:
        with self.db_connection.cursor() as cursor:
            sql = "DELETE FROM film WHERE judul = %s"
            cursor.execute(sql, (judul_film,))
            self.db_connection.commit()
            QMessageBox.information(self, 'Hapus Film', 'Film berhasil dihapus dari database.')
    except Exception as e:
        QMessageBox.warning(self, 'Error', f'Gagal menghapus film dari database: {str(e)}')

def show_search_film_dialog(self):
    search_film_dialog = SearchFilmDialog(self.film_list, self)
    search_film_dialog.exec_()

```

Tujuan: Mewakili jendela antarmuka admin utama.

Kegunaan dan Fungsi:

Menampilkan daftar film dan menyediakan tombol untuk tindakan seperti memasukkan, mengedit, dan menghapus film. Memungkinkan admin untuk berinteraksi dengan database film. Memanfaatkan dialog untuk input dan pengeditan.



### 3.1.6 Class SearchFilmDialog

```
261 class SearchFilmDialog(QDialog):
262     def __init__(self, film_list, db_connection=None, parent=None):
263         super(SearchFilmDialog, self).__init__(parent)
264         self.film_list = film_list
265         self.db_connection = db_connection
266
267         self.setWindowTitle('Cari Film')
268         self.setGeometry(200, 200, 300, 150)
269
270         self.setup_ui()
271
272     def setup_ui(self):
273         layout = QVBoxLayout()
274
275         self.film_list_widget = QListWidget(self)
276         self.film_list_widget.itemClicked.connect(self.show_selected_film)
277         layout.addWidget(self.film_list_widget)
278
279         self.search_bar = QLineEdit(self)
280         layout.addWidget(self.search_bar)
281
282         button_search = QPushButton('Cari', self)
283         button_search.clicked.connect(self.load_film_from_database)
284         layout.addWidget(button_search)
285
286         self.load_film_from_database()
287         self.setLayout(layout)
288
289     def load_film_from_database(self):
290         self.film_list_widget.clear()
291         title = self.search_bar.text().strip()
292
293         try:
294             with self.db_connection.cursor(pymysql.cursors.DictCursor) as cursor:
295                 sql = "SELECT * FROM film WHERE LOWER(judul) LIKE LOWER(%s)"
296                 cursor.execute(sql, ('%' + title + '%',))
297                 films = cursor.fetchall()
298
299                 for film in films:
300                     item_text = f"Judul:{film['judul']}"
301                     item = QListWidgetItem(item_text)
302                     self.film_list_widget.addItem(item)
303
304         except Exception as e:
305             print(f"Exception in load_film_from_database: {e}")
306             self.show_database_error(e)
307
308     def search_in_database(self, title):
309         try:
310             with self.db_connection.cursor() as cursor:
311                 sql = "SELECT * FROM film WHERE judul LIKE %s"
312                 cursor.execute(sql, ('%' + title + '%',))
313                 result = cursor.fetchone()
314
315                 print("Query executed:", sql)
316                 print("Query parameters:", (title,))
317                 print("Result from database:", result)
318
319                 return result
320
321
```

```

321
322         except pymysql.Error as e:
323             self.show_database_error(e)
324             return None
325
326     def show_selected_film(self, item):
327         try:
328             split_result = item.text().split(":")
329             if len(split_result) >= 2:
330                 title = split_result[1].strip()
331                 film_data = self.search_in_database(title)
332
333                 if film_data:
334                     result_window = BookingSeatDialog(item, film_data, self.db_connection)
335                     result_window.exec_()
336                 else:
337                     QMessageBox.warning(self, 'Film Tidak Ditemukan', 'Maaf, film tidak ditemukan. Silakan coba lagi.')
338             else:
339                 QMessageBox.warning(self, 'Error', 'Format item tidak valid. Silakan pilih item film yang valid.')
340         except IndexError:
341             QMessageBox.warning(self, 'Error', 'Format item tidak valid. Silakan pilih item film yang valid.')
342

```

Tujuan: Mewakili dialog pencarian film di database.

Kegunaan dan Fungsi: Memungkinkan admin mencari film berdasarkan judulnya.

Menampilkan daftar hasil pencarian dan memungkinkan tindakan lebih lanjut seperti memesan kursi untuk film yang dipilih.

### 3.1.7 Class BookingSeatDialog

```

class BookingSeatDialog(QDialog):
    def __init__(self, selected_item, film_data, db_connection):
        super().__init__()
        self.selected_item = selected_item
        self.film_data = film_data
        self.db_connection = db_connection
        self.selected_seats = set()

        self.setWindowTitle('Booking Kursi')
        self.setGeometry(200, 200, 300, 150)

        # Pemanggilan langsung ke show_seat_selection saat instance dibuat
        self.show_seat_selection()

    def show_seat_selection(self):
        seat_selection_dialog = QDialog(self)
        seat_selection_dialog.setWindowTitle("Pilih Kursi")

        layout = QVBoxLayout()

        grid_layout = QGridLayout()

        for row in range(5):
            for col in range(8):
                seat_button = QPushButton(f"Seat {row * 8 + col + 1}")
                seat_button.setStyleSheet("background-color: green;")
                seat_button.clicked.connect(self.toggle_seat)

```

```

        # Create a placeholder widget to hold the button in the grid
        placeholder_widget = QWidget()
        placeholder_widget.setSizePolicy(seat_button.sizePolicy())
        placeholder_widget.setObjectName(seat_button.objectName())
        placeholder_layout = QVBoxLayout(placeholder_widget)
        placeholder_layout.addWidget(seat_button)

        grid_layout.addWidget(placeholder_widget, row, col)

    layout.addLayout(grid_layout)

    confirm_button = QPushButton("Next")
    confirm_button.clicked.connect(seat_selection_dialog.accept)
    layout.addWidget(confirm_button)

    seat_selection_dialog.setLayout(layout)

    result = seat_selection_dialog.exec_()
    if result == QDialog.Accepted:
        self.show_price_page()

def toggle_seat(self):
    button = self.sender()
    current_color = button.palette().button().color().name()

    if current_color == "green":
        if len(self.selected_seats) < int(self.film_data['seats']):
            # Button is green and available, book the seat
            button.setStyleSheet("background-color: green;") # Change to red to indicate selected
            self.selected_seats.add(button.text())
        else:
            QMessageBox.warning(self, 'Warning', 'All seats are already booked.', QMessageBox.Ok)
            QMessageBox.warning(self, 'Warning', 'All seats are already booked.', QMessageBox.Ok)
    else:
        # Button is red, unbook the seat
        button.setStyleSheet("background-color: red;")
        if button.text() in self.selected_seats:
            self.selected_seats.remove(button.text())

def show_price_page(self):
    result_dialog = ResultDialog(self.film_data, self.selected_seats, self.db_connection)
    result = result_dialog.exec_()

    if result == QDialog.Accepted:
        # Perform additional actions if needed after the result dialog is accepted
        self.show_seat_selection() # Call the show_seat_selection method again to go back to the seat selection

```

Tujuan: Mewakili dialog pemesanan kursi untuk film yang dipilih.

Kegunaan dan Fungsi: Menampilkan kisi kursi yang tersedia untuk film yang dipilih. Memungkinkan pengguna untuk memilih dan memesan kursi. Lanjutkan ke halaman pembayaran setelah pemilihan kursi.

### 3.1.8 Class Result Dialog

```
class ResultDialog(QDialog):
    def __init__(self, film_data, selected_seats, db_connection):
        super().__init__()
        self.db_connection = db_connection
        self.film_data = film_data
        self.selected_seats = set()
        self.selected_seats = selected_seats

        self.setWindowTitle('Strukt Pembayaran')
        self.setGeometry(200, 200, 300, 150)

        self.setup_ui()

    def setup_ui(self):
        layout = QVBoxLayout()

        title_label = QLabel(f"Judul: {self.film_data['judul']}")
        layout.addWidget(title_label)

        time_label = QLabel(f"Jam Tayang: {self.film_data['jam_tayang']}")
        layout.addWidget(time_label)

        seats_label = QLabel()
        layout.addWidget(seats_label)
        self.update_seats_label(seats_label)

        price_label = QLabel("Harga: Rp 50,000") # Adjust as needed
        layout.addWidget(price_label)

        ok_button = QPushButton("Oke")
        ok_button.clicked.connect(self.purchase_ticket) # Connect to self.accept
        layout.addWidget(ok_button)

        self.setLayout(layout)

    def update_seats_label(self, label):
        available_seats = int(self.film_data['seats'])
        if available_seats > 0:
            label.setText(f"Seats: {available_seats} (Available)")
            label.setStyleSheet("color: green; font-weight: bold;")
        else:
            label.setText("Seats: Sold Out")
            label.setStyleSheet("color: red; font-weight: bold;")

    def purchase_ticket(self):
        try:
            with self.db_connection.cursor() as cursor:
                # Check available seats
                sql_check_seats = "SELECT seats FROM film WHERE judul = %s"
                cursor.execute(sql_check_seats, (self.film_data['judul'],))
                result = cursor.fetchone()

                if result and int(result['seats']) > 0:
                    # Update available seats
                    sql_update_seats = "UPDATE film SET seats = seats - 1 WHERE judul = %s"
                    cursor.execute(sql_update_seats, (self.film_data['judul'],))

                    # Insert purchase data into the 'purchases' table (example table name)
                    sql_insert_purchase = """
                    INSERT INTO purchases (judul, genre, jam_tayang, purchased_at)
                    VALUES (%s, %s, %s, %s)
                    """
```

```

        INSERT INTO purchases (judul, genre, jam_tayang, purchased_at)
        VALUES (%s, %s, %s, %s)
        """
        purchase_data = (
            self.film_data['judul'],
            self.film_data['genre'],
            self.film_data['jam_tayang'],
            datetime.now(),
        )
        cursor.execute(sql_insert_purchase, purchase_data)

        self.db_connection.commit()
        QMessageBox.information(self, 'Success', f'Ticket purchased for {self.film_data["judul"]}.'.)
    else:
        QMessageBox.warning(self, 'Warning', 'No available seats for the selected time.')

except pymysql.Error as e:
    self.show_database_error(e)

```

Tujuan: Merupakan dialog yang menunjukkan hasil pemilihan kursi dan rincian pembayaran.

Kegunaan dan Fungsi: Menampilkan detail seperti judul film, waktu pemutaran, kursi yang dipilih, dan total harga. Memberikan opsi untuk membeli tiket, memperbarui database yang sesuai.

### 3.1.9 Class Pemesanan Tiket Bioskop

```

class PemesananTiketBioskop(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle('Pemesanan Tiket Bioskop')
        self.setGeometry(100, 100, 400, 200)

        self.customer_name = ''
        self.customer_email = ''
        self.film_list = QListWidget(self)

        self.init_ui()

    def init_ui(self):
        layout = QVBoxLayout()

        label_name = QLabel("Name:")
        self.input_name = QLineEdit()
        layout.addWidget(label_name)
        layout.addWidget(self.input_name)

        label_email = QLabel("Email:")
        self.input_email = QLineEdit()
        layout.addWidget(label_email)
        layout.addWidget(self.input_email)

        login_button = QPushButton("Masuk")
        login_button.clicked.connect(self.on_masuk_click)
        layout.addWidget(login_button)

        button_keluar = QPushButton('Keluar', self)
        button_keluar.clicked.connect(self.close)

```

```

        button_keluar.clicked.connect(self.close)
        layout.addWidget(button_keluar)

        central_widget = QWidget()
        central_widget.setLayout(layout)
        self.setCentralWidget(central_widget)

        menubar = self.menuBar()
        admin_menu = menubar.addMenu('Admin')

        admin_login_action = QAction('Admin Login', self)
        admin_login_action.triggered.connect(self.show_admin_login_dialog)
        admin_menu.addAction(admin_login_action)

        self.admin_interface = None

        self.db_connection = pymysql.connect(
            host='localhost',
            user='root',
            password='',
            database='tiket_bioskop',
            charset='utf8mb4',
            cursorclass=pymysql.cursors.DictCursor
        )

    def show_admin_login_dialog(self):
        admin_login_dialog = AdminLoginDialog(self)
        if admin_login_dialog.exec_() == QDialog.Accepted:
            self.admin_interface = AdminInterface(self.db_connection)
            self.admin_interface.show()

```

```

    def on_masuk_click(self):
        self.customer_name = self.input_name.text()
        self.customer_email = self.input_email.text()

        if self.customer_name and self.customer_email:
            message = f"Selamat datang, {self.customer_name}!\n" \
                    f"Email: {self.customer_email}"
            QMessageBox.information(self, 'Informasi', message)
            self.show_search_film_dialog()
        else:
            QMessageBox.warning(self, 'Peringatan', 'Mohon isi nama dan email terlebih dahulu.')

    def show_search_film_dialog(self):
        search_film_dialog = SearchFilmDialog(self.film_list, self.db_connection, self)
        search_film_dialog.exec_()

    def closeEvent(self, event):
        if hasattr(self, 'db_connection') and self.db_connection:
            self.db_connection.close()
        event.accept()

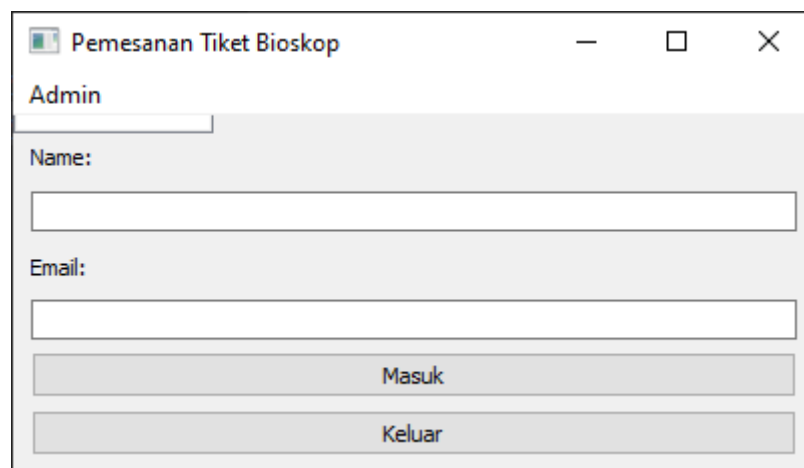
__name__ == '__main__':
    app = QApplication(sys.argv)
    window = PemesananTiketBioskop()
    window.show()
    sys.exit(app.exec_())

```

Tujuan: Mewakili jendela antarmuka pelanggan utama. Kegunaan dan Fungsi: Memungkinkan pelanggan memasukkan nama dan email mereka. Menyediakan tombol untuk login sebagai admin. Memungkinkan pelanggan mencari dan memesan tiket film.

### 3.2 Implementasi Hasil

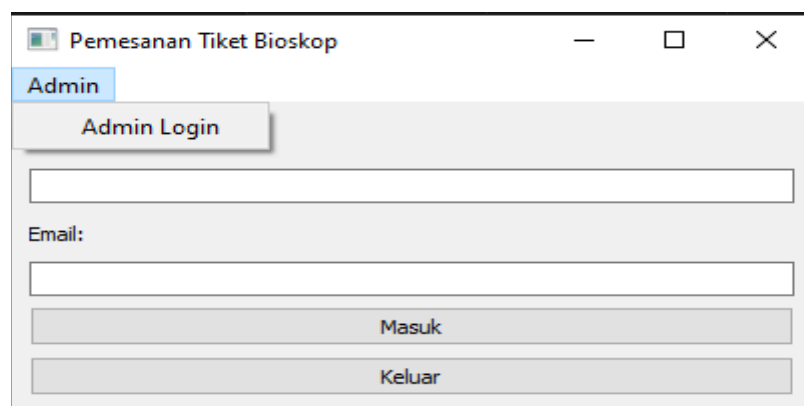
Implementasi hasil merupakan representasi dari keseluruhan kegiatan dan metode ilmiah yang diterapkan selama penelitian. Hasil dari penelitian ini adalah program pemesanan tiket bioskop,



The screenshot shows a window titled "Pemesanan Tiket Bioskop". Inside, there is a section labeled "Admin" with a text input field. Below this, there are two more text input fields labeled "Name:" and "Email:". At the bottom of this section, there are two buttons: "Masuk" (Login) and "Keluar" (Logout).

#### 3.1.1 Menu Admin

Menu admin adalah sebuah fitur dimana hanya admin bioskop yang dapat login kedalam fitur ini. Dengan cara menekan aksi Admin Login

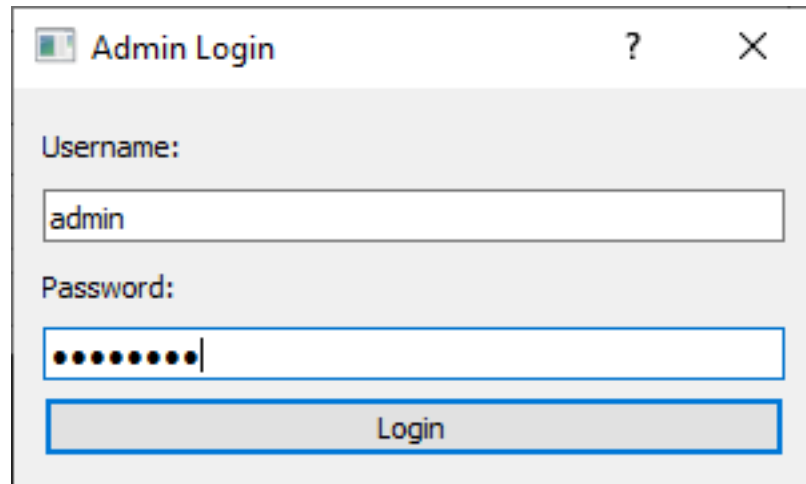


The screenshot shows the same window as before, but with a blue highlight on the "Admin" section. A dropdown menu is visible, showing the option "Admin Login". The "Name:" and "Email:" input fields and the "Masuk" and "Keluar" buttons are still present below.

Gambar 3.1.1 Menubar fitur admin login.

#### 3.1.2 Admin Login

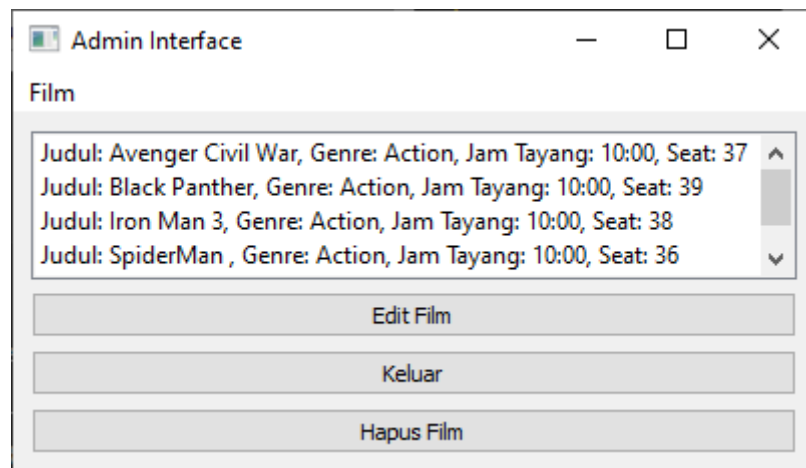
Admin login adalah halaman dimana kita sebagai admin harus memasukkan username dan password yang dimiliki oleh admin yang digunakan untuk login. bila data yang dimasukkan benar maka kita sebagai admin dapat login dan masuk ke halaman selanjutnya..



Gambar 3.2.1 admin melakukan login

### 3.1.3 Admin Interface

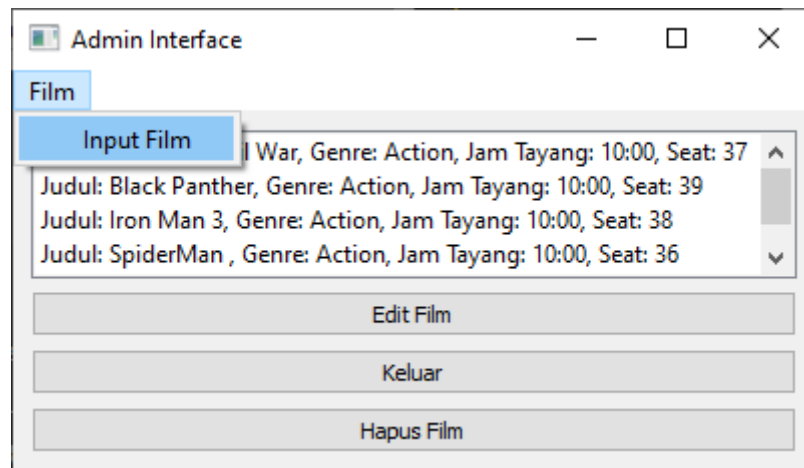
Admin interface adalah halaman dashboard yang hanya dapat diakses oleh admin apabila sudah berhasil login. terdapat tombol keluar yang berguna untuk kembali ke halaman admin login.



### 3.1.4 Input Film

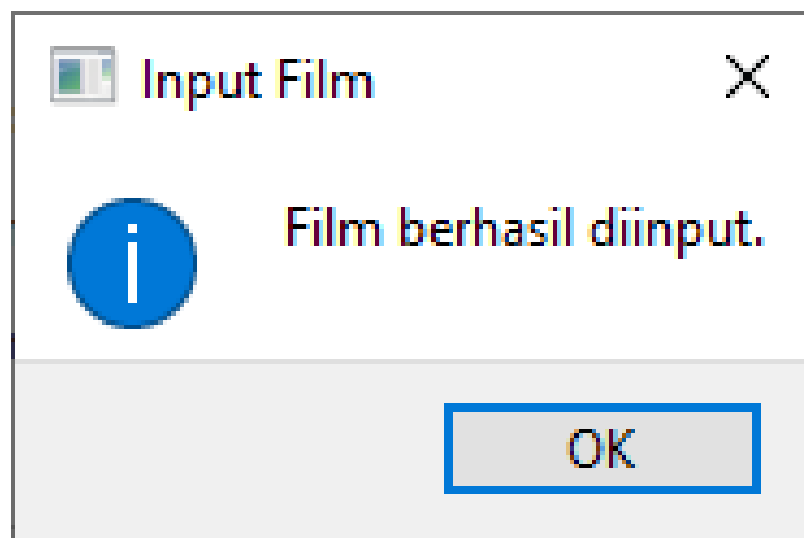
Input film merupakan fitur yang dapat diakses oleh admin, dimana fitur ini digunakan oleh admin untuk menambahkan film, genre film, jam tayang dan jumlah tempat duduk. pada fitur ini admin juga bisa melakukan edit dan hapus pada data yang telah diinputkan sebelumnya. data yang telah di input juga tersimpan pada database.





Gambar 3.4.1 Tampilan fitur Input Film.

Gambar 3.4.2 Menambahkan film yang akan ditayangkan



Gambar 3.4.3 Film berhasil di input.

**Admin Interface**

Film

Judul: Avenger Civil War, Genre: Action, Jam Tayang: 10:00, Seat: 37  
 Judul: Black Panther, Genre: Action, Jam Tayang: 10:00, Seat: 39  
 Judul: Budi Pekerti, Genre: Drama, Jam Tayang: 15.30, Seats: 40  
 Judul: Iron Man 3, Genre: Action, Jam Tayang: 10:00, Seat: 38

Edit Film

Keluar

Hapus Film

Gambar 3.4.4 Film yang di inputkan masuk ke dalam data.

Server: 127.0.0.1 - Database: tiket\_bioskop - Tabel: tbl\_film

Menampilkan baris 0 - 5 (total 6, Pencarian dilakukan dalam 0.0009 detik)

SELECT \* FROM `tbl\_film`

Profil [ Edit direktori ] [ Ubah ] [ Jalankan SQL ] [ Buat kode PHP ] [ Segarkan ]

Tampilkan semua Jumlah baris: 25 Saring baris: Cari di tabel ini Sort by key: Tidak ada

	id	Judul	genre	Jam tayang	seats
Ubah Salin Hapus	1	Avenger Civil War	Action	10.00	37
Ubah Salin Hapus	2	Black Panther	Action	10.00	39
Ubah Salin Hapus	3	Iron Man 3	Action	10.00	38
Ubah Salin Hapus	4	SpiderMan	Action	10.00	36
Ubah Salin Hapus	6	Siksa Neraka	Horror	12.00	40
Ubah Salin Hapus	7	Budi Pekerti	Drama	15.30	40

Pilih Semua Dengan pilihan: Ubah Salin Hapus Ekspor

Tampilkan semua Jumlah baris: 25 Saring baris: Cari di tabel ini Sort by key: Tidak ada

Kontrol hasil query

Gambar 3.4.5 Data film masuk ke dalam database.

**Edit Film**

Judul Film: Budi Pekerti

Genre Film: Drama

Jam Tayang: 15.30

Seat: 40

Simpan

Gambar 3.4.6 Sebelum melakukan pengeditan

**Edit Film**

Judul Film: Budi Pekerti

Genre Film: Drama

Jam Tayang: 16.00

Seat: 40

Simpan

Gambar 3.4.7 Melakukan pengeditan pada jam tayang

**Success**

Information icon: Film data updated successfully.

OK

Gambar 3.4.8 Berhasil melakukan pengeditan

**Admin Interface**

Film

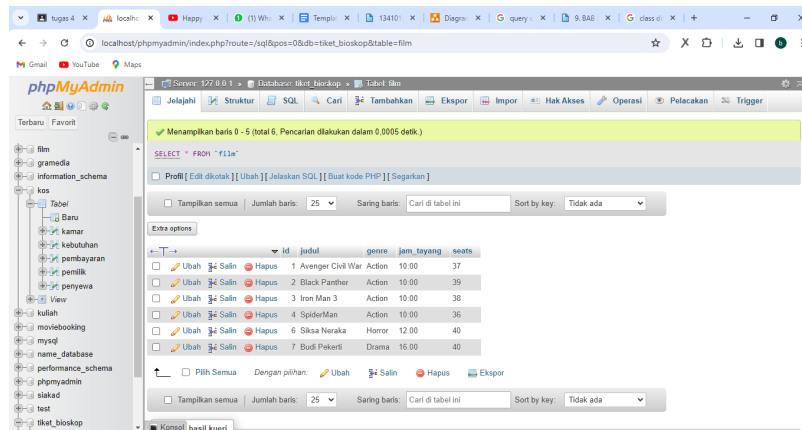
Judul : Budi Pekerti, Genre : Drama, Jam Tayang : 16.00, Seats : 40
Judul: Avenger Civil War, Genre: Action, Jam Tayang: 10:00, Seat: 37
Judul: Black Panther, Genre: Action, Jam Tayang: 10:00, Seat: 39
Judul: Iron Man 3, Genre: Action, Jam Tayang: 10:00, Seat: 38

Edit Film

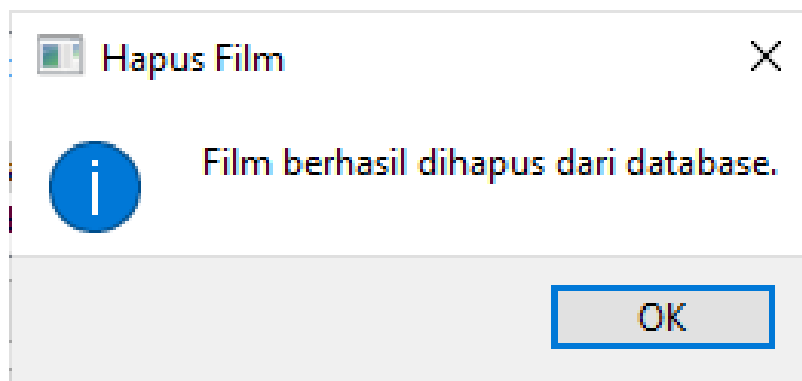
Keluar

Hapus Film

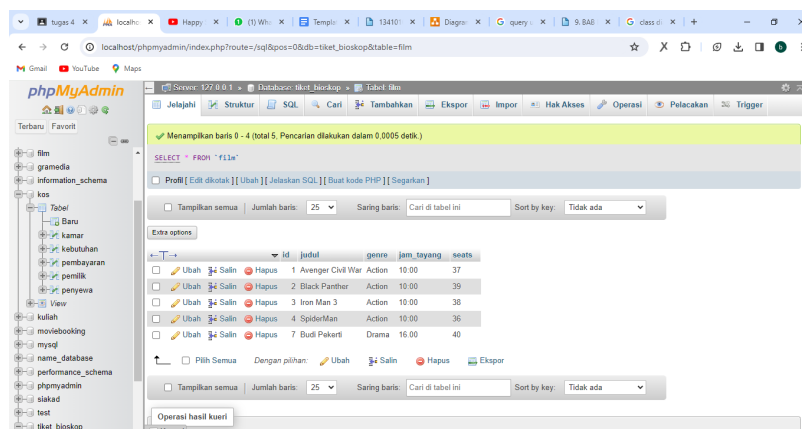
Gambar 3.4.9 Tampilan data film setelah di edit.



Gambar 3.4.10 Tampilan database setelah mengalami pengeditan pada program.



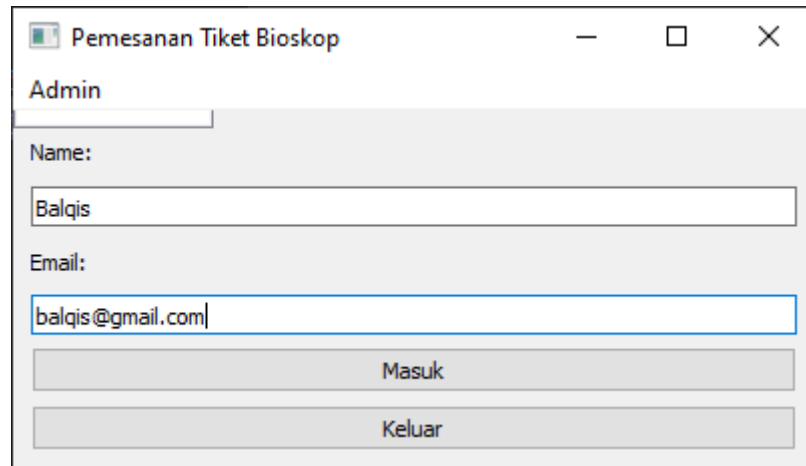
Gambar 3.4.11 Menghapus film



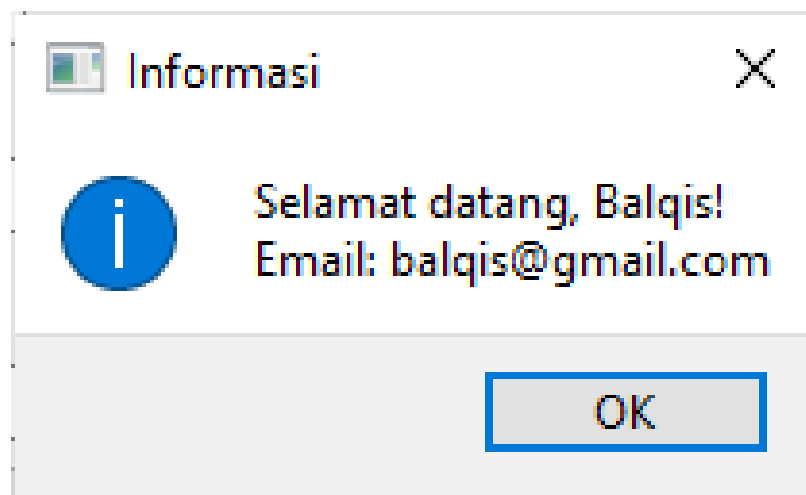
Gambar 3.4.12 Tampilan database setelah film berhasil dihapus.

### 3.1.5 Login Pembeli

Login pembeli adalah halaman dimana pembeli tiket bioskop harus mengisi nama dan email pembeli sebelum melakukan transaksi pembelian tiket. Bila pembeli telah mengisi nama dan email maka pembeli dapat menekan tombol masuk dan akan keluar pesan bahwa pembeli atas nama dan email yang dimiliki berhasil masuk..



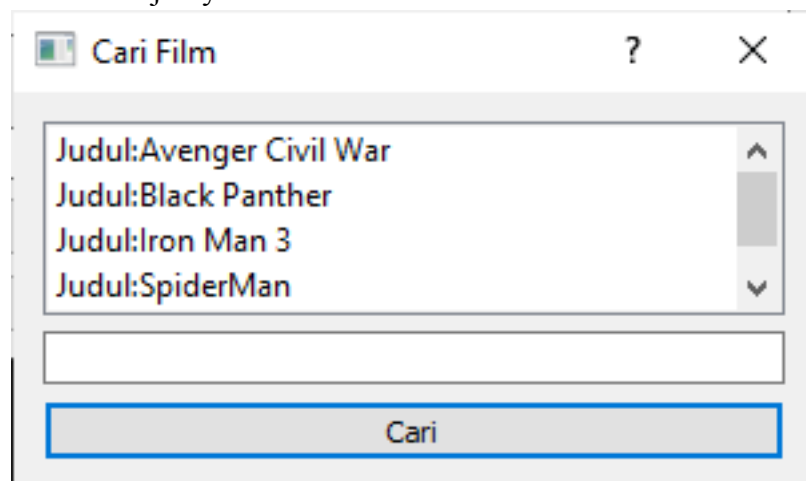
Gambar...pembeli mengisi data diri untuk memesan tiket.



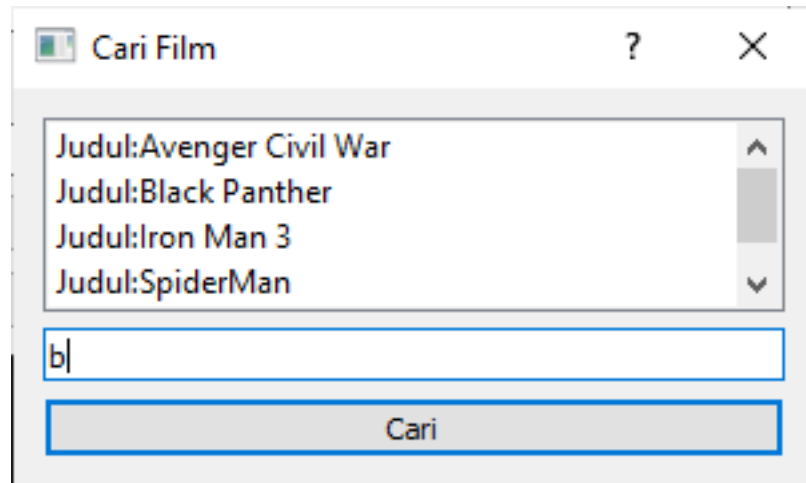
Gambar ... Pembeli berhasil masuk.

### 3.1.6 Cari Film

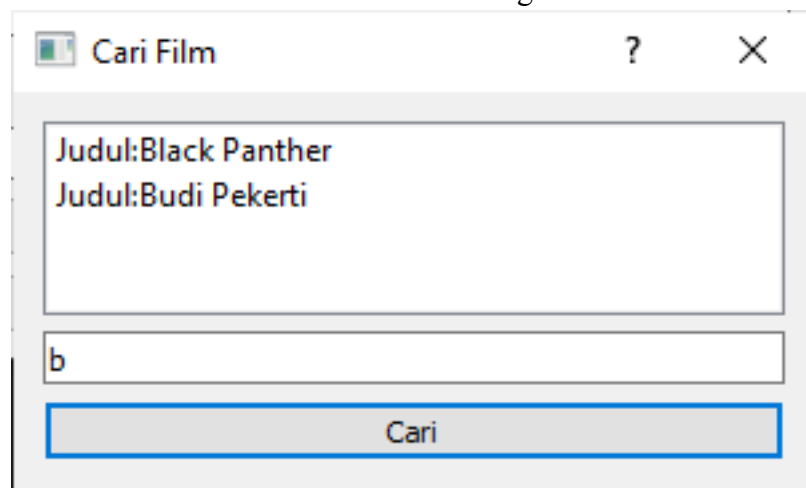
Cari film adalah halaman dimana pembeli dapat melihat apa saja film yang ditayangkan oleh bioskop. Pada halaman ini terdapat fitur cari yang berguna untuk memudahkan pembeli untuk mencari film yang akan di tonton. Jika film yang diinginkan ada pembeli cukup menekan satu kali pada film yang dinginkan dengan begitu pembeli akan dibawa masuk ke halaman selanjutnya.



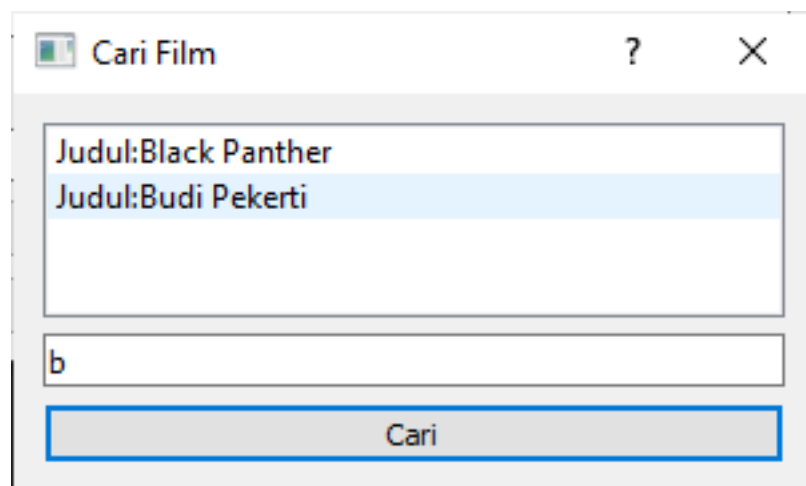
Gambar .. Pilihan film.



Gambar... mencari film dengan huruf b.



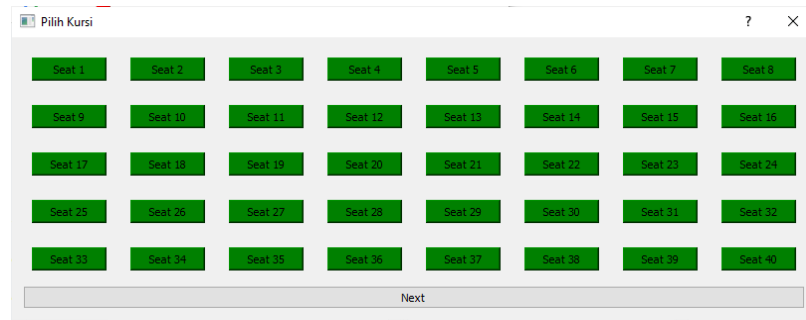
Gambar ... Menampilkan hasil pencarian film yang memiliki huruf b.



Gambar ... Memilih film.

### 3.1.7 Pilih Kursi

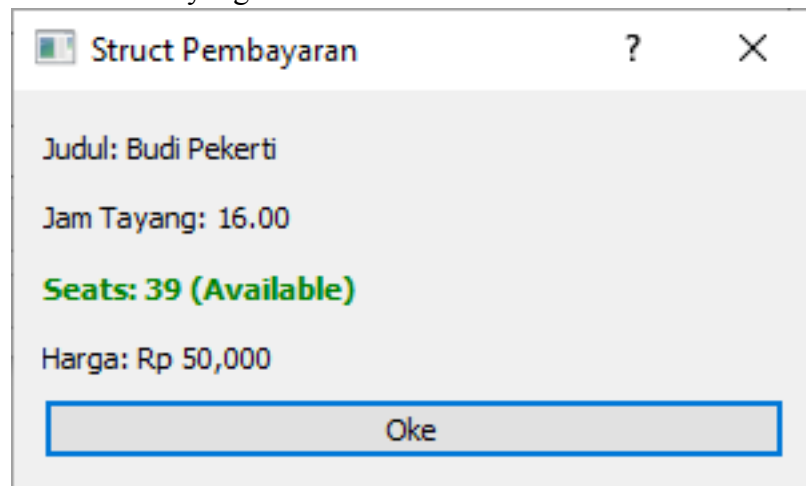
Pilih kursi adalah halaman dimana pembeli yang sebelumnya telah memilih film yang akan ditonton dapat memilih tempat duduk yang diinginkan. Dengan cara menekan kursi yang berwarna hijau lalu menekan tombol next pembeli berhasil memilih tempat duduk.



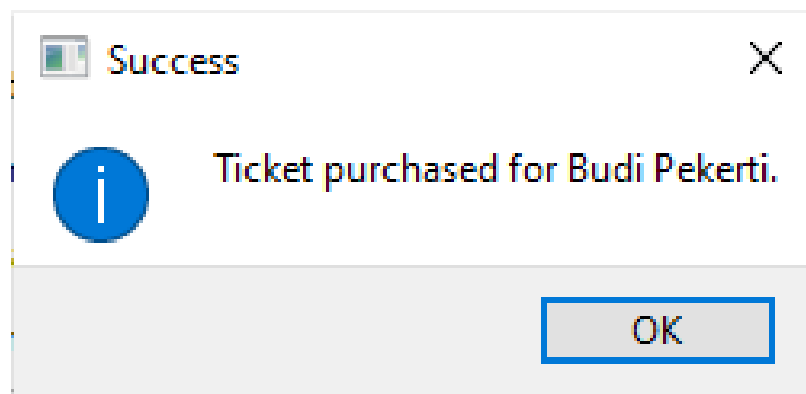
Gambar ... Memilih Kursi.

### 3.1.8 Struk Pembayaran

Struk pembayaran merupakan hasil dari pembeli yang telah memilih film dan tempat duduk tiket bioskop. Disini juga terdapat tampilan pembayaran yang harus dibayar oleh pembeli. Bila pembeli telah menekan tombol oke maka akan keluar pesan bahwa tiket telah berhasil dibeli. Dan data tiket film yang dibeli masuk ke dalam database.



Gambar ...



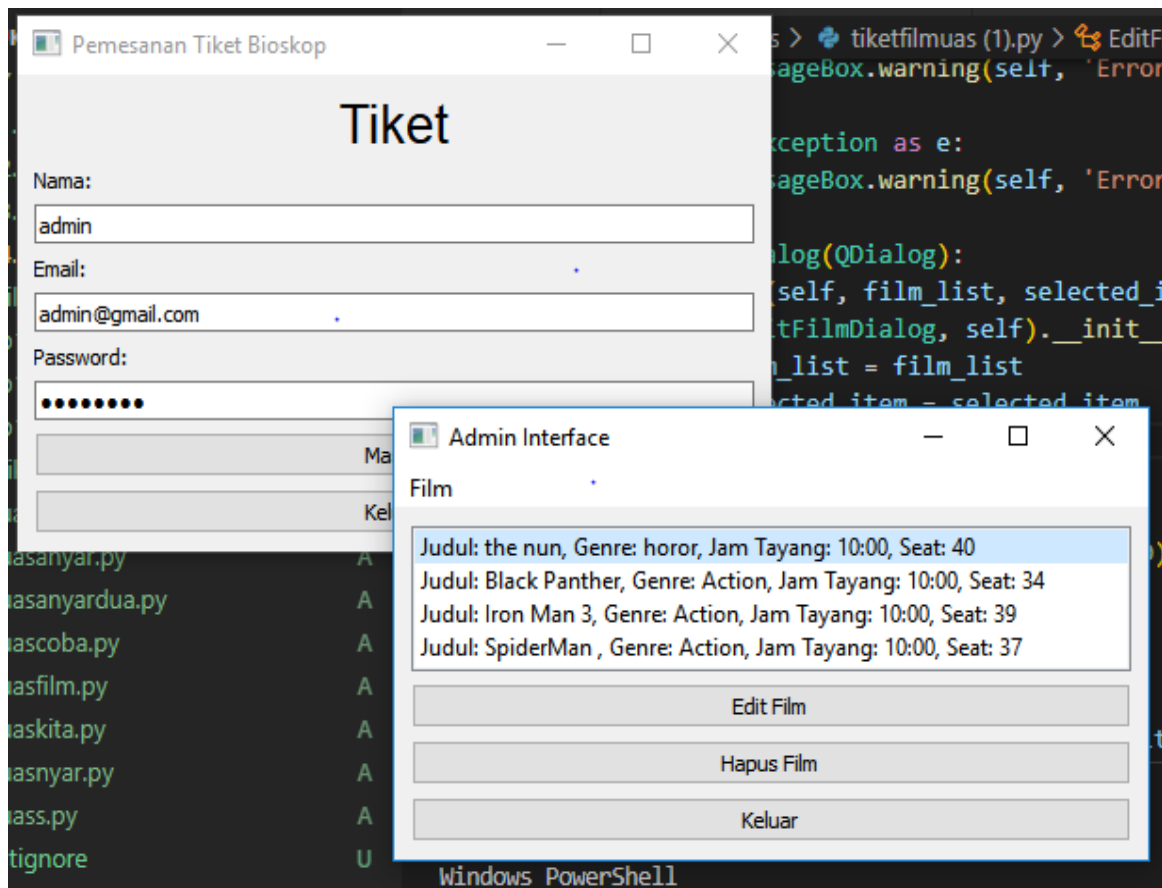
Gambar ... Tiket berhasil di pesan,

id	judul	genre	jam_tayang	purchased_at
1	Avenger Civil War	Action	10:00	2023-12-27 21:01:50
2	Avenger Civil War	Action	10:00	2023-12-27 21:13:42
3	Iron Man 3	Action	10:00	2023-12-29 18:26:35
4	Iron Man 3	Action	10:00	2023-12-29 18:26:40
5	SpiderMan	Action	10:00	2023-12-29 18:27:11
6	SpiderMan	Action	10:00	2023-12-29 18:28:01
7	SpiderMan	Action	10:00	2023-12-29 18:28:31
8	SpiderMan	Action	10:00	2023-12-29 18:28:36
9	Budi Pekerti	Drama	16:00	2024-01-02 23:45:11
10	Iron Man 3	Action	10:00	2024-01-03 00:01:39
11	SpiderMan	Action	10:00	2024-01-03 00:13:37
12	Budi Pekerti	Drama	16:00	2024-01-03 10:10:07

Gambar ... Tampilan database tiket yang berhasil dibeli.

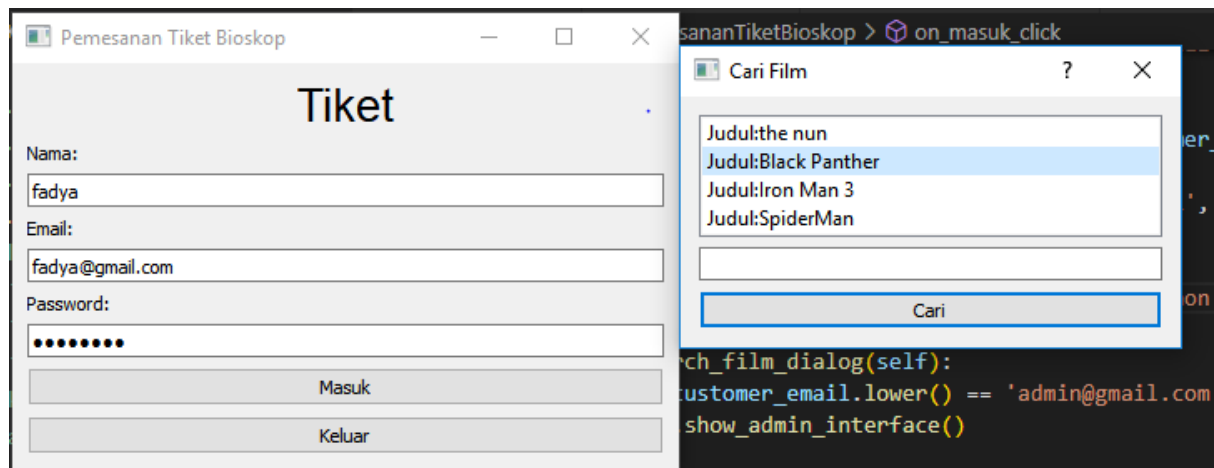
### 3.2 Challenge Mengubah Tampilan

Mengubah dua layout yakni admin dan pemesan, menjadi satu tampilan layout



Gambar 3.2.1 Tampilan jika admin akan masuk.





Gambar 3.2.2 Tampilan jika pemesan akan masuk.

## **BAB IV PENUTUP**

### **4.1 Kesimpulan**

Dari hasil implementasi dan pengujian program pemesanan tiket bioskop online maka dapat disimpulkan dengan adanya

### **4.2 Saran**

Pada program pemesanan tiket bioskop online ini masih belum sempurna disebabkan beberapa hal yang pertama saat admin melakukan input seats yaitu jumlah tempat duduk penonton, tidak dapat sama dengan tampilan output. Yang kedua untuk pembelian tiket lebih dari satu kami belum bisa melakukan penghitungan secara otomatis pada program. Yang ketiga pada saat kursi telah dipilih oleh pembeli sebelumnya tampilan warna untuk kursi yang telah terisi tidak berwarna merah dan kursi tetap masih bisa dipilih oleh pembeli lain.

## DAFTAR PUSTAKA

**Ludfiandy Romadhony.** 2018. *“Rancangan Bangunan Sistem Informasi Rumah Kost Online Berbasis Web Pada Startup Borhouse”*.  
<https://repository.dinamika.ac.id/id/eprint/3772/1/13410100197-2018-STIKOMSURABAYA.pdf> pada tanggal 25 desember 2023. CONTOH PENULISAN DAFPUS

[https://repository.uin-suska.ac.id/16744/9/9.%20BAB%20IV\\_201860SIF.pdf](https://repository.uin-suska.ac.id/16744/9/9.%20BAB%20IV_201860SIF.pdf)

[https://repository.maranatha.edu/14817/2/0872033\\_Chapter1.pdf](https://repository.maranatha.edu/14817/2/0872033_Chapter1.pdf)