

# **PROPOSAL PROJECT APLIKASI TIKET BIOSKOP**

7



**Disusun Oleh :**

- 1. Fitria Suci Aliya Wijayanti (25031554071)**
- 2. Mochammad Fawwaz Muslih (25031554126)**
- 3. Salum Nur Anisa Septriani (25031554046)**

**Dosen Pengampu:**

**Hasanuddin Al-Habib, S.Si., M.Si**

**PROGRAM STUDI SAINS DATA  
FAKULTAS MATEMATIKA DAN ILMU PENGETEHUDAN ALAM  
UNIVERSITAS NEGERI SURABAYA  
2025**

# DAFTAR ISI

<b>BAB 1</b>	<b>2</b>
<b>PENDAHULUAN</b>	<b>2</b>
1.1 LATAR BELAKANG	2
1.2 RUMUSAN MASALAH	3
1.3 TUJUAN	4
<b>BAB 2</b>	<b>5</b>
<b>ANALISIS DAN PERANCANGAN</b>	<b>5</b>
2.1 ANALISIS KEBUTUHAN APLIKASI	5
2.2 FLOWCHART	7
<b>BAB 3</b>	<b>8</b>
<b>IMPLEMENTASI</b>	<b>8</b>
3.1 PENJELASAN KODE PROGRAM	8
3.2 SCREENSHOOT APLIKASI	44
<b>BAB 4</b>	<b>48</b>
<b>LAMPIRAN</b>	<b>48</b>
4.1 KODE PROGRAM LENGKAP	48
<b>DAFTAR PUSAKA</b>	<b>49</b>

# **BAB 1**

## **PENDAHULUAN**

### **1.1 LATAR BELAKANG**

Perkembangan teknologi digital yang semakin pesat telah membawa perubahan signifikan dalam berbagai aspek kehidupan manusia, termasuk dalam industri hiburan, khususnya pada layanan bioskop. Seiring meningkatnya kebutuhan akan kecepatan, ketepatan, dan kenyamanan, sistem pemesanan tiket bioskop dituntut untuk beralih dari metode konvensional menuju sistem berbasis digital. Sistem pemesanan tiket secara manual yang masih digunakan pada beberapa tempat sering kali menimbulkan berbagai permasalahan, seperti antrean panjang, keterbatasan waktu pelayanan, kesalahan pencatatan data, serta kurang efisien dalam pengelolaan transaksi dan pemilihan kursi. Kondisi ini dapat berdampak pada menurunnya kepuasan pelanggan dan efektivitas operasional pihak pengelola bioskop.

Untuk mengatasi permasalahan tersebut, diperlukan sebuah sistem aplikasi yang mampu mengotomasi dan mendigitalisasi seluruh proses pemesanan tiket bioskop. Sistem ini mencakup proses pemilihan film, penentuan jadwal tayang, pemilihan kursi, perhitungan biaya secara otomatis, hingga pencetakan tiket sebagai bukti transaksi. Dengan adanya sistem yang terkomputerisasi, diharapkan proses pemesanan tiket dapat dilakukan dengan lebih cepat, akurat, dan terstruktur, sehingga meminimalkan kesalahan data serta meningkatkan efisiensi layanan.

Berdasarkan latar belakang tersebut, proyek ini bertujuan untuk merancang dan membangun sebuah aplikasi pencetakan tiket bioskop berbasis Graphical User Interface (GUI) menggunakan bahasa pemrograman Python dengan bantuan library. Aplikasi ini dirancang sebagai simulasi sistem pemesanan tiket yang interaktif dan mudah dioperasikan oleh pengguna. Selain itu, aplikasi ini juga menampilkan detail pemesanan tiket secara digital, seperti informasi film, jadwal tayang, nomor kursi, total harga, serta menyediakan fitur pencetakan tiket. Dengan adanya aplikasi ini, diharapkan dapat memberikan gambaran nyata mengenai penerapan teknologi pemrograman dalam mendukung digitalisasi layanan bioskop serta menjadi sarana pembelajaran dalam pengembangan aplikasi berbasis GUI.

## **1.2 RUMUSAN MASALAH**

Rumusan masalah dalam project aplikasi pencetakan tiket bioskop adalah bagaimana merancang sebuah aplikasi yang mudah digunakan oleh pengguna dalam melakukan pemesanan tiket bioskop. Aplikasi ini diharapkan mampu menampilkan informasi film secara jelas, seperti judul film, jadwal tayang, studio, harga tiket, serta menyediakan fitur pemilihan kursi agar tidak terjadi pemesanan ganda. Selain itu, diperlukan sistem yang dapat mengelola proses transaksi pembelian tiket secara otomatis dan akurat, mulai dari pemesanan hingga pencetakan tiket. Permasalahan lain yang dikaji adalah bagaimana aplikasi dapat mencetak tiket dengan format yang rapi dan informatif serta menyimpan riwayat pembelian tiket untuk keperluan pengelolaan data dan pelaporan, sehingga keseluruhan sistem dapat berjalan secara efektif dan efisien.

### **1.3 TUJUAN**

Tujuan dari pembuatan aplikasi tiket bioskop ini adalah untuk membangun sebuah

- 1.3.1 antarmuka pengguna grafis (GUI) yang dapat mensimulasikan proses pemesanan dan
- 1.3.2 pencetakan tiket secara digital. Sistem ini dirancang untuk:Memfasilitasi proses pemilihan film, jadwal, dan kursi..
- 1.3.3 menyediakan validasi data pemesanan untuk memastikan kesesuaian informasi.
- 1.3.4 menghasilkan keluaran berupa tiket digital yang menampilkan detail pemesanan.

## **BAB 2**

### **ANALISIS DAN PERANCANGAN**

#### **2.1 ANALISIS KEBUTUHAN APLIKASI**

Aplikasi Tiket Bioskop dibuat sebagai bentuk pemanfaatan teknologi untuk mendukung kegiatan pemesanan tiket film secara digital. Seiring berkembangnya teknologi dan meningkatnya kebutuhan akan layanan yang praktis, diperlukan sebuah aplikasi yang dapat membantu proses pemesanan tiket menjadi lebih mudah, tertata, dan efisien. Oleh karena itu, dilakukan analisis kebutuhan aplikasi sebagai dasar pengembangan sistem ini. Sebagai berikut :

##### **2.1.1 Perkembangan Teknologi dan Kebutuhan Layanan Digital**

- > Perkembangan teknologi mendorong perubahan cara masyarakat mengakses layanan hiburan.
- > Penggunaan aplikasi menjadi solusi yang umum dalam berbagai aktivitas sehari-hari.
- > Pemesanan tiket bioskop secara digital dinilai lebih praktis dan efisien.
- > Aplikasi tiket bioskop menjadi relevan dengan kebutuhan masyarakat saat ini.
- > Pengembangan aplikasi ini mengikuti tren pemanfaatan teknologi dalam kehidupan sehari-hari.

##### **2.1.2 Kemudahan Akses Informasi dan Pemesanan Tiket**

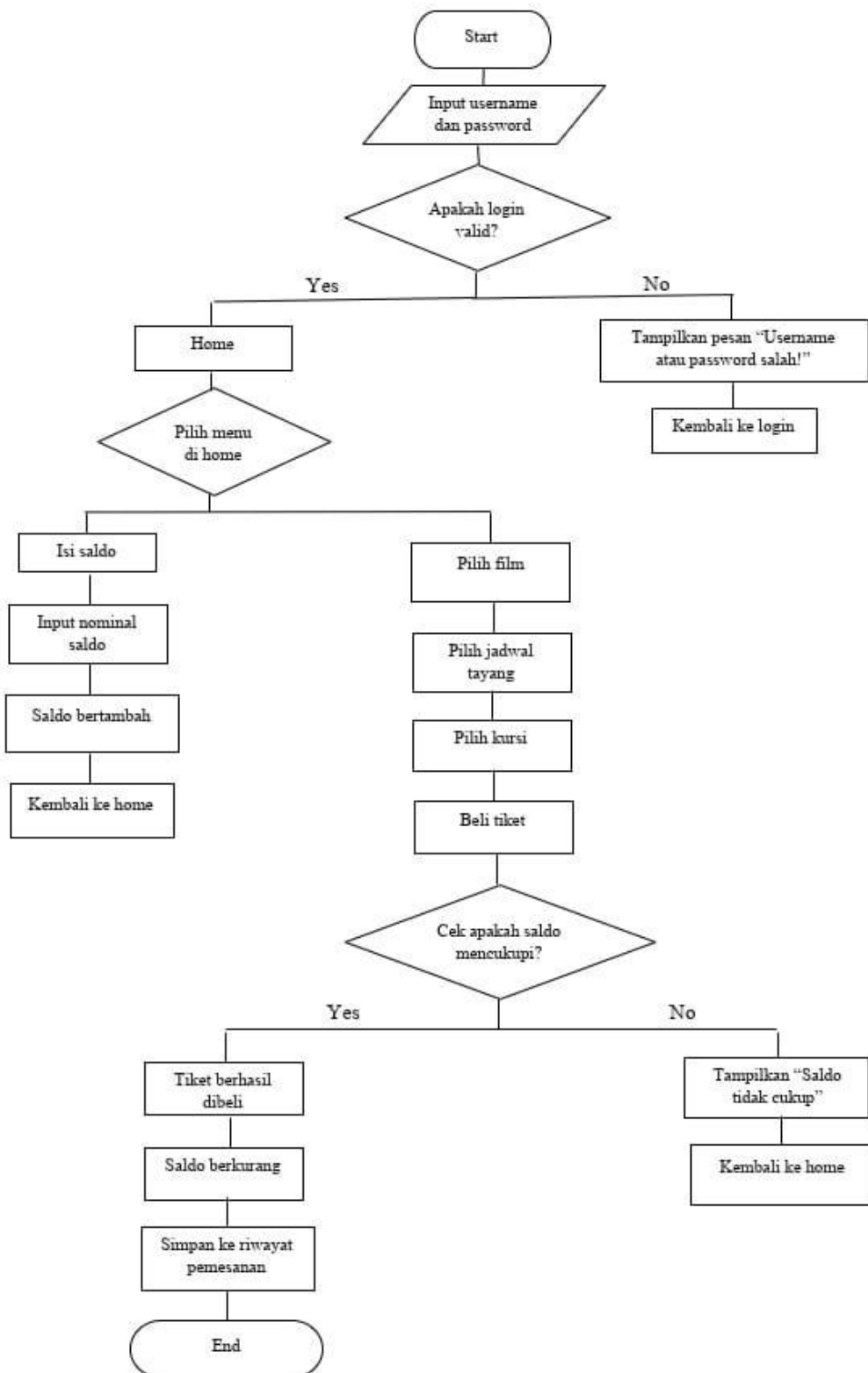
- > Pengguna membutuhkan kemudahan dalam memperoleh informasi film.
- > Akses informasi yang cepat membantu pengguna menentukan pilihan film.
- > Proses pemesanan tiket yang terpusat dinilai lebih sederhana.
- > Aplikasi membantu mengurangi kerumitan dalam proses pemesanan.
- > Kemudahan akses menjadi salah satu alasan utama aplikasi ini dikembangkan.

##### **2.1.3 Pengelolaan Data Pemesanan yang Lebih Tertata**

- >Pemesanan tiket menghasilkan data yang perlu dikelola dengan baik.

- >Pencatatan manual berpotensi menimbulkan kesalahan dan ketidakteraturan.
- >Aplikasi membantu menyimpan data pemesanan secara rapi.
- >Riwayat pemesanan memudahkan penelusuran transaksi yang telah dilakukan.
- >Pengelolaan data yang tertata menjadi alasan penting dalam pembuatan aplikasi.

## 2.2 FLOWCHART



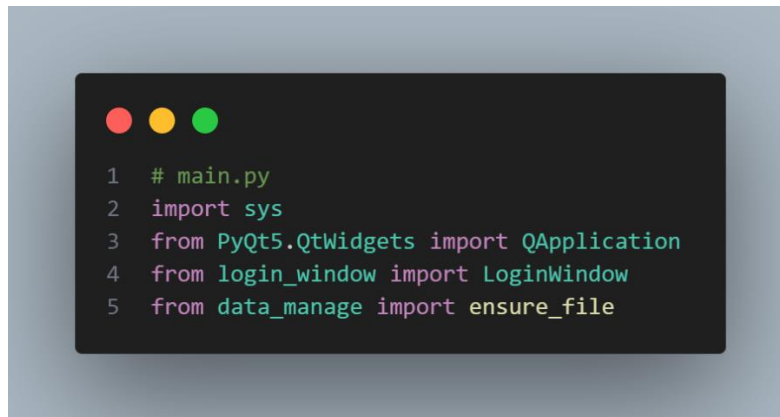
## BAB 3

# IMPLEMENTASI

### 3.1 PENJELASAN KODE PROGRAM

Project aplikasi pencetakam tiket bioskop ini melibatkan beberapa modul dan fitur, yang masing-masing dirancang untuk berkontribusi pada fungsionalitas keseluruhan. Berikut adalah penjelasan rinci tentang bagian-bagian penting dari kode tersebut:

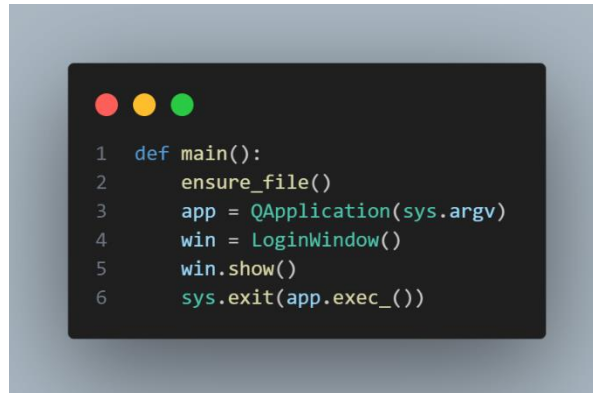
#### 3.1.1. MAIN.PY



```
1 # main.py
2 import sys
3 from PyQt5.QtWidgets import QApplication
4 from login_window import LoginWindow
5 from data_manage import ensure_file
```

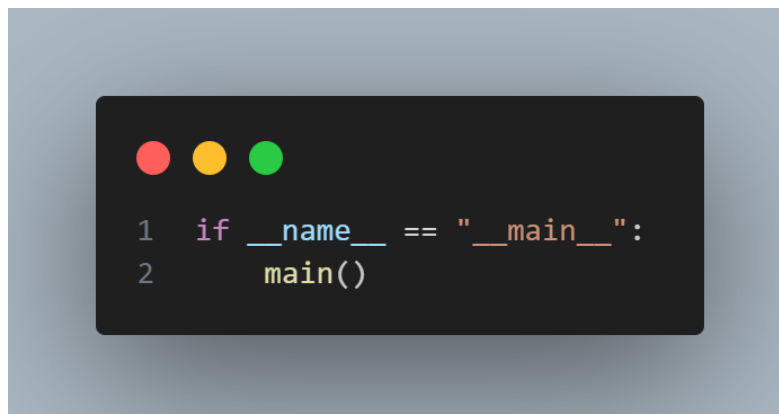
Gambar 3. 1.1 Main py

Kode pada gambar tersebut merupakan bagian awal dari file **main.py** yang berfungsi sebagai titik masuk (entry point) aplikasi pencetak tiket bioskop berbasis **PyQt5**. Baris `import sys` digunakan untuk mengakses fungsi sistem yang dibutuhkan oleh aplikasi Qt saat dijalankan. Selanjutnya, `from PyQt5.QtWidgets import QApplication` berfungsi untuk mengimpor kelas **QApplication**, yaitu komponen utama yang wajib ada untuk menjalankan aplikasi berbasis GUI di PyQt5. Baris `from login_window import LoginWindow` digunakan untuk memanggil kelas **LoginWindow** dari file lain, yang berfungsi menampilkan halaman login saat aplikasi dibuka. Terakhir, `from data_manage import ensure_file` berfungsi untuk mengimpor fungsi yang bertugas memastikan file data (seperti data pengguna atau tiket) sudah tersedia sebelum aplikasi dijalankan, sehingga aplikasi dapat berjalan dengan aman dan terstruktur.



Gambar 3.1.1 Main py

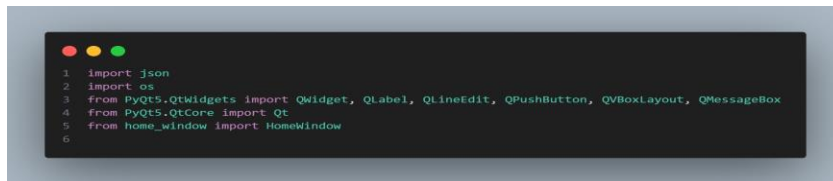
Potongan kode tersebut merupakan fungsi **main()** yang berperan sebagai pengendali utama jalannya aplikasi pencetak tiket bioskop. Baris `ensure_file()` digunakan untuk memastikan bahwa file data yang dibutuhkan aplikasi sudah tersedia sebelum program dijalankan. Selanjutnya, `QApplication(sys.argv)` berfungsi untuk menginisialisasi aplikasi berbasis PyQt5 dan menangkap argumen dari sistem operasi. Baris `win = LoginWindow()` digunakan untuk membuat objek jendela login sebagai tampilan awal aplikasi, kemudian `win.show()` berfungsi untuk menampilkan jendela tersebut ke layar. Terakhir, `sys.exit(app.exec_())` digunakan untuk menjalankan event loop aplikasi dan memastikan program dapat ditutup dengan benar ketika pengguna menutup aplikasi.



Gambar 3.1.1 Main py

Potongan kode tersebut berfungsi sebagai penentu eksekusi utama program dalam aplikasi pencetak tiket bioskop. Pernyataan `if __name__ == "__main__":` digunakan untuk memastikan bahwa fungsi `main()` hanya akan dijalankan ketika file ini dieksekusi secara langsung, bukan ketika diimpor sebagai modul oleh file Python lain. Dengan adanya struktur ini, alur program menjadi lebih terkontrol dan aman, karena proses inisialisasi aplikasi seperti pengecekan file data dan pemanggilan jendela login hanya berjalan saat aplikasi benar-benar dijalankan oleh pengguna.

### 3.1.2. LOGIN WINDOW



Gambar 3.1.2 Login Window

Kode tersebut digunakan untuk mengimpor modul pengelolaan data dan sistem file (json dan os), komponen antarmuka grafis berbasis **PyQt5**, serta kelas HomeWindow sebagai jendela utama aplikasi yang diakses setelah proses autentikasi atau navigasi berhasil.



Gambar 3.1.2 Login Window

digunakan untuk menyimpan nama file **JSON** yang berfungsi sebagai tempat penyimpanan data pengguna (user), seperti informasi akun login, sehingga memudahkan proses pembacaan dan penulisan data pengguna secara terstruktur dan permanen.

```

1 class RegisterWindow(QWidget):
2     def __init__(self, parent_login):
3         super().__init__()
4         self.parent_login = parent_login
5         self.setWindowTitle("Register User Baru")
6         self.setFixedSize(360, 260)
7         self.setStyleSheet("background: #1E90FF; color: white;") # biru
8         self.init_ui()
9
10    def init_ui(self):
11        layout = QVBoxLayout()
12        layout.setContentsMargins(24, 24, 24, 24)
13
14        lbl = QLabel("REGISTER USER BARU")
15        lbl.setAlignment(Qt.AlignCenter)
16        lbl.setStyleSheet("font-size:18pt; font-weight:bold;")
17        layout.addWidget(lbl)
18
19        self.user_in = QLineEdit()
20        self.user_in.setPlaceholderText("Username")
21        self.pass_in = QLineEdit()
22        self.pass_in.setPlaceholderText("Password")
23        self.pass_in.setEchoMode(QLineEdit.Password)
24
25        self.user_in.setStyleSheet("padding:8px; border-radius:6px;")
26        self.pass_in.setStyleSheet("padding:8px; border-radius:6px;")
27
28        layout.addWidget(self.user_in)
29        layout.addWidget(self.pass_in)
30
31        # Tombol CONFIRM REGISTER
32        btn_confirm = QPushButton("CONFIRM REGISTER")
33        btn_confirm.setStyleSheet("background:white; color:#1E90FF; font-weight:bold; height:40px;")
34        btn_confirm.clicked.connect(self.do_register)
35        layout.addWidget(btn_confirm)
36
37        # Tombol kembali login
38        btn_back = QPushButton("Back to login")
39        btn_back.setStyleSheet("background:white; color:#1E90FF; font-weight:bold; height:40px;")
40        btn_back.clicked.connect(self.back_to_login)
41        layout.addWidget(btn_back)
42
43        self.setLayout(layout)
44
45    def do_register(self):
46        user = self.user_in.text().strip()
47        pw = self.pass_in.text().strip()
48        if not user or not pw:
49            QMessageBox.warning(self, "Gagal", "Username dan password tidak boleh kosong!")
50            return
51        if user in self.parent_login.users:
52            QMessageBox.warning(self, "Gagal", "Username sudah ada!")
53            return
54        #menyimpan user baru
55        self.parent_login.users[user] = pw
56        self.parent_login.save_users()
57        QMessageBox.information(self, "Berhasil", f"User {user} berhasil dibuat! Silakan login.")
58        self.back_to_login() # kembali ke login window
59
60    def back_to_login(self):
61        self.parent_login.show()
62        self.close()

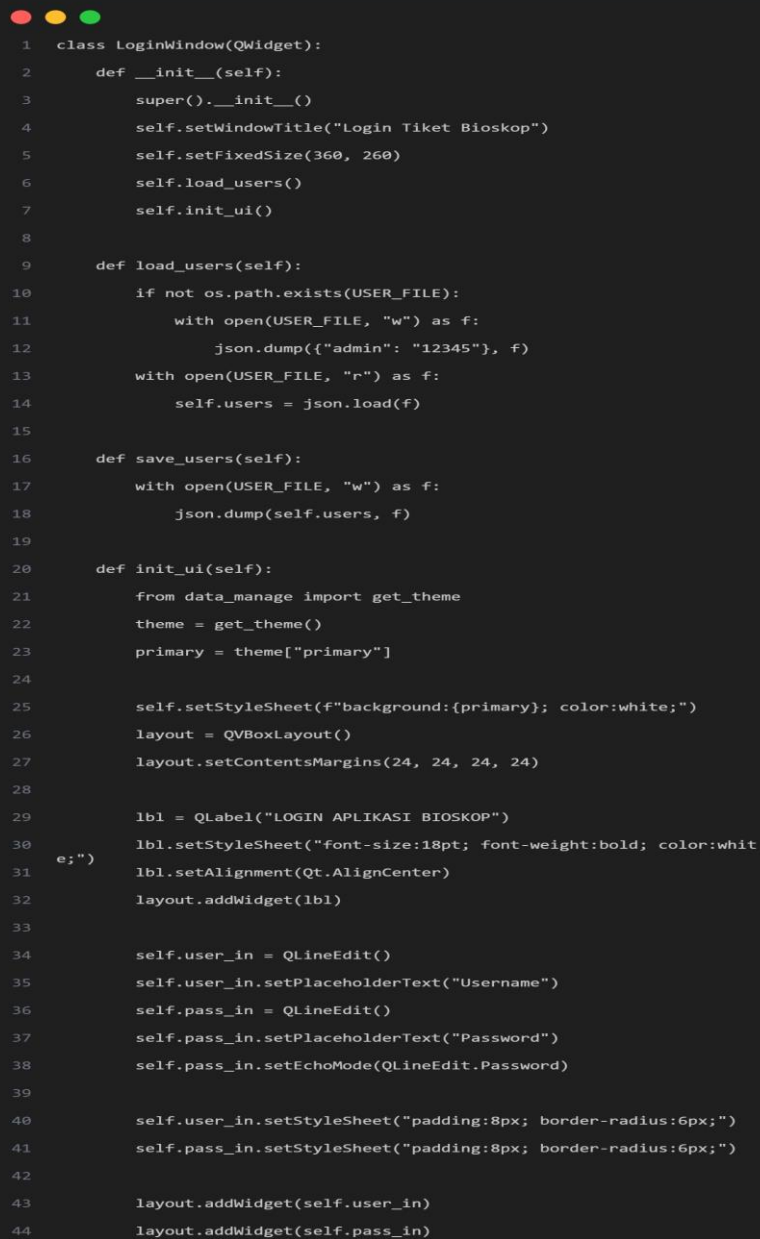
```

Gambar 3.1.2 Login Window

Kelas **RegisterWindow** merupakan kelas turunan dari **QWidget** yang berfungsi sebagai jendela pendaftaran pengguna baru pada aplikasi berbasis PyQt5. Kelas ini dirancang untuk menangani proses registrasi akun sekaligus mengatur tampilan antarmuka pengguna secara terstruktur dan interaktif. Pada metode `__init__`, jendela registrasi diinisialisasi dengan menerima parameter `parent_login` sebagai referensi ke jendela login. Referensi ini digunakan untuk mengakses dan menyimpan data pengguna serta memungkinkan navigasi kembali ke halaman login. Selain itu, metode ini juga mengatur judul jendela, ukuran tetap, warna latar belakang biru, serta memanggil metode `init_ui` untuk membangun komponen antarmuka.

Metode `init_ui` bertanggung jawab dalam menyusun tampilan antarmuka pendaftaran. Komponen disusun secara vertikal menggunakan **QVBoxLayout**, dimulai dari label judul, kolom input username dan password, hingga tombol aksi. Input password disembunyikan menggunakan Password Echo Mode untuk menjaga keamanan data, sementara tombol “Confirm Register” dan “Back to login” disediakan untuk memproses pendaftaran dan kembali ke halaman login.

Metode `do_register` berfungsi sebagai logika utama pendaftaran pengguna. Metode ini melakukan validasi terhadap input agar tidak kosong, mengecek keberadaan username untuk mencegah duplikasi akun, serta menyimpan data pengguna baru ke dalam sistem melalui jendela login. Setelah pendaftaran berhasil, aplikasi menampilkan pesan informasi dan mengarahkan pengguna kembali ke halaman login. Terakhir, metode `back_to_login` digunakan untuk menutup jendela registrasi dan menampilkan kembali jendela login tanpa menutup aplikasi. Dengan demikian, alur navigasi aplikasi tetap berjalan dengan baik dan pengalaman pengguna menjadi lebih terstruktur.



```

1  class LoginWindow(QWidget):
2      def __init__(self):
3          super().__init__()
4          self.setWindowTitle("Login Tiket Bioskop")
5          self.setFixedSize(360, 260)
6          self.load_users()
7          self.init_ui()
8
9      def load_users(self):
10         if not os.path.exists(USER_FILE):
11             with open(USER_FILE, "w") as f:
12                 json.dump({"admin": "12345"}, f)
13         with open(USER_FILE, "r") as f:
14             self.users = json.load(f)
15
16     def save_users(self):
17         with open(USER_FILE, "w") as f:
18             json.dump(self.users, f)
19
20     def init_ui(self):
21         from data_manage import get_theme
22         theme = get_theme()
23         primary = theme["primary"]
24
25         self.setStyleSheet(f"background:{primary}; color:white;")
26         layout = QVBoxLayout()
27         layout.setContentsMargins(24, 24, 24, 24)
28
29         lbl = QLabel("LOGIN APLIKASI BIOSKOP")
30         lbl.setStyleSheet("font-size:18pt; font-weight:bold; color:whit
31 e;")
32         lbl.setAlignment(Qt.AlignCenter)
33         layout.addWidget(lbl)
34
35         self.user_in = QLineEdit()
36         self.user_in.setPlaceholderText("Username")
37         self.pass_in = QLineEdit()
38         self.pass_in.setPlaceholderText("Password")
39         self.pass_in.setEchoMode(QLineEdit.Password)
40
41         self.user_in.setStyleSheet("padding:8px; border-radius:6px;")
42         self.pass_in.setStyleSheet("padding:8px; border-radius:6px;")
43
44         layout.addWidget(self.user_in)
45         layout.addWidget(self.pass_in)

```

Gambar 3.1.2 Login Window

Kelas **LoginWindow** merupakan kelas turunan dari QWidget yang berfungsi sebagai jendela login utama pada aplikasi pemesanan tiket bioskop berbasis PyQt5. Kelas ini menangani proses autentikasi pengguna sekaligus pengelolaan data akun yang tersimpan dalam file JSON.

Pada metode `__init__`, jendela login diinisialisasi dengan mengatur judul dan ukuran jendela agar tetap. Selanjutnya, metode `load_users` dipanggil untuk memuat data pengguna dari file penyimpanan, kemudian metode `init_ui` dijalankan untuk membangun tampilan antarmuka login.

Metode `load_users` berfungsi untuk memuat data pengguna dari file `users.json`. Apabila file tersebut belum tersedia, sistem akan membuat file baru dengan akun default (admin) sebagai inisialisasi awal. Setelah itu, data pengguna dibaca dan disimpan ke dalam variabel `self.users` dalam bentuk struktur data dictionary. Metode `save_users` digunakan untuk menyimpan perubahan data pengguna ke dalam file `users.json`. Metode ini dipanggil ketika terjadi penambahan atau perubahan data akun, sehingga data tetap tersimpan secara permanen.

Metode `init_ui` bertanggung jawab dalam membangun tampilan antarmuka login. Tema warna aplikasi diambil dari modul `data_manage` melalui fungsi `get_theme`, kemudian diterapkan sebagai warna latar belakang jendela. Komponen antarmuka disusun menggunakan `QVBoxLayout`, yang terdiri dari label judul, kolom input username, dan kolom input password. Kolom password disembunyikan menggunakan Password Echo Mode untuk menjaga keamanan data pengguna. Secara keseluruhan, kelas `LoginWindow` berfungsi sebagai modul autentikasi yang mengintegrasikan antarmuka pengguna, pengelolaan data akun, serta penerapan tema tampilan secara terstruktur dalam aplikasi bioskop.

```

1 btn_login = QPushButton("LOGIN")
2 btn_login.setFixedHeight(40)
3 btn_login.clicked.connect(self.do_login)
4 layout.addWidget(btn_login)
5
6
7 btn_register = QPushButton("REGISTER")
8 btn_register.setFixedHeight(40)
9 btn_register.clicked.connect(self.show_register_window)
10 layout.addWidget(btn_register)
11
12 self.setLayout(layout)
13
14 def do_login(self):
15     user = self.user_in.text().strip()
16     pw = self.pass_in.text().strip()
17     if user in self.users:
18         if self.users[user] == pw:
19             self.open_home(user, role="admin" if user=="admin" else "user")
20         else:
21             QMessageBox.warning(self, "Gagal", "Password salah!")
22     else:
23         QMessageBox.warning(self, "Gagal", "User tidak ditemukan. Silakan register terlebih dahulu.")
24
25 def show_register_window(self):
26     # menyembunyikan login window
27     self.hide()
28     # membuat register window
29     self.register_window = RegisterWindow(self)
30     self.register_window.move(self.x(), self.y())
31     self.register_window.show()
32
33 def open_home(self, username, role="user"):
34     self.hide()
35     self.home = HomeWindow(username=username, role=role)
36     self.home.show()
37

```

Gambar 3.1.2 Login Window

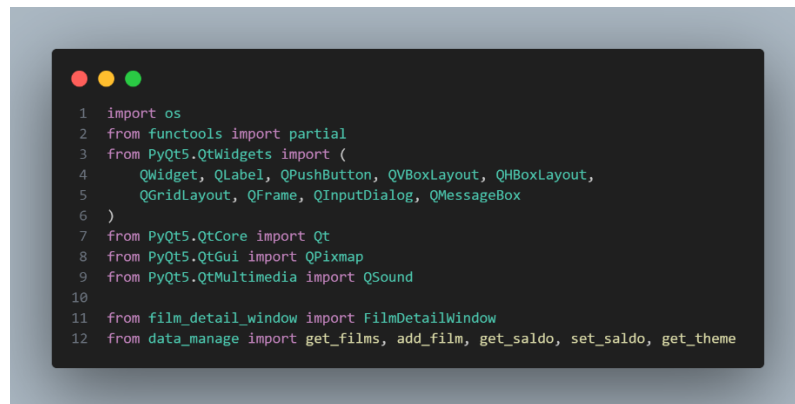
Potongan kode tersebut merupakan bagian dari kelas LoginWindow yang mengatur tombol login, tombol registrasi, serta alur proses autentikasi dan perpindahan antar jendela dalam aplikasi.

Tombol LOGIN dibuat menggunakan QPushButton dan dihubungkan dengan metode do\_login, yang akan dijalankan saat tombol ditekan. Tombol REGISTER juga dibuat dengan cara yang sama dan dihubungkan dengan metode show\_register\_window untuk membuka jendela pendaftaran pengguna baru. Kedua tombol ditambahkan ke dalam layout agar tampil rapi pada jendela login.

Metode do\_login berfungsi untuk memproses autentikasi pengguna dengan mengambil input username dan password, kemudian mencocokkannya dengan data pengguna yang tersimpan. Jika username dan password sesuai, aplikasi akan membuka halaman utama (home) dengan menentukan peran pengguna sebagai admin atau user, sedangkan jika terjadi kesalahan, sistem akan menampilkan pesan peringatan menggunakan QMessageBox.

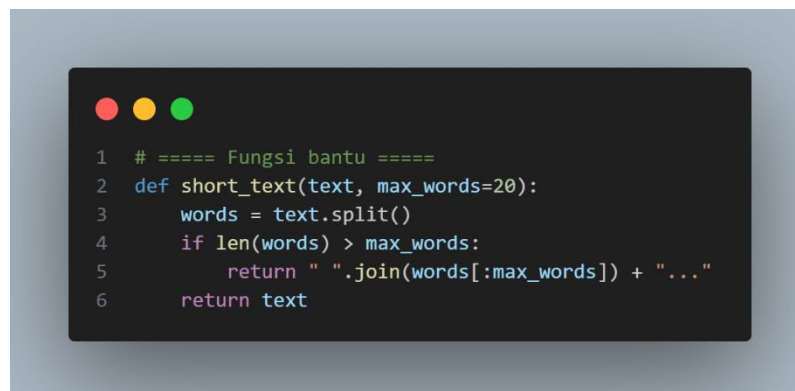
Metode show\_register\_window digunakan untuk menampilkan jendela registrasi dengan cara menyembunyikan jendela login terlebih dahulu, kemudian membuka jendela pendaftaran pada posisi yang sama. Selanjutnya, metode open\_home berfungsi untuk menutup jendela login dan menampilkan jendela utama aplikasi (HomeWindow) setelah proses login berhasil, sehingga alur navigasi aplikasi berjalan dengan baik.

### 3.1.3. HOME WINDOW



Gambar 3.1.3 Home Window

Kode tersebut merupakan bagian import dan inisialisasi dependensi pada aplikasi tiket bioskop berbasis PyQt5. Modul `os` digunakan untuk mengelola path file dan direktori, sedangkan `partial` dari `functools` membantu mengirim parameter ke fungsi callback (misalnya saat tombol ditekan). Berbagai komponen dari `PyQt5.QtWidgets` seperti `QWidget`, `QLabel`, `QPushButton`, dan layout (`QVBoxLayout`, `QHBoxLayout`, `QGridLayout`) digunakan untuk membangun tampilan antarmuka, sementara `QFrame`, `QInputDialog`, dan `QMessageBox` mendukung elemen pemisah serta interaksi input dan notifikasi. Modul `Qt` mengatur properti tampilan, `QPixmap` digunakan untuk memuat dan menampilkan gambar seperti poster film, dan `QSound` berfungsi memutar efek suara. Selain itu, `FilmDetailWindow` diimpor untuk menampilkan detail film, sedangkan fungsi-fungsi dari `data_manage` seperti `get_films`, `add_film`, `get_saldo`, `set_saldo`, dan `get_theme` digunakan untuk mengelola data film, saldo pengguna, serta tema tampilan aplikasi secara terpusat.



Gambar 3.1.3 Home Window

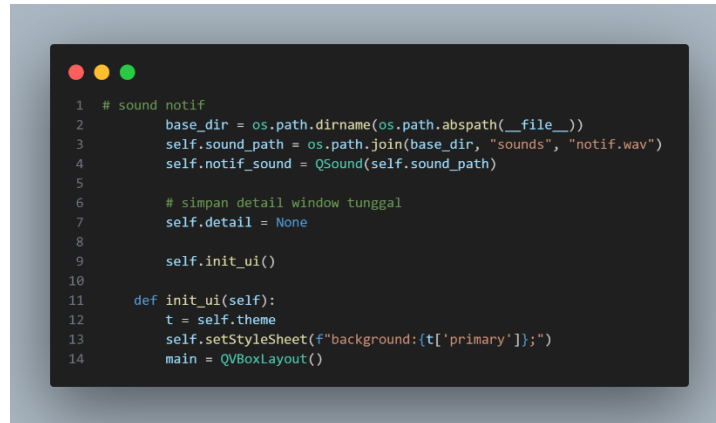
Kode tersebut merupakan `**fungsi bantu**` bernama ``short_text`` yang digunakan untuk `**memotong teks agar tidak terlalu panjang**` saat ditampilkan pada

antarmuka aplikasi. Fungsi ini menerima dua parameter, yaitu `text` sebagai teks yang akan diproses dan `max\_words` sebagai batas maksimum jumlah kata (default 20 kata). Teks dipecah menjadi daftar kata menggunakan `split()`, kemudian jumlah katanya diperiksa. Jika jumlah kata melebihi batas yang ditentukan, fungsi akan menggabungkan kembali kata-kata hingga batas `max\_words` dan menambahkan tanda elipsis (...) di akhir untuk menandakan teks terpotong. Jika jumlah kata tidak melebihi batas, teks akan dikembalikan apa adanya. Fungsi ini berguna untuk menampilkan deskripsi film secara ringkas agar tampilan aplikasi tetap rapi dan tidak memakan banyak ruang.

```
1 class HomeWindow(QWidget):
2     def __init__(self, username="guest", role="user"):
3         super().__init__()
4         self.username = username
5         self.role = role
6         self.setWindowTitle("Home - Tiket Bioskop")
7         self.setFixedSize(920, 720)
8         self.theme = get_theme()
```

Gambar 3.1.3 Home Window

Kode tersebut mendefinisikan **kelas `HomeWindow`** yang merupakan halaman utama (home) pada aplikasi tiket bioskop berbasis PyQt5. Kelas ini mewarisi `QWidget`, sehingga berfungsi sebagai sebuah jendela antarmuka. Pada metode `\_\_init\_\_`, parameter `username` dan `role` digunakan untuk menyimpan identitas serta hak akses pengguna, dengan nilai default `"guest"` dan `"user"`. Pemanggilan `super().\_\_init\_\_()` bertujuan untuk menginisialisasi kelas induk `QWidget`. Selanjutnya, judul jendela diatur menjadi **"Home - Tiket Bioskop"** dan ukuran jendela ditetapkan tetap sebesar **920 x 720 piksel** agar tampilan konsisten. Variabel `self.theme` diisi dengan nilai dari fungsi `get\_theme()`, yang berfungsi mengambil pengaturan tema (seperti warna atau gaya tampilan) sehingga antarmuka aplikasi dapat menyesuaikan tema yang sedang digunakan.



Gambar 3.1.3 Home Window

Potongan kode tersebut berfungsi untuk **\*\*mengatur suara notifikasi, inialisasi variabel, dan mempersiapkan tampilan antarmuka\*\*** pada jendela utama aplikasi. Pertama, `base_dir` digunakan untuk mendapatkan direktori tempat file Python berada, sehingga path file dapat diakses dengan aman di berbagai perangkat. Selanjutnya, `self.sound_path` menyusun lokasi file suara `*notif.wav*` yang berada di dalam folder `sounds`, dan `QSound` digunakan untuk memuat file tersebut sebagai suara notifikasi yang dapat diputar saat terjadi suatu aksi (misalnya klik tombol atau transaksi berhasil). Variabel `self.detail` diinisialisasi dengan nilai `None` untuk menyimpan satu instance `FilmDetailWindow`, sehingga aplikasi mencegah terbukanya banyak jendela detail film secara bersamaan. Terakhir, pemanggilan `self.init_ui()` berfungsi untuk menjalankan metode yang membangun dan menampilkan seluruh komponen antarmuka pengguna pada jendela Home.

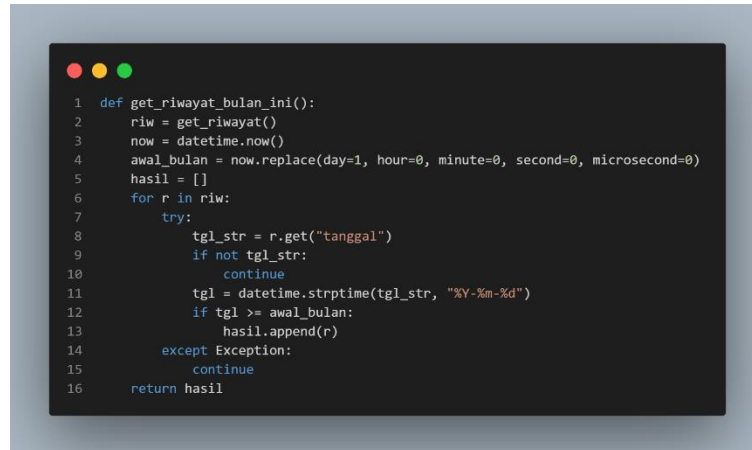
```

1  # ===== HEADER =====
2      header = QHBoxLayout()
3      lbl = QLabel(f"Selamat datang, {self.username}")
4      lbl.setStyleSheet("font-size:14pt; font-weight:bold; color:maroon;")
5      header.addWidget(lbl)
6      header.addStretch()
7
8      self.btn_saldo = QPushButton(f"Saldo: Rp {get_saldo()}")
9      self.btn_saldo.clicked.connect(self.isi_saldo)
10     self.btn_saldo.setStyleSheet(
11         f"background:{t['card']}; color:{t['primary']}; font-weight:bold; padding:8px;"
12     )
13     header.addWidget(self.btn_saldo)
14
15     btn_riwayat = QPushButton("Riwayat")
16     btn_riwayat.clicked.connect(self.open_riwayat)
17     btn_riwayat.setStyleSheet(f"background:{t['card']}; color:{t['primary']}; padding:8px;")
18     header.addWidget(btn_riwayat)
19
20     if self.role == "admin":
21         btn_add = QPushButton("Tambah Film")
22         btn_add.clicked.connect(self.add_film_dialog)
23         btn_add.setStyleSheet(
24             f"background:{t['accent']}; color:black; padding:8px; font-weight:bold;"
25         )
26         header.addWidget(btn_add)
27
28     main.addLayout(header)
29     main.addSpacing(12)
30

```

Gambar 3.1.3 Home Window

Kode tersebut membangun **bagian header (bagian atas)** pada halaman utama aplikasi tiket bioskop. Layout `QHBoxLayout` digunakan agar komponen tersusun secara horizontal. Label `QLabel` menampilkan pesan sambutan **"Selamat datang"** yang disesuaikan dengan `username` pengguna, serta diberi gaya huruf tebal dan warna maroon agar terlihat menonjol. Fungsi `addStretch()` digunakan untuk memberi ruang kosong fleksibel sehingga elemen berikutnya terdorong ke sisi kanan. Selanjutnya, tombol saldo (`btn_saldo`) menampilkan jumlah saldo pengguna yang diambil dari fungsi `get_saldo()` dan ketika diklik akan memanggil fungsi `isi_saldo` untuk melakukan pengisian saldo, dengan tampilan warna mengikuti tema aplikasi. Tombol **Riwayat** disediakan untuk membuka halaman riwayat transaksi atau tiket melalui fungsi `open_riwayat`. Jika peran pengguna adalah **admin**, maka tombol **Tambah Film** akan ditampilkan, yang berfungsi membuka dialog penambahan film melalui `add_film_dialog`. Terakhir, seluruh header dimasukkan ke layout utama (`main`) dan diberi jarak vertikal agar tampilan lebih rapi dan tidak terlalu rapat dengan konten di bawahnya.



Gambar 3.1.3 Home Window

Kode tersebut berfungsi untuk \*membuat dan menata tampilan daftar film dalam bentuk grid (kartu film)\* pada halaman utama aplikasi. QFrame dan QGridLayout digunakan sebagai wadah untuk menyusun kartu film secara teratur dengan jarak antar item yang diatur menggunakan setSpacing(14). Data film diambil melalui fungsi get\_films(), kemudian ditentukan jumlah kolom grid sebanyak dua kolom (cols = 2). Variabel base\_dir digunakan untuk menyimpan lokasi direktori file agar memudahkan pemanggilan file pendukung seperti poster. Melalui perulangan enumerate, setiap film diberi indeks untuk menentukan posisi baris (r) dan kolom (c) pada grid. Setiap film ditampilkan dalam sebuah QFrame berbentuk kartu dengan ukuran tetap 420 × 300 piksel dan gaya visual sesuai tema aplikasi. Di dalam kartu tersebut, QVBoxLayout digunakan untuk menyusun konten secara vertikal dengan margin dan jarak yang rapi, sehingga informasi film dapat ditampilkan dengan tampilan yang konsisten dan mudah dibaca.



Gambar 3.1.3 Home Window

Kode tersebut berfungsi untuk \*menampilkan poster film\* pada setiap kartu film di halaman utama aplikasi. Sebuah QLabel digunakan sebagai wadah poster, dengan ukuran tetap  $400 \times 140$  piksel dan posisi konten di tengah. Path poster diambil dari data film menggunakan `film.get("poster")`, kemudian digabungkan dengan direktori dasar aplikasi (`base_dir`) agar menghasilkan path lengkap file gambar. Jika path poster tersedia dan file gambarnya benar-benar ada, gambar akan dimuat menggunakan `QPixmap` lalu diskalakan agar menyesuaikan ukuran label tanpa merusak proporsi gambar, serta menggunakan `SmoothTransformation` agar hasilnya lebih halus. Jika poster tidak ditemukan atau tidak tersedia, label akan menampilkan teks "Poster tidak tersedia" dengan garis tepi sebagai penanda. Terakhir, komponen poster tersebut ditambahkan ke layout vertikal kartu film sehingga tampil rapi di dalam card.



Gambar 3.1.3 Home Window

Kode tersebut berfungsi untuk \*menampilkan judul dan deskripsi singkat film\* di dalam kartu film. Label QLabel pertama digunakan untuk menampilkan \*judul film\*, yang diambil dari data `film["judul"]`, lalu diberi gaya huruf tebal dan ukuran font lebih besar agar mudah dikenali, serta diratakan ke tengah. Selanjutnya, bagian \*\*deskripsi film\* diambil dari data deskripsi dan diproses menggunakan fungsi `short_text` untuk membatasi panjang teks maksimal 20 kata agar tampilan tetap rapi. Label deskripsi diatur agar teks dapat berpindah baris (`setWordWrap(True)`) dan disejajarkan ke tengah. Pemanggilan `v.addStretch()` di bagian akhir berfungsi menambahkan ruang kosong fleksibel di bawah konten, sehingga tata letak kartu film terlihat seimbang dan tidak terlalu padat.



```

1 # BUTTON DETAIL
2 btn = QPushButton("DETAIL & JADWAL")
3 btn.clicked.connect(lambda checked, f=film: self.open_detail(f))
4 btn.setStyleSheet(f"background:{t['primary']}; color:white; padding:8px; border-radius:6px;")
5 v.addWidget(btn)
6
7 card.setLayout(v)
8 grid.addWidget(card, r, c)
9
10 grid_frame.setLayout(grid)
11 main.addWidget(grid_frame)
12 main.addStretch()
13 self.setLayout(main)

```

Gambar 3.1.3 Home Window

Kode tersebut berfungsi untuk \*menambahkan tombol detail, menyusun kartu film ke dalam grid, dan menutup pengaturan layout halaman utama. Tombol DETAIL & JADWAL dibuat untuk memungkinkan pengguna melihat informasi lengkap film beserta jadwal penayangannya, dan dihubungkan dengan fungsi `open_detail` menggunakan \*lambda agar data film yang dipilih dapat dikirim sebagai parameter. Tombol ini juga diberi gaya warna sesuai tema agar tampil menarik dan konsisten. Setelah itu, layout vertikal (v) yang berisi poster, judul, deskripsi, dan tombol diterapkan ke dalam kartu film (card), lalu kartu tersebut ditempatkan pada posisi baris (r) dan kolom (c) yang sesuai di `QGridLayout`. Setelah seluruh film ditambahkan, grid dimasukkan ke dalam `grid_frame`, kemudian `grid_frame` ditambahkan ke layout utama (main). Pemanggilan `addStretch()` memberi ruang kosong di bagian bawah agar tampilan lebih seimbang, dan `self.setLayout(main)` menetapkan seluruh susunan layout tersebut sebagai tampilan resmi jendela Home.

```

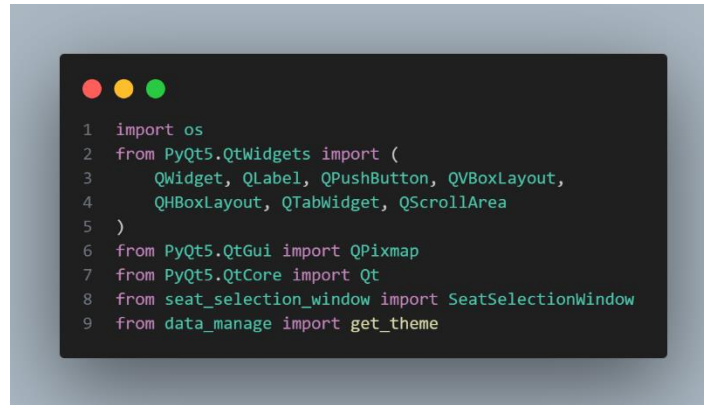
1 # ===== FUNCTIONS =====
2 def refresh(self):
3     self.layout().deleteLater()
4     self.init_ui()
5
6 def add_film_dialog(self):
7     judul, ok = QInputDialog.getText(self, "Tambah Film", "Judul film:")
8     if not ok or not judul.strip():
9         return
10    des, ok2 = QInputDialog.getMultilineText(self, "Deskripsi", "Deskripsi singkat:")
11    if not ok2:
12        des = ""
13    add_film(judul.strip(), des.strip())
14    QMessageBox.information(self, "OK", "Film berhasil ditambahkan.")
15    self.refresh()
16
17 def open_detail(self, film):
18     if self.detail and self.detail.isVisible():
19         self.detail.update_film(film)
20         self.detail.raise_()
21         self.detail.activateWindow()
22     else:
23         self.detail = FilmDetailWindow(film, parent=self)
24         self.detail.show()
25
26 def isi_saldo(self):
27     dialog = QInputDialog(self)
28     dialog.setWindowTitle("Isi Saldo")
29     dialog.setLabelText("Masukkan nominal (Rp):")
30     dialog.setInputMode(QInputDialog.IntInput)
31     dialog.setIntrange(1, 100000000)
32     dialog.setIntStep(1000)
33
34     if not dialog.exec_():
35         return
36     amt = dialog.intValue()
37     set_saldo(get_saldo() + amt)
38     if os.path.exists(self.sound_path):
39         self.notif_sound.play()
40     QMessageBox.information(self, "Berhasil", f"Saldo bertambah Rp {amt}")
41     self.btn_saldo.setText(f"Saldo: Rp {get_saldo()}")
42
43 def open_riwayat(self):
44     from riwayat_window import RiwayatWindow
45     self.riw = RiwayatWindow()
46     self.riw.show()
47

```

Gambar 3.1.3 Home Window

Kode tersebut berisi \*kumpulan fungsi (method)\* yang mengatur interaksi dan logika utama pada halaman Home aplikasi tiket bioskop. Fungsi refresh digunakan untuk memuat ulang tampilan dengan cara menghapus layout lama lalu membangun ulang antarmuka melalui `init_ui()`, sehingga perubahan data dapat langsung terlihat. Fungsi `add_film_dialog` menampilkan dialog input untuk menambahkan film baru, dimulai dari judul dan deskripsi, kemudian menyimpannya menggunakan `add_film`, menampilkan pesan konfirmasi, dan memanggil refresh agar film baru muncul di daftar. Fungsi `open_detail` bertugas membuka jendela detail film; jika jendela detail sudah terbuka, maka data film diperbarui dan jendela difokuskan, sehingga tidak terjadi pembukaan banyak jendela detail sekaligus. Fungsi `isi_saldo` menyediakan dialog pengisian saldo dengan batas nominal tertentu, memperbarui saldo menggunakan `set_saldo`, memutar suara notifikasi jika file tersedia, menampilkan pesan keberhasilan, serta memperbarui teks tombol saldo. Terakhir, fungsi `open_riwayat` digunakan untuk membuka jendela riwayat tiket atau transaksi dengan memanggil `RiwayatWindow`, sehingga pengguna dapat melihat histori aktivitasnya dalam aplikasi.

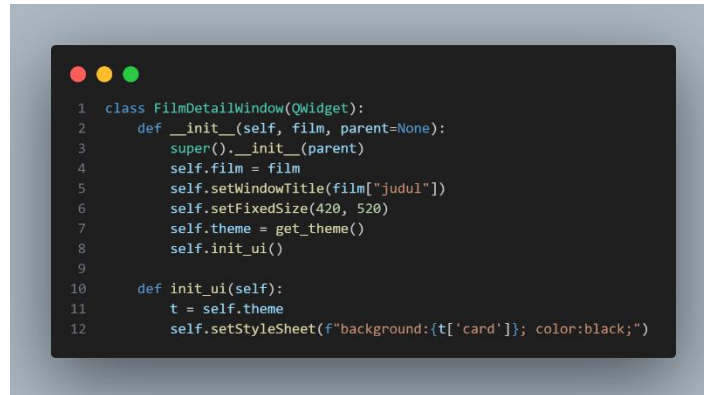
### 3.1.4. FILM DETAIL

A screenshot of a code editor window with a dark background and light-colored text. The code is a Python script showing imports for a PyQt5 application. The imports include the 'os' module, various Qt widgets and layouts from 'PyQt5.QtWidgets', 'QPixmap' from 'PyQt5.QtGui', 'Qt' from 'PyQt5.QtCore', 'SeatSelectionWindow' from 'seat\_selection\_window', and 'get\_theme' from 'data\_manage'. The code is numbered from 1 to 9.

```
1 import os
2 from PyQt5.QtWidgets import (
3     QWidget, QLabel, QPushButton, QVBoxLayout,
4     QHBoxLayout, QTabWidget, QScrollArea
5 )
6 from PyQt5.QtGui import QPixmap
7 from PyQt5.QtCore import Qt
8 from seat_selection_window import SeatSelectionWindow
9 from data_manage import get_theme
```

Gambar 3.1.4 Film Detail

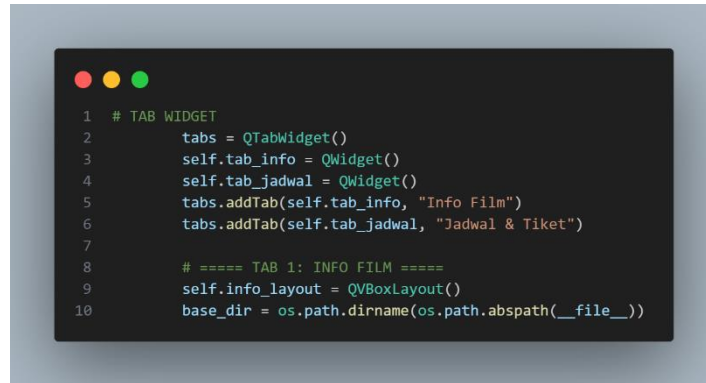
Kode tersebut merupakan \*bagian import modul dan komponen utama\* yang digunakan untuk membangun jendela detail film pada aplikasi tiket bioskop berbasis PyQt5. Modul `os` digunakan untuk mengelola path file dan memastikan akses gambar atau resource lain berjalan dengan benar. Berbagai widget dari `PyQt5.QtWidgets` seperti `QWidget`, `QLabel`, dan `QPushButton` digunakan untuk membuat elemen tampilan, sedangkan `QVBoxLayout` dan `HBoxLayout` berfungsi mengatur tata letak komponen secara vertikal dan horizontal. `QTabWidget` dimanfaatkan untuk menampilkan informasi dalam bentuk tab, sementara `QScrollArea` memungkinkan konten yang panjang tetap dapat dilihat dengan cara digulir. Modul `QPixmap` digunakan untuk memuat dan menampilkan gambar seperti poster film, dan `Qt` mengatur perataan serta properti visual lainnya. Selain itu, `SeatSelectionWindow` diimpor untuk menangani proses pemilihan kursi saat pemesanan tiket, dan `get_theme` digunakan untuk mengambil pengaturan tema agar tampilan jendela detail film konsisten dengan tema aplikasi secara keseluruhan.

A screenshot of a code editor window with a dark background and light-colored text. The code is written in Python and defines a class named FilmDetailWindow that inherits from QWidget. The class has two methods: \_\_init\_\_ and init\_ui. The \_\_init\_\_ method takes 'self', 'film', and 'parent' as arguments. It calls super().\_\_init\_\_(parent), sets self.film = film, sets the window title to film['judul'], sets a fixed size of (420, 520), gets the theme using get\_theme(), and calls self.init\_ui(). The init\_ui method takes 'self' as an argument, assigns self.theme to t, and sets the stylesheet to f"background:{t['card']}; color:black;".

```
1 class FilmDetailWindow(QWidget):
2     def __init__(self, film, parent=None):
3         super().__init__(parent)
4         self.film = film
5         self.setWindowTitle(film["judul"])
6         self.setFixedSize(420, 520)
7         self.theme = get_theme()
8         self.init_ui()
9
10    def init_ui(self):
11        t = self.theme
12        self.setStyleSheet(f"background:{t['card']}; color:black;")
```

Gambar 3.1.4 Film Detail

Kode tersebut mendefinisikan \*kelas FilmDetailWindow, yaitu jendela yang digunakan untuk menampilkan \*\*detail informasi sebuah film\* pada aplikasi tiket bioskop. Kelas ini mewarisi QWidget, sehingga berfungsi sebagai jendela terpisah yang dapat dipanggil dari halaman utama. Pada metode \_\_init\_\_, data film yang dipilih diterima melalui parameter film dan disimpan ke dalam atribut self.film, kemudian judul jendela diatur sesuai dengan judul film tersebut. Ukuran jendela ditetapkan tetap sebesar  $420 \times 520$  piksel\* agar tampilannya konsisten. Selanjutnya, tema aplikasi diambil menggunakan fungsi get\_theme() dan disimpan ke dalam self.theme. Metode init\_ui() kemudian dipanggil untuk membangun komponen antarmuka. Di dalam init\_ui, variabel t digunakan sebagai singkatan dari tema, dan setStyleSheet diterapkan untuk mengatur warna latar belakang serta warna teks jendela agar sesuai dengan tema aplikasi dan terlihat serasi.



Gambar 3.1.4 Film Detail

Kode tersebut berfungsi untuk \*membuat tampilan berbasis tab\* pada jendela detail film agar informasi tersusun rapi dan mudah diakses. QTabWidget digunakan sebagai wadah utama tab, kemudian dua widget tab dibuat, yaitu tab\_info untuk menampilkan \*informasi film\* dan tab\_jadwal untuk menampilkan \*jadwal serta pemesanan tiket. Kedua tab tersebut ditambahkan ke dalam QTabWidget dengan judul yang sesuai agar pengguna mudah memahami fungsinya. Selanjutnya, pada bagian \*\*Tab Info Film\*, dibuat QVBoxLayout sebagai layout utama untuk menyusun konten secara vertikal. Variabel base\_dir digunakan untuk menyimpan direktori dasar file program, sehingga memudahkan pemanggilan resource seperti poster film atau file pendukung lainnya secara aman dan konsisten.



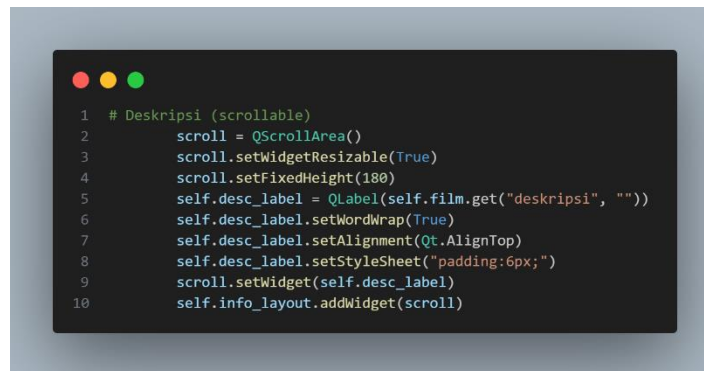
Gambar 3.1.4 Film Detail

Kode tersebut berfungsi untuk \*menampilkan poster film pada tab Info Film\* di jendela detail. Sebuah QLabel digunakan sebagai wadah poster dan posisinya diatur agar berada di tengah. Path poster dibentuk dengan menggabungkan direktori dasar aplikasi (base\_dir) dan lokasi poster yang diambil dari data film. Jika file poster ditemukan, gambar dimuat menggunakan QPixmap lalu diskalakan ke ukuran  $220 \times 320$  piksel dengan menjaga rasio aspek dan kualitas tampilan agar tetap halus. Jika file poster tidak tersedia, label akan menampilkan teks "Poster tidak tersedia" disertai garis tepi sebagai penanda. Terakhir, komponen poster tersebut ditambahkan ke dalam layout info\_layout sehingga tampil rapi di dalam tab informasi film.



Gambar 3.1.4 Film Detail

Kode tersebut digunakan untuk \*menampilkan judul film\* pada tab Info Film di jendela detail. QLabel dibuat dengan teks yang diambil dari data judul film (self.film["judul"]), kemudian posisinya diratakan ke tengah agar tampil lebih rapi dan fokus. Gaya tampilan judul diatur menggunakan setStyleSheet dengan ukuran font lebih besar dan huruf tebal, sehingga judul film terlihat menonjol dibandingkan informasi lainnya. Setelah itu, label judul ditambahkan ke dalam info\_layout agar tersusun secara vertikal bersama poster dan elemen informasi film lainnya.



Gambar 3.1.4 Film Detail

Kode tersebut berfungsi untuk \*menampilkan deskripsi film yang dapat digulir (scrollable)\* pada tab Info Film. QScrollArea digunakan agar teks deskripsi yang panjang tetap dapat dibaca tanpa memperbesar ukuran jendela, dengan tinggi area yang dibatasi sebesar 180 piksel dan pengaturan agar isinya menyesuaikan ukuran area. QLabel digunakan untuk menampilkan teks deskripsi film yang diambil dari data deskripsi, kemudian diatur agar teks dapat berpindah baris (setWordWrap(True)), diratakan ke bagian atas, dan diberi padding supaya lebih nyaman dibaca. Label deskripsi ini kemudian dimasukkan ke dalam QScrollArea, lalu area scroll tersebut ditambahkan ke info\_layout sehingga informasi film tetap rapi dan mudah dinavigasi oleh pengguna.



```

1 # Tombol EXIT/TUTUP
2 btn_exit = QPushButton("TUTUP")
3 btn_exit.setStyleSheet(f"background:{t['primary']}; color:white; padding:6px; border-radius:4px;")
4 btn_exit.clicked.connect(self.close)
5 self.info_layout.addWidget(btn_exit)
6
7 self.info_layout.addStretch()
8 self.tab_info.setLayout(self.info_layout)

```

Gambar 3.1.4 Film Detail

Kode tersebut berfungsi untuk \*menambahkan tombol keluar (TUTUP)\* pada tab Info Film di jendela detail. Tombol QPushButton dibuat dengan teks \*"TUTUP"\* dan diberi gaya warna sesuai tema aplikasi agar tampil konsisten dan mudah dikenali. Ketika tombol ditekan, sinyal clicked dihubungkan ke metode self.close, sehingga jendela detail film akan langsung ditutup. Setelah itu, tombol ditambahkan ke dalam info\_layout, diikuti dengan addStretch() untuk memberikan ruang kosong fleksibel di bagian bawah agar tata letak terlihat lebih seimbang. Terakhir, info\_layout ditetapkan sebagai layout resmi untuk tab\_info, sehingga seluruh komponen pada tab informasi film tersusun dengan rapi.



```

1 # ===== TAB 2: JADWAL =====
2 jadwal_layout = QVBoxLayout()
3 lbl = QLabel("Jadwal Tayang (Default demo):")
4 jadwal_layout.addWidget(lbl)
5
6 studios = [("Studio 1", "20:00"), ("Studio 2", "18:30"), ("Studio 3", "21:30")]
7 for s, j in studios:
8     row = QHBoxLayout()
9     lbls = QLabel(f"{s} | Jam {j}")
10    btn = QPushButton("Pilih Kursi & Beli")
11    btn.setStyleSheet(f"background:{t['primary']}; color:white; padding:6px;")
12    btn.clicked.connect(lambda checked, s=s, j=j: self.open_seat(s, j))
13    row.addWidget(lbls)
14    row.addStretch()
15    row.addWidget(btn)
16    jadwal_layout.addLayout(row)
17
18 jadwal_layout.addStretch()
19 self.tab_jadwal.setLayout(jadwal_layout)
20

```

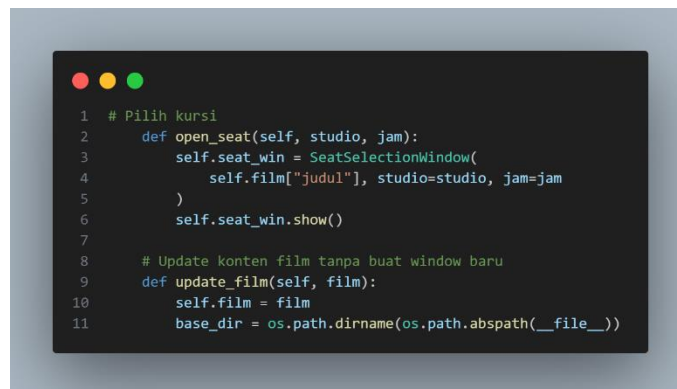
Gambar 3.1.4 Film Detail

Kode tersebut membangun \*Tab 2: Jadwal & Tiket\* pada jendela detail film. QVBoxLayout digunakan sebagai layout utama untuk menyusun elemen secara vertikal, dimulai dengan label yang menampilkan judul "Jadwal Tayang (Default demo)" sebagai informasi kepada pengguna. Daftar jadwal disimpan dalam variabel studios yang berisi nama studio dan jam tayang, lalu ditampilkan menggunakan perulangan. Untuk setiap jadwal, dibuat baris QHBoxLayout yang berisi label informasi studio dan jam tayang, serta tombol \*"Pilih Kursi & Beli"\*. Tombol ini diberi gaya sesuai tema aplikasi dan dihubungkan ke fungsi open\_seat menggunakan \*lambda\* agar data studio dan jam yang dipilih dapat dikirim sebagai parameter. addStretch() digunakan untuk mendorong tombol ke sisi kanan dan memberi ruang kosong di bagian bawah agar tampilan lebih rapi. Terakhir, layout jadwal tersebut diterapkan ke tab\_jadwal, sehingga pengguna dapat memilih jadwal dan melanjutkan ke proses pemilihan kursi dengan mudah.



Gambar 3.1.4 Film Detail

Kode tersebut berfungsi untuk \*menyusun layout utama jendela detail film. QVBoxLayout digunakan sebagai layout utama agar komponen disusun secara vertikal, dengan QTabWidget (tabs) sebagai elemen inti yang berisi tab \*Info Film dan Jadwal & Tiket. Dengan menambahkan tabs ke dalam main\_layout, seluruh konten tab dapat ditampilkan dalam satu wadah utama. Terakhir, self.setLayout(main\_layout) menetapkan layout tersebut sebagai tata letak resmi jendela, sehingga seluruh komponen antarmuka dapat ditampilkan dengan rapi dan terstruktur.



Gambar 3.1.4 Film Detail

Kode tersebut berisi \*fungsi pendukung\* pada jendela detail film yang berkaitan dengan pemilihan kursi dan pembaruan data film. Fungsi open\_seat digunakan untuk membuka jendela \*pemilihan kursi, dengan membuat instance SeatSelectionWindow dan mengirimkan parameter judul film, studio, serta jam tayang yang dipilih pengguna, kemudian menampilkannya ke layar. Sementara itu, fungsi update\_film berfungsi untuk \*\*memperbarui konten film yang ditampilkan\* tanpa harus membuat jendela detail baru, sehingga lebih efisien dan nyaman bagi pengguna. Pada fungsi ini, data film baru disimpan ke self.film, dan base\_dir kembali ditentukan untuk memastikan akses terhadap resource seperti poster tetap berjalan dengan benar saat konten film diperbarui.



```

1  # Poster
2  poster_path = os.path.join(base_dir, self.film.get("poster", ""))
3  if os.path.exists(posters_path):
4      pixmap = QPixmap(posters_path)
5      self.poster_label.setPixmap(
6          pixmap.scaled(220, 320, Qt.KeepAspectRatio, Qt.SmoothTransformation)
7      )
8  else:
9      self.poster_label.setText("Poster tidak tersedia")
10
11  # Judul
12  self.title_label.setText(self.film["judul"])
13
14  # Deskripsi
15  self.desc_label.setText(self.film.get("deskripsi", ""))

```

Gambar 3.1.4 Film Detail

Kode tersebut digunakan untuk **\*\*memperbarui tampilan poster, judul, dan deskripsi film\*\*** ketika data film diganti melalui fungsi `update_film`, tanpa perlu membuat jendela baru. Pertama, path poster dibentuk dari direktori dasar dan data poster film, lalu dicek keberadaannya; jika file poster tersedia, gambar dimuat dan ditampilkan kembali pada `poster_label` dengan ukuran serta rasio yang sesuai, sedangkan jika tidak tersedia akan ditampilkan teks **“Poster tidak tersedia”**. Selanjutnya, teks pada `title_label` diperbarui sesuai judul film yang baru, dan bagian deskripsi juga diperbarui dengan mengubah isi `desc_label` menggunakan data deskripsi film terbaru. Dengan mekanisme ini, jendela detail dapat menampilkan informasi film yang berbeda secara dinamis dan lebih efisien.

### 3.1.5. DATA MANAGE

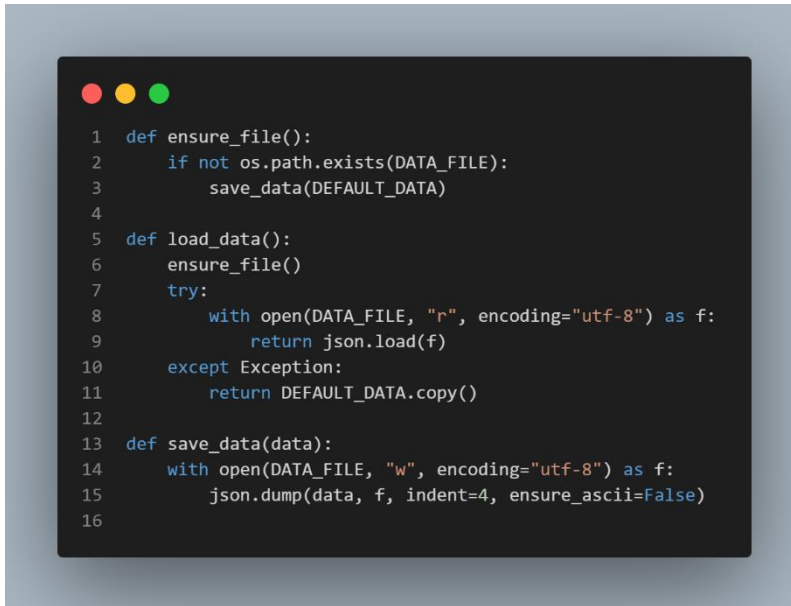
A screenshot of a code editor with a dark background and light-colored text. The code is written in Python and defines several constants and data structures. It includes imports for 'json', 'os', 'datetime', and 'openpyxl'. It defines 'DATA\_FILE' as 'cinema\_data.json', 'DEFAULT\_THEME' as a dictionary with color codes, and 'DEFAULT\_DATA' as a dictionary containing 'saldo', 'riwayat\_tiket', 'kursi\_terisi', 'films', 'next\_film\_id', and 'theme'. The 'films' list contains four movie entries with their IDs, titles, and descriptions.

```
1 import json
2 import os
3 from datetime import datetime
4 from openpyxl import Workbook
5
6 DATA_FILE = "cinema_data.json"
7 DEFAULT_THEME = {
8     "primary": "#800000",
9     "accent": "#FFD700",
10    "bg": "#000000",
11    "card": "#FFFFFF",
12    "sold": "#4D0000",
13    "available": "#FFFFFF",
14    "selected": "#FFD700"
15 }
16
17 DEFAULT_DATA = {
18     "saldo": 0,
19     "riwayat_tiket": [],
20     "kursi_terisi": {},
21     "films": [
22         {"id": 1, "judul": "Film A: Interstellar", "deskripsi": "Sci-fi epic"},
23         {"id": 2, "judul": "Film B: The Dark Knight", "deskripsi": "Action/Crime"},
24         {"id": 3, "judul": "Film C: Avatar 3D", "deskripsi": "Fantasy/Sci-fi"},
25         {"id": 4, "judul": "Film D: Titanic", "deskripsi": "Drama/Romance"}
26     ],
27     "next_film_id": 5,
28     "theme": DEFAULT_THEME
29 }
30
```

Gambar 3.1.5 Data Manage

Pada bagian awal program, dilakukan proses import beberapa library yang dibutuhkan untuk mendukung jalannya aplikasi. Library json digunakan untuk membaca dan menyimpan data aplikasi dalam format JSON, sedangkan os berfungsi untuk mengelola file dan direktori pada sistem. Library datetime dimanfaatkan untuk mencatat waktu transaksi pemesanan tiket, sehingga setiap pemesanan memiliki informasi tanggal yang jelas. Selain itu, library openpyxl digunakan untuk membuat dan mengelola file Excel sebagai media ekspor laporan riwayat pemesanan.

Selanjutnya, didefinisikan beberapa konstanta penting seperti data\_file yang menunjukkan nama file penyimpanan data utama aplikasi, serta default\_theme yang berisi pengaturan warna antarmuka. Penggunaan tema ini bertujuan agar tampilan aplikasi konsisten di setiap halaman dan mudah disesuaikan apabila di kemudian hari diperlukan perubahan desain.




```

1  def ensure_file():
2      if not os.path.exists(DATA_FILE):
3          save_data(DEFAULT_DATA)
4
5  def load_data():
6      ensure_file()
7      try:
8          with open(DATA_FILE, "r", encoding="utf-8") as f:
9              return json.load(f)
10     except Exception:
11         return DEFAULT_DATA.copy()
12
13 def save_data(data):
14     with open(DATA_FILE, "w", encoding="utf-8") as f:
15         json.dump(data, f, indent=4, ensure_ascii=False)
16

```

Gambar 3.1.5 Data Manage

Bagian ini bertanggung jawab terhadap proses membaca dan menyimpan data aplikasi ke dalam file JSON. Fungsi yang disediakan memastikan bahwa file data selalu tersedia, dapat dibaca dengan aman, dan dapat diperbarui setiap kali terjadi perubahan data. Dengan mekanisme ini, data aplikasi bersifat persistensi, artinya tetap tersimpan meskipun aplikasi ditutup dan dijalankan kembali.



```

1  def ensure_file():
2      if not os.path.exists(DATA_FILE):
3          save_data(DEFAULT_DATA)
4
5  def load_data():
6      ensure_file()
7      try:
8          with open(DATA_FILE, "r", encoding="utf-8") as f:
9              return json.load(f)
10     except Exception:
11         return DEFAULT_DATA.copy()
12
13 def save_data(data):
14     with open(DATA_FILE, "w", encoding="utf-8") as f:
15         json.dump(data, f, indent=4, ensure_ascii=False)
16

```

Gambar 3.1.5 Data Manage

Pada bagian ini diatur proses pengambilan dan pembaruan saldo pengguna, serta pencatatan riwayat pemesanan tiket. Setiap transaksi pemesanan akan otomatis

disimpan beserta tanggal transaksi. Pencatatan tanggal ini penting sebagai dasar untuk pelacakan transaksi dan pembuatan laporan periodik, sehingga aplikasi mampu merekam aktivitas pengguna secara kronologis dan terstruktur.

```
1 def get_riwayat_bulan_ini():
2     riw = get_riwayat()
3     now = datetime.now()
4     awal_bulan = now.replace(day=1, hour=0, minute=0, second=0, microsecond=0)
5     hasil = []
6     for r in riw:
7         try:
8             tgl_str = r.get("tanggal")
9             if not tgl_str:
10                 continue
11             tgl = datetime.strptime(tgl_str, "%Y-%m-%d")
12             if tgl >= awal_bulan:
13                 hasil.append(r)
14         except Exception:
15             continue
16     return hasil
17 def export_riwayat_excel():
18     data = get_riwayat_bulan_ini()
19     wb = Workbook()
20     ws = wb.active
21     ws.title = "Riwayat Pemesanan"
22     ws.append(["Tanggal", "Judul Film", "Studio", "Jam", "Kursi", "Total"])
23     if not data:
24         print("Peringatan: Tidak ada data riwayat untuk bulan ini.")
25     else:
26         for d in data:
27             kursi = d.get("kursi", [])
28             kursi_str = ", ".join(kursi) if isinstance(kursi, list) else str(kursi)
29             ws.append([
30                 str(d.get("tanggal", "-")),
31                 str(d.get("judul", "-")),
32                 str(d.get("studio", "-")),
33                 str(d.get("jam", "-")),
34                 kursi_str,
35                 d.get("total", 0)
36             ])
37     output_file = "riwayat_pemesanan_bulan_ini.xlsx"
38     try:
39         wb.save(output_file)
40         print(f'Sukses! File "{output_file}" berhasil dibuat.')
41     except Exception as e:
42         print(f'Gagal menyimpan file: {e}')
```

Gambar 3.1.5 Data Manage

Bagian ini berfungsi untuk menyaring data riwayat pemesanan berdasarkan periode waktu tertentu, Data yang telah disaring kemudian dapat diekspor ke dalam file Excel menggunakan library openpyxl. Fitur ini memudahkan pengguna atau pengelola aplikasi dalam membuat laporan pemesanan yang rapi, mudah dibaca, dan siap digunakan untuk keperluan dokumentasi atau evaluasi.

```

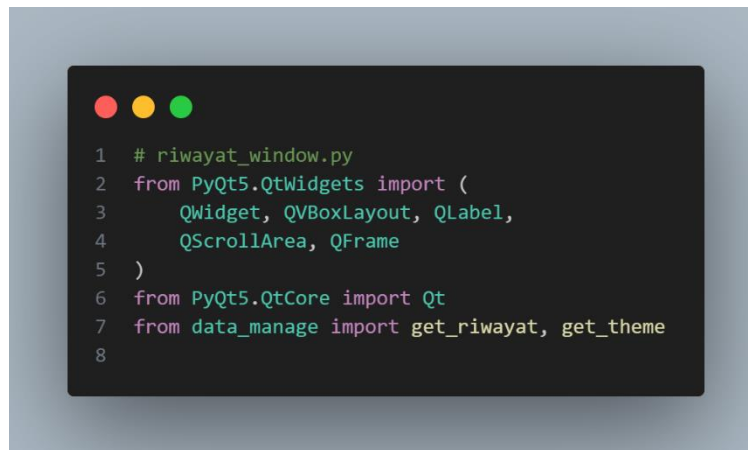
1 def get_kursi_terisi(key):
2     return load_data().get("kursi_terisi", {}).get(key, [])
3
4 def add_kursi_terisi(key, seats):
5     d = load_data()
6     d.setdefault("kursi_terisi", {}).setdefault(key, []).extend(seats)
7     d["kursi_terisi"][key] = sorted(list(set(d["kursi_terisi"][key])))
8     save_data(d)
9
10 def get_films():
11     return load_data().get("films", [])
12
13 def add_film(judul, deskripsi=""):
14     d = load_data()
15     fid = d.get("next_film_id", 1)
16     f = {"id": fid, "judul": judul, "deskripsi": deskripsi}
17     d.setdefault("films", []).append(f)
18     d["next_film_id"] = fid + 1
19     save_data(d)
20     return f
21
22 def find_film(film_id):
23     for f in get_films():
24         if f["id"] == film_id:
25             return f
26     return None
27
28 def get_theme():
29     return load_data().get("theme", DEFAULT_THEME.copy())

```

Gambar 3.1.5 Data Manage

Bagian terakhir mengatur data film yang tersedia, pengelolaan kursi agar tidak terjadi pemesanan ganda, serta pengambilan tema tampilan aplikasi. Dengan pemisahan fungsi ini, aplikasi menjadi lebih modular dan mudah dikembangkan. Perubahan pada data film, kursi, maupun tampilan dapat dilakukan tanpa memengaruhi bagian lain dari sistem.

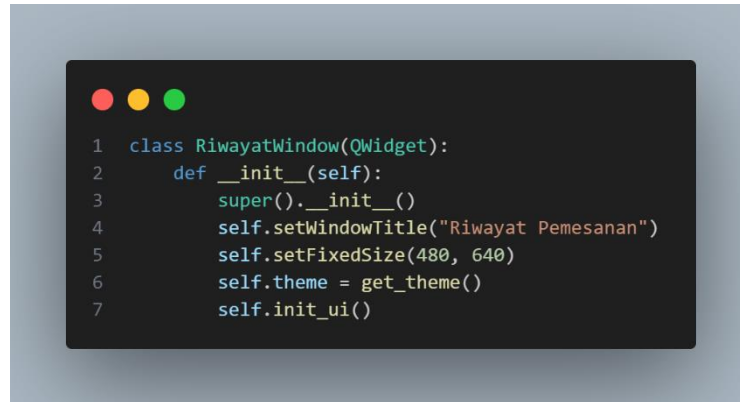
### 3.1.6 . RIWAYAT WINDOW

A screenshot of a code editor window with a dark background and light-colored text. The code is written in Python and shows imports for PyQt5.QtWidgets and PyQt5.QtCore, as well as functions from a data\_manage module. The code is numbered from 1 to 8.

```
1 # riwayat_window.py
2 from PyQt5.QtWidgets import (
3     QWidget, QVBoxLayout, QLabel,
4     QScrollArea, QFrame
5 )
6 from PyQt5.QtCore import Qt
7 from data_manage import get_riwayat, get_theme
8
```

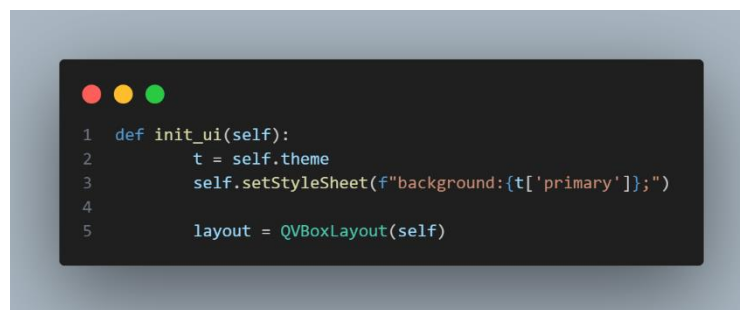
Gambar 3.1.6 Riwayat Window

Kode **riwayat\_window.py** ini berfungsi sebagai bagian dari aplikasi berbasis PyQt5 yang digunakan untuk menampilkan jendela riwayat data. Pada bagian awal, kode mengimpor berbagai komponen antarmuka dari modul PyQt5.QtWidgets seperti QWidget, QVBoxLayout, QLabel, QScrollArea, dan QFrame yang digunakan untuk membangun tampilan grafis, serta Qt dari PyQt5.QtCore untuk pengaturan perilaku dan perataan elemen antarmuka. Selain itu, kode juga mengimpor fungsi `get_riwayat` dan `get_theme` dari modul `data_manage`, yang kemungkinan digunakan untuk mengambil data riwayat pengguna dan pengaturan tema tampilan aplikasi. Secara keseluruhan, file ini berperan sebagai modul antarmuka yang mengelola tampilan riwayat dengan dukungan layout vertikal dan area gulir agar data dapat ditampilkan secara rapi dan mudah diakses oleh pengguna.



Gambar 3.1.6 Riwayat Window

Potongan kode ini mendefinisikan sebuah kelas bernama **RiwayatWindow** yang merupakan turunan dari **QWidget**, sehingga berfungsi sebagai jendela antarmuka grafis dalam aplikasi PyQt5. Pada metode `__init__`, konstruktor induk dipanggil menggunakan `super().__init__()` agar properti dasar **QWidget** dapat diinisialisasi dengan benar. Selanjutnya, judul jendela diatur menjadi "**Riwayat Pemesanan**", dan ukuran jendela dikunci pada lebar 480 piksel serta tinggi 640 piksel agar tidak dapat diubah oleh pengguna. Variabel `self.theme` digunakan untuk menyimpan pengaturan tema yang diambil melalui fungsi `get_theme()`, yang kemungkinan berisi informasi seperti warna atau gaya tampilan. Terakhir, metode `init_ui()` dipanggil untuk membangun dan mengatur seluruh komponen antarmuka pengguna yang akan ditampilkan pada jendela riwayat tersebut.



Gambar 3.1.6 Riwayat Window

Potongan kode ini mendefinisikan metode **init\_ui** yang berfungsi untuk mengatur tampilan antarmuka pada jendela riwayat. Variabel `t` digunakan sebagai penyingkat dari `self.theme` agar lebih mudah diakses, kemudian warna latar belakang jendela diatur menggunakan `setStyleSheet` dengan mengambil nilai warna utama (`primary`) dari tema tersebut. Setelah itu, sebuah objek **QVBoxLayout** dibuat dan langsung diterapkan ke widget utama, yang berfungsi untuk menyusun seluruh komponen antarmuka secara vertikal dari atas ke bawah sehingga tampilan menjadi lebih rapi dan terstruktur.



```

1 # Header
2 header = QLabel("RIWAYAT TIKET")
3 header.setAlignment(Qt.AlignCenter)
4 header.setStyleSheet(
5     "font-size:16pt; color:white; font-weight:bold;"
6 )
7 layout.addWidget(header)
8
9 scroll = QScrollArea()
10 scroll.setWidgetResizable(True)
11
12 container = QFrame()
13 v = QVBoxLayout(container)
14
15 riw = get_riwayat()
16
17 if not riw:
18     lbl = QLabel("Belum ada pemesanan.")
19     lbl.setAlignment(Qt.AlignCenter)
20     lbl.setStyleSheet("color:white;")
21     v.addWidget(lbl)
22 else:
23     for i, tkt in enumerate(reversed(riw), start=1):
24         struk = QFrame()
25         struk.setStyleSheet("""
26             background: white;
27             border-radius: 8px;
28             padding: 6px;
29         """)
30
31         s_layout = QVBoxLayout(struk)

```

Gambar 3.1.6 Riwayat Window

Bagian kode ini berfungsi untuk membangun tampilan utama isi jendela riwayat tiket. Pertama, sebuah label header bertuliskan **"RIWAYAT TIKET"** dibuat, disejajarkan ke tengah, dan diberi gaya teks berwarna putih, tebal, serta berukuran besar, lalu ditambahkan ke layout utama. Selanjutnya, dibuat `QScrollArea` agar daftar riwayat dapat digulir ketika jumlah data banyak, dengan pengaturan agar ukurannya menyesuaikan konten. Sebuah `QFrame` digunakan sebagai wadah konten scroll, dan `QVBoxLayout` diterapkan untuk menyusun setiap item riwayat secara vertikal. Data riwayat kemudian diambil menggunakan fungsi `get_riwayat()`. Jika data tersebut kosong, aplikasi menampilkan pesan **"Belum ada pemesanan."** di tengah layar dengan teks berwarna putih. Namun, jika data tersedia, program akan melakukan perulangan terhadap riwayat yang dibalik urutannya agar data terbaru tampil lebih dulu. Untuk setiap tiket, dibuat sebuah `QFrame` bernama `struk` yang diberi latar belakang putih, sudut membulat, dan padding, sehingga setiap riwayat tiket terlihat seperti kartu atau struk yang rapi, kemudian disiapkan layout vertikal khusus untuk menampung detail tiket tersebut.



```

1 # Judul Struk
2 title = QLabel(f"STRUK TIKET #{i}")
3 title.setAlignment(Qt.AlignCenter)
4 title.setStyleSheet("font-weight:bold;")
5 s_layout.addWidget(title)
6
7 s_layout.addWidget(QLabel("====="))

```

Gambar 3.1.6 Riwayat Window

Potongan kode ini digunakan untuk menambahkan bagian judul pada setiap kartu atau struk tiket yang ditampilkan. Sebuah QLabel dibuat dengan teks **"STRUK TIKET #i"**, di mana nilai *i* menunjukkan nomor urutan tiket, kemudian teks tersebut disejajarkan ke tengah dan diberi gaya huruf tebal agar terlihat menonjol sebagai judul. Label judul ini lalu dimasukkan ke dalam layout vertikal *s\_layout* milik kartu struk. Setelah itu, sebuah QLabel tambahan berisi garis pemisah ditambahkan untuk memberikan batas visual antara judul dan isi detail tiket, sehingga tampilan struk menjadi lebih rapi dan mudah dibaca.



```

1 # Isi Struk
2 s_layout.addWidget(QLabel(f"Film : {tkt.get('film')} or tkt.get('judul')}"))
3 s_layout.addWidget(QLabel(f"Jam : {tkt.get('jam')}"))
4 s_layout.addWidget(
5     QLabel(f"Kursi : {' , '.join(tkt.get('kursi', []))}")
6 )
7 s_layout.addWidget(
8     QLabel(f"Total : Rp {tkt.get('total', 0):,}")
9 )
10
11 v.addWidget(struk)
12
13 scroll.setWidget(container)
14 layout.addWidget(scroll)
15

```

Gambar 3.1.6 Riwayat Window

Bagian kode ini berfungsi untuk menampilkan isi detail pada setiap struk tiket yang ada di riwayat pemesanan. Informasi film ditampilkan menggunakan QLabel dengan mengambil data judul film dari kunci film atau judul pada data tiket, sehingga tetap fleksibel jika salah satu tidak tersedia. Selanjutnya, jam penayangan ditampilkan, diikuti dengan daftar kursi yang digabung menjadi satu teks menggunakan tanda koma agar mudah dibaca. Total harga tiket juga ditampilkan dalam format rupiah dengan pemisah ribuan untuk meningkatkan keterbacaan. Setiap label detail ini ditambahkan ke dalam layout vertikal *s\_layout* milik kartu struk, lalu kartu struk tersebut dimasukkan ke layout utama *v* agar tersusun rapi secara vertikal. Setelah semua data dimasukkan, container ditetapkan sebagai widget di dalam QScrollArea, dan area gulir tersebut ditambahkan ke layout utama jendela, sehingga seluruh riwayat tiket dapat digulir jika jumlahnya melebihi tinggi layar.


### 3.1.7 . SEAT\_SELECTION



```
1 # seat_selection_window.py
2 from PyQt5.QtWidgets import QWidget, QLabel, QPushButton, QGridLayout, QVBoxLayout, QHBoxLayout, QMessageBox
3 from PyQt5.QtCore import Qt
4 from data_manage import get_saldo, set_saldo, get_kursi_terisi, add_kursi_terisi, add_riwayat, get_theme
5 import math
6 from datetime import datetime
7 from data_manage import add_riwayat
```

Gambar 3.1.7 Seat Selection

Bagian kode import ini digunakan pada Seat Selection Window untuk menyediakan komponen antarmuka PyQt5, mengatur tata letak tampilan pemilihan kursi, serta menampilkan pesan kepada pengguna. Selain itu, kode ini memanfaatkan modul data\_manage untuk mengelola saldo, data kursi yang telah terisi, riwayat pembelian tiket, dan tema tampilan aplikasi, serta menggunakan datetime dan math untuk mendukung pencatatan waktu transaksi dan perhitungan yang diperlukan dalam proses pemesanan kursi.



```
1 class SeatSelectionWindow(QWidget):
2     def __init__(self, judul, studio="Studio 1", jam="20:00"):
3         super().__init__()
4         self.judul = judul
5         self.studio = studio
6         self.jam = jam
7         self.setWindowTitle(f"Pilih Kursi - {judul}")
8         self.setFixedSize(720, 720)
9         self.theme = get_theme()
10        self.selected = set()
11        self.init_ui()
```

Gambar 3.1.7 Seat Selection

digunakan untuk menampilkan dan mengelola pemilihan kursi bioskop. Pada metode `__init__`, parameter judul, studio, dan jam digunakan untuk menyimpan informasi film yang dipilih oleh pengguna. Jendela kemudian diberi judul sesuai nama film dan diatur dengan ukuran tetap agar tampilan konsisten. Selain itu, kelas ini mengambil tema warna aplikasi melalui fungsi `get_theme`, menyiapkan variabel `selected` untuk menyimpan kursi yang dipilih pengguna, serta memanggil metode `init_ui` untuk membangun tampilan antarmuka pemilihan kursi.

```

1 def init_ui(self):
2     t = self.theme
3     self.setStyleSheet(f"background:{t['primary']}; color:white;")
4     v = QVBoxLayout()
5     title = QLabel(f"Pilih Kursi - {self.judul} | {self.studio} | {self.jam}")
6     title.setStyleSheet("font-size:14pt; font-weight:bold; color:white;")
7     v.addWidget(title)
8
9     screen_lbl = QLabel("[ LAYAR ]")
10    screen_lbl.setAlignment(Qt.AlignCenter)
11    screen_lbl.setStyleSheet("background:#000; color:#ccc; padding:6px;")
12    v.addWidget(screen_lbl)
13
14    grid_frame = QWidget()
15    grid = QGridLayout()
16    grid.setSpacing(8)
17
18    key = f"{self.judul}|{self.studio}|{self.jam}"
19    sold = set(get_kursi_terisi(key))
20    rows = [chr(i) for i in range(65, 65+8)] # A..H
21    cols = range(1, 9) # 1..8
22
23    self.btns = {}
24    for r_idx, r in enumerate(rows):
25        for c_idx, c in enumerate(cols):
26            kode = f"{r}{c}"
27            btn = QPushButton(kode)
28            btn.setFixedSize(64, 48)
29            if kode in sold:
30                btn.setStyleSheet(f"background:{t['sold']}; color:white; border-radius:6px;")
31                btn.setEnabled(False)
32            else:
33                btn.setStyleSheet(f"background:{t['available']}; color:black; border-radius:6px;")
34                btn.clicked.connect(lambda checked, k=kode: self.toggle_seat(k))
35            grid.addWidget(btn, r_idx, c_idx)
36            self.btns[kode] = btn
37
38    grid_frame.setLayout(grid)
39    v.addWidget(grid_frame)

```

Gambar 3.1.7 Seat Selection

Metode `init_ui` berfungsi untuk membangun tampilan antarmuka pemilihan kursi pada Seat Selection Window. Pada awal metode, tema warna aplikasi diterapkan sebagai latar belakang jendela, kemudian dibuat layout vertikal (`QVBoxLayout`) yang menampung seluruh komponen tampilan. Label judul ditampilkan untuk menunjukkan informasi film, studio, dan jam tayang, diikuti dengan label “[LAYAR]” sebagai penanda posisi layar bioskop.

Selanjutnya, dibuat layout grid (`QGridLayout`) untuk menampilkan kursi bioskop dalam bentuk baris dan kolom. Data kursi yang sudah terjual diambil menggunakan fungsi `get_kursi_terisi`, kemudian setiap kursi dibuat sebagai tombol (`QPushButton`) dengan kode kursi seperti A1 hingga H8. Kursi yang sudah terisi akan diberi warna khusus, dinonaktifkan, dan tidak dapat dipilih, sedangkan kursi yang masih tersedia dapat diklik oleh pengguna.

Setiap tombol kursi yang tersedia dihubungkan dengan fungsi `toggle_seat` untuk mengatur proses pemilihan dan pembatalan kursi. Semua tombol kursi disimpan dalam dictionary `self.btns` agar mudah dikelola, lalu grid kursi ditambahkan ke dalam layout utama sehingga seluruh tampilan pemilihan kursi dapat ditampilkan secara rapi dalam jendela aplikasi.

```

1
2 # total & button
3 h = QHBoxLayout()
4 self.total_lbl = QLabel("Total Harga: Rp 0")
5 self.total_lbl.setStyleSheet("font-weight:bold; color:white;")
6 h.addWidget(self.total_lbl)
7 h.addStretch()
8 buy_btn = QPushButton("Beli Tiket")
9 buy_btn.setFixedSize(140, 40)
10 buy_btn.setStyleSheet(f"background:{t['accent']}; color:black; font-weight:bold;")
11 buy_btn.clicked.connect(self.buy)
12 h.addWidget(buy_btn)
13 v.addLayout(h)
14
15 self.setLayout(v)
16
17 def toggle_seat(self, kode):
18     t = self.theme
19     if kode in self.selected:
20         self.selected.remove(kode)
21         self.btns[kode].setStyleSheet(f"background:{t['available']}; color:black; border-radius:6px;")
22     else:
23         self.selected.add(kode)
24         self.btns[kode].setStyleSheet(f"background:{t['selected']}; color:black; border-radius:6px;")
25     self.total_lbl.setText(f"Total Harga: Rp {len(self.selected) * HARGA_TIKET}")
26
27 def buy(self):
28     if len(self.selected) == 0:
29         QMessageBox.warning(self, "Info", "Pilih minimal 1 kursi.")
30         return
31     total = len(self.selected) * HARGA_TIKET
32     s = get_saldo()
33     if s < total:
34         QMessageBox.warning(self, "Saldo", f"Saldo tidak cukup. Total Rp {total}, Saldo Rp {s}")
35         return

```

Gambar 3.1.7 Seat Selection

Bagian kode ini digunakan untuk menampilkan total harga tiket dan tombol pembelian pada Seat Selection Window. Layout horizontal (QHBoxLayout) mengatur posisi label total harga dan tombol Beli Tiket agar sejajar. Total harga akan berubah secara otomatis sesuai jumlah kursi yang dipilih, sedangkan tombol Beli Tiket dihubungkan dengan fungsi buy untuk memproses transaksi pembelian.

Fungsi toggle\_seat mengatur logika pemilihan dan pembatalan kursi, termasuk perubahan warna tombol serta perhitungan total harga berdasarkan jumlah kursi yang dipilih. Sementara itu, fungsi buy melakukan validasi pembelian, yaitu memastikan pengguna telah memilih kursi dan memiliki saldo yang cukup sebelum proses pembelian dapat dilanjutkan

```

1 # deduct, add riwayat, mark seats sold
2     set_saldo(s - total)
3     key = f"{self.judul}|{self.studio}|{self.jam}"
4     add_kursi_terisi(key, list(self.selected))
5     add_riwayat({
6         "judul": self.judul,
7         "studio": self.studio,
8         "jam": self.jam,
9         "kursi": sorted(list(self.selected)),
10        "total": total,
11        "waktu": __import__("datetime").datetime.utcnow().isoformat()
12    })
13    QMessageBox.information(self, "Sukses", "Tiket berhasil dibeli.")
14    # update UI: disable purchased buttons
15    for k in list(self.selected):
16        self.btns[k].setEnabled(False)
17        self.btns[k].setStyleSheet(f"background:{self.theme['sold']}; color:white; border-radius:6px;")
18    self.selected.clear()
19    self.total_lbl.setText("Total Harga: Rp 0")

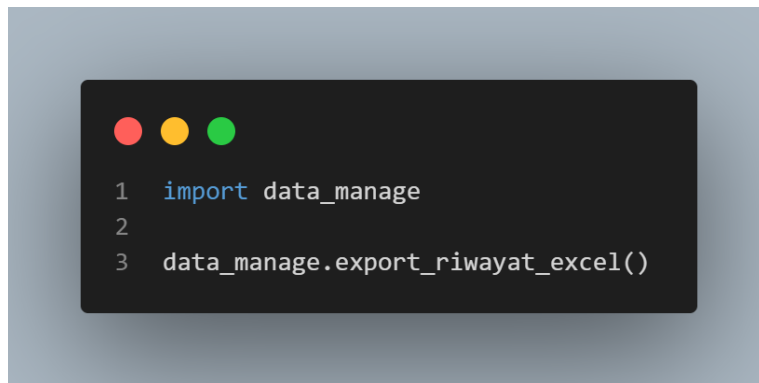
```

Gambar 3.1.7 Seat Selection

Bagian kode ini berfungsi untuk menyelesaikan proses pembelian tiket setelah semua validasi terpenuhi. Saldo pengguna dikurangi sesuai total harga tiket menggunakan fungsi `set_saldo`, kemudian data kursi yang dibeli disimpan sebagai kursi terisi berdasarkan kombinasi judul film, studio, dan jam tayang. Selain itu, detail transaksi seperti judul film, studio, jam, kursi yang dipilih, total pembayaran, dan waktu transaksi dicatat ke dalam riwayat pembelian melalui fungsi `add_riwayat`.

Setelah transaksi berhasil, sistem menampilkan pesan konfirmasi kepada pengguna dan memperbarui tampilan antarmuka dengan menonaktifkan tombol kursi yang telah dibeli serta mengubah warnanya menjadi tanda kursi terjual. Terakhir, daftar kursi yang dipilih dikosongkan dan total harga dikembalikan ke nol agar aplikasi siap digunakan untuk transaksi berikutnya.

### 3.1.8 . Export Excel



Gambar 3.1.8 Export Excel

Kode ini berfungsi untuk menjalankan proses ekspor data riwayat pemesanan tiket ke dalam file Excel. Kode ini harus di-run secara terpisah (manual), tidak otomatis saat aplikasi utama dijalankan. Setelah dijalankan, sistem akan membuat file Excel berisi riwayat pemesanan tiket sesuai dengan transaksi yang telah dilakukan pengguna. File Excel yang dihasilkan tidak terbuka langsung di aplikasi, melainkan harus dibuka melalui File Explorer. Isi file Excel tersebut menampilkan data pemesanan seperti tanggal, judul film, studio, jam tayang, kursi, dan total pembayaran, sesuai dengan riwayat pembelian tiket pengguna.

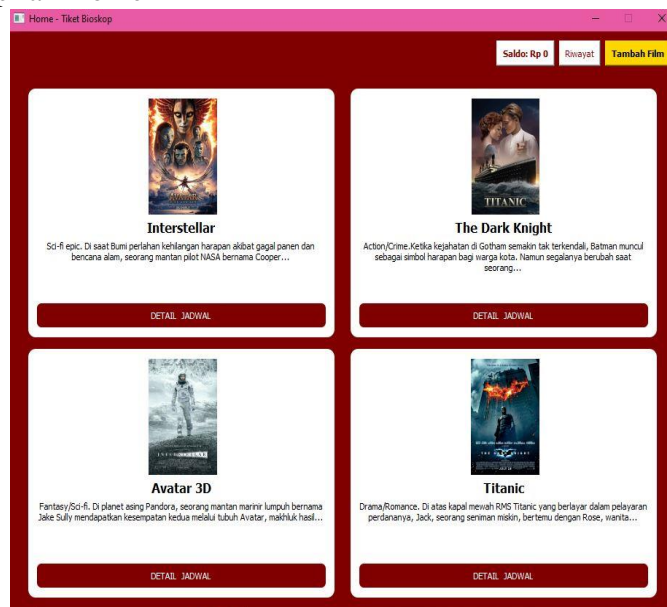
## 3.2 SCREENSHOT APLIKASI

### 3.2.1 Menu Login



Gambar 3.2.1 Menu Login

### 3.2.2 Tampilan Home



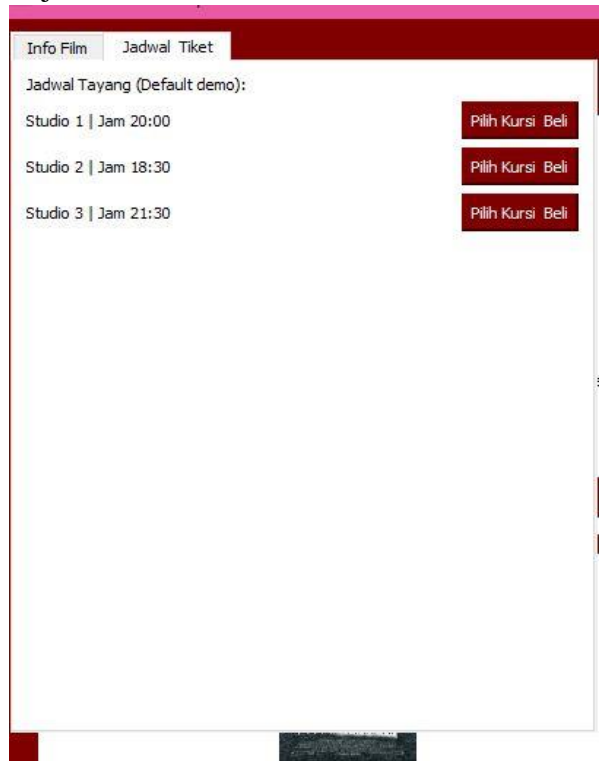
Gambar 3.2.2 Tampilan Home

3.2.3 Dekripsi Film



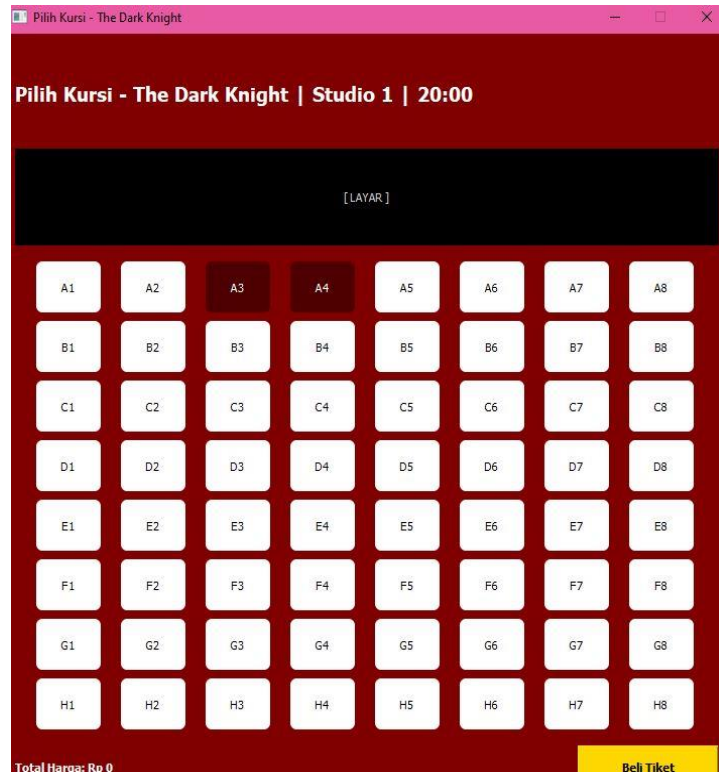
Gambar 3.2.3 Deskripsi Film

3.2.4 Pemilihan jadwal



Gambar 3.2.4 Pemilihan jadwal

### 3.2.5 Pemilihan Kursi



Gambar 3.2.5 Pemilihan Kursi

### 3.2.6 Riwayat Pemesanan

**RIWAYAT PEMESANAN**

**STRUK TIKET #1**

=====

Film : Interstellar

Jam : 21:30

Kursi : A5, A6

Total : Rp 70,000

**STRUK TIKET #2**

=====

Film : Interstellar

Jam : 20:00

Kursi : A5, A6

Total : Rp 70,000

**STRUK TIKET #3**

=====

Film : Titanic

Jam : 18:30

**DETAIL JADWAL**

Gambar 3.2.6 Riwayat Pemesanan

## **BAB 4**

### **LAMPIRAN**

#### **4.1 KODE PROGRAM LENGKAP**

Berikut adalah folder lengkap berisi code aplikasi tiket bioskop.

[https://drive.google.com/drive/folders/1GdOWToDdtcnnLreZgkhvrfGgUlpTWsS?usp=drive\\_link](https://drive.google.com/drive/folders/1GdOWToDdtcnnLreZgkhvrfGgUlpTWsS?usp=drive_link)

## DAFTAR PUSAKA

Codementor. (n.d.). Building desktop applications with PyQt. Diakses dari

<https://www.codementor.io/blog/desktop-app-pyqt-8jwfnotuly>

Google. (n.d.). Python GUIs. Diakses dari <https://share.google/bXp0qQkOIjC1c1agZ>

Python GUIs. (n.d.). Tutorial PyQt5. Diakses dari [https://www.pythonguis-com.translate.goog/pyqt5/tutorial/?x\\_tr\\_sl=en&x\\_tr\\_tl=id&x\\_tr\\_hl=id&x\\_tr\\_pto=tc](https://www.pythonguis-com.translate.goog/pyqt5/tutorial/?x_tr_sl=en&x_tr_tl=id&x_tr_hl=id&x_tr_pto=tc)

YouTube. (n.d.). Tutorial pengembangan aplikasi desktop menggunakan Python. Diakses dari

<https://share.google/plBLhDs7HUiMzXaY8>