LAPORAN PRAKTIKUM

POSTTEST 6

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Danendra Hazzel Putra Wahana 2409106096

Kelas C1’24

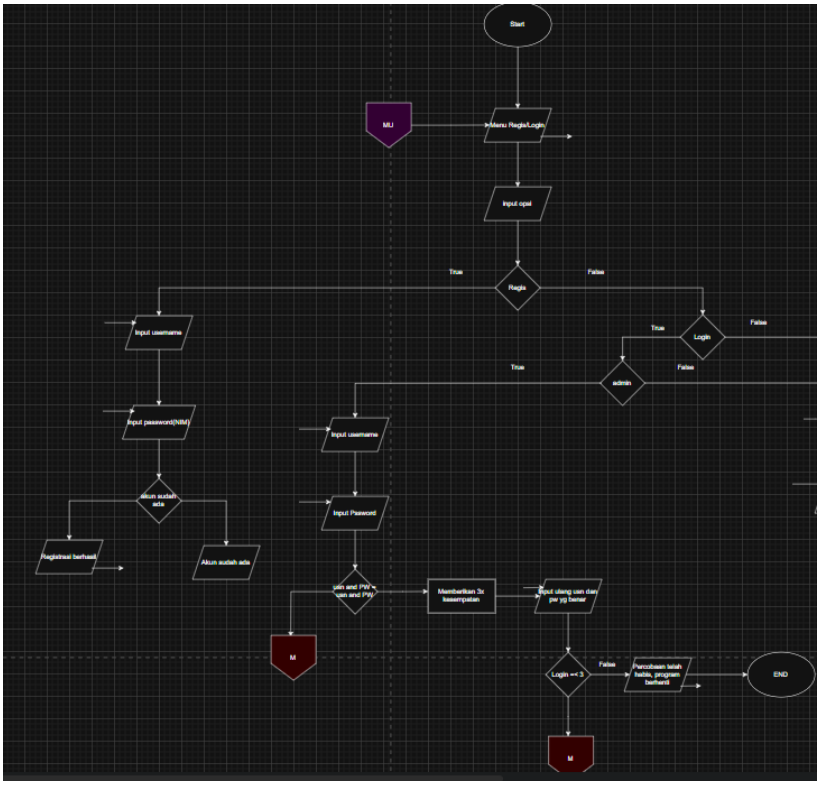
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

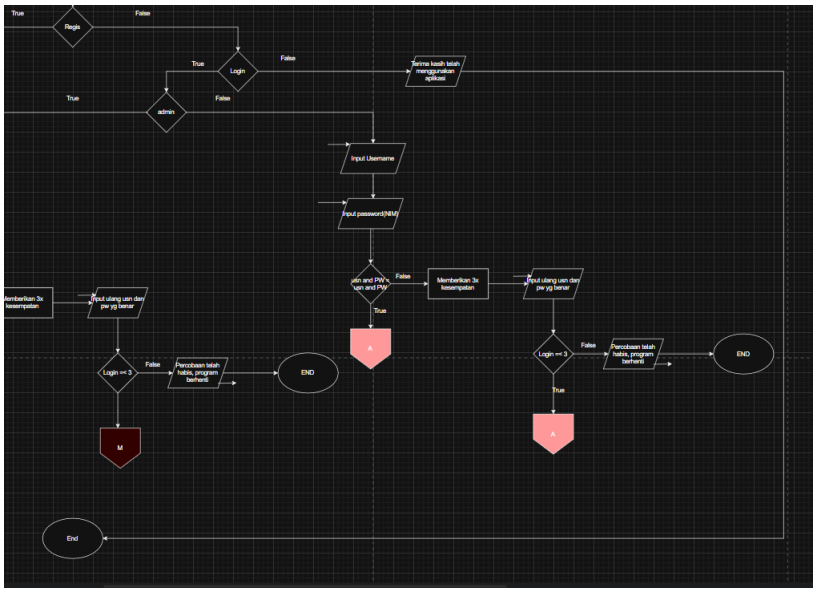
SAMARINDA

2025

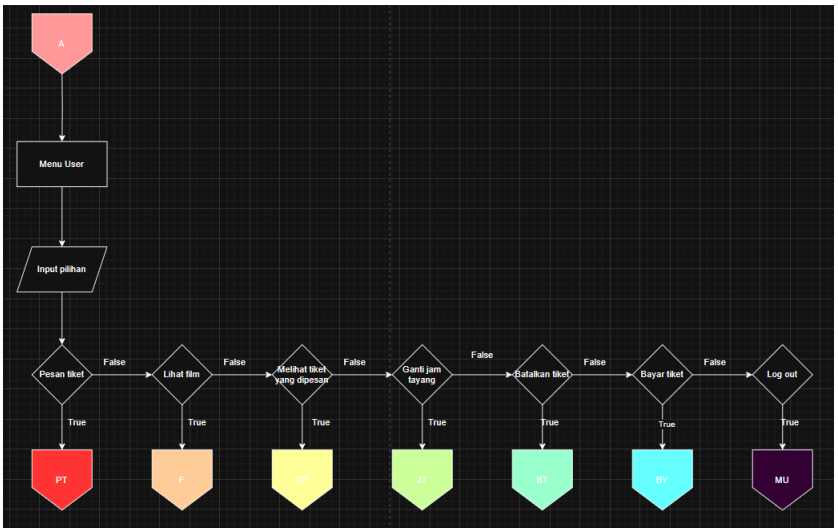
1. Flowchart



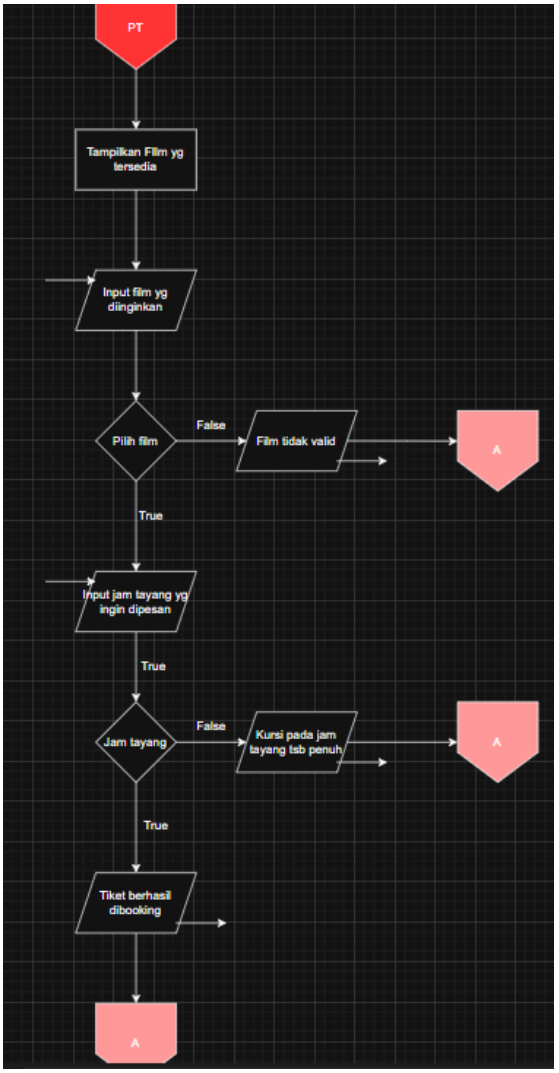
Gambar 1.1 Bagian menu regis



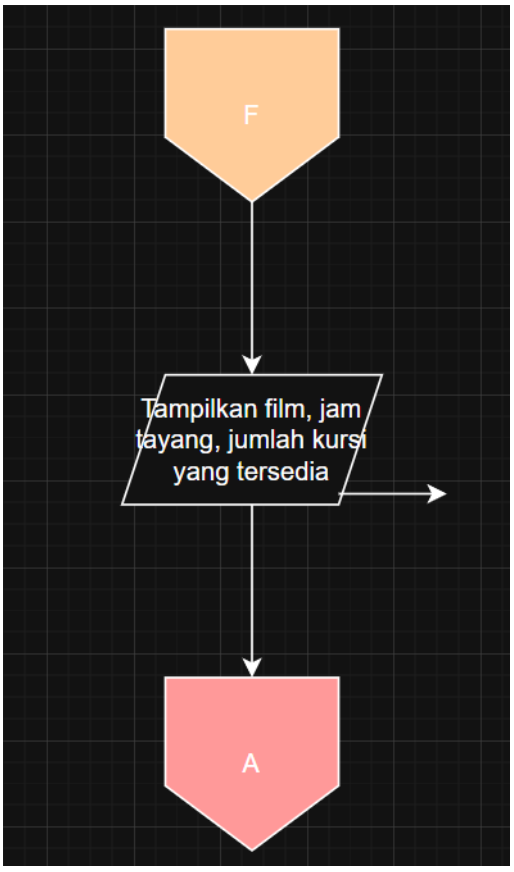
Gambar 1.2 Bagian menu regis bagian 2



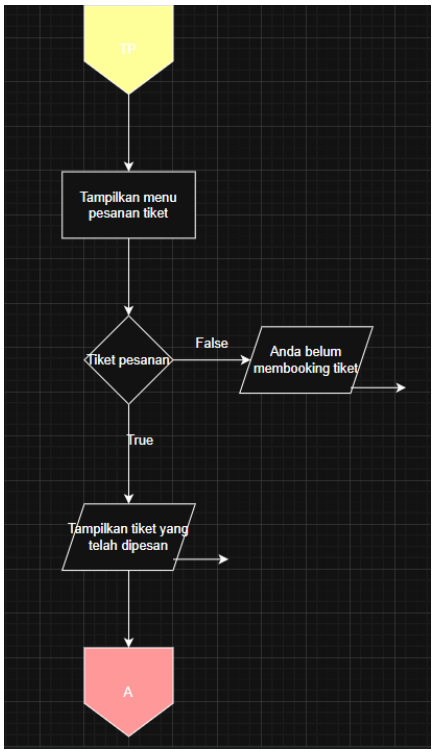
Gambar 1.3 Bagian menu user



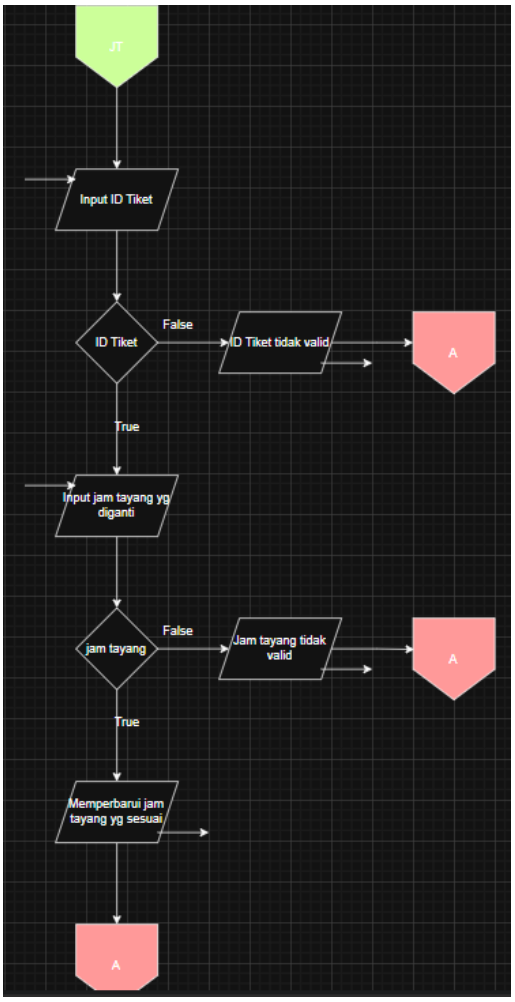
Gambar 1.4 Bagian pesan tiket



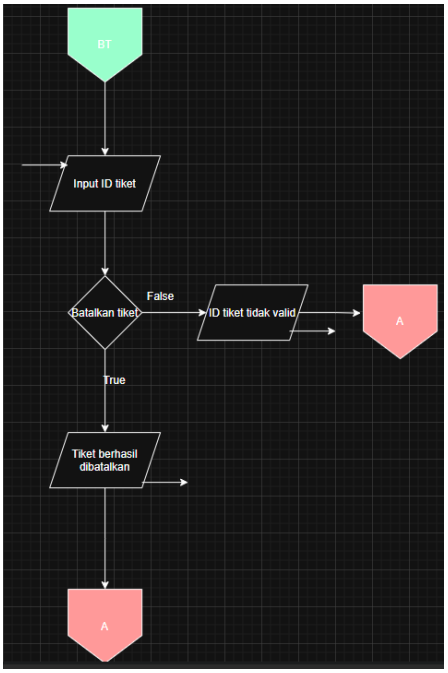
Gambar 1.5 Bagian menampilkan film yang tersedia



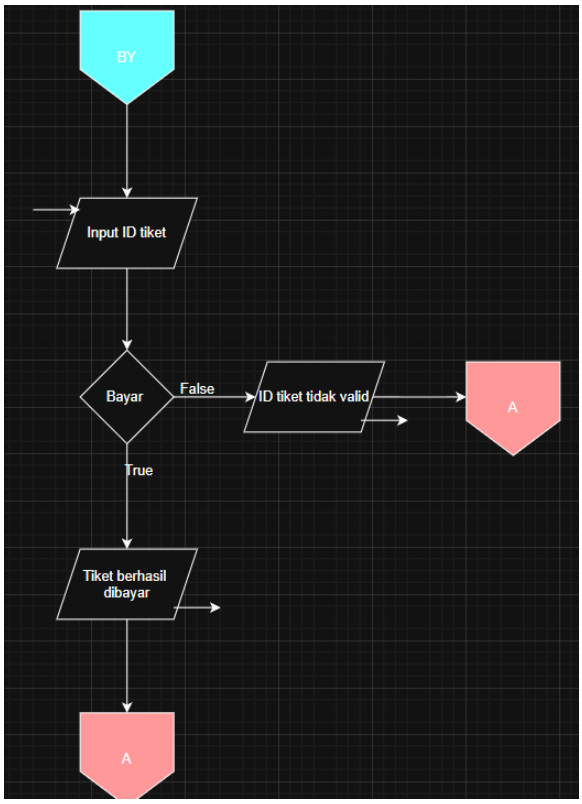
Gambar 1.6 Bagian menampilkan tiket yang sudah dibeli.



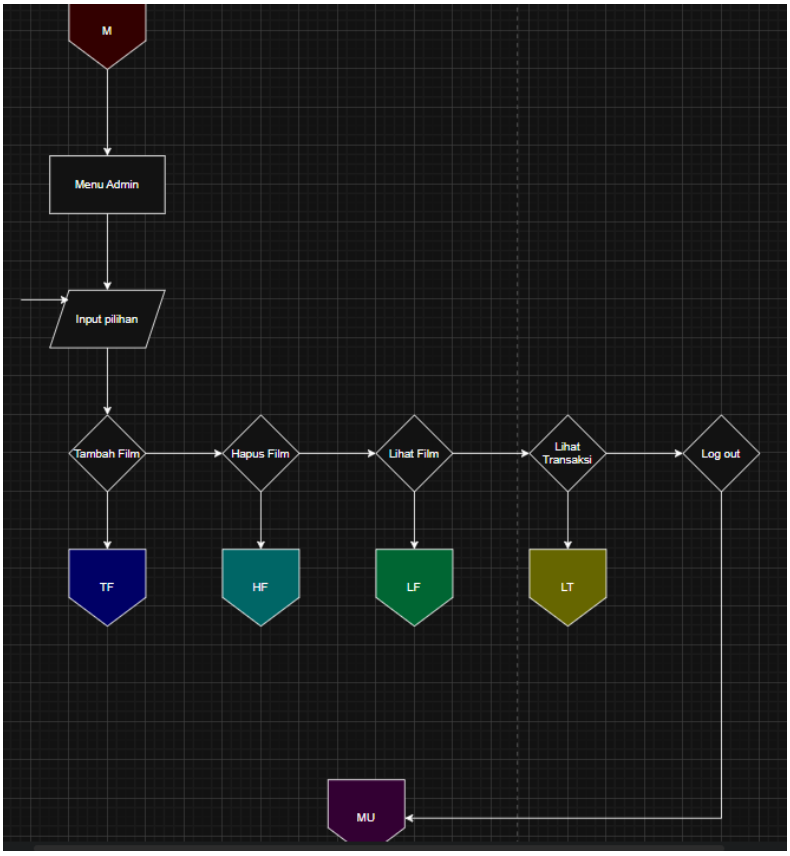
Gambar 1.7 Bagian mengganti jam tayang



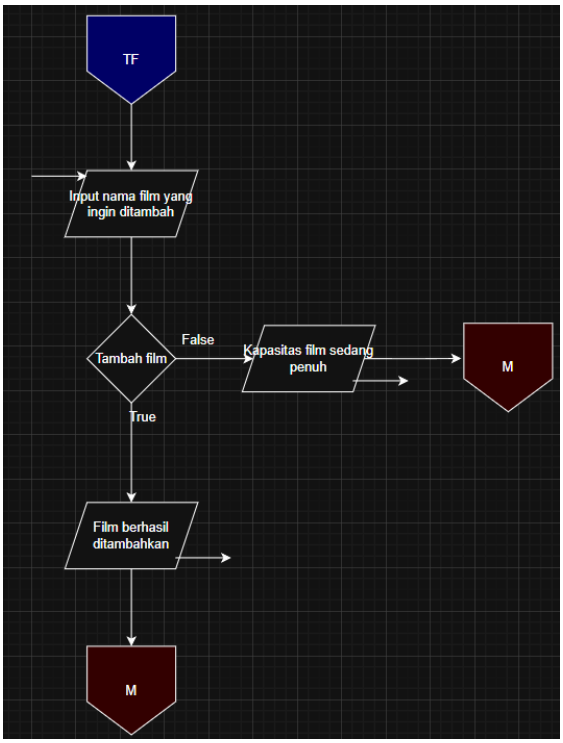
Gambar 1.8 Bagian pembatalan tiket



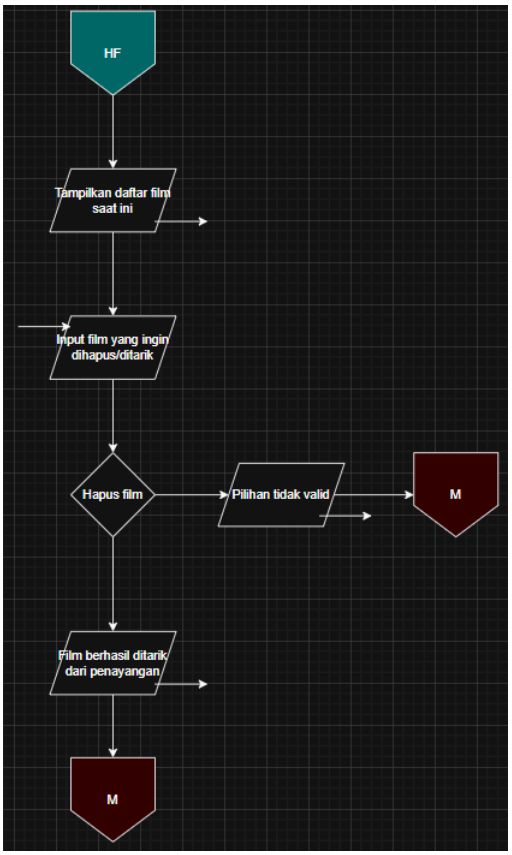
Gambar 1.9 Bagian bayar tiket



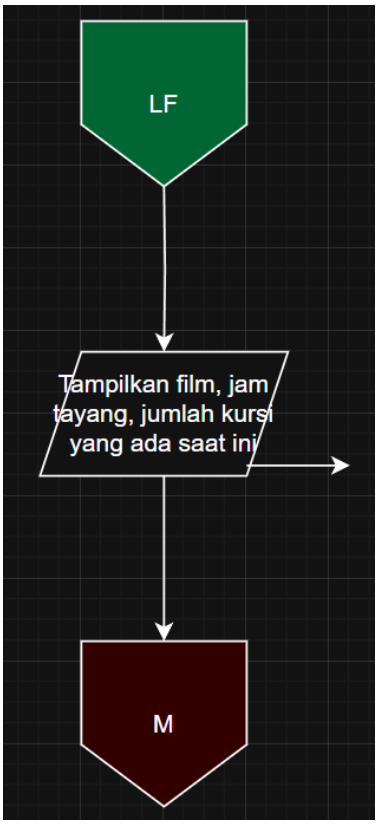
Gambar 1.10 Bagian menu admin



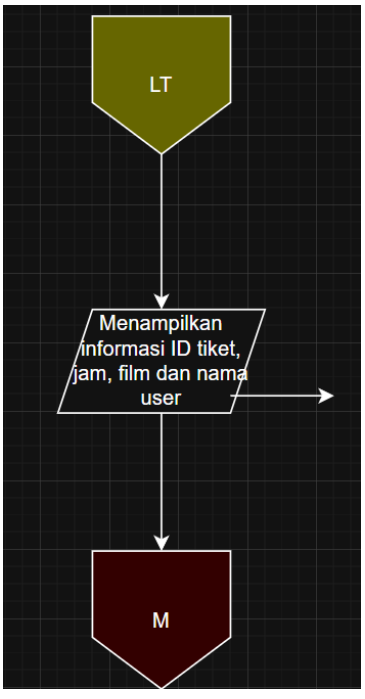
Gambar 1.11 Bagian menambah film



Gambar 1.12 Bagian penarikan film



Gambar 1.13 Bagian melihat film saat ini



Gambar 1.14 Bagian informasi pemesanan tiket user

2. Analisis Program

Program ini adalah sebuah pemesanan tiket online sederhana yang memungkinkan Pengguna dapat login sebagai pembeli maupun admin. Tentu fitur admin maupun user memilki perbedaannya masing-masing. Berikut adalah penjelasan fitur dari admin maupun pembeli :

a) Admin dapat menambahkan atau menghapus film, melihat daftar film yang tersedia, serta melihat transaksi pemesanan tiket yang dilakukan beberapa user.

b) User dapat memesan tiket, melihat film yang tersedia, melihat tiket yang sudah dipesan, mengganti jam tayang, membatalkan tiket, dan melakukan pembayaran.

Program ini bertujuan untuk memudahkan para user untuk memesan/membooking tiket dengan mudah secara online yang dapat dilakukan kapan saja dan dimana saja serta admin yang dapat mengkoordinasi pemesanan dan informasi pada bioskop saat ini dengan mudah hanya melalui dari aplikasi.

3. Source Code

```cpp
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

// Nested struct untuk menyimpan data jam tayang dan kursi terisi
struct JamTayang {
    string jam;
    int kursiTerisi;
};

// Struct untuk menyimpan data film
struct FilmData {
    string namaFilm;
    JamTayang jamTayang[5];
};

// Struct untuk menyimpan data user
struct User {
    string username;
    string password;
    bool isAdmin;
};

// Struct untuk menyimpan data booking
struct Booking {
    int filmId;
    int jamId;
    bool isPaid;
    int userId;  // Menyimpan ID user yang melakukan booking
};

// Deklarasi fungsi dan prosedur dengan pointer
void tampilkanMenuUtama();
void registrasiUser(User *users, int *userCount);
bool login(User *users, int userCount, bool *isAdmin, int *currentUserId);
bool menuAdmin(FilmData *films, int *filmCount, Booking *bookings, int *bookingCount, const int *maksKursi);
bool menuUser(FilmData *films, int filmCount, Booking *bookings, int *bookingCount, int *currentUserId, User *users, const int *maksKursi);
void tambahFilm(FilmData *films, int *filmCount);
void hapusFilm(FilmData *films, int *filmCount);
void lihatFilm(const FilmData *films, int filmCount, const int *maksKursi);
void lihatTransaksi(const Booking *bookings, int bookingCount, const FilmData *films);
void pesanTiket(FilmData *films, int filmCount, Booking *bookings, int *bookingCount, int *currentUserId, const int *maksKursi);
void lihatTiketUser(const Booking *bookings, int bookingCount, const FilmData *films, int currentUserId);
void gantiJamTayang(FilmData *films, Booking *bookings, int bookingCount, int currentUserId, const int *maksKursi);
void batalkanTiket(FilmData *films, Booking *bookings, int *bookingCount);
void bayarTiket(Booking *bookings, int *bookingCount, const FilmData *films);
void tampilkanFilmTersedia(const FilmData *films, int filmCount, int index, const int *maksKursi);
void tampilkanJamTayang(const JamTayang *jamTayang, int index, const int *maksKursi);
void tampilkanJamTayang(const JamTayang &jamTayang);
```

```
// Fungsi sorting baru
void selectionSortFilmDescending(FilmData *films, int filmCount) {
    for (int i = 0; i < filmCount - 1; i++) {
        int max_idx = i;
        for (int j = i + 1; j < filmCount; j++) {
            if (films[j].namaFilm > films[max_idx].namaFilm) {
                max_idx = j;
            }
        }
        if (max_idx != i) {
            swap(films[i], films[max_idx]);
        }
    }
}

void selectionSortJamAscending(JamTayang *jamTayang, int jamCount) {
    for (int i = 0; i < jamCount - 1; i++) {
        int min_idx = i;
        for (int j = i + 1; j < jamCount; j++) {
            if (jamTayang[j].jam < jamTayang[min_idx].jam) {
                min_idx = j;
            }
        }
        if (min_idx != i) {
            swap(jamTayang[i], jamTayang[min_idx]);
        }
    }
}

int main() {
    // Array of struct untuk menyimpan data film
    FilmData films[10] = {
        {"Kimi No Nawa", {{"10:00", 0}, {"12:00", 0}, {"15:00", 0}, {"17:00", 0}, {"19:00", 0}}},
        {"Godzilla : Minus one", {{"10:00", 0}, {"12:00", 0}, {"15:00", 0}, {"17:00", 0}, {"19:00", 0}}},
        {"Avengers : secret wars", {{"10:00", 0}, {"12:00", 0}, {"15:00", 0}, {"17:00", 0}, {"19:00", 0}}},
        {"Your Lie In April", {{"10:00", 0}, {"12:00", 0}, {"15:00", 0}, {"17:00", 0}, {"19:00", 0}}},
        {"Cek Toko Sebelah", {{"10:00", 0}, {"12:00", 0}, {"15:00", 0}, {"17:00", 0}, {"19:00", 0}}}
    };

    int filmCount = 5; // Jumlah film awal

    // Array untuk menyimpan data user
    User users[10];
    int userCount = 0;

    // Array untuk menyimpan booking
    Booking bookings[10];
    int bookingCount = 0;

    // Kapasitas maksimum per film dan jam
```

```
const int maksKursi = 20;

 // Inisialisasi admin dan user menggunakan pointer
 User *adminPtr = &users[userCount++];
 *adminPtr = {"admin", "selalubenar", true};  // Admin

 User *userPtr = &users[userCount++];
 *userPtr = {"Danendra Hazzel PW", "2409106096", false};  // User biasa

 int pilihan = 0;
 bool isRunning = true;
 int loginAttempts = 0;
 bool isAdmin = false;
 int currentUserId = -1;

 // Menu utama
 while (isRunning) {
     tampilkanMenuUtama();
     cin >> pilihan;

     switch (pilihan) {
         case 1: {
             registrasiUser(users, &userCount);
             break;
         }
         case 2: {
             if (login(users, userCount, &isAdmin, &currentUserId)) {
                 cout << "Login berhasil!\n";
                 cout << "Selamat Datang " << (isAdmin ? "Bos" : "Tuan") << " " << users[currentUserId].username << endl;

                 // Menu setelah login
                 bool isLoggedIn = true;
                 while (isLoggedIn) {
                     if (isAdmin) {
                         isLoggedIn = menuAdmin(films, &filmCount, bookings, &bookingCount, &maksKursi);
                     } else {
                         isLoggedIn = menuUser(films, filmCount, bookings, &bookingCount, &currentUserId, users, &maksKursi);
                     }
                 }
             } else {
                 loginAttempts++;
                 cout << "Login gagal, Username atau Password salah! Sisa percobaan anda sebanyak " << 3 - loginAttempts << endl;

                 if (loginAttempts >= 3) {
                     cout << "Kesempatan anda telah habis, silahkan mencoba lagi dalam beberapa saat.\n";
                     isRunning = false;
                 }
             }
             break;
         }
         case 3: {
```

```
                cout << "Terima kasih telah menggunakan aplikasi ini, enjoy the movie!\n";
                isRunning = false;
                break;
            }
            default: {
                cout << "Pilihan tidak valid!\n";
                break;
            }
        }
    }

    return 0;
}

void tampilkanMenuUtama() {
    cout << "\n===== MENU UTAMA =====\n";
    cout << "1. Registrasi\n";
    cout << "2. Login\n";
    cout << "3. Keluar\n";
    cout << "Pilihan Anda: ";
}

void registrasiUser(User *users, int *userCount) {
    if (*userCount >= 10) {
        cout << "Maaf, kapasitas user sudah penuh!\n";
        return;
    }

    cout << "\n===== REGISTRASI =====\n";
    string username, password;
    cout << "Masukkan Username: ";
    cin >> ws;
    getline(cin, username);
    cout << "Masukkan Password: ";
    cin >> password;

    // Menggunakan pointer untuk mengecek username
    bool usernameExists = false;
    User *current = users;
    for (int i = 0; i < *userCount; i++) {
        if (current->username == username) {
            usernameExists = true;
            break;
        }
        current++;
    }

    if (usernameExists) {
        cout << "Registrasi gagal! Akun telah tersedia.\n";
    } else {
        User *newUser = &users[*userCount];
```

```
        *newUser = {username, password, false};
        (*userCount)++;
        cout << "Registrasi berhasil!\n";
    }
}

bool login(User *users, int userCount, bool *isAdmin, int *currentUserId) {
    cout << "\n===== LOGIN =====\n";
    cout << "Ingin login sebagai apa?\n";
    cout << "1. Admin\n";
    cout << "2. User\n";
    cout << "Pilihan Anda: ";
    int loginType;
    cin >> loginType;

    if (loginType < 1 || loginType > 2) {
        cout << "Pilihan tidak valid!\n";
        return false;
    }

    string username, password;
    cout << "Masukkan Username: ";
    cin >> ws;
    getline(cin, username);
    cout << "Masukkan Password: ";
    cin >> password;

    // Menggunakan pointer untuk iterasi
    User *current = users;
    for (int i = 0; i < userCount; i++) {
        if (current->username == username && current->password == password) {
            if ((loginType == 1 && current->isAdmin) || (loginType == 2 && !current->isAdmin)) {
                *isAdmin = current->isAdmin;
                *currentUserId = i;
                return true;
            }
        }
        current++;
    }
    return false;
}

bool menuAdmin(FilmData *films, int *filmCount, Booking *bookings, int *bookingCount, const int *maksKursi) {
    cout << "\n===== MENU ADMIN =====\n";
    cout << "1. Tambah Film\n";
    cout << "2. Hapus Film\n";
    cout << "3. Lihat Film\n";
    cout << "4. Lihat Transaksi\n";
    cout << "5. Log out\n";
    cout << "Pilihan Anda: ";

```

```
    int pilihan;
    cin >> pilihan;

    switch (pilihan) {
        case 1: tambahFilm(films, filmCount); break;
        case 2: hapusFilm(films, filmCount); break;
        case 3: lihatFilm(films, *filmCount, maksKursi); break;
        case 4: lihatTransaksi(bookings, *bookingCount, films); break;
        case 5: return false;
        default: cout << "Pilihan tidak valid!\n"; break;
    }
    return true;
}

bool menuUser(FilmData *films, int filmCount, Booking *bookings, int *bookingCount, int *currentUserId, User *users, const int *maksKursi) {
    cout << "\n===== MENU USER =====\n";
    cout << "1. Pesan Tiket\n";
    cout << "2. Lihat Film yang Tersedia\n";
    cout << "3. Lihat Tiket yang Sudah Dipesan\n";
    cout << "4. Ganti Jam Tayang\n";
    cout << "5. Batalkan Tiket\n";
    cout << "6. Bayar Tiket\n";
    cout << "7. Log out\n";
    cout << "Pilihan Anda: ";

    int pilihan;
    cin >> pilihan;

    switch (pilihan) {
        case 1: pesanTiket(films, filmCount, bookings, bookingCount, currentUserId, maksKursi); break;
        case 2: lihatFilm(films, filmCount, maksKursi); break;
        case 3: lihatTiketUser(bookings, *bookingCount, films, *currentUserId); break;
        case 4: gantiJamTayang(films, bookings, *bookingCount, *currentUserId, maksKursi); break;
        case 5: batalkanTiket(films, bookings, bookingCount); break;
        case 6: bayarTiket(bookings, bookingCount, films); break;
        case 7: {
            cout << "Kembali ke menu utama\n";
            return false;
        }
        default: cout << "Pilihan tidak valid!\n"; break;
    }
    return true;
}

void tambahFilm(FilmData *films, int *filmCount) {
    if (*filmCount >= 10) {
        cout << "Maaf, kapasitas film sudah penuh!\n";
        return;
    }
```

```
    cout << "Masukkan Nama Film: ";
    cin >> ws;
    getline(cin, films[*filmCount].namaFilm);

    // Menggunakan pointer untuk mengakses jam tayang
    JamTayang *jamPtr = films[*filmCount].jamTayang;
    for (int i = 0; i < 5; i++) {
        jamPtr->jam = to_string(10 + i * 2) + ":00";
        jamPtr->kursiTerisi = 0;
        jamPtr++;
    }

    cout << "Film berhasil ditambahkan!\n";
    (*filmCount)++;
}

void hapusFilm(FilmData *films, int *filmCount) {
    cout << "\n===== HAPUS FILM =====\n";
    cout << "Daftar Film:\n";
    for (int i = 0; i < *filmCount; i++) {
        cout << i + 1 << ". " << films[i].namaFilm << endl;
    }

    int filmPilihan;
    cout << "Pilih film yang ingin ditarik (1-" << *filmCount << "): ";
    cin >> filmPilihan;

    if (filmPilihan < 1 || filmPilihan > *filmCount) {
        cout << "Pilihan tidak valid!\n";
        return;
    }

    // Menggunakan pointer untuk operasi penghapusan
    FilmData *target = films + (filmPilihan - 1);
    FilmData *end = films + (*filmCount - 1);

    while (target < end) {
        *target = *(target + 1);
        target++;
    }

    (*filmCount)--;
    cout << "Film berhasil ditarik dari penayangan!\n";
}

void lihatFilm(const FilmData *films, int filmCount, const int *maksKursi) {
    cout << "\n===== DAFTAR FILM =====\n";

    // Buat salinan untuk sorting
    FilmData sortedFilms[10];
    for (int i = 0; i < filmCount; i++) {
        sortedFilms[i] = films[i];
    }

    // Urutkan film descending
    selectionSortFilmDescending(sortedFilms, filmCount);

    // Tampilkan film yang sudah diurutkan
    for (int i = 0; i < filmCount; i++) {
        cout << i + 1 <
```

```
        sortedFilms[i] = films[i];
    }

    // Urutkan film descending
    selectionSortFilmDescending(sortedFilms, filmCount);

    // Tampilkan film yang sudah diurutkan
    for (int i = 0; i < filmCount; i++) {
        cout << i + 1 << ". " << sortedFilms[i].namaFilm << endl;
        cout << "   Jam Tayang: ";

        // Buat salinan jam tayang untuk sorting
        JamTayang sortedJam[5];
        for (int j = 0; j < 5; j++) {
            sortedJam[j] = sortedFilms[i].jamTayang[j];
        }

        // Urutkan jam ascending
        selectionSortJamAscending(sortedJam, 5);

        // Tampilkan jam yang sudah diurutkan
        for (int j = 0; j < 5; j++) {
            int kursiTersisa = *maksKursi - sortedJam[j].kursiTerisi;
            cout << sortedJam[j].jam << " (" << kursiTersisa << " kursi) ";
        }
        cout << endl << endl;
    }
}

void lihatTransaksi(const Booking *bookings, int bookingCount, const FilmData *films) {
    cout << "\n===== TRANSAKSI =====\n";
    for (int i = 0; i < bookingCount; i++) {
        cout << "ID Booking: " << i << endl;
        cout << "Film: " << films[bookings[i].filmId].namaFilm << endl;
        cout << "Jam: " << films[bookings[i].filmId].jamTayang[bookings[i].jamId].jam << endl;
        cout << "Status: " << (bookings[i].isPaid ? "Sudah dibayar" : "Belum dibayar") << endl;
        cout << "----------------------\n";
    }
}

void pesanTiket(FilmData *films, int filmCount, Booking *bookings, int *bookingCount, int *currentUserId, const int *maksKursi) {
    cout << "\n===== PESAN TIKET =====\n";

    // Tampilkan film yang tersedia
    cout << "Film yang tersedia:\n";
    for (int i = 0; i < filmCount; i++) {
        cout << i + 1 << ". " << films[i].namaFilm << endl;
    }

    int filmPilihan;
    cout << "Pilih film (1-" << filmCount << "): ";
```

```
    cin >> filmPilihan;

    if (filmPilihan < 1 || filmPilihan > filmCount) {
        cout << "Film tidak valid!\n";
        return;
    }

    // Menggunakan pointer untuk mengakses film yang dipilih
    FilmData *selectedFilm = films + (filmPilihan - 1);

    // Tampilkan jam tayang
    cout << "Jam tayang yang tersedia:\n";
    for (int i = 0; i < 5; i++) {
        int kursiTersisa = *maksKursi - selectedFilm->jamTayang[i].kursiTerisi;
        cout << i + 1 << ". " << selectedFilm->jamTayang[i].jam << " (Kursi tersisa: " << kursiTersisa << ")\n";
    }

    int jamPilihan;
    cout << "Pilih jam tayang (1-5): ";
    cin >> jamPilihan;

    if (jamPilihan < 1 || jamPilihan > 5) {
        cout << "Jam tayang tidak valid!\n";
        return;
    }

    // Menggunakan pointer untuk mengakses jam tayang yang dipilih
    JamTayang *selectedJam = selectedFilm->jamTayang + (jamPilihan - 1);

    // Cek apakah kursi masih tersedia
    if (selectedJam->kursiTerisi >= *maksKursi) {
        cout << "Maaf, kursi untuk film dan jam tersebut sudah penuh!\n";
        return;
    }

    // Menggunakan pointer untuk menyimpan booking baru
    Booking *newBooking = bookings + (*bookingCount);
    newBooking->filmId = filmPilihan - 1;
    newBooking->jamId = jamPilihan - 1;
    newBooking->isPaid = false;
    newBooking->userId = *currentUserId;

    // Update kursi terisi menggunakan pointer
    selectedJam->kursiTerisi++;

    cout << "Booking berhasil!\n";
    cout << "ID Booking Anda: " << *bookingCount << endl;
    cout << "Film: " << selectedFilm->namaFilm << endl;
    cout << "Jam: " << selectedJam->jam << endl;
    cout << "Status: Belum dibayar\n";

    (*bookingCount)++;
}

void lihatTiketUser(const Booking *book
```

```
    (*bookingCount)++;
}

void lihatTiketUser(const Booking *bookings, int bookingCount, const FilmData *films, int currentUserId) {
    cout << "\n===== TIKET YANG SUDAH DIPESAN =====\n";
    bool adaTiket = false;

    for (int i = 0; i < bookingCount; i++) {
        // Menggunakan dereference untuk mengakses data booking
        const Booking &currentBooking = *(bookings + i);
        if (currentBooking.filmId != -1 && currentBooking.userId == currentUserId) {
            adaTiket = true;
            cout << "ID Booking: " << i << endl;
            cout << "Film: " << films[currentBooking.filmId].namaFilm << endl;
            cout << "Jam: " << films[currentBooking.filmId].jamTayang[currentBooking.jamId].jam << endl;
            cout << "Status: " << (currentBooking.isPaid ? "Sudah dibayar" : "Belum dibayar") << endl;
            cout << "----------------------\n";
        }
    }

    if (!adaTiket) {
        cout << "Belum ada tiket yang dipesan!\n";
    }
}

void gantiJamTayang(FilmData *films, Booking *bookings, int bookingCount, int currentUserId, const int *maksKursi) {
    cout << "\n===== GANTI JAM TAYANG =====\n";

    int idBooking;
    cout << "Masukkan ID Booking: ";
    cin >> idBooking;

    if (idBooking < 0 || idBooking >= bookingCount || bookings[idBooking].filmId == -1) {
        cout << "ID Booking tidak valid!\n";
        return;
    }

    // Menggunakan pointer untuk mengakses booking
    Booking *selectedBooking = bookings + idBooking;

    if (selectedBooking->userId != currentUserId) {
        cout << "Anda tidak memiliki akses untuk mengubah booking ini!\n";
        return;
    }

    if (selectedBooking->isPaid) {
        cout << "Tidak dapat mengubah jam tayang tiket yang sudah dibayar!\n";
        return;
    }

    int filmId = selectedBooking->filmId;
```

```
    int jamLama = selectedBooking->jamId;

    // Menggunakan pointer untuk mengakses film
    FilmData *selectedFilm = films + filmId;

    // Tampilkan jam tayang yang tersedia
    cout << "Jam tayang saat ini: " << selectedFilm->jamTayang[jamLama].jam << endl;
    cout << "Jam tayang baru yang tersedia:\n";

    for (int i = 0; i < 5; i++) {
        int kursiTersisa = *maksKursi - selectedFilm->jamTayang[i].kursiTerisi;
        cout << i + 1 << ". " << selectedFilm->jamTayang[i].jam << " (Kursi tersisa: " << kursiTersisa << ")\n";
    }

    int jamBaru;
    cout << "Pilih jam tayang baru (1-5): ";
    cin >> jamBaru;

    if (jamBaru < 1 || jamBaru > 5) {
        cout << "Jam tayang tidak valid!\n";
        return;
    }

    // Menggunakan pointer untuk mengakses jam tayang baru
    JamTayang *newJam = selectedFilm->jamTayang + (jamBaru - 1);

    if (newJam->kursiTerisi >= *maksKursi) {
        cout << "Maaf, kursi untuk jam tersebut sudah penuh!\n";
        return;
    }

    // Menggunakan pointer untuk mengakses jam tayang lama
    JamTayang *oldJam = selectedFilm->jamTayang + jamLama;

    // Update kursi terisi menggunakan pointer
    oldJam->kursiTerisi--;
    newJam->kursiTerisi++;

    // Update booking menggunakan pointer
    selectedBooking->jamId = jamBaru - 1;

    cout << "Jam tayang berhasil diubah!\n";
    cout << "Film: " << selectedFilm->namaFilm << endl;
    cout << "Jam tayang baru: " << newJam->jam << endl;
}

void batalkanTiket(FilmData *films, Booking *bookings, int *bookingCount) {
    cout << "\n===== BATALKAN TIKET =====\n";

    int idBooking;
    cout << "Masukkan ID Booking: ";
```

```
    cin >> idBooking;

    if (idBooking < 0 || idBooking >= *bookingCount || bookings[idBooking].filmId == -1) {
        cout << "ID Booking tidak valid!\n";
        return;
    }

    // Menggunakan pointer untuk mengakses booking
    Booking *selectedBooking = bookings + idBooking;

    if (selectedBooking->isPaid) {
        cout << "Tidak dapat membatalkan tiket yang sudah dibayar!\n";
        return;
    }

    int filmId = selectedBooking->filmId;
    int jamId = selectedBooking->jamId;

    // Menggunakan pointer untuk mengakses film dan jam tayang
    FilmData *selectedFilm = films + filmId;
    JamTayang *selectedJam = selectedFilm->jamTayang + jamId;

    // Update kursi terisi menggunakan pointer
    selectedJam->kursiTerisi--;

    // Hapus booking menggunakan pointer
    selectedBooking->filmId = -1;
    selectedBooking->jamId = -1;
    selectedBooking->isPaid = false;
    selectedBooking->userId = -1;

    cout << "Tiket berhasil dibatalkan!\n";
}

void bayarTiket(Booking *bookings, int *bookingCount, const FilmData *films) {
    cout << "\n===== BAYAR TIKET =====\n";
    int idBooking;
    cout << "Masukkan ID Booking: ";
    cin >> idBooking;

    if (idBooking < 0 || idBooking >= *bookingCount || bookings[idBooking].filmId == -1) {
        cout << "ID Booking tidak valid!\n";
        return;
    }

    bookings[idBooking].isPaid = true;
    cout << "Pembayaran berhasil!\n";
    cout << "Film: " << films[bookings[idBooking].filmId].namaFilm << endl;
    cout << "Jam: " << films[bookings[idBooking].filmId].jamTayang[bookings[idBooking].jamId].jam << endl;
    cout << "Status: Sudah dibayar\n";
}
```

```
void tampilkanFilmTersedia(const FilmData *films, int filmCount, int index, const int *maksKursi) {
    if (index >= filmCount) return;

    // Menggunakan dereference untuk mengakses data film
    const FilmData &currentFilm = *(films + index);
    cout << index + 1 << ". " << currentFilm.namaFilm << endl;
    cout << "   Jam Tayang: ";
    tampilkanJamTayang(currentFilm.jamTayang, 0, maksKursi);
    cout << endl;

    tampilkanFilmTersedia(films, filmCount, index + 1, maksKursi);
}

void tampilkanJamTayang(const JamTayang *jamTayang, int index, const int *maksKursi) {
    if (index >= 5) return;

    // Menggunakan dereference untuk mengakses data jam tayang
    const JamTayang &currentJam = *(jamTayang + index);
    int kursiTersisa = *maksKursi - currentJam.kursiTerisi;
    cout << currentJam.jam << " (" << kursiTersisa << " kursi) ";

    tampilkanJamTayang(jamTayang, index + 1, maksKursi);
}

void tampilkanJamTayang(const JamTayang &jamTayang) {
    cout << jamTayang.jam;
}
```

4. Uji Coba dan Hasil Output

```
PS C:\Users\HazzelPW> cd "c

===== MENU UTAMA =====
1. Registrasi
2. Login
3. Keluar
Pilihan Anda:
```

Gambar 4.1 Output menu utama

```
===== MENU UTAMA =====
1. Registrasi
2. Login
3. Keluar
Pilihan Anda: 1

===== REGISTRASI =====
Masukkan Username: hajel
Masukkan Password: 23
Registrasi berhasil!
```

Gambar 4.2 Output Registrasi

```
===== MENU UTAMA =====
1. Registrasi
2. Login
3. Keluar
Pilihan Anda: 2

===== LOGIN =====
Ingin login sebagai apa?
1. Admin
2. User
Pilihan Anda: 1
Masukkan Username: admin
Masukkan Password: selalubenar
Login berhasil!
Selamat Datang Bos admin

===== MENU ADMIN =====
1. Tambah Film
2. Hapus Film
3. Lihat Film
4. Lihat Transaksi
5. Log out
Pilihan Anda:
```

Gambar 4.3 Output login sebagai admin dan menu admin

```
===== MENU ADMIN =====
1. Tambah Film
2. Hapus Film
3. Lihat Film
4. Lihat Transaksi
5. Log out
Pilihan Anda: 1

===== TAMBAH FILM =====
Masukkan Nama Film: God of War
Film berhasil ditambahkan!

===== MENU ADMIN =====
1. Tambah Film
2. Hapus Film
3. Lihat Film
4. Lihat Transaksi
5. Log out
Pilihan Anda: 2

===== HAPUS FILM =====
Daftar Film:
1. Kimi No Nawa
2. Godzilla : Minus one
3. Avengers : secret wars
4. Your Lie In April
5. Cek Toko Sebelah
6. God of War
Pilih film yang ingin ditarik (1-6): 6
Film berhasil ditarik dari penayangan!
```

Gambar 4.4 Output menambah dan menghapus film

```
===== MENU ADMIN =====
1. Tambah Film
2. Hapus Film
3. Lihat Film
4. Lihat Transaksi
5. Log out
Pilihan Anda: 3

===== DAFTAR FILM =====
1. Kimi No Nawa
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)
2. Godzilla : Minus one
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)
3. Avengers : secret wars
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)
4. Your Lie In April
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)
5. Cek Toko Sebelah
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)

===== MENU ADMIN =====
1. Tambah Film
2. Hapus Film
3. Lihat Film
4. Lihat Transaksi
5. Log out
Pilihan Anda: 4

===== TRANSAKSI =====

===== MENU ADMIN =====
1. Tambah Film
2. Hapus Film
3. Lihat Film
4. Lihat Transaksi
5. Log out
Pilihan Anda: 5
Kembali ke menu utama
```

Gambar 4.5 Output melihat film, melihat transaksi dan log out dari akun admin

```
===== MENU UTAMA =====
1. Registrasi
2. Login
3. Keluar
Pilihan Anda: 2

===== LOGIN =====
Ingin login sebagai apa?
1. Admin
2. User
Pilihan Anda: 2
Masukkan Username: Danendra Hazzel PW
Masukkan Password: 2409106096
Login berhasil!
Selamat Datang Tuan Danendra Hazzel PW

===== MENU USER =====
1. Pesan Tiket
2. Lihat Film yang Tersedia
3. Lihat Tiket yang Sudah Dipesan
4. Ganti Jam Tayang
5. Batalkan Tiket
6. Bayar Tiket
7. Log out
Pilihan Anda:
```

Gambar 4.6 Output login sebagai user dan menu utama user

```
===== MENU USER =====
1. Pesan Tiket
2. Lihat Film yang Tersedia
3. Lihat Tiket yang Sudah Dipesan
4. Ganti Jam Tayang
5. Batalkan Tiket
6. Bayar Tiket
7. Log out
Pilihan Anda: 1

===== PESAN TIKET =====
Film yang tersedia:
1. Kimi No Nawa
2. Godzilla : Minus one
3. Avengers : secret wars
4. Your Lie In April
5. Cek Toko Sebelah
Pilih film (1-5): 4
Jam tayang yang tersedia:
1. 10:00 (Kursi tersisa: 20)
2. 12:00 (Kursi tersisa: 20)
3. 15:00 (Kursi tersisa: 20)
4. 17:00 (Kursi tersisa: 20)
5. 19:00 (Kursi tersisa: 20)
Pilih jam tayang (1-5): 2
Booking berhasil!
ID Booking Anda: 0
Film: Your Lie In April
Jam: 12:00
Status: Belum dibayar
```

Gambar 4.7 Output memesan tiket

```
===== MENU USER =====
1. Pesan Tiket
2. Lihat Film yang Tersedia
3. Lihat Tiket yang Sudah Dipesan
4. Ganti Jam Tayang
5. Batalkan Tiket
6. Bayar Tiket
7. Log out
Pilihan Anda: 2

===== FILM YANG TERSEDIA =====
1. Kimi No Nawa
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)
2. Godzilla : Minus one
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)
3. Avengers : secret wars
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)
4. Your Lie In April
   Jam Tayang: 10:00 (20 kursi) 12:00 (19 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)
5. Cek Toko Sebelah
   Jam Tayang: 10:00 (20 kursi) 12:00 (20 kursi) 15:00 (20 kursi) 17:00 (20 kursi) 19:00 (20 kursi)

===== MENU USER =====
1. Pesan Tiket
2. Lihat Film yang Tersedia
3. Lihat Tiket yang Sudah Dipesan
4. Ganti Jam Tayang
5. Batalkan Tiket
6. Bayar Tiket
7. Log out
Pilihan Anda: 3

===== TIKET YANG SUDAH DIPESAN =====
ID Booking: 0
Film: Your Lie In April
Jam: 12:00
Status: Belum dibayar
----------------------
```

Gambar 4.8 Output melihat film yang tersedia dan melihat tiket yang telah dipesan

```
===== MENU USER =====
1. Pesan Tiket
2. Lihat Film yang Tersedia
3. Lihat Tiket yang Sudah Dipesan
4. Ganti Jam Tayang
5. Batalkan Tiket
6. Bayar Tiket
7. Log out
Pilihan Anda: 4

===== GANTI JAM TAYANG =====
Masukkan ID Booking: 0
Jam tayang saat ini: 12:00
Jam tayang baru yang tersedia:
1. 10:00 (Kursi tersisa: 20)
2. 12:00 (Kursi tersisa: 19)
3. 15:00 (Kursi tersisa: 20)
4. 17:00 (Kursi tersisa: 20)
5. 19:00 (Kursi tersisa: 20)
Pilih jam tayang baru (1-5): 1
Jam tayang berhasil diubah!
Film: Your Lie In April
Jam tayang baru: 10:00

===== MENU USER =====
1. Pesan Tiket
2. Lihat Film yang Tersedia
3. Lihat Tiket yang Sudah Dipesan
4. Ganti Jam Tayang
5. Batalkan Tiket
6. Bayar Tiket
7. Log out
Pilihan Anda: 5

===== BATALKAN TIKET =====
Masukkan ID Booking: 1
Tiket berhasil dibatalkan!
```

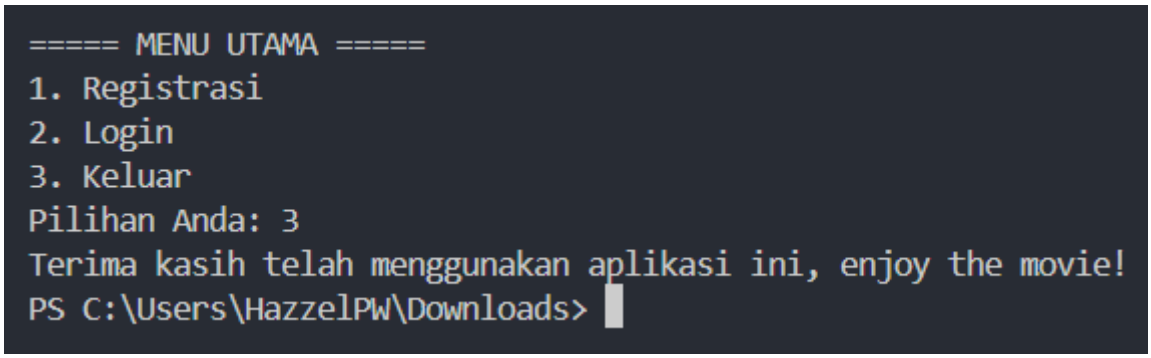
Gambar 4.9 Output ganti jam tayang dan membatalkan tiket

```
===== MENU USER =====
1. Pesan Tiket
2. Lihat Film yang Tersedia
3. Lihat Tiket yang Sudah Dipesan
4. Ganti Jam Tayang
5. Batalkan Tiket
6. Bayar Tiket
7. Log out
Pilihan Anda: 6

===== BAYAR TIKET =====
Masukkan ID Booking: 0
Pembayaran berhasil!
Film: Your Lie In April
Jam: 10:00
Status: Sudah dibayar

===== MENU USER =====
1. Pesan Tiket
2. Lihat Film yang Tersedia
3. Lihat Tiket yang Sudah Dipesan
4. Ganti Jam Tayang
5. Batalkan Tiket
6. Bayar Tiket
7. Log out
Pilihan Anda: 7
Kembali ke menu utama
```

Gambar 4.10 Bayar tiket dan log out dari akun user

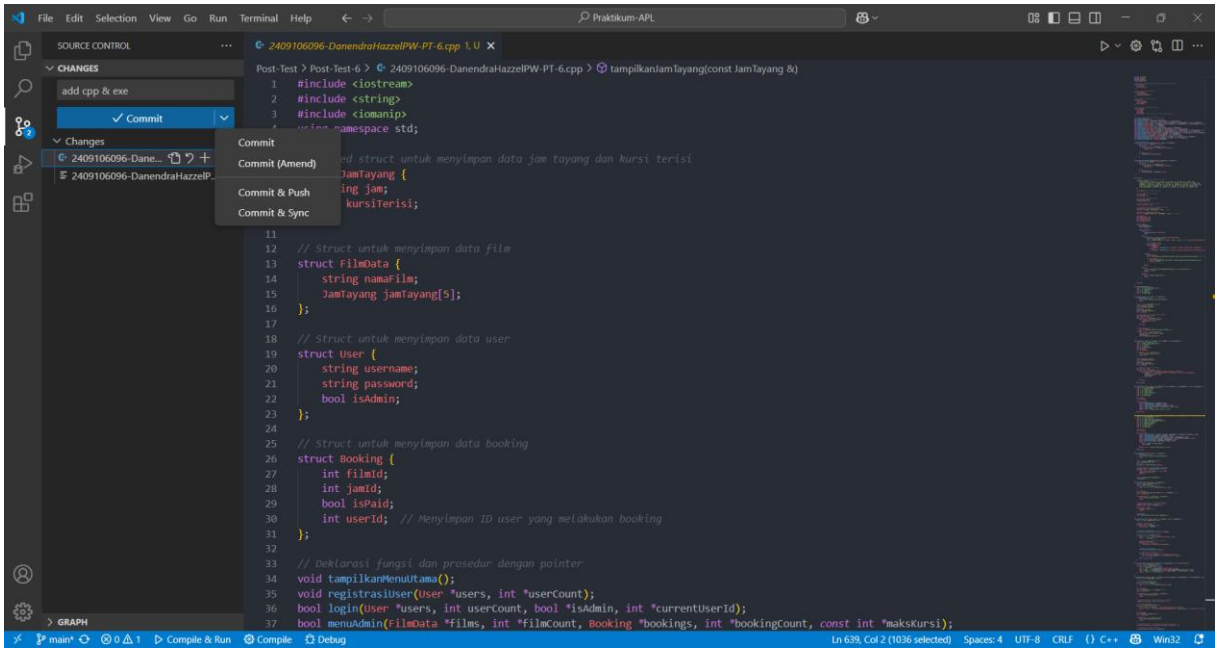
```
===== MENU ADMIN =====
1. Tambah Film
2. Hapus Film
3. Lihat Film
4. Lihat Transaksi
5. Log out
Pilihan Anda: 4

===== TRANSAKSI =====
ID Booking: 0
Film: Your Lie In April
Jam: 10:00
User: Danendra Hazzel PW
Status: Sudah dibayar
----------------------
ID Booking: 1
```

Gambar 4.11 Output informasi pada admin jika user melakukan transaksi



Gambar 4.12 Output keluar dari program

5. Langkah-Langkah Git pada VSCode



1. Pastikan VSCode terhubung dengan github
2. Buka folder yang ingin di commit ke github dengan VSCode
3. Masuk kedalam source control(ctrl+shift+g)
4. Ketik pesan pada folder yang ingin di push, sebagai contoh disini mengetik “Last Update cpp & exe”
5. Tekan “more action” dan pilih bagian “commit & push” yang berguna untuk menyimpan dan mendorong folder kita ke github.
6. Output update akan tampil pada layar dan juga pada “Graph” dibawah