LAPORAN PRAKTIKUM

POSTTEST 3

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Muhammad Rizky Adha Putra (2409106113)

Kelas (C2 ‘24)

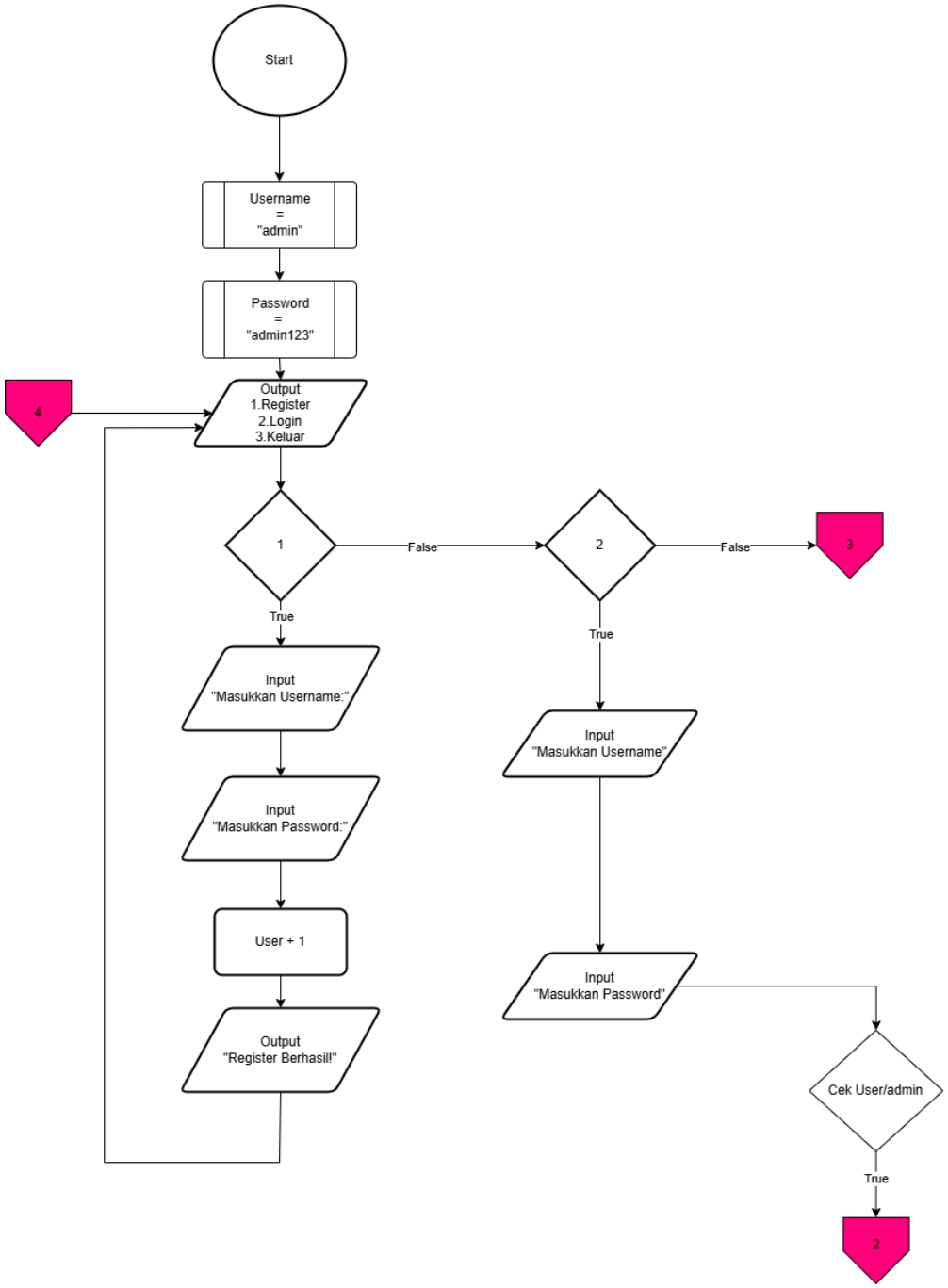
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

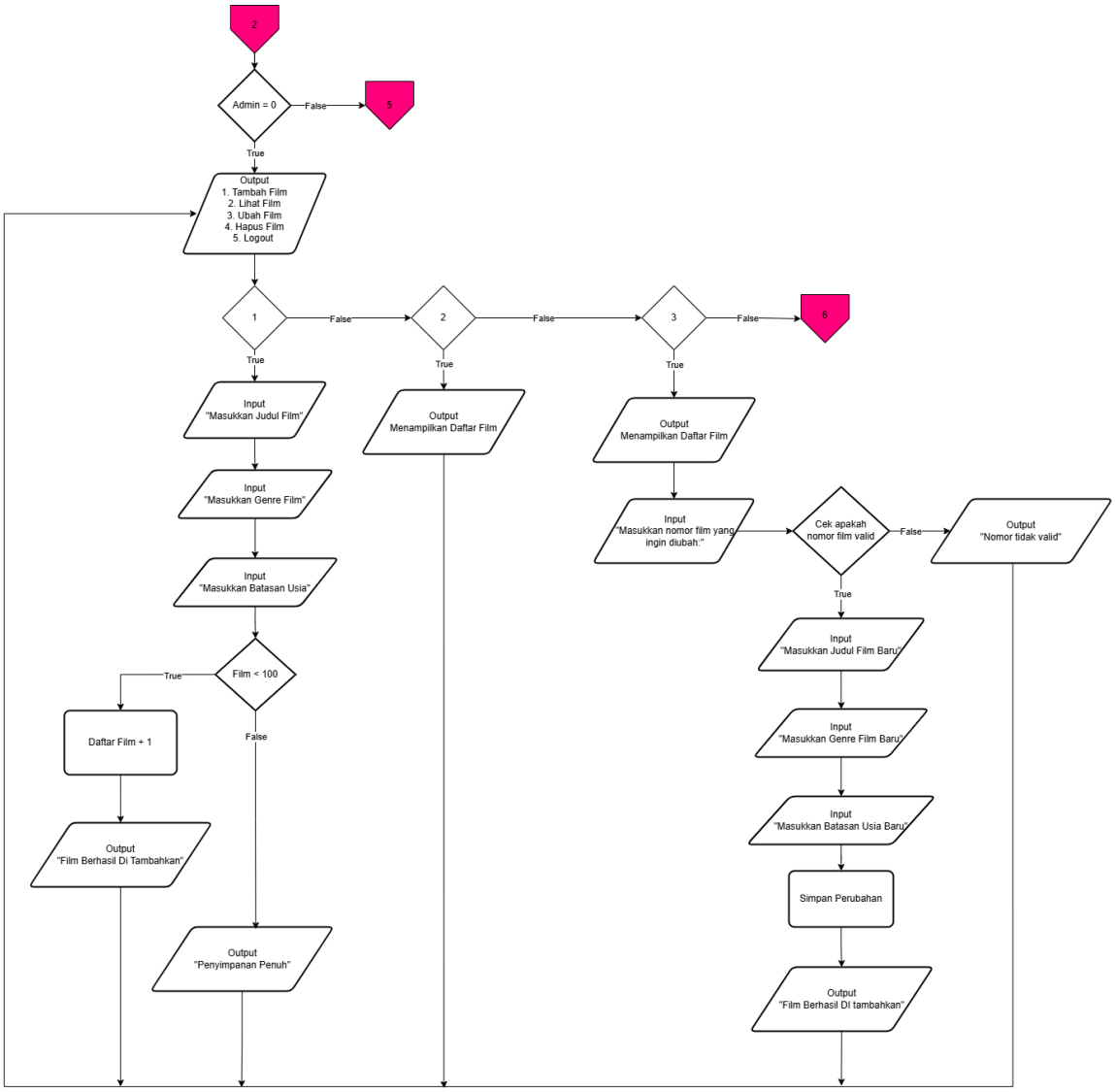
SAMARINDA

2025

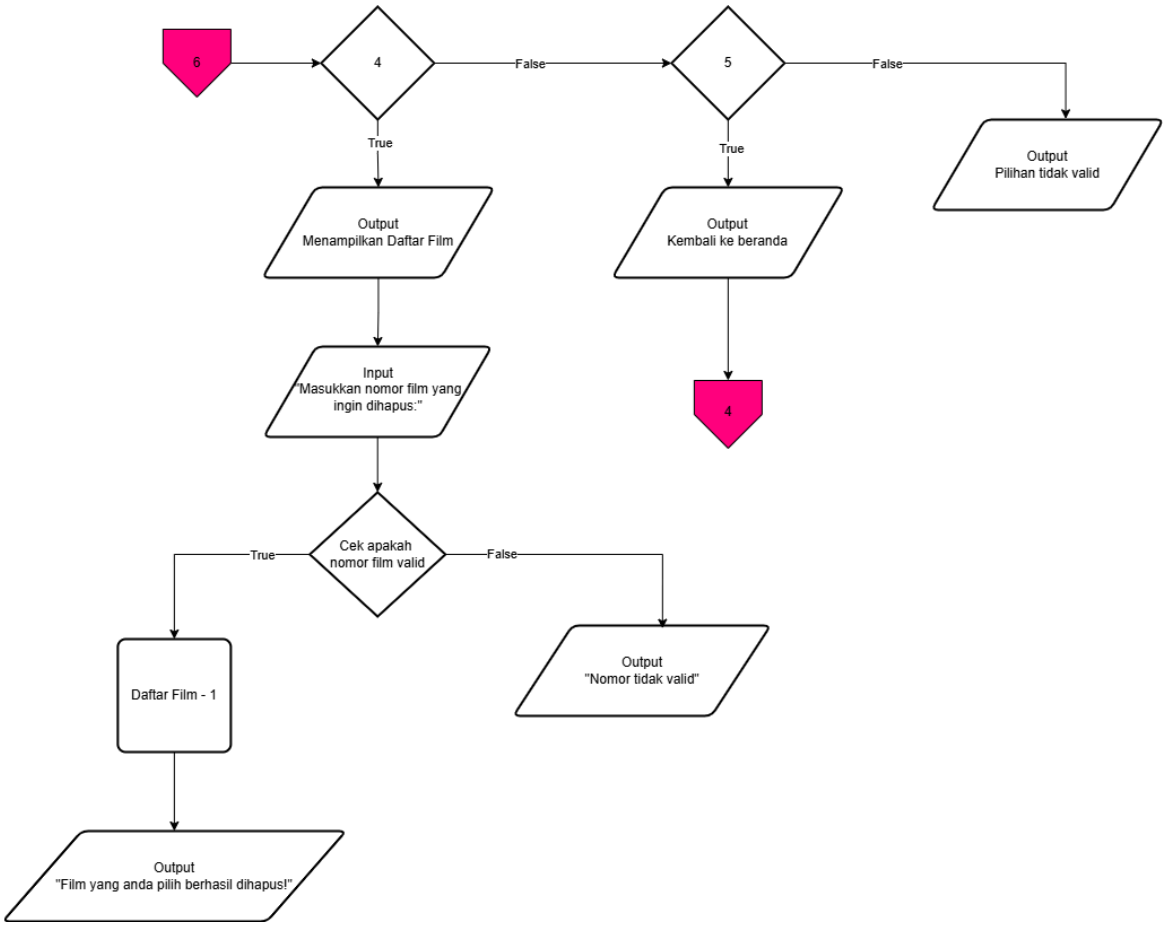
1. Flowchart



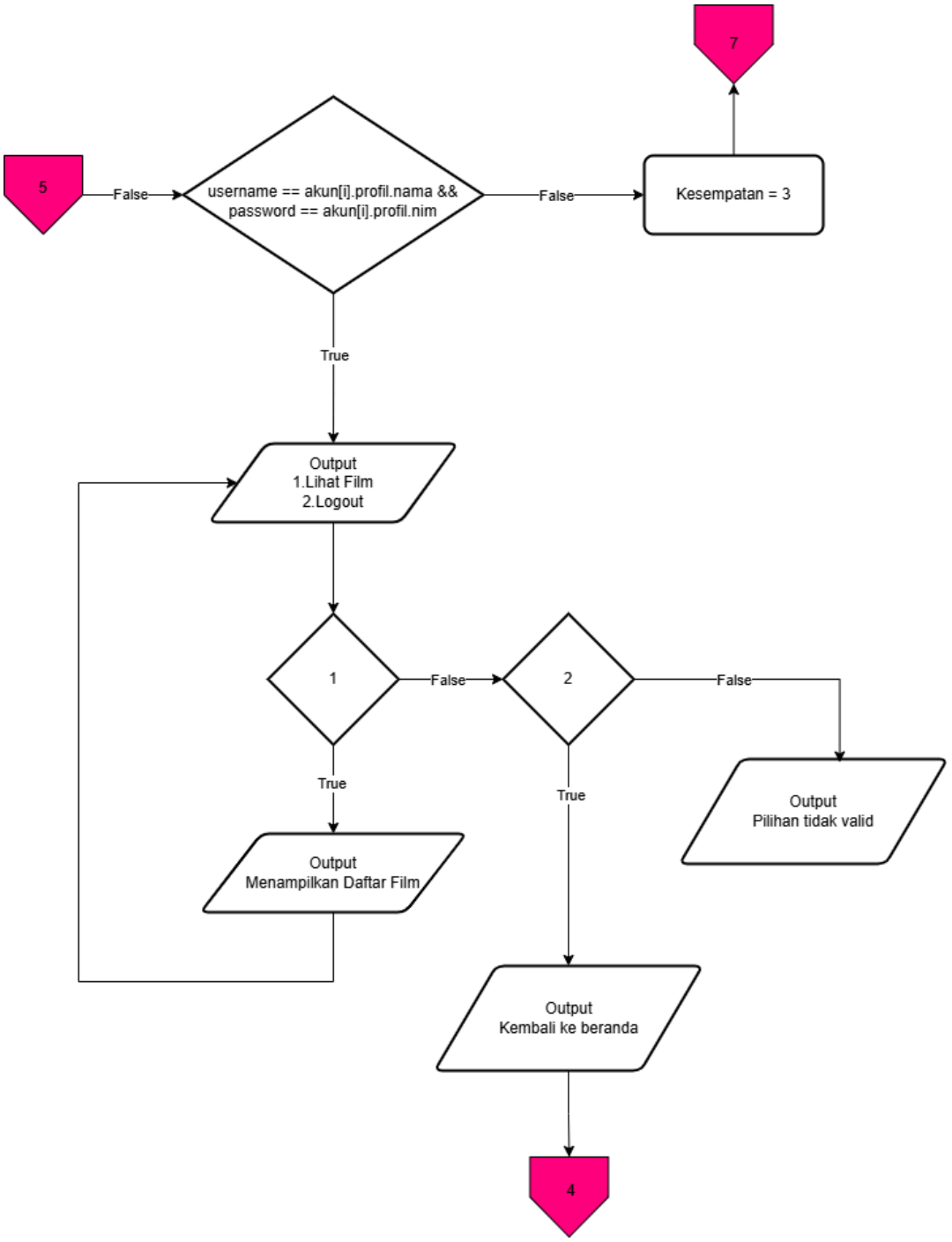
Gambar 1.1
Flowchart



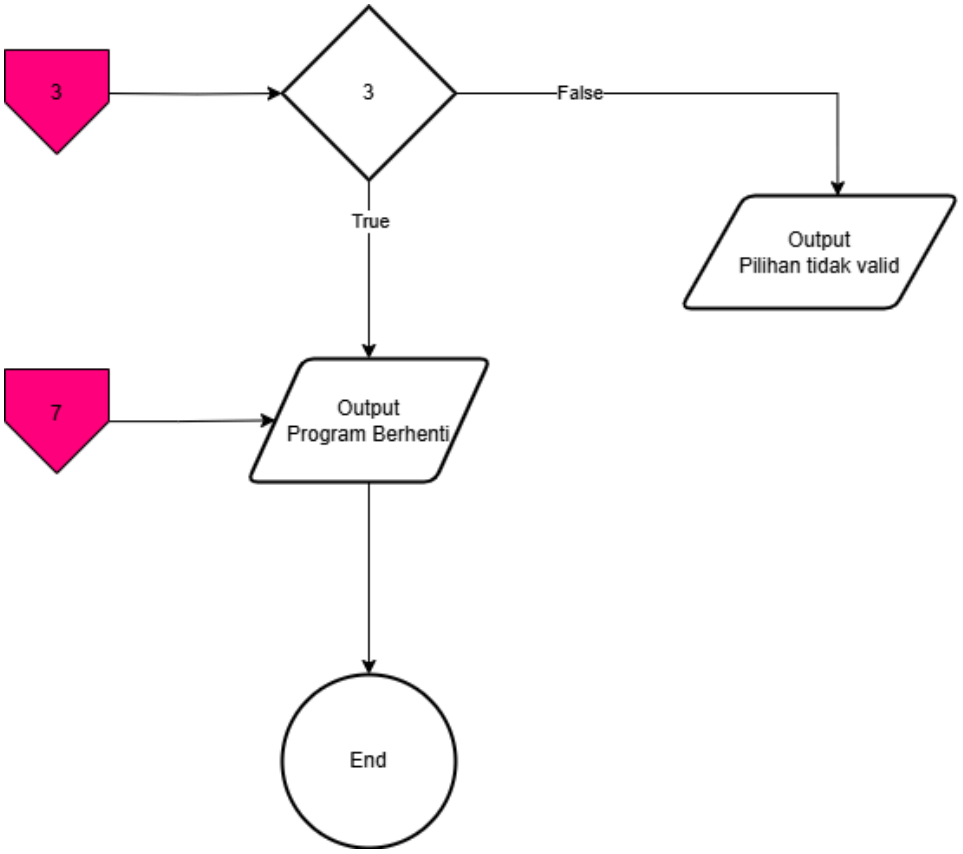
Gambar 1.2
Flowchart



Gambar 1.3
Flowchart



Gambar 1.4
Flowchart



Gambar 1.5
Flowchart

2. Analisis Program

Pada post test 3 ini saya membuat APLIX sebuah program berbasis C++ yang digunakan untuk mengelola daftar film. Program ini memungkinkan dua jenis pengguna, yaitu Admin dan User, untuk berinteraksi dengan data film. Admin memiliki kontrol penuh atau CRUD yang dimana biasanya untuk menambahkan, mengedit, dan menghapus film, sedangkan User ada fitur tambahan yaitu register dulu sebelum login dan ketika sudah selesai register maka baru bisa login lalu melihat daftar film yang tersedia di APLIX.

3. Source Code

A. Perulangan Program

Source Code:

```
while (true) {
    int role;
    cout << "+----------------------+" << endl;
    cout << "|        APLIX         |" << endl;
    cout << "+----------------------+" << endl;
    cout << "| 1. Register Akun     |" << endl;
    cout << "| 2. Login             |" << endl;
    cout << "| 3. Keluar            |" << endl;
    cout << "+----------------------+" << endl;
    cout << "Pilih: ";
    cin >> role;
    cin.ignore();

    if (role == 3) {
        cout << "Terima Kasih Sudah Berkunjung Di APLIX.\n";
        return 0;
    }
}
```

B. Fitur Register dan Login Multiuser

Source Code:

```
struct Profil {
    string nama;
    string nim;
};

struct User {
    Profil profil;
};

User akun[10] = { { {"admin", "admin123"} } };
int userC = 1;

if (role == 1) { // Register
    if (userC < 10) {
        cout << "\n== Registrasi Akun Baru ==" << endl;
        cout << "Masukkan Nama: ";
        getline(cin, akun[userC].profil.nama);
        cout << "Masukkan NIM: ";
        getline(cin, akun[userC].profil.nim);
        userC++;
        cout << "Registrasi berhasil!\n";
    } else {
        cout << "Maksimum user tercapai!\n";
    }
}
```

```
bool loginBerhasil = false;
int kesempatan = 3;
int loginIndex = -1;

while (kesempatan > 0) {
    string username, password;
    cout << "\nMasukkan Nama: ";
    getline(cin, username);
    cout << "Masukkan NIM: ";
    getline(cin, password);

    for (int i = 0; i < userC; i++) {
        if (username == akun[i].profil.nama && password ==
akun[i].profil.nim) {
            loginBerhasil = true;
            loginIndex = i;
            break;
        }
    }

    if (loginBerhasil) break;

    kesempatan--;
    cout << "User atau password salah! Sisa percobaan: " <<
kesempatan << endl;
}

if (!loginBerhasil) {
    cout << "Anda telah gagal login 3 kali. Program berhenti.\n";
    return 0;
}

cout << "\nLogin berhasil! Selamat datang, " <<
akun[loginIndex].profil.nama << "!\n";
```

C. Fitur Register dan Login Multiuser

Source Code:

```
struct Film {
    string judul;
    string genre;
    int usia;
};

Film daftarFilm[100] = {
    {"Up", "Comedy", 13},
    {"The Lion King", "Animasi", 0},
    {"Titanic", "Romance", 13},
    {"Avengers", "Aksi", 13},
    {"The Godfather", "Kriminal", 17}
};

int filmC = 5;
```

D. Operasi CRUD pada Data Film

Create

Source Code:

```
if (filmC < 100) {
    cout << "Masukkan Judul Film: ";
    getline(cin, daftarFilm[filmC].judul);
    cout << "Masukkan Genre Film: ";
    getline(cin, daftarFilm[filmC].genre);
    cout << "Masukkan Usia Minimal: ";
    cin >> daftarFilm[filmC].usia;
    cin.ignore();
    filmC++;
    cout << "Film berhasil ditambahkan!\n";
}
```

Read

Source Code:

```
for (int i = 0; i < filmC; ++i) {
    cout << i + 1 << ". " << daftarFilm[i].judul
         << " (" << daftarFilm[i].genre
         << ", " << daftarFilm[i].usia << "+)\n";
}
```

Update
Source Code:

```
int index;
cout << "Masukkan nomor film yang ingin diubah: ";
cin >> index;
cin.ignore();
if (index > 0 && index <= filmC) {
    cout << "Masukkan judul baru: ";
    getline(cin, daftarFilm[index - 1].judul);
    cout << "Masukkan genre baru: ";
    getline(cin, daftarFilm[index - 1].genre);
    cout << "Masukkan usia minimal baru: ";
    cin >> daftarFilm[index - 1].usia;
    cin.ignore();
    cout << "Film berhasil diubah!\n";
}
```

Delete
Source Code:

```
cout << "Masukkan nomor film yang ingin dihapus: ";
cin >> index;
cin.ignore();
if (index > 0 && index <= filmC) {
    for (int i = index - 1; i < filmC - 1; ++i) {
        daftarFilm[i] = daftarFilm[i + 1];
    }
    filmC--;
    cout << "Film berhasil dihapus!\n";
}
```

E. Tampilan Data yang Rapi

Source Code:

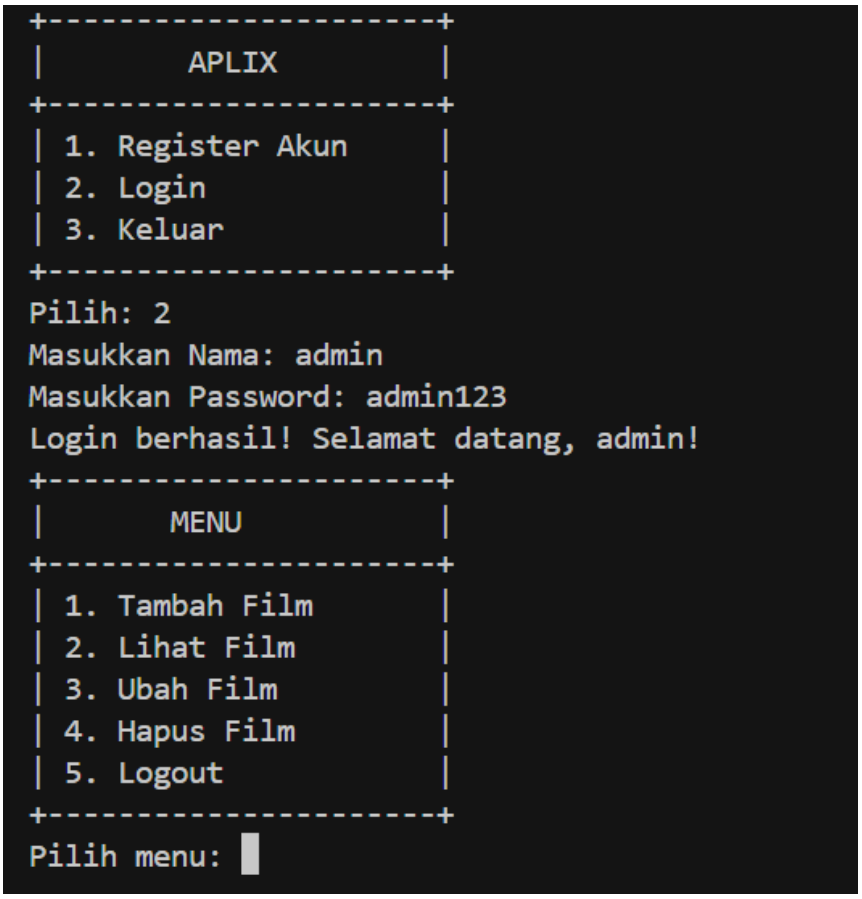
```
cout << "\nDaftar Film:\n";
cout << "+-------------------------------------+\n";
cout << "| No | Judul          | Genre  | Usia |\n";
cout << "+-------------------------------------+\n";
for (int i = 0; i < filmC; ++i) {
    cout << "| " << setw(2) << i + 1 << " | " << setw(14) << 
daftarFilm[i].judul
         << " | " << setw(6) << daftarFilm[i].genre
         << " | " << setw(4) << daftarFilm[i].usia << " |\n";
}
cout << "+-------------------------------------+\n";
```

4. Uji Coba dan Hasil Output

A. Login Gagal

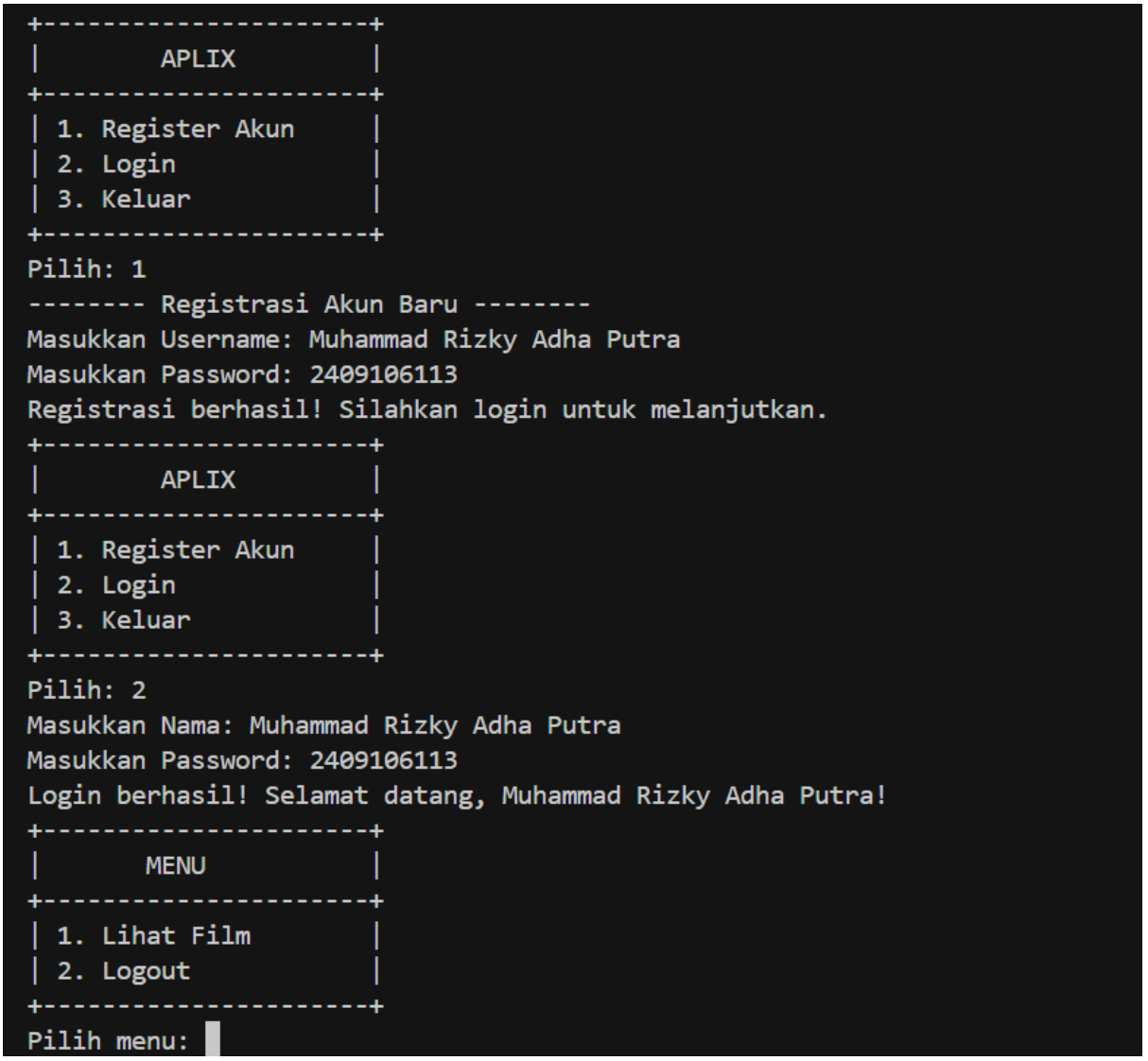
Gambar 4.1
Login Gagal

B. Login Berhasil Admin



Gambar 4.2
Login Berhasil Admin

C. Register dan Login Untuk User



Gambar 4.3
Register dan Login User

5. Langkah-Langkah Git pada VSCode

```
Asus@LAPTOP-F4V7N70I MINGW64 /c/GITHUB/Praktikum-Apl (main|REBASE)
$ git fetch origin

Asus@LAPTOP-F4V7N70I MINGW64 /c/GITHUB/Praktikum-Apl (main|REBASE)
$ git checkout main
M       Post-test/Post-test-3/2409106113-MuhammadRizkyAdhaPutra-PT-3.exe
Already on 'main'
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

Asus@LAPTOP-F4V7N70I MINGW64 /c/GITHUB/Praktikum-Apl (main|REBASE)
$ git reser --hard origin/main
git: 'reser' is not a git command. See 'git --help'.

The most similar commands are
        reset
        restore

Asus@LAPTOP-F4V7N70I MINGW64 /c/GITHUB/Praktikum-Apl (main|REBASE)
$ git push -f origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 713.26 KiB | 10.19 MiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/113Rizky/praktikum-apl.git
 + 0caddf6...134d8fd main -> main (forced update)
```

1. **Fetch Remote Update:**

Ambil pembaruan dari remote menggunakan git fetch origin untuk memastikan data terbaru dari origin sudah ada di lokal.

2. **Checkout ke Main:**

Pindah ke branch main dengan git checkout main agar perintah berikutnya dijalankan pada branch yang tepat.

3. **Reset Hard:**

Jalankan git reset --hard origin/main untuk membuat branch main lokal menjadi identik dengan branch origin/main, sehingga semua commit lokal yang berbeda akan dihapus.

4. **Force Push (Opsional):**

Jika perlu menyamakan branch remote dengan kondisi lokal yang baru, gunakan git push -f origin main untuk memaksa update pada remote.