D’MOV1E SEBAGAI APLIKASI MANAJEMEN TIKET BIOSKOP BERBASIS DESKTOP

Diajukan Sebagai Tugas Mata Kuliah

Struktur Data

Anggota Kelompok:

1. Putu Widyantara Artanta Wibawa (2108561005)
2. Putu Putri Pratiwi (2108561010)
3. Gede Krisnawa Sandhya Wandhana (2108561017)

PRODI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

BUKIT JIMBARAN

2022

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Bioskop merupakan tempat yang mempertunjukkan film untuk dinikmati oleh masyarakat publik. Walaupun perkembangan teknologi kian pesat dimana masyarakat saat ini dapat menikmati sebuah film melalui siaran televisi maupun laptop atau *handphone* secara online, keberadaan bioskop masih banyak diminati. Bahkan penjualan tiket bioskop kerap meningkat pesat ketika dirilisnya suatu film baru atau ketika musim liburan tiba.

Untuk menikmati film dalam bioskop, pelanggan diharuskan membeli tiket di loket bioskop yang akan dituju. Dalam pembelian tiket, pelanggan diharuskan mengantri. Selanjutnya pelanggan dapat memilih film, jadwal tayang, dan tempat duduk yang tersedia, pelanggan melakukan pembayaran langsung pada loket bioskop. Namun proses pembelian pada bioskop ini sering menimbulkan permasalahan seperti antrian panjang pada loket sehingga mengakibatkan pembatalan tiket karena jam tayang yang terlewat. Selain itu karena antrian yang panjang, tidak sedikit pula pelanggan yang kehabisan tiket sehingga tidak bisa menikmati film. Pandemi Covid-19 juga turut andil dalam permasalahan pada bioskop. Pihak bioskop harus membatasi jumlah pengunjung untuk meminimalisir penularan virus Covid-19. Hal tersebut menyebabkan pelanggan menjadi kesulitan dalam membeli tiket di bioskop.

Dari permasalahan tersebut, diperlukan solusi berupa pembelian tiket bioskop via *online* melalui suatu aplikasi agar pelanggan dapat membeli tiket bioskop dengan nyaman tanpa adanya antrian panjang pada loket bioskop, ketinggalan jam tayang, dan meminimalisir kerumunan pada bioskop di tengah pandemic Covid-19. Oleh karena itu, diperlukan sebuah aplikasi yang mampu untuk melakukan manajemen pembelian tiket di bioskop, sehingga proses pembelian tiket, pemilihan kursi, dan pembayaran dapat berlangsung dengan mudah.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, dapat ditarik rumusan masalah sebagai berikut:

1. Bagaimana aplikasi manajemen pembelian tiket bioskop tersebut dibuat?
2. Bagaimana tampilan dari aplikasi manajemen pembelian tiket bioskop tersebut?

1.3 Tujuan

Adapun tujuan dibuatnya aplikasi ini selain memenuhi tugas akhir mata kuliah Struktur Data adalah untuk menciptakan suatu solusi dari permasalahan pembelian tiket bioskop secara langsung.

1.4 Manfaat

Pembuatan aplikasi ini akan memberikan manfaat kepada pelanggan maupun pihak bioskop karena proses pemesanan tiket ini bersifat praktis dan tidak menimbulkan antrian panjang pada bioskop**.**

BAB II

ANALISIS DAN RANCANGAN

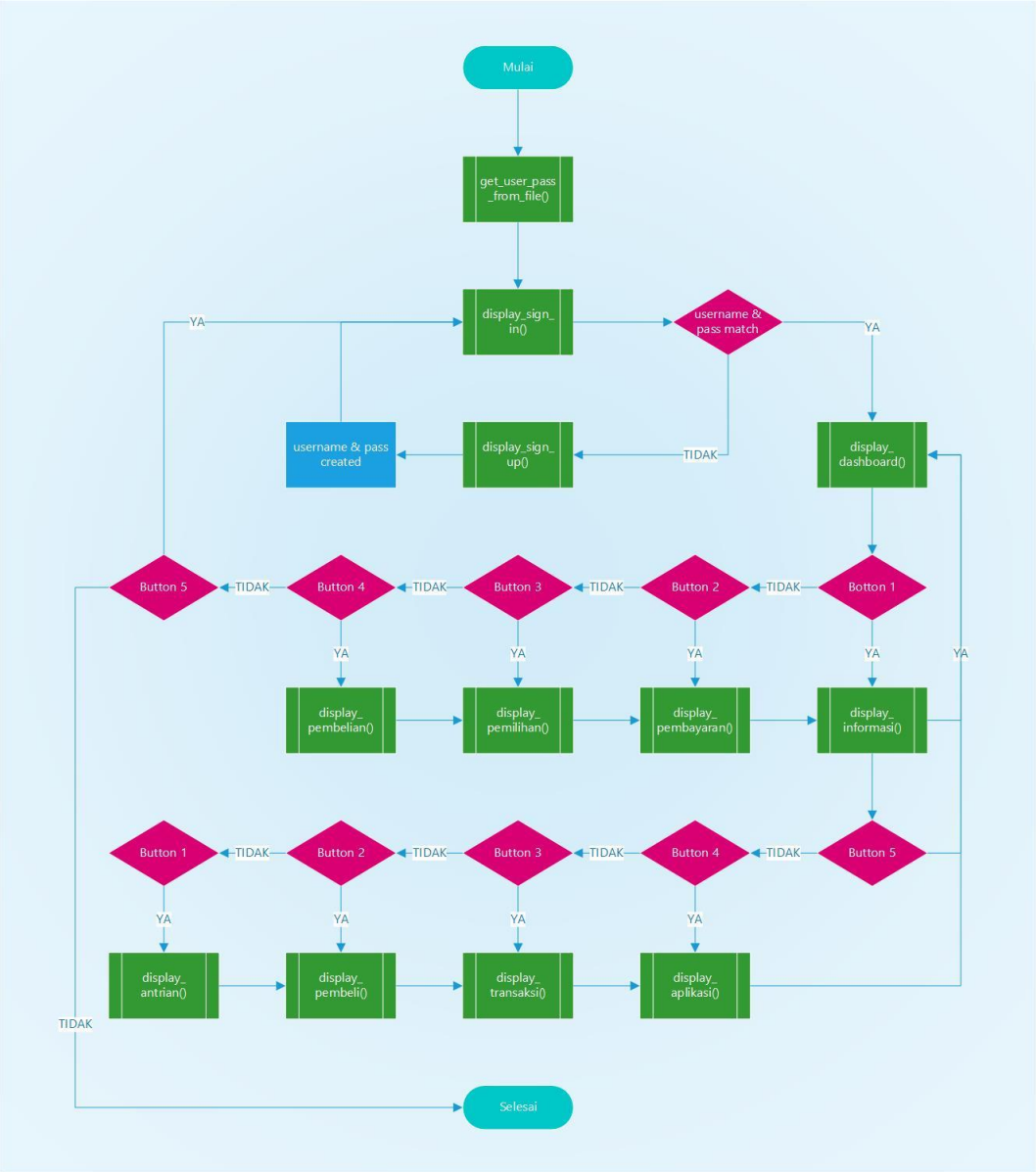
2.1 Algoritma

Berdasarkan permasalahan yang ada, dapat dianalisis dan dibuatkan sebuah alur algoritma pemecahan masalah.

1. Masukkan username dan password
2. Jika username dan password sesuai maka masuk ke dashboard
   2.1 Jika tidak buat username dan password baru
3. Masuk ke menu pembelian
4. Masukkan nama pembeli, jumlah tiket, nama film, dan waktu film
5. Jika seluruh informasi telah sesuai masuk ke menu pemilihan
   5.1 Jika tidak tampilkan pesan error
6. Pilih kursi sesuai dengan jumlah tiket yang dipesan
7. Jika seluruh pilihan kursi telah sesuai masuk ke menu pembayaran
   7.1 Jika tidak tampilkan pesan error
8. Masukkan jumlah uang yang dibayar
9. Jika jumlah uang yang dibayar telah sesuai maka konfirmasi pembayaran
   9.1 Jika tidak tampilkan pesan error
10. Masuk ke menu informasi
11. Masuk ke menu informasi antrian untuk melihat antrian pemilihan kursi dan antrian pembayaran
12. Masuk ke menu informasi pembeli untuk melihat pembeli terbaik
13. Masuk ke menu informasi transaksi untuk melihat transaksi terakhir
14. Masuk ke menu informasi aplikasi untuk melihat informasi tentang aplikasi

2.2 Rancangan

Berdasarkan analisis pemecahan masalah dengan alur algoritma yang sudah ada, dapat dibuat sebuah *flowchart* atau diagram alir yang menggambarkan rancangan dari program.



BAB III

IMPLEMENTASI DAN HASIL CAPTURE

3.1 Implementasi

Berikut adalah implementasi dari flowchart atau diagram alir program manajemen tiket bioskop ke dalam source code bahasa pemrograman C. Source code langkap dapat diakses pada: https://github.com/putuwaw/d-mov1e

main.c

```
#include "lib/d_mov1e.h"

int main(){
    get_user_pass_from_file();
    display_sign_in();
    return 0;
}
```

d_mov1e.c

```
#include "d_mov1e.h"

// LINKED LIST
UserInfo *User, *Head;

// QUEUE
Queue *qUser, *qFront, *qFrontPay;;

// STACK
Stack *sChair, *sTop, *sTopChair;

// BST
BST *bRoot, *bNode;

char FilmChair[16][3] ={
    "A1","A2","A3","A4",
    "B1","B2","B3","B4",
    "C1","C2", "C3", "C4",
    "D1", "D2", "D3", "D4"
};

char FilmName[4][50] = {
    "KKN Desa Penari",
    "The Throne",
    "Ready Player One",
    "Stand by Me Doraemon 2"
};

char FilmTime[4][3][20] = {
    {
        "08.00-10.00",
        "10.00-12.00",
        "13.30-15.30"
    },
    {
        "08.30-10.00",
        "11.30-13.00",
        "13.00-14.30"
    },
    {
        "10.30-11.30",
```

```
        "11.30-12.30",
        "12.30-13.30"
    },
    {
        "09.00-10.30",
        "10.00-11.30",
        "12.00-13.30"
    }
};

char StudioTime[3][4][20] = {
    {
        "08.00-10.00",
        "10.00-12.00",
        "12.00-13.30",
        "13.30-15.30"
    },
    {
        "08.30-10.00",
        "10.00-11.30",
        "11.30-13.00",
        "13.00-14.30"
    },
    {
        "09.00-10.30",
        "10.30-11.30",
        "11.30-12.30",
        "12.30-13.30"
    }
};

char StudioChair[3][4][16];

int TicketPrice[4] = {80000, 60000, 65000, 70000};

int AvailableChair[3][4];

int MoneyCurrency[7] = {100000, 50000, 20000, 10000, 5000, 2000, 1000};

bool admitLogin;

void reset_string(char *str, int len){
    int i;
    for (i = 0; i < len; i++){
        str[i] = '\0';
    }
}

void display_info_dialog(const gchar *info, GtkWidget *window){
    GtkWidget *dialog;
    dialog = gtk_message_dialog_new(GTK_WINDOW(window),
            GTK_DIALOG_DESTROY_WITH_PARENT,
            GTK_MESSAGE_INFO,
            GTK_BUTTONS_OK,
            info);
    gtk_widget_set_name(dialog, "infoDialog");
    gtk_window_set_title(GTK_WINDOW(dialog), "Information");
    gtk_dialog_run(GTK_DIALOG(dialog));
    gtk_widget_destroy(dialog);
}

void display_error_dialog(const gchar *error, GtkWidget *window){
    GtkWidget *dialog;
    dialog = gtk_message_dialog_new(GTK_WINDOW(window),
            GTK_DIALOG_DESTROY_WITH_PARENT,
            GTK_MESSAGE_ERROR,
            GTK_BUTTONS_OK,
            error);
    gtk_widget_set_name(dialog, "errorDialog");
    gtk_window_set_title(GTK_WINDOW(dialog), "Error");
```

```
    gtk_dialog_run(GTK_DIALOG(dialog));
    gtk_widget_destroy(dialog);
}

void display_warn_dialog(const gchar *warn, GtkWidget *window){
    GtkWidget *dialog;
    dialog = gtk_message_dialog_new(GTK_WINDOW(window),
            GTK_DIALOG_DESTROY_WITH_PARENT,
            GTK_MESSAGE_ERROR,
            GTK_BUTTONS_YES_NO,
            warn);
    gtk_widget_set_name(dialog, "warnDialog");
    gtk_window_set_title(GTK_WINDOW(dialog), "Warning");
    gtk_dialog_run(GTK_DIALOG(dialog));
    gtk_widget_destroy(dialog);
}

void get_user_pass_from_file(){
    char compUsr[100], compPwd[100];
    reset_string(compUsr, 100);
    reset_string(compPwd, 100);

    Head = NULL;
    FILE *f;
    f = fopen("data/user_pass_data.txt", "r");
    char ch;
    int idx = 1, counter = 0;
    while(!feof(f)){
        ch = fgetc(f);
        if (ch == '|'){
            counter = 0;
            idx = 2;
        }
        else if (ch == '\n'){
            User = malloc(sizeof(UserInfo));
            strcpy(User->Username, compUsr);
            strcpy(User->Password, compPwd);
            User->Next = NULL;
            if (Head == NULL){
                Head = User;
            }
            else{
                UserInfo *prev;
                prev = Head;
                while(prev->Next != NULL){
                    prev = prev->Next;
                }
                prev->Next = User;
            }
            // RESET
            reset_string(compUsr, 100);
            reset_string(compPwd, 100);
            counter = 0;
            idx = 1;
        }
        else{
            if (idx == 1){
                compUsr[counter] = ch;
                counter++;
            }
            else if(idx == 2){
                compPwd[counter] = ch;
                counter++;
            }
        }
    }
    fclose(f);
}

char* do_password_hash(char *password){
```

```
    static char result[100];
    reset_string(result, 100);

    SHA256_CTX context;
    unsigned char md[SHA256_DIGEST_LENGTH];
    SHA256_Init(&context);
    SHA256_Update(&context, (unsigned char *)password, sizeof(password));
    SHA256_Final(md, &context);

    int i = 0;
    while(md[i] != '\0'){
        int temp = md[i];
        int modTemp = temp % 16;
        if (modTemp < 10){
            result[i] = modTemp + '0';
        }
        else{
            modTemp -=  10;
            result[i] = modTemp + 'a';
        }
        i++;
    }
    return result;
}

char *timeToStr(struct tm *timeInfo){
    static const char weekDay[][10] = {
        "Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"};
    static const char monthDay[][15] = {
        "Januari", "Februari", "Maret", "April", "Mei", "Juni",
        "Juli", "Agustus", "September", "Oktober", "November", "Desember"};
    static char timeResult[100];
    reset_string(timeResult, 100);
    sprintf(timeResult, "%.2d:%.2d:%.2d - %s, %d %s %d",
            timeInfo->tm_hour,
            timeInfo->tm_min,
            timeInfo->tm_sec,
            weekDay[timeInfo->tm_wday],
            timeInfo->tm_mday,
            monthDay[timeInfo->tm_mon],
            1900 + timeInfo->tm_year);
    return timeResult;
}
```

sign_in.c

```
#include "sign_in.h"

// WINDOW
GtkWidget *windowSignIn;

// INPUT
GtkWidget *entryUsernameSignIn;
GtkWidget *entryPasswordSignIn;

void check_login(){
    char pass[100], username[100], passHashing[100];
    reset_string(pass, 100);
    reset_string(username, 100);
    reset_string(passHashing, 100);

    const gchar *gtextUsername =
gtk_entry_get_text(GTK_ENTRY(entryUsernameSignIn));
    const gchar *gtextPass = gtk_entry_get_text(GTK_ENTRY(entryPasswordSignIn));
    sprintf(username, "%s", gtextUsername);
    sprintf(pass, "%s", gtextPass);

    strcpy(passHashing, pass);
    strcpy(passHashing, do_password_hash(passHashing));

    // CHECK
```

```
    admitLogin = false;
    UserInfo *temp = Head;
    while(temp != NULL){
        if (strcmp(username, temp->Username) == 0 && strcmp(passHashing, temp-
>Password) == 0){
            admitLogin = true;
            break;
        }
        temp = temp->Next;
    }
    if (admitLogin){
        display_info_dialog("Login Berhasil!", windowSignIn);
        display_dashboard(windowSignIn);
    }
    else{
        display_error_dialog("Username atau Password Salah!", windowSignIn);
    }
}

void display_sign_in(){
    // WIDGET
    GtkWidget *layoutSignIn;
    GtkWidget *bgSignIn;

    // ICON
    GdkPixbuf *iconSignIn;

    // BUTTON
    GtkWidget *buttonSignIn;
    GtkWidget *buttonSignUp;

    // LABEL
    GtkWidget *labelTitle;
    GtkWidget *labelContent;
    GtkWidget *labelUsername;
    GtkWidget *labelPassword;
    GtkWidget *labelNewAccount;

    gtk_init(NULL, NULL);

    // WINDOW
    windowSignIn = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowSignIn), "D'MOV1E");
    gtk_window_set_default_size(GTK_WINDOW(windowSignIn), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowSignIn), GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutSignIn = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowSignIn), layoutSignIn);

    // ICON
    iconSignIn = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowSignIn), iconSignIn);

    // BACKGROUND
    bgSignIn = gtk_image_new_from_file("src/image/login.png");
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), bgSignIn, 0, 0);

    // LABEL
    labelTitle = gtk_label_new("Welcome to D'MOV1E");
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), labelTitle, 900, 100);
    gtk_widget_set_name(labelTitle, "labelTitle");

    labelContent = gtk_label_new("Sign In");
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), labelContent, 900, 150);
    gtk_widget_set_name(labelContent, "labelContent");

    labelNewAccount = gtk_label_new("Belum punya akun?");
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), labelNewAccount, 900, 500);
```

```
    labelUsername = gtk_label_new("Username");
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), labelUsername, 900, 270);

    labelPassword = gtk_label_new("Password");
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), labelPassword, 900, 355);

    // ENTRY
    entryUsernameSignIn = gtk_entry_new();
    gtk_widget_set_size_request(entryUsernameSignIn, 250, 40);
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), entryUsernameSignIn, 900, 300);

    entryPasswordSignIn = gtk_entry_new();
    gtk_widget_set_size_request(entryPasswordSignIn, 250, 40);
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), entryPasswordSignIn, 900, 390);
    gtk_entry_set_visibility(GTK_ENTRY(entryPasswordSignIn), false); // password
hidden
    gtk_widget_set_name(entryPasswordSignIn, "entryPassSignIn");

    // BUTTON
    buttonSignIn = gtk_button_new_with_label("Login");
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), buttonSignIn, 900, 450);

    buttonSignUp = gtk_button_new_with_label("Buat Akun");
    gtk_layout_put(GTK_LAYOUT(layoutSignIn), buttonSignUp, 1070, 495);

    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css, "src/css/sign_in.css", NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowSignIn);

    // SIGNAL
    g_signal_connect(windowSignIn, "destroy", G_CALLBACK(gtk_main_quit), NULL);
    g_signal_connect(buttonSignUp, "clicked",
G_CALLBACK(handle_display_sign_up), windowSignIn);
    g_signal_connect(buttonSignIn, "clicked", G_CALLBACK(check_login), NULL);

    gtk_main();
}
void handle_display_sign_in(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_sign_in();
}
```

sign_up.c

```
#include "sign_up.h"

// WIDGET
GtkWidget *windowSignUp;

// INPUT
GtkWidget *entryUsernameSignUp;
GtkWidget *entryPasswordSignUp;
GtkWidget *entryRepeatPass;

bool search_exist_acc(char *user, char *hash){
    UserInfo *temp;
    temp = Head;
    while(temp != NULL){
        if(strcmp(temp->Username, user) == 0 && strcmp(temp->Password, hash) ==
0){
            return true;
        }
```

```
        temp = temp->Next;
    }
    return false;
}

void create_new_acc(){
    char pass[100], passRepeat[100], passHashing[100], username[100];
    // RESET
    reset_string(pass, 100);
    reset_string(passRepeat, 100);
    reset_string(username, 100);
    reset_string(passHashing, 100);

    const gchar *gtextPass = gtk_entry_get_text(GTK_ENTRY(entryPasswordSignUp));
    const gchar *gtextPassRepeat =
gtk_entry_get_text(GTK_ENTRY(entryRepeatPass));
    const gchar *gtextUsername =
gtk_entry_get_text(GTK_ENTRY(entryUsernameSignUp));

    sprintf(pass, "%s", gtextPass);
    sprintf(passRepeat, "%s", gtextPassRepeat);
    sprintf(username, "%s", gtextUsername);

    strcpy(passHashing, pass);
    strcpy(passHashing, do_password_hash(passHashing));

    bool isFound;
    isFound = search_exist_acc(username, passHashing);

    if (strcmp(pass, "") == 0 || strcmp(username, "") == 0 ){
        display_error_dialog("Username dan Password Harus Diisi!",
windowSignUp);
    }
    else if(strcmp(pass, passRepeat) != 0){
        display_error_dialog("Password Tidak Sama!", windowSignUp);
    }
    else if(isFound){
        display_error_dialog("Akun Sudah Ada!", windowSignUp);
    }
    else{
        // FILE OPERATION
        FILE *f;
        f = fopen("data/user_pass_data.txt", "a");
        fprintf(f,"%s|%s\n", username, passHashing);
        fclose(f);

        User = malloc(sizeof(UserInfo));
        strcpy(User->Username, username);
        strcpy(User->Password, passHashing);
        User->Next = NULL;
        if (Head == NULL){
            Head = User;
        }
        else{
            UserInfo *prev;
            prev = Head;
            while(prev->Next != NULL){
                prev = prev->Next;
            }
            prev->Next = User;
        }
        // RESET INPUT
        gtk_entry_set_text(GTK_ENTRY(entryUsernameSignUp), "");
        gtk_entry_set_text(GTK_ENTRY(entryPasswordSignUp), "");
        gtk_entry_set_text(GTK_ENTRY(entryRepeatPass), "");
        // DISPLAY SUCCESS
        display_info_dialog("Akun Berhasi Dibuat!", windowSignUp);
    }
    // RESET
    reset_string(pass, 100);
```

```
    reset_string(passRepeat, 100);
    reset_string(username, 100);
    reset_string(passHashing, 100);
}

void display_sign_up(){
    // WIDGET
    GtkWidget *layoutSignUp;
    GtkWidget *bgSignUp;

    // LABEL
    GtkWidget *labelUsername;
    GtkWidget *labelPassword;
    GtkWidget *labelRepeatPass;
    GtkWidget *labelAlreadyAcc;

    // BUTTON
    GtkWidget *buttonCreateNewAcc;
    GtkWidget *buttonBackToLogin;

    // ICON
    GdkPixbuf *iconSignUp;
    gtk_init(NULL, NULL);

    // WINDOW
    windowSignUp = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowSignUp), "SIGN UP");
    gtk_window_set_default_size(GTK_WINDOW(windowSignUp), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowSignUp), GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutSignUp = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowSignUp), layoutSignUp);

    // ICON
    iconSignUp = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowSignUp), iconSignUp);

    // BACKGROUND
    bgSignUp = gtk_image_new_from_file("src/image/new_account.png");
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), bgSignUp, 0, 0);

    // LABEL
    labelUsername = gtk_label_new("Username");
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), labelUsername, 200, 172);

    labelPassword = gtk_label_new("Password");
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), labelPassword, 200, 258);

    labelRepeatPass = gtk_label_new("Ulangi Password");
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), labelRepeatPass, 200, 344);

    // ENTRY
    entryUsernameSignUp = gtk_entry_new();
    gtk_widget_set_size_request(entryUsernameSignUp, 250, 40);
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), entryUsernameSignUp, 200, 210);

    entryPasswordSignUp = gtk_entry_new();
    gtk_widget_set_size_request(entryPasswordSignUp, 250, 40);
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), entryPasswordSignUp, 200, 290);
    gtk_entry_set_visibility(GTK_ENTRY(entryPasswordSignUp), false); // password
hidden

    entryRepeatPass = gtk_entry_new();
    gtk_widget_set_size_request(entryRepeatPass, 250, 40);
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), entryRepeatPass, 200, 380);
    gtk_entry_set_visibility(GTK_ENTRY(entryRepeatPass), false); // password
hidden

    labelAlreadyAcc = gtk_label_new("Sudah punya akun?");
```

```
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), labelAlreadyAcc, 200, 500);

    // BUTTON
    buttonCreateNewAcc = gtk_button_new_with_label("Daftar");
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), buttonCreateNewAcc, 200, 440);

    buttonBackToLogin = gtk_button_new_with_label("Login");
    gtk_layout_put(GTK_LAYOUT(layoutSignUp), buttonBackToLogin, 375, 495);

    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css, "src/css/sign_up.css", NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowSignUp);

    // SIGNAL
    g_signal_connect(windowSignUp, "destroy", G_CALLBACK(gtk_main_quit), NULL);
    g_signal_connect(buttonBackToLogin, "clicked",
G_CALLBACK(handle_display_sign_in), windowSignUp);
    g_signal_connect(buttonCreateNewAcc, "clicked", G_CALLBACK(create_new_acc),
NULL);

    gtk_main();
}
void handle_display_sign_up(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_sign_up();
}
```

dashboard.c

```
#include "dashboard.h"

void open_website(GtkWidget *widget, GtkWidget *window){
    if (g_strcmp0(gtk_widget_get_name(widget), "trailer1") == 0){
        gtk_show_uri_on_window(NULL, "https://www.youtube.com/watch?v=jtDRXPTZT-
M", GDK_CURRENT_TIME, NULL);
    }
    else if (g_strcmp0(gtk_widget_get_name(widget), "trailer2") == 0){
        gtk_show_uri_on_window(NULL,
"https://www.youtube.com/watch?v=06pl59umDlw", GDK_CURRENT_TIME, NULL);
    }
    else if (g_strcmp0(gtk_widget_get_name(widget), "trailer3") == 0){
        gtk_show_uri_on_window(NULL,
"https://www.youtube.com/watch?v=cSp1dM2Vj48", GDK_CURRENT_TIME, NULL);
    }
    else if (g_strcmp0(gtk_widget_get_name(widget), "trailer4") == 0){
        gtk_show_uri_on_window(NULL,
"https://www.youtube.com/watch?v=ZjIquMdpsSA&t=5s", GDK_CURRENT_TIME, NULL);
    }
}

void display_dashboard(GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));

    // WIDGET
    GtkWidget *windowDashboard;
    GtkWidget *layoutDashboard;
    GtkWidget *bgDashboard;

    // LABEL
    GtkWidget *labelTitle;
    GtkWidget *labelPlaying;
```

```
    // BUTTON
    GtkWidget *buttonDashtoLogin;
    GtkWidget *buttonPembelian;
    GtkWidget *buttonPemilihanKursi;
    GtkWidget *buttonPembayaran;
    GtkWidget *buttonInformasi;

    GtkWidget *buttonTrailerFilm1;
    GtkWidget *buttonTrailerFilm2;
    GtkWidget *buttonTrailerFilm3;
    GtkWidget *buttonTrailerFilm4;

    // ICON
    GdkPixbuf *iconDashboard;

    gtk_init(NULL, NULL);

    // WINDOW
    windowDashboard = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowDashboard), "Dashboard D'MOV1E");
    gtk_window_set_default_size(GTK_WINDOW(windowDashboard), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowDashboard), GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutDashboard = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowDashboard), layoutDashboard);

    // ICON
    iconDashboard = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowDashboard), iconDashboard);

    // BACKGROUND
    bgDashboard = gtk_image_new_from_file("src/image/dashboard.png");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), bgDashboard, 0, 0);

    // LABEL
    labelTitle = gtk_label_new("Dashboard");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), labelTitle, 270, 50);
    gtk_widget_set_name(labelTitle, "labelTitle");

    labelPlaying = gtk_label_new("Playing Now!");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), labelPlaying, 880, 50);
    gtk_widget_set_name(labelPlaying, "labelPlaying");

    // BUTTON
    buttonDashtoLogin = gtk_button_new_with_label("Logout");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonDashtoLogin, 140, 565);
    gtk_widget_set_size_request(buttonDashtoLogin, 400, 75);


    buttonPembelian = gtk_button_new_with_label("Pembelian");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonPembelian, 140, 165);
    gtk_widget_set_size_request(buttonPembelian, 400, 75);

    buttonPemilihanKursi = gtk_button_new_with_label("Pemilihan Kursi");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonPemilihanKursi, 140, 265);
    gtk_widget_set_size_request(buttonPemilihanKursi, 400, 75);

    buttonPembayaran = gtk_button_new_with_label("Pembayaran");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonPembayaran, 140, 365);
    gtk_widget_set_size_request(buttonPembayaran, 400, 75);

    buttonInformasi = gtk_button_new_with_label("Informasi");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonInformasi, 140, 465);
    gtk_widget_set_size_request(buttonInformasi, 400, 75);

    // FILM POSTER
    buttonTrailerFilm1 = gtk_button_new_with_label("");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonTrailerFilm1, 800, 170);
    gtk_widget_set_name(buttonTrailerFilm1, "trailer1");
```

```
    gtk_widget_set_size_request(buttonTrailerFilm1, 141, 210);

    buttonTrailerFilm2 = gtk_button_new_with_label("");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonTrailerFilm2, 980, 170);
    gtk_widget_set_name(buttonTrailerFilm2, "trailer2");
    gtk_widget_set_size_request(buttonTrailerFilm2, 141, 210);

    buttonTrailerFilm3 = gtk_button_new_with_label("");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonTrailerFilm3, 800, 425);
    gtk_widget_set_name(buttonTrailerFilm3, "trailer3");
    gtk_widget_set_size_request(buttonTrailerFilm3, 141, 210);

    buttonTrailerFilm4 = gtk_button_new_with_label("");
    gtk_layout_put(GTK_LAYOUT(layoutDashboard), buttonTrailerFilm4, 980, 425);
    gtk_widget_set_name(buttonTrailerFilm4, "trailer4");
    gtk_widget_set_size_request(buttonTrailerFilm4, 141, 210);

    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css, "src/css/dashboard.css", NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowDashboard);

    // SIGNAL
    g_signal_connect(windowDashboard, "destroy", G_CALLBACK(gtk_main_quit),
NULL);
    g_signal_connect(buttonDashtoLogin, "clicked",
G_CALLBACK(handle_display_sign_in), windowDashboard);
    g_signal_connect(buttonPembelian, "clicked",
G_CALLBACK(handle_display_dashboard_pembelian), windowDashboard);
    g_signal_connect(buttonPemilihanKursi, "clicked",
G_CALLBACK(handle_display_dashboard_pemilihan_kursi), windowDashboard);
    g_signal_connect(buttonPembayaran, "clicked",
G_CALLBACK(handle_display_dashboard_pembayaran), windowDashboard);
    g_signal_connect(buttonInformasi, "clicked",
G_CALLBACK(handle_display_dashboard_informasi), windowDashboard);
    // TRAILER SIGNAL
    g_signal_connect(buttonTrailerFilm1, "clicked", G_CALLBACK(open_website),
windowDashboard);
    g_signal_connect(buttonTrailerFilm2, "clicked", G_CALLBACK(open_website),
windowDashboard);
    g_signal_connect(buttonTrailerFilm3, "clicked", G_CALLBACK(open_website),
windowDashboard);
    g_signal_connect(buttonTrailerFilm4, "clicked", G_CALLBACK(open_website),
windowDashboard);

    gtk_main();
}
void handle_display_dashboard(GtkWidget *widget, GtkWidget *window){
    display_dashboard(window);
}
```

dashboard_informasi.c

```
#include "dashboard_informasi.h"

void display_dashboard_informasi(){
    // WIDGET
    GtkWidget *windowDashInformasi;
    GtkWidget *layoutDashInformasi;
    GtkWidget *bgDashInformasi;

    // BUTTON
    GtkWidget *buttonAbout;
```

```
    GtkWidget *buttonInfoQueue;
    GtkWidget *buttonHighTransaction;
    GtkWidget *buttonTransactionHistory;

    GtkWidget *buttonToDash;
    GtkWidget *labelTitle;

    // ICON
    GdkPixbuf *iconDashInformasi;

    gtk_init(NULL, NULL);

    // WINDOW
    windowDashInformasi = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowDashInformasi), "Informasi");
    gtk_window_set_default_size(GTK_WINDOW(windowDashInformasi), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowDashInformasi),
GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutDashInformasi = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowDashInformasi), layoutDashInformasi);

    // ICON
    iconDashInformasi = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowDashInformasi), iconDashInformasi);

    // BACKGROUND
    bgDashInformasi = gtk_image_new_from_file("src/image/informasi.png");
    gtk_layout_put(GTK_LAYOUT(layoutDashInformasi), bgDashInformasi, 0, 0);

     // label
    labelTitle = gtk_label_new("Informasi");
    gtk_layout_put(GTK_LAYOUT(layoutDashInformasi), labelTitle, 570, 50);
    gtk_widget_set_name(labelTitle, "labelTitle");


    // BUTTON
    buttonToDash = gtk_button_new_with_label("Dashboard");
    gtk_layout_put(GTK_LAYOUT(layoutDashInformasi), buttonToDash, 50,550);
    gtk_widget_set_size_request(buttonToDash, 540, 70);

    buttonInfoQueue = gtk_button_new_with_label("Info Antrian");
    gtk_layout_put(GTK_LAYOUT(layoutDashInformasi), buttonInfoQueue, 50,150);
    gtk_widget_set_size_request(buttonInfoQueue, 540, 70);


    buttonHighTransaction = gtk_button_new_with_label("Pembeli Terbaik");
    gtk_layout_put(GTK_LAYOUT(layoutDashInformasi), buttonHighTransaction,
50,350);
    gtk_widget_set_size_request(buttonHighTransaction, 550, 70);


    buttonAbout = gtk_button_new_with_label("Tentang Aplikasi");
    gtk_layout_put(GTK_LAYOUT(layoutDashInformasi), buttonAbout, 50,450);
    gtk_widget_set_size_request(buttonAbout, 540, 70);


    buttonTransactionHistory = gtk_button_new_with_label("Riwayat Transaksi");
    gtk_layout_put(GTK_LAYOUT(layoutDashInformasi), buttonTransactionHistory,
50,250);
    gtk_widget_set_size_request(buttonTransactionHistory, 550, 70);


    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
```

```
    gtk_css_provider_load_from_path(css, "src/css/dashboard_informasi.css",
NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowDashInformasi);

    // SIGNAL
    g_signal_connect(windowDashInformasi, "destroy", G_CALLBACK(gtk_main_quit),
NULL);
    g_signal_connect(buttonToDash, "clicked",
G_CALLBACK(handle_display_dashboard), windowDashInformasi);

    g_signal_connect(buttonHighTransaction, "clicked",
G_CALLBACK(handle_display_informasi_pembeli), windowDashInformasi);
    g_signal_connect(buttonInfoQueue, "clicked",
G_CALLBACK(handle_display_informasi_antrian), windowDashInformasi);
    g_signal_connect(buttonAbout, "clicked",
G_CALLBACK(handle_display_informasi_aplikasi), windowDashInformasi);
    g_signal_connect(buttonTransactionHistory, "clicked",
G_CALLBACK(handle_display_informasi_transaksi), windowDashInformasi);

    gtk_main();
}

void handle_display_dashboard_informasi(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_dashboard_informasi();
}
```

dashboard_informasi_antrian.c

```
#include "dashboard_informasi_antrian.h"

void display_informasi_antrian(){
     // WIDGET
    GtkWidget *windowInfoAntrian;
    GtkWidget *layoutInfoAntrian;
    GtkWidget *bgInfoAntrian;

    GtkWidget *buttonToInformasi;

    // LABEL
    GtkWidget *labelQueueSelect;
    GtkWidget *labelQueuePay;
    GtkWidget *labelTitle;

    GtkWidget *labelQueueName;
    GtkWidget *labelEmptyData;

    // ICON
    GdkPixbuf *iconInfoAntrian;

    gtk_init(NULL, NULL);

    // WINDOW
    windowInfoAntrian = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowInfoAntrian), "Informasi Antrian");
    gtk_window_set_default_size(GTK_WINDOW(windowInfoAntrian), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowInfoAntrian), GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutInfoAntrian = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowInfoAntrian), layoutInfoAntrian);

    // ICON
    iconInfoAntrian = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowInfoAntrian), iconInfoAntrian);

    // BACKGROUND
```

```
    bgInfoAntrian = gtk_image_new_from_file("src/image/informasi_antrian.png");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), bgInfoAntrian, 0, 0);

    // LABEL
    labelTitle = gtk_label_new("Informasi Antrian");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), labelTitle, 500, 50);
    gtk_widget_set_name(labelTitle, "labelTitle");

    // BUTTON
    buttonToInformasi = gtk_button_new_with_label("Informasi");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), buttonToInformasi, 570,660);

    labelQueueName = gtk_label_new("Antrian Pemilihan Kursi");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), labelQueueName, 300,200);

    labelQueueName = gtk_label_new("Antrian Pembayaran Tiket");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), labelQueueName, 750,200);

    labelEmptyData = gtk_label_new("Belum ada data");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), labelEmptyData, 300,250);

    int x = 300, y = 250;

    FILE *f;
    f = fopen("data/user_ticket_data.txt", "r");
    int counterLine;

    char ch, compUsr[100];
    int counter, idx;

    reset_string(compUsr, 100);
    counter = 0;
    counterLine = 0;
    idx = 1;
    bool hasDeleted = false;

    while(!feof(f) && counterLine < 10){
        ch = fgetc(f);
        if (ch == '\n'){
            counterLine++;
            labelQueueSelect = gtk_label_new(compUsr);
            if (!hasDeleted){
                gtk_widget_destroy(labelEmptyData);
                hasDeleted = true;
            }
            gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), labelQueueSelect,
x,y);
            y += 50;

            reset_string(compUsr, 100);
            counter = 0;
            idx = 1;
        }
        else if (ch == '|'){
            idx++;
            counter = 0;
        }
        else{
            if (idx == 1){
                compUsr[counter] = ch;
                counter++;
            }
        }

    }
    fclose(f);

    labelEmptyData = gtk_label_new("Belum ada data");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), labelEmptyData, 750,250);
    hasDeleted = false;
```

```
    x = 750, y = 250;
    f = fopen("data/user_invoice_data.txt", "r");

    reset_string(compUsr, 100);
    counter = 0;
    counterLine = 0;
    idx = 1;

    while(!feof(f) && counterLine < 10){
        ch = fgetc(f);
        if (ch == '\n'){
            counterLine++;
            labelQueuePay = gtk_label_new(compUsr);
            if (!hasDeleted){
                gtk_widget_destroy(labelEmptyData);
                hasDeleted = true;
            }
            gtk_layout_put(GTK_LAYOUT(layoutInfoAntrian), labelQueuePay, x,y);
            y += 50;

            reset_string(compUsr, 100);
            counter = 0;
            idx = 1;
        }
        else if (ch == '|'){
            idx++;
            counter = 0;
        }
        else{
            if (idx == 1){
                compUsr[counter] = ch;
                counter++;
            }
        }

    }
    fclose(f);


    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css,
"src/css/dashboard_informasi_antrian.css", NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowInfoAntrian);

    // SIGNAL
    g_signal_connect(windowInfoAntrian, "destroy", G_CALLBACK(gtk_main_quit),
NULL);
    g_signal_connect(buttonToInformasi, "clicked",
G_CALLBACK(handle_display_dashboard_informasi), windowInfoAntrian);

    gtk_main();
}

void handle_display_informasi_antrian(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_informasi_antrian();
}
```

dashboard_informasi_aplikasi.c

```
#include "dashboard_informasi_aplikasi.h"
```

```
void display_informasi_aplikasi(){
    GtkWidget *windowInfoAplikasi;
    GtkWidget *layoutInfoAplikasi;
    GtkWidget *bgInfoAplikasi;

    GtkWidget *buttonToInformasi;


    // ICON
    GdkPixbuf *iconInfoAplikasi;

    gtk_init(NULL, NULL);

    // WINDOW
    windowInfoAplikasi = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowInfoAplikasi), "Informasi Aplikasi");
    gtk_window_set_default_size(GTK_WINDOW(windowInfoAplikasi), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowInfoAplikasi), GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutInfoAplikasi = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowInfoAplikasi), layoutInfoAplikasi);

    // ICON
    iconInfoAplikasi = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowInfoAplikasi), iconInfoAplikasi);

    // BACKGROUND
    bgInfoAplikasi = gtk_image_new_from_file("src/image/Tentang Aplikasi.png");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAplikasi), bgInfoAplikasi, 0, 0);

    // BUTTON
    buttonToInformasi = gtk_button_new_with_label("Informasi");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAplikasi), buttonToInformasi, 50,660);

    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css,
"src/css/dashboard_informasi_aplikasi.css", NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowInfoAplikasi);

    // SIGNAL
    g_signal_connect(windowInfoAplikasi, "destroy", G_CALLBACK(gtk_main_quit),
NULL);
    g_signal_connect(buttonToInformasi, "clicked",
G_CALLBACK(handle_display_dashboard_informasi), windowInfoAplikasi);

    gtk_main();
}

void handle_display_informasi_aplikasi(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_informasi_aplikasi();
}
```

dashboard_informasi_transaksi.c

```
#include "dashboard_informasi_aplikasi.h"

void display_informasi_aplikasi(){
    GtkWidget *windowInfoAplikasi;
    GtkWidget *layoutInfoAplikasi;
    GtkWidget *bgInfoAplikasi;
```

```
    GtkWidget *buttonToInformasi;


    // ICON
    GdkPixbuf *iconInfoAplikasi;

    gtk_init(NULL, NULL);

    // WINDOW
    windowInfoAplikasi = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowInfoAplikasi), "Informasi Aplikasi");
    gtk_window_set_default_size(GTK_WINDOW(windowInfoAplikasi), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowInfoAplikasi), GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutInfoAplikasi = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowInfoAplikasi), layoutInfoAplikasi);

    // ICON
    iconInfoAplikasi = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowInfoAplikasi), iconInfoAplikasi);

    // BACKGROUND
    bgInfoAplikasi = gtk_image_new_from_file("src/image/Tentang Aplikasi.png");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAplikasi), bgInfoAplikasi, 0, 0);

    // BUTTON
    buttonToInformasi = gtk_button_new_with_label("Informasi");
    gtk_layout_put(GTK_LAYOUT(layoutInfoAplikasi), buttonToInformasi, 50,660);

    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css,
"src/css/dashboard_informasi_aplikasi.css", NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowInfoAplikasi);

    // SIGNAL
    g_signal_connect(windowInfoAplikasi, "destroy", G_CALLBACK(gtk_main_quit),
NULL);
    g_signal_connect(buttonToInformasi, "clicked",
G_CALLBACK(handle_display_dashboard_informasi), windowInfoAplikasi);

    gtk_main();
}

void handle_display_informasi_aplikasi(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_informasi_aplikasi();
}
```

dashboard_antrian_pembeli.c

```
#include "dashboard_informasi_pembeli.h"

int foundHighest[3];
char chrUsernameFoundHighest[3][100] = {"data kosong", "data kosong", "data
kosong"};
char chrFilmNameFoundHighest[3][50] = {"data kosong", "data kosong", "data
kosong"};
int counterFoundHigh = -1;

void find_highest(BST *curr){
```

```
    if (curr != NULL){
        find_highest(curr->bRight);

        counterFoundHigh++;
        if (counterFoundHigh < 3){
            strcpy(chrUsernameFoundHighest[counterFoundHigh], curr->bUsername);
            strcpy(chrFilmNameFoundHighest[counterFoundHigh], curr->bFilmName);
            foundHighest[counterFoundHigh] = curr->bPayment;
        }

        find_highest(curr->bLeft);
    }
}

void display_informasi_pembeli(){
    bRoot = NULL;
     // WIDGET
    GtkWidget *windowInfoBST;
    GtkWidget *layoutInfoBST;
    GtkWidget *bgInfoBST;

    GtkWidget *buttonToInformasi;

    // LABEL
    GtkWidget *labelPembeliTerbaik;
    GtkWidget *labelTitle;

    // ICON
    GdkPixbuf *iconInfoBST;

    gtk_init(NULL, NULL);

    // WINDOW
    windowInfoBST = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowInfoBST), "Informasi Pembeli
Terbaik");
    gtk_window_set_default_size(GTK_WINDOW(windowInfoBST), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowInfoBST), GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutInfoBST = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowInfoBST), layoutInfoBST);

    // ICON
    iconInfoBST = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowInfoBST), iconInfoBST);

    // BACKGROUND
    bgInfoBST = gtk_image_new_from_file("src/image/PEMBELI TERSULTAN.png");
    gtk_layout_put(GTK_LAYOUT(layoutInfoBST), bgInfoBST, 0, 0);

     // label
    labelTitle = gtk_label_new("Pembeli Terbaik");
    gtk_layout_put(GTK_LAYOUT(layoutInfoBST), labelTitle, 550, 50);
    gtk_widget_set_name(labelTitle, "labelTitle");

    // BUTTON
    buttonToInformasi = gtk_button_new_with_label("Informasi");
    gtk_layout_put(GTK_LAYOUT(layoutInfoBST), buttonToInformasi, 600,650);

    // load data from file
    FILE *f;
    f = fopen("data/user_transaction.txt", "r");
    // make binary search tree from file
    char compUsr[100], compFilmName[50];
    int compPayment;

    reset_string(compUsr, 100);
    reset_string(compFilmName, 50);
    compPayment = 0;
```

```
int counter = 0, idx = 1;
char ch;
while(!feof(f)){
    ch = fgetc(f);
    if (ch == '\n'){
        // make tree
        bNode = (BST*)malloc(sizeof(BST));
        strcpy(bNode->bUsername, compUsr);
        strcpy(bNode->bFilmName, compFilmName);
        bNode->bPayment = compPayment;
        bNode->bLeft = NULL;
        bNode->bRight = NULL;

        if (bRoot == NULL){
            bRoot = bNode;
        }
        else{
            BST *curr = bRoot, *parent = NULL;
            while(curr != NULL){
                parent = curr;
                if (bNode->bPayment > curr->bPayment){
                    curr = curr->bRight;
                }
                else{
                    curr = curr->bLeft;
                }
            }
            if (bNode->bPayment > parent->bPayment){
                parent->bRight = bNode;
            }
            else{
                parent->bLeft = bNode;
            }
        }
        // reset
        reset_string(compUsr, 100);
        reset_string(compFilmName, 50);
        compPayment = 0;

        counter = 0;
        idx = 1;
    }
    else if (ch == '|'){
        counter = 0;
        idx++;
    }
    else{
        if (idx == 1){
            compUsr[counter] = ch;
            counter++;
        }
        else if (idx == 2){
            compFilmName[counter] = ch;
            counter++;
        }
        else if (idx == 3){
            int temp = ch - '0';
            compPayment = (compPayment * 10) + temp;
        }
    }
}
fclose(f);

// search top highest 3
if (bRoot != NULL){
        find_highest(bRoot);

}
// display it
```

```
    int i, x, y;
    for (i = 0; i < 3; i++){
        if (i == 0){
            x = 570;
            y = 300;
        }
        else if (i == 1){
            x = 300;
            y = 430;
        }
        else{
            x = 830;
            y = 520;
        }
        labelPembeliTerbaik = gtk_label_new(chrUsernameFoundHighest[i]);
        gtk_layout_put(GTK_LAYOUT(layoutInfoBST), labelPembeliTerbaik, x,y);
        y += 20;
        labelPembeliTerbaik = gtk_label_new(chrFilmNameFoundHighest[i]);
        gtk_layout_put(GTK_LAYOUT(layoutInfoBST), labelPembeliTerbaik, x,y);
        y += 20;
        char temp[100];
        reset_string(temp, 100);
        if (foundHighest[i] == 0){
            strcpy(temp, "data kosong");
        }
        else{
            sprintf(temp, "%d", foundHighest[i]);
        }

        labelPembeliTerbaik = gtk_label_new(temp);
        gtk_layout_put(GTK_LAYOUT(layoutInfoBST), labelPembeliTerbaik, x,y);
    }

    counterFoundHigh = -1;

    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css,
"src/css/dashboard_informasi_pembeli.css", NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowInfoBST);

    // SIGNAL
    g_signal_connect(windowInfoBST, "destroy", G_CALLBACK(gtk_main_quit), NULL);
    g_signal_connect(buttonToInformasi, "clicked",
G_CALLBACK(handle_display_dashboard_informasi), windowInfoBST);

    gtk_main();
}

void handle_display_informasi_pembeli(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_informasi_pembeli();
}
```

dashboard_pembayaran.c

```
#include "dashboard_pembayaran.h"

// input
GtkWidget *entryDibayarPelanggan;
GtkWidget *layoutDashPembayaran;
```

```
GtkWidget *labelPecahanUangTersedia;
GtkWidget *labelPecahanUangKembali;

int arrUangKembalian[7];

void get_user_invoice_from_file(){
    qFrontPay = NULL;

    char compUsr[100], compBuyTime[50], compFilmName[50], compFilmTime[20],
compFilmChair[50];
    int compTicket, compStudioFilm, compStudioTime;
    reset_string(compUsr, 100);
    reset_string(compBuyTime, 50);
    reset_string(compFilmName, 50);
    reset_string(compFilmTime, 20);
    reset_string(compFilmChair, 50);

    compTicket = 0;
    compStudioFilm = 0;
    compStudioTime = 0;

    FILE *f;
    f = fopen("data/user_invoice_data.txt", "r");
    char ch;
    int idx = 1, counter = 0;
    while(!feof(f)){
        ch = fgetc(f);
        if (ch == '|'){
            counter = 0;
            idx++;
        }
        else if (ch == '\n'){

            qUser = malloc(sizeof(Queue));
            strcpy(qUser->qUsername, compUsr);
            qUser->qTiketDibeli = compTicket;
            strcpy(qUser->qBuyTime, compBuyTime);

            qUser->qStudioFilm = compStudioFilm;
            qUser->qStudioTime = compStudioTime;

            strcpy(qUser->qFilmName, compFilmName);
            strcpy(qUser->qFilmTime, compFilmTime);
            strcpy(qUser->qFilmChair, compFilmChair);

            qUser->qNext = NULL;

            if (qFrontPay == NULL){
                qFrontPay = qUser;
            }
            else{
                Queue *prev;
                prev = qFrontPay;
                while(prev->qNext != NULL){
                    prev = prev->qNext;
                }
                prev->qNext = qUser;
            }

            // reset
            reset_string(compUsr, 100);
            reset_string(compBuyTime, 50);
            reset_string(compFilmName, 50);
            reset_string(compFilmTime, 20);
            reset_string(compFilmChair, 50);

            compTicket = 0;
            compStudioFilm = 0;
            compStudioTime = 0;
```

```
            counter = 0;
            idx = 1;
        }
        else{
            if (idx == 1){
                compUsr[counter] = ch;
                counter++;
            }
            else if(idx == 2){
                int tempCompTicket = ch - '0';
                compTicket = compTicket*10 + tempCompTicket;
                counter++;
            }
            else if (idx == 3){
                compBuyTime[counter] = ch;
                counter++;
            }
            else if (idx == 4){
                int tempCompStudioFilm = ch - '0';
                compStudioFilm = tempCompStudioFilm;
                counter++;
            }
            else if (idx == 5){
                int tempCompStudioTime = ch - '0';
                compStudioTime = tempCompStudioTime;
                counter++;
            }
            else if (idx == 6){
                compFilmName[counter] = ch;
                counter++;
            }
            else if (idx == 7){
                compFilmTime[counter] = ch;
                counter++;
            }
            else if (idx == 8){
                compFilmChair[counter] = ch;
                counter++;
            }
        }
    }
    fclose(f);
}

int get_uang_kembalian_rec(int money, int depth){
    if (money == 0){
        return 0;
    }
    else if(money < MoneyCurrency[depth]){
        get_uang_kembalian_rec(money, depth+1);
    }
    else{
        arrUangKembalian[depth] = money / MoneyCurrency[depth];
        get_uang_kembalian_rec(money % MoneyCurrency[depth], depth);
    }
    return 0;
}


void count_cash_back(GtkWidget *widget, GtkWidget *window){
    char tempUangPelanggan[100];
    reset_string(tempUangPelanggan, 100);
    const gchar *gUangPelanggan =
gtk_entry_get_text(GTK_ENTRY(entryDibayarPelanggan));
    strcpy(tempUangPelanggan, gUangPelanggan);

    // UANG FROM STRING TO INT
    int i = 0, iUangPelanggan = 0;
    while(tempUangPelanggan[i] != '\0'){
        int temp = tempUangPelanggan[i] - '0';
```

```
        iUangPelanggan = iUangPelanggan * 10 + temp;
        i++;
    }

    // GET TICKET PRICE
    int saveIndex;
    for (i = 0; i < 4; i++){
        if (strcmp(qFrontPay->qFilmName, FilmName[i]) == 0){
            saveIndex = i;
            break;
        }
    }
    if (iUangPelanggan < (qFrontPay->qTiketDibeli * TicketPrice[saveIndex])){
        display_error_dialog("Uang tidak cukup!", window);
    }
    else{
        gtk_widget_destroy(labelPecahanUangTersedia);
        gtk_widget_destroy(labelPecahanUangKembali);

        // COUNT MONEY BACK
        labelPecahanUangTersedia = gtk_label_new(
                                    "Rp 100.000 : \n\n"
                                    "Rp 50.000 : \n\n"
                                    "Rp 20.000 : \n\n"
                                    "Rp 10.000 : \n\n"
                                    "Rp 5.000 : \n\n"
                                    "Rp 2.000 : \n\n"
                                    "Rp 1.000 : "
                                    );
        char tempPecahanUangKembali[50];
        reset_string(tempPecahanUangKembali, 50);
        int uangKembalian = iUangPelanggan - (qFrontPay->qTiketDibeli *
TicketPrice[saveIndex]);
        int i;
        for (i = 0; i < 7; i++){
            arrUangKembalian[i] = 0;
        }

        // RECURSIVE
        get_uang_kembalian_rec(uangKembalian, 0);

        // DISPLAY MONEY BACK
        for (i = 0; i < 7; i++){
            char strTemp[100];
            reset_string(strTemp, 100);
            sprintf(strTemp, "%d lembar\n\n", arrUangKembalian[i]);
            strcat(tempPecahanUangKembali, strTemp);
        }
        labelPecahanUangKembali = gtk_label_new(tempPecahanUangKembali);
        GtkWidget *labelTitleKembalian;
        labelTitleKembalian = gtk_label_new("Uang Kembalian");
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelTitleKembalian,
950, 200);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran),
labelPecahanUangTersedia, 950, 250);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran),
labelPecahanUangKembali, 1065, 250);
        gtk_widget_show_all(window);
    }
}

void display_warn_pembayaran(GtkWidget *widget, GtkWidget *window){
    bool isDeleted = false;
    if (qFrontPay == NULL){
        display_error_dialog("Data pembayaran kosong", window);
    }
    else{
        GtkWidget *dialog;
        dialog = gtk_message_dialog_new(GTK_WINDOW(window),
                GTK_DIALOG_DESTROY_WITH_PARENT,
```

```
                GTK_MESSAGE_ERROR,
                GTK_BUTTONS_YES_NO,
                "Konfirmasi Pembayaran?");
        gtk_window_set_title(GTK_WINDOW(dialog), "Warning");
        int res = gtk_dialog_run(GTK_DIALOG(dialog));
        if (res == -8){ // YES
            isDeleted = true;
            // ADD TO TRANSACTION HISTORY
            FILE *f;
            f = fopen("data/transaction_history.txt", "a");
            fprintf(f, "%s|%d|%s|%d|%d|%s|%s|%s\n",
                        qFrontPay->qUsername, qFrontPay->qTiketDibeli,
qFrontPay->qBuyTime,
                        qFrontPay->qStudioFilm, qFrontPay->qStudioTime,
                        qFrontPay->qFilmName, qFrontPay->qFilmTime, qFrontPay-
>qFilmChair);
            fclose(f);

            // ADD TO USER TRANSACTION TO FIND BEST USER
            f = fopen("data/user_transaction.txt", "a");
            int saveIndex, i;
            for (i = 0; i < 4; i++){
                if (strcmp(qFrontPay->qFilmName, FilmName[i]) == 0){
                    saveIndex = i;
                    break;
                }
            }
            fprintf(f, "%s|%s|%d\n", qFrontPay->qUsername, qFrontPay->qFilmName,
(qFrontPay->qTiketDibeli * TicketPrice[saveIndex]));
            fclose(f);

            // DELETE IN QUEUE
            if (qFrontPay->qNext == NULL){
                free(qFrontPay);
                qFrontPay = NULL;
            }
            else{
                Queue *curr = qFrontPay;
                qFrontPay = qFrontPay->qNext;
                free(curr);
                curr = NULL;
            }

            // UPDATE DATA IN USER INVOICE
            f = fopen("data/user_invoice_data.txt", "w");
            Queue *curr = qFrontPay;
            while(curr != NULL){
                fprintf(f, "%s|%d|%s|%d|%d|%s|%s|%s\n",
                        curr->qUsername, curr->qTiketDibeli, curr->qBuyTime,
                        curr->qStudioFilm, curr->qStudioTime,
                        curr->qFilmName, curr->qFilmTime, curr->qFilmChair);
                curr = curr->qNext;
            }
            fclose(f);

            // REMOVE ALL QUEUE
            while(qFrontPay != NULL){
                Queue *curr = qFrontPay;
                if (qFrontPay->qNext == NULL){
                    free(qFrontPay);
                    qFrontPay = NULL;
                }
                else{
                    qFrontPay = qFrontPay->qNext;
                    free(curr);
                    curr = NULL;
                }
            }
        }
        gtk_widget_destroy(dialog);
```

```
    }
    if(isDeleted){
        // RELOAD TO READ DATA
        gtk_window_close(GTK_WINDOW(window));
        display_dashboard_pembayaran();
    }
}

void display_dashboard_pembayaran(){
    get_user_invoice_from_file();

    // WINDOW
    GtkWidget *windowDashPembayaran;

    // ICON
    GdkPixbuf *iconDashPembayaran;

    // BACKGROUND
    GtkWidget *bgDashPembayaran;

    // BUTTON
    GtkWidget *buttonBackToDash;
    GtkWidget *buttonConfirmBayar;
    GtkWidget *buttonCountCashBack;

    // LABEL
    GtkWidget *labelPembayaranNamaPembeli;
    GtkWidget *labelPembayaranJumlahTiket;
    GtkWidget *labelPembayaranWaktuPembelian;
    GtkWidget *labelPembayaranNamaFilm;
    GtkWidget *labelPembayaranStudioFilm;
    GtkWidget *labelPembayaranWaktuFilm;
    GtkWidget *labelPembayaranKursiDipilih;
    GtkWidget *labelPembayaranTotalHarga;
    GtkWidget *labelPembayaranDibayar;
    GtkWidget *labelUserPayInfo;
    GtkWidget *labelEmptyData;
    GtkWidget *labelTitle;

    gtk_init(NULL, NULL);

    // WINDOW
    windowDashPembayaran = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowDashPembayaran), "Pembayaran Tiket");
    gtk_window_set_default_size(GTK_WINDOW(windowDashPembayaran), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowDashPembayaran),
GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutDashPembayaran = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowDashPembayaran),
layoutDashPembayaran);

    // ICON
    iconDashPembayaran = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowDashPembayaran), iconDashPembayaran);

    // BACKGROUND
    bgDashPembayaran = gtk_image_new_from_file("src/image/pembayaran.png");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), bgDashPembayaran, 0, 0);

    // label
    labelTitle = gtk_label_new("Pembayaran");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelTitle, 550, 50);
    gtk_widget_set_name(labelTitle, "labelTitle");

    labelPecahanUangTersedia = gtk_label_new("");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPecahanUangTersedia,
800, 100);
```

```
    labelPecahanUangKembali = gtk_label_new("");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPecahanUangKembali,
900, 100);

    // label
    labelPembayaranNamaPembeli = gtk_label_new("Nama: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPembayaranNamaPembeli,
100, 200);
    labelPembayaranJumlahTiket = gtk_label_new("Jumlah: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPembayaranJumlahTiket,
100, 250);
    labelPembayaranWaktuPembelian = gtk_label_new("Waktu beli: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran),
labelPembayaranWaktuPembelian, 100, 300);

    labelPembayaranNamaFilm = gtk_label_new("Film: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPembayaranNamaFilm,
100, 350);
    labelPembayaranStudioFilm = gtk_label_new("Studio: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPembayaranStudioFilm,
100, 400);
    labelPembayaranWaktuFilm = gtk_label_new("Waktu: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPembayaranWaktuFilm,
100, 450);

    labelPembayaranKursiDipilih = gtk_label_new("Kursi: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran),
labelPembayaranKursiDipilih, 100, 500);

    labelPembayaranTotalHarga = gtk_label_new("Total Harga: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPembayaranTotalHarga,
500, 200);

    labelPembayaranDibayar = gtk_label_new("Pembayaran: ");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelPembayaranDibayar,
500, 250);

    // BUTTON
    buttonBackToDash = gtk_button_new_with_label("Dashboard");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), buttonBackToDash, 150,
550);

    buttonConfirmBayar = gtk_button_new_with_label("Konfirmasi Pembayaran");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), buttonConfirmBayar, 950,
550);

    buttonCountCashBack = gtk_button_new_with_label("");

    // DISPLAY IF THERE IS NO DATA
    if (qFrontPay == NULL){
        int i, y = 200;
        for (i = 0; i < 7; i++){
            labelEmptyData = gtk_label_new("Belum ada data");
            gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelEmptyData,
200, y);
            y += 50;
        }
        y = 200;
        for (i = 0; i < 2; i++){
            labelEmptyData = gtk_label_new("Belum ada data");
            gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelEmptyData,
610, y);
            y += 50;
        }
    }

    // DISPLAY IF THERE IS DATA
    if (qFrontPay != NULL){
        gtk_widget_destroy(buttonCountCashBack);
```

```
        buttonCountCashBack = gtk_button_new_with_label("Hitung Kembalian");
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), buttonCountCashBack,
550, 550);

        entryDibayarPelanggan = gtk_entry_new();
        gtk_widget_set_size_request(entryDibayarPelanggan, 250, 40);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), entryDibayarPelanggan,
500, 300);

        // USER INFORMATION
        labelUserPayInfo = gtk_label_new(qFrontPay->qUsername);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelUserPayInfo, 200,
200);

        char tempUserInfo[50];
        reset_string(tempUserInfo, 50);
        sprintf(tempUserInfo, "%d", qFrontPay->qTiketDibeli);
        labelUserPayInfo = gtk_label_new(tempUserInfo);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelUserPayInfo, 200,
250);

        char tempBuyTime[10];
        reset_string(tempBuyTime, 10);
        int a = 0;
        while(a < 9){
            tempBuyTime[a] = qFrontPay->qBuyTime[a];
            a++;
        }
        labelUserPayInfo = gtk_label_new(tempBuyTime);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelUserPayInfo, 200,
300);

        labelUserPayInfo = gtk_label_new(qFrontPay->qFilmName);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelUserPayInfo, 200,
350);

        reset_string(tempUserInfo, 50);
        sprintf(tempUserInfo, "%d", qFrontPay->qStudioFilm);
        labelUserPayInfo = gtk_label_new(tempUserInfo);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelUserPayInfo, 200,
400);

        reset_string(tempUserInfo, 50);
        strcpy(tempUserInfo, StudioTime[(qFrontPay->qStudioFilm -
1)][(qFrontPay->qStudioTime - 1)]);
        labelUserPayInfo = gtk_label_new(tempUserInfo);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelUserPayInfo, 200,
450);

        labelUserPayInfo = gtk_label_new(qFrontPay->qFilmChair);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelUserPayInfo, 200,
500);

        int i, saveIndex;
        for (i = 0; i < 4; i++){
            if (strcmp(qFrontPay->qFilmName, FilmName[i]) == 0){
                saveIndex = i;
                break;
            }
        }
        reset_string(tempUserInfo, 50);
        sprintf(tempUserInfo, "%d", qFrontPay->qTiketDibeli *
TicketPrice[saveIndex]);
        labelUserPayInfo = gtk_label_new(tempUserInfo);
        gtk_layout_put(GTK_LAYOUT(layoutDashPembayaran), labelUserPayInfo, 610,
200);
    }

    // CSS
    GdkDisplay *display;
```

```
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css, "src/css/dashboard_pembayaran.css",
NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowDashPembayaran);

    // SIGNAL
    g_signal_connect(windowDashPembayaran, "destroy", G_CALLBACK(gtk_main_quit),
NULL);
    g_signal_connect(buttonBackToDash, "clicked",
G_CALLBACK(handle_display_dashboard), windowDashPembayaran);
    g_signal_connect(buttonCountCashBack, "clicked",
G_CALLBACK(count_cash_back), windowDashPembayaran);
    g_signal_connect(buttonConfirmBayar, "clicked",
G_CALLBACK(display_warn_pembayaran), windowDashPembayaran);

    gtk_main();
}

void handle_display_dashboard_pembayaran(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_dashboard_pembayaran();
}
```

dashboard_pembelian.c

```
#include "dashboard_pembelian.h"

// WINDOW
GtkWidget *windowDashPembelian;

// LAYOUT
GtkWidget *layoutDashPembelian;

// LABEL
GtkWidget *labelTimeNow;

// INPUT
GtkWidget *entryNamaPembeli;
GtkWidget *entryTiketDibeli;

// COMBOBOX
GtkWidget *comboBox;
GtkWidget *comboBoxWaktu;

bool exitDashPembelian;

void reset_window(GtkWidget *widget){
    exitDashPembelian = true;
}

void combo_selected(GtkWidget *comboBox){
    // GET ACTIVE TEXT FROM COMBOBOX
    gchar *getFilmDipilih =
gtk_combo_box_text_get_active_text(GTK_COMBO_BOX_TEXT(comboBox));
    char chrGetFilmDipilih[50];
    reset_string(chrGetFilmDipilih, 50);
    sprintf(chrGetFilmDipilih, "%s", getFilmDipilih);

    // RESET COMBOBOX WAKTU
    gtk_combo_box_text_remove_all(GTK_COMBO_BOX_TEXT(comboBoxWaktu));

    // ADD COMBOBOX WAKTU WITH NEW DATA ACCORDING TO ACTIVE TEXT
    int i,j;
    for (i =0 ; i< 4; i++){
```

```
        // FIND THE ACTIVE TEXT IN THE ARRAY
        char compFilmDipilih[50];
        reset_string(compFilmDipilih, 50);
        strcpy(compFilmDipilih, FilmName[i]);
        if(strcmp(compFilmDipilih, chrGetFilmDipilih) == 0){
            // IF FOUND
            for (j =0; j < 3 ; j++){
                // ADD COMBOBOX WAKTU WITH THE DATA
                char appendText[20];
                reset_string(appendText, 20);
                strcpy(appendText, FilmTime[i][j]);

gtk_combo_box_text_append_text(GTK_COMBO_BOX_TEXT(comboBoxWaktu), appendText);
            }
            break;
        }
    }
    gtk_widget_show_all(windowDashPembelian);
}

void display_warn_pembelian(GtkWidget *widget, GtkWidget *window){
    GtkWidget *dialog;
    dialog = gtk_message_dialog_new(GTK_WINDOW(window),
            GTK_DIALOG_DESTROY_WITH_PARENT,
            GTK_MESSAGE_ERROR,
            GTK_BUTTONS_YES_NO,
            "Konfirmasi Pembelian?");
    gtk_widget_set_name(dialog, "warnDialog");
    gtk_window_set_title(GTK_WINDOW(dialog), "Warning");
    int res = gtk_dialog_run(GTK_DIALOG(dialog));

    // GET USER INPUT IN GCHAR
    const gchar *gNamaPembeli = gtk_entry_get_text(GTK_ENTRY(entryNamaPembeli));
    const gchar *gTiketDibeli = gtk_entry_get_text(GTK_ENTRY(entryTiketDibeli));

    if (res == -8){ // YES
        char chrWaktuFilm[20];
        reset_string(chrWaktuFilm, 20);

        // TIME CALC
        time_t rawtime;
        struct tm *timeInfo;
        time(&rawtime);
        timeInfo = localtime(&rawtime);
        int timeNow, timeCompare = 0;
        timeNow = (10000 * timeInfo->tm_hour) + (100*timeInfo->tm_min) +
timeInfo->tm_sec;

        // GET TIME FILM
        strcpy(chrWaktuFilm,
gtk_combo_box_text_get_active_text(GTK_COMBO_BOX_TEXT(comboBoxWaktu)));

        // CONVERT TIME STRING TO INT FOR COMPARISON WITH TIME NOW
        int i;
        for (i = 0; i < 5; i++){
            if (chrWaktuFilm[i] == '.'){
                continue;
            }
            else{
                int temp = chrWaktuFilm[i] - '0';
                timeCompare = timeCompare * 10 + temp;
            }
        }
        timeCompare *= 100;

        // ERROR DETECTION
        if (g_strcmp0(gNamaPembeli, "") == 0 || g_strcmp0(gTiketDibeli, "") ==
0){
            display_error_dialog("Nama atau tiket tidak boleh kosong!", window);
            gtk_widget_destroy(dialog);
```

```
        }
        else if(g_strcmp0(gTiketDibeli, "0") == 0){
            display_error_dialog("Pembelian tiket minimal satu!", window);
            gtk_widget_destroy(dialog);
        }
        else if(timeCompare < timeNow){
            display_error_dialog("Waktu pembelian tiket telah ditutup!",
window);
            gtk_widget_destroy(dialog);
        }
        else{ // NO ERROR DETECTED
            char chrTiketDibeli[5];
            char chrFilmDipilih[50];
            reset_string(chrFilmDipilih, 50);
            reset_string(chrTiketDibeli, 5);

            // CONVERT GCHAR TO CHAR
            strcpy(chrFilmDipilih,
gtk_combo_box_text_get_active_text(GTK_COMBO_BOX_TEXT(comboBox)));
            sprintf(chrTiketDibeli, "%s", gTiketDibeli);

            // CONVERT STRING TIKET TO INT
            int iTiketDibeli = 0, i = 0;
            while(chrTiketDibeli[i] != '\0'){
                int temp = chrTiketDibeli[i] - '0';
                iTiketDibeli = iTiketDibeli * 10 + temp;
                i++;
            }

            /* COUNT AVAILABLE CHAIR */
            FILE *f;
            f = fopen("data/studio_chair_data.txt", "r");
            char ch;
            int counterLine = 0, counterStudio = 0, temp = 0;
            while(!feof(f)){
                ch = fgetc(f);
                if (ch == '\n'){
                    AvailableChair[counterStudio][counterLine] = temp;
                    temp = 0;
                    counterLine++;
                    if (counterLine % 4 == 0){
                        counterLine = 0;
                        counterStudio++;
                    }
                }
                else{
                    if (ch == '0'){
                        temp++;
                    }
                }
            }
            /* END OF COUNT AVAILABLE CHAIR */

            int tempStudioFilm, tempStdTime;
            bool isDoraemon = false;

            // COMPARISON
            if (strcmp("KKN Desa Penari", chrFilmDipilih) == 0){
                tempStudioFilm = 0;
            }
            else if (strcmp("The Throne", chrFilmDipilih) == 0){
                tempStudioFilm = 1;
            }
            else if (strcmp("Ready Player One", chrFilmDipilih) == 0){
                tempStudioFilm = 2;
            }
            else{
                isDoraemon = true;
                if (strcmp("09.00-10.30", chrWaktuFilm) == 0){
                    tempStudioFilm = 2;
```

```
                    tempStdTime = 0;
                }
                else if(strcmp("10.00-11.30", chrWaktuFilm) == 0){
                    tempStudioFilm = 1;
                    tempStdTime = 1;
                }
                else if(strcmp("12.00-13.30", chrWaktuFilm) == 0){
                    tempStudioFilm = 0;
                    tempStdTime = 2;
                }
            }

            // DORAEMON PLAYS IN DIFFERENT STUDIO
            if (!isDoraemon){
                int i;
                for (i = 0; i < 4; i++){
                    if (strcmp(chrWaktuFilm, StudioTime[tempStudioFilm][i]) ==
0){
                        tempStdTime = i;
                        break;
                    }
                }
            }

            // COMPARE TICKET WITH CHAIR AVAILABLE
            if (AvailableChair[tempStudioFilm][tempStdTime] < iTiketDibeli){
                display_error_dialog("Jumlah tiket melebihi batas!", window);
                gtk_widget_destroy(dialog);
            }
            else{ // AVAILABLE
                // CREATE NEW QUEUE
                qUser = (Queue*)malloc(sizeof(Queue));

                // USERNAME DAN TIKET
                strcpy(qUser->qUsername, gNamaPembeli);
                qUser->qTiketDibeli = iTiketDibeli;

                // TIME
                time_t rawtime;
                struct tm *timeInfo;
                time(&rawtime);
                timeInfo = localtime(&rawtime);
                char chrTimeNow[100];
                reset_string(chrTimeNow, 100);
                strcpy(chrTimeNow, timeToStr(timeInfo));
                strcpy(qUser->qBuyTime, chrTimeNow);

                // STUDIO FILM
                bool isDoraemon = false;
                if (strcmp("KKN Desa Penari", chrFilmDipilih) == 0){
                    qUser->qStudioFilm = 1;
                }
                else if (strcmp("The Throne", chrFilmDipilih) == 0){
                    qUser->qStudioFilm = 2;
                }
                else if (strcmp("Ready Player One", chrFilmDipilih) == 0){
                    qUser->qStudioFilm = 3;
                }
                else{
                    isDoraemon = true;
                    if (strcmp("09.00-10.30", chrWaktuFilm) == 0){
                        qUser->qStudioFilm = 3;
                        qUser->qStudioTime = 1;
                    }
                    else if(strcmp("10.00-11.30", chrWaktuFilm) == 0){
                        qUser->qStudioFilm = 2;
                        qUser->qStudioTime = 2;
                    }
                    else if(strcmp("12.00-13.30", chrWaktuFilm) == 0){
                        qUser->qStudioFilm = 1;
```

```
                        qUser->qStudioTime = 3;
                    }
                }

                // SEARCH FOR STUDIO TIME
                int tempStudioTime  = qUser->qStudioFilm;
                tempStudioTime--;

                // DORAEMON PLAYS IN DIFFERENT STUDIO
                if (!isDoraemon){
                    int i;
                    for (i = 0; i < 4; i++){
                        if (strcmp(chrWaktuFilm, StudioTime[tempStudioTime][i])
== 0){
                            qUser->qStudioTime = (i+1);
                            break;
                        }
                    }
                }

                // COPY OTHER INFORMATION
                strcpy(qUser->qFilmName, chrFilmDipilih);
                strcpy(qUser->qFilmTime, chrWaktuFilm);
                strcpy(qUser->qFilmChair, "");
                // NEXT
                qUser->qNext = NULL;

                // ENQUEUE
                if (qFront == NULL){
                    qFront = qUser;
                }
                else{
                    Queue *prev = qFront;
                    while(prev->qNext != NULL){
                        prev = prev->qNext;
                    }
                    prev->qNext = qUser;
                }

                /* FILE OPERATION */
                FILE *f;
                f = fopen("data/user_ticket_data.txt", "a");
                fprintf(f, "%s|%d|%s|%d|%d|%s|%s|%s\n",
                        qUser->qUsername, qUser->qTiketDibeli, qUser->qBuyTime,
                        qUser->qStudioFilm, qUser->qStudioTime,
                        qUser->qFilmName, qUser->qFilmTime, qUser->qFilmChair);
                fclose(f);
                /* END OF FILE OPERATION */

                // RESET INPUT
                gtk_entry_set_text(GTK_ENTRY(entryNamaPembeli), "");
                gtk_entry_set_text(GTK_ENTRY(entryTiketDibeli), "");
                gtk_widget_destroy(dialog);

                // SUCCESS
                display_info_dialog("Pembelian berhasil", window);
            }
        }
    }
    else{
        gtk_widget_destroy(dialog);
    }
}

gboolean time_handler(){
    gchar buf[100];
    if (exitDashPembelian){
        return false;
    }
    GDateTime *now = g_date_time_new_now_local();
```

```
    gchar *my_time = g_date_time_format(now, "%H:%M:%S WITA");
    sprintf(buf, "%s", my_time);
    gtk_widget_destroy(labelTimeNow);
    labelTimeNow = gtk_label_new(buf);
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), labelTimeNow, 1000, 100);
    gtk_widget_show_all(windowDashPembelian);
    g_free(my_time);
    g_date_time_unref(now);
    return true;
}

void display_dashboard_pembelian(){
    exitDashPembelian = false;

    // TIME CALC
    time_t rawtime;
    struct tm *timeInfo;
    time(&rawtime);
    timeInfo = localtime(&rawtime);
    char chrTimeNow[50];
    reset_string(chrTimeNow, 20);
    sprintf(chrTimeNow, "%.2d:%.2d.%.2d WITA", timeInfo->tm_hour, timeInfo-
>tm_min, timeInfo->tm_sec);
    int timeNow;
    timeNow = (10000 * timeInfo->tm_hour) + (100*timeInfo->tm_min) + timeInfo-
>tm_sec;

    // TIME STATUS
    char status[10];
    reset_string(status, 10);

    // BACKGROUND
    GtkWidget *bgDashPembelian;

    // LABEL
    GtkWidget *labelTitle;
    GtkWidget *labelNamaPembeli;
    GtkWidget *labelTiketDibeli;
    GtkWidget *labelFilmDipilih;
    GtkWidget *labelWaktuDipilih;

    // BUTTON
    GtkWidget *buttonBackToDash;
    GtkWidget *buttonConfirmBeli;
    GtkWidget *buttonFilmTitle;
    GtkWidget *buttonFilmTime;

    // ICON
    GdkPixbuf *iconDashPembelian;

    gtk_init(NULL, NULL);

    // WINDOW
    windowDashPembelian = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowDashPembelian), "PEMBELIAN TIKET");
    gtk_window_set_default_size(GTK_WINDOW(windowDashPembelian), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowDashPembelian),
GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutDashPembelian = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowDashPembelian), layoutDashPembelian);

    // ICON
    iconDashPembelian = gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowDashPembelian), iconDashPembelian);

    // BACKGROUND
    bgDashPembelian = gtk_image_new_from_file("src/image/Pembelian.png");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), bgDashPembelian, 0, 0);
```

```
    // LABEL
    labelTitle = gtk_label_new("Menu Pembelian");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), labelTitle, 550, 75);
    gtk_widget_set_name(labelTitle, "labelTitle");

    labelNamaPembeli = gtk_label_new("Nama");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), labelNamaPembeli, 150, 150);

    labelTiketDibeli = gtk_label_new("Jumlah Tiket:");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), labelTiketDibeli, 150, 250);

    labelTimeNow = gtk_label_new(chrTimeNow);
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), labelTimeNow, 1000, 100);

    // ENTRY
    entryNamaPembeli = gtk_entry_new();
    gtk_widget_set_size_request(entryNamaPembeli, 250, 40);
    gtk_entry_set_placeholder_text(GTK_ENTRY(entryNamaPembeli), "Masukkan nama
pembeli...");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), entryNamaPembeli, 150, 190);

    entryTiketDibeli = gtk_entry_new();
    gtk_widget_set_size_request(entryTiketDibeli, 250, 40);
    gtk_entry_set_placeholder_text(GTK_ENTRY(entryTiketDibeli), "Masukkan jumlah
tiket...");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), entryTiketDibeli, 150, 290);

    // BUTTON
    buttonBackToDash = gtk_button_new_with_label("Dashboard");
    gtk_widget_set_name(buttonBackToDash, "buttonAction");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), buttonBackToDash, 150, 600);

    buttonConfirmBeli = gtk_button_new_with_label("Buat Pesanan");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), buttonConfirmBeli, 1000,
600);
    gtk_widget_set_name(buttonConfirmBeli, "buttonAction");

    /* FILM TIME DISPLAY */
    int i,j;
    gint x = 550, y = 200; // POSITION
    for (i =0; i < 4; i++){
        x = 550;
        for (j = 0; j < 3; j++){
            char filmTime[10];
            reset_string(filmTime, 10);
            strcpy(filmTime, FilmTime[i][j]);
            // MAKE NEW BUTTON
            buttonFilmTime = gtk_button_new_with_label(filmTime);
            gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), buttonFilmTime, x,
y);
            gtk_widget_set_size_request(buttonFilmTime, 35, 20);
            // COMPARISON TIME
            int k, compareTime = 0;
            for (k = 0; k < 5; k++){
                if (filmTime[k] == '.'){
                    continue;
                }
                else{
                    int temp = filmTime[k] - '0';
                    compareTime = compareTime * 10 + temp;
                }
            }
            compareTime *= 100;
            reset_string(status, 10);
            strcpy(status, "avail");
            if (compareTime < timeNow){
                reset_string(status, 10);
                strcpy(status, "late");
            }
```

```
            // GIVE COLOR TO BUTTON
            gtk_widget_set_name(buttonFilmTime, status);
            x += 145;
        }
        y += 100;
    }
    /* END OF FILM TIME DISPLAY */

    /* FILM NAME DISPLAY */
    y = 150;
    x = 550;
    for (i = 0; i < 4; i++){
        char filmName[50], chrTicketPrice[50];
        reset_string(filmName, 50);
        reset_string(chrTicketPrice, 50);
        strcpy(filmName, FilmName[i]);
        strcat(filmName, " - Rp. ");
        sprintf(chrTicketPrice, "%d", TicketPrice[i]);
        strcat(filmName, chrTicketPrice);
        buttonFilmTitle = gtk_button_new_with_label(filmName);
        gtk_widget_set_name(buttonFilmTitle, "namaFilm");
        gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), buttonFilmTitle, x, y);
        gtk_widget_set_size_request(buttonFilmTitle, 425, 50);
        y += 100;
    }
    /* END OF FILM NAME DISPLAY */

    labelFilmDipilih = gtk_label_new("Pilih Film:");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), labelFilmDipilih, 150, 350);

    // COMBOBOX
    comboBox = gtk_combo_box_text_new();
    for (i =0 ; i< 4; i++){
        char appendText[50];
        reset_string(appendText, 50);
        strcpy(appendText, FilmName[i]);
        gtk_combo_box_text_append_text(GTK_COMBO_BOX_TEXT(comboBox),
appendText);
    }
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian),comboBox, 150,390);
    labelWaktuDipilih = gtk_label_new("Pilih Waktu:");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian), labelWaktuDipilih, 150,
440);

    // COMBOBOX WAKTU
    comboBoxWaktu = gtk_combo_box_text_new();
    gtk_combo_box_text_append_text(GTK_COMBO_BOX_TEXT(comboBoxWaktu), "Silakan
pilih film dahulu!");
    gtk_layout_put(GTK_LAYOUT(layoutDashPembelian),comboBoxWaktu, 150,480);

    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css, "src/css/dashboard_pembelian.css",
NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    g_timeout_add(1000, (GSourceFunc)time_handler, NULL);
    gtk_widget_show_all(windowDashPembelian);

    // SIGNAL
    g_signal_connect(windowDashPembelian, "destroy", G_CALLBACK(gtk_main_quit),
NULL);
    g_signal_connect(windowDashPembelian, "destroy", G_CALLBACK(reset_window),
NULL);
```

```
    g_signal_connect(buttonBackToDash, "clicked",
G_CALLBACK(handle_display_dashboard), windowDashPembelian);
    g_signal_connect(buttonConfirmBeli, "clicked",
G_CALLBACK(display_warn_pembelian), windowDashPembelian);
    g_signal_connect(comboBox, "changed", G_CALLBACK(combo_selected), NULL);
    gtk_main();
}

void handle_display_dashboard_pembelian(GtkWidget *widget, GtkWidget *window){
    gtk_window_close(GTK_WINDOW(window));
    display_dashboard_pembelian();
}
```

dashboard_pemilihan.c

```
#include "dashboard_pembelian.h"

// LAYOUT
GtkWidget *layoutPemilihanKursi;

// LABEL
GtkWidget *servePerson;
GtkWidget *labelCountTicket;
GtkWidget *labelStudioName;
GtkWidget *labelFilmTime;
GtkWidget *labelTiketDipilih;
GtkWidget *labelEmptyData;

// BUTTON
GtkWidget *buttonAddKursi;
GtkWidget *buttonUndoKursi;
GtkWidget *buttonKursi;

// COMBO BOX
GtkWidget *comboSelectKursi;

int ChairSelected;

void get_user_ticket_from_file(){
    char compUsr[100], compBuyTime[50], compFilmName[50], compFilmTime[20],
compFilmChair[50];
    int compTicket, compStudioFilm, compStudioTime;
    reset_string(compUsr, 100);
    reset_string(compBuyTime, 50);
    reset_string(compFilmName, 50);
    reset_string(compFilmTime, 20);
    reset_string(compFilmChair, 50);

    compTicket = 0;
    compStudioFilm = 0;
    compStudioTime = 0;

    qFront = NULL;
    FILE *f;
    f = fopen("data/user_ticket_data.txt", "r");
    char ch;
    int idx = 1, counter = 0;
    while(!feof(f)){
        ch = fgetc(f);
        if (ch == '|'){
            counter = 0;
            idx++;
        }
        else if (ch == '\n'){
            qUser = malloc(sizeof(Queue));
            strcpy(qUser->qUsername, compUsr);
            qUser->qTiketDibeli = compTicket;
            strcpy(qUser->qBuyTime, compBuyTime);

            qUser->qStudioFilm = compStudioFilm;
            qUser->qStudioTime = compStudioTime;
```

```
            strcpy(qUser->qFilmName, compFilmName);
            strcpy(qUser->qFilmTime, compFilmTime);
            strcpy(qUser->qFilmChair, compFilmChair);

            qUser->qNext = NULL;
            if (qFront == NULL){
                qFront = qUser;
            }
            else{
                Queue *prev;
                prev = qFront;
                while(prev->qNext != NULL){
                    prev = prev->qNext;
                }
                prev->qNext = qUser;
            }

            // reset
            reset_string(compUsr, 100);
            reset_string(compBuyTime, 50);
            reset_string(compFilmName, 50);
            reset_string(compFilmTime, 20);
            reset_string(compFilmChair, 50);

            compTicket = 0;
            compStudioFilm = 0;
            compStudioTime = 0;

            counter = 0;
            idx = 1;
        }
        else{
            if (idx == 1){
                compUsr[counter] = ch;
                counter++;
            }
            else if(idx == 2){
                int tempCompTicket = ch - '0';
                compTicket = compTicket*10 + tempCompTicket;
                counter++;
            }
            else if (idx == 3){
                compBuyTime[counter] = ch;
                counter++;
            }
            else if (idx == 4){
                int tempCompStudioFilm = ch - '0';
                compStudioFilm = tempCompStudioFilm;
                counter++;
            }
            else if (idx == 5){
                int tempCompStudioTime = ch - '0';
                compStudioTime = tempCompStudioTime;
                counter++;
            }
            else if (idx == 6){
                compFilmName[counter] = ch;
                counter++;
            }
            else if (idx == 7){
                compFilmTime[counter] = ch;
                counter++;
            }
            else if (idx == 8){
                compFilmChair[counter] = ch;
                counter++;
            }
        }
    }
```

```
    fclose(f);
}

void write_user_ticket_to_file(){
    FILE *f;
    f = fopen("data/user_ticket_data.txt", "w");
    Queue *curr;
    curr = qFront;
    while(curr != NULL){
        fprintf(f, "%s|%d|%s|%d|%d|%s|%s|%s\n",curr->qUsername, curr-
>qTiketDibeli, curr->qBuyTime, curr->qStudioFilm, curr->qStudioTime, curr-
>qFilmName, curr->qFilmTime, curr->qFilmChair);
        curr = curr->qNext;
    }
    fclose(f);
}

void display_warn_pemilihan_kursi(GtkWidget *widget, GtkWidget *window){
    bool afterDel = false;
    if (qFront == NULL){
        display_error_dialog("Data pelanggan kosong", window);
    }
    else if(qFront->qTiketDibeli != ChairSelected){
        display_error_dialog("Jumlah tiket tidak sama", window);
    }
    else{ // NO ERROR
        afterDel = true;
        GtkWidget *dialog;
        dialog = gtk_message_dialog_new(GTK_WINDOW(window),
                GTK_DIALOG_DESTROY_WITH_PARENT,
                GTK_MESSAGE_ERROR,
                GTK_BUTTONS_YES_NO,
                "Konfirmasi Pemilihan Kursi?");
        gtk_widget_set_name(dialog, "warnConfirm");
        gtk_window_set_title(GTK_WINDOW(dialog), "Warning");
        int res = gtk_dialog_run(GTK_DIALOG(dialog));
        if (res == -8){ // YES
            char tempWriteToData[100];
            char tempFilmName[3];
            reset_string(tempWriteToData, 100);
            reset_string(tempFilmName, 3);

            // GET DATA FROM TIKET DIPILIH
            strcpy(tempWriteToData,
gtk_label_get_label(GTK_LABEL(labelTiketDipilih)));
            int i = 0, counter = 0;
            while(tempWriteToData[i] != '\0'){
                if (tempWriteToData[i] == '\n'){
                    reset_string(tempFilmName, 3);
                    counter = 0;
                }
                else{
                    tempFilmName[counter] = tempWriteToData[i];
                    counter++;
                    if (counter == 2){
                        // SEARCH FILM CHAIR
                        int j, savePos;
                        for (j = 0; j < 16; j++){
                            if (strcmp(FilmChair[j], tempFilmName) == 0){
                                savePos = j;
                                j += 16; // BREAK LOOP
                            }
                        }
                        // CHANGE ARRAY
                        StudioChair[(qFront->qStudioFilm) - 1][(qFront-
>qStudioTime - 1)][savePos] = 1;
                    }
                }
                i++;
            }
```

```
            /* FILE OPERATION STUDIO CHAIR DATA */
            FILE *f;
            f = fopen("data/studio_chair_data.txt", "w");
            int a, b, c;
            for (a = 0; a < 3; a++){
                for (b = 0; b < 4; b++){
                    for (c = 0; c < 16; c++){
                        fprintf(f, "%d", StudioChair[a][b][c]);
                    }
                    fprintf(f, "\n");
                }
            }
            fclose(f);
            /* END OF FILE OPERATION STUDIO CHAIR DATA */

            // COPY DATA TIKET TO QUEUE
            char tempChairSelected[100];
            reset_string(tempChairSelected, 100);
            Stack *now = sTop;
            while(now != NULL){
                strcat(tempChairSelected, now->sChairName);
                strcat(tempChairSelected, " ");
                now = now->sNext;
            }
            strcpy(qFront->qFilmChair, tempChairSelected);

            // WRITE TO FILE
            f = fopen("data/user_invoice_data.txt", "a");
            fprintf(f, "%s|%d|%s|%d|%d|%s|%s|%s\n",
                    qFront->qUsername, qFront->qTiketDibeli, qFront->qBuyTime,
                    qFront->qStudioFilm, qFront->qStudioTime,
                    qFront->qFilmName, qFront->qFilmTime, qFront->qFilmChair);
            fclose(f);
            // END OF WRITE TO FILE

            // DEQUEUE
            Queue *curr;
            curr = qFront;
            while(curr->qNext != NULL){
                curr = curr->qNext;
            }
            if(curr == qFront){
                qFront = NULL;
                free(curr);
                curr = NULL;
            }
            else{
                curr = qFront;
                qFront = qFront->qNext;
                free(curr);
                curr = NULL;
            }
            // WRITE AGAIN
            write_user_ticket_to_file();
            gtk_widget_destroy(dialog);
            display_info_dialog("Pemilihan Kursi Berhasil!", window);
        }
        else{
            gtk_widget_destroy(dialog);
        }
    }
    if(qFront == NULL){
        gtk_label_set_text(GTK_LABEL(labelCountTicket), "Data kosong");
        gtk_label_set_text(GTK_LABEL(servePerson), "Data kosong");
        gtk_label_set_text(GTK_LABEL(labelStudioName), "Data kosong");
        gtk_label_set_text(GTK_LABEL(labelFilmTime), "Data kosong");
        gtk_label_set_text(GTK_LABEL(labelEmptyData), "Data kosong");
        if (afterDel){
            gtk_window_close(GTK_WINDOW(window));
```

```
            display_dashboard_pemilihan_kursi();
        }
    }
    else{
        gtk_widget_destroy(labelTiketDipilih);

        // REMOVE FROM STACK
        while(sTop != NULL){
            Stack *curr = sTop;
            sTop= sTop->sNext;
            free(curr);
            curr = NULL;
        }
        free(sTop);
        sTop = NULL;

        // CHANGE DISPLAY
        char temp[3];
        reset_string(temp, 3);
        sprintf(temp, "%d", qFront->qTiketDibeli);
        gtk_label_set_text(GTK_LABEL(labelCountTicket), temp);
        gtk_label_set_text(GTK_LABEL(servePerson), qFront->qUsername);
        char tempStudioFilm[5];
        reset_string(tempStudioFilm, 5);
        sprintf(tempStudioFilm, "%d", qFront->qStudioFilm);
        gtk_label_set_text(GTK_LABEL(labelStudioName),tempStudioFilm);
        gtk_label_set_text(GTK_LABEL(labelFilmTime), qFront->qFilmTime);

        // RESET COMBO BOX
        gtk_combo_box_text_remove_all(GTK_COMBO_BOX_TEXT(comboSelectKursi));

        // ADD DATA TO COMBO BOX
        int counter;
        int i, j, delpos;
        // COMBO BOX
        for (i = 0; i < 16; i++){
            char temp[5];
            reset_string(temp, 5);
            strcpy(temp, FilmChair[i]);
            gtk_combo_box_text_append_text(GTK_COMBO_BOX_TEXT(comboSelectKursi),
temp);
        }

        // READ STUDIO CHAIR FROM FILE
        FILE *f;
        f = fopen("data/studio_chair_data.txt", "r");
        char ch;
        int studio = 0, waktu = 0, kursi = 0;
        while(!feof(f)){
            ch = fgetc(f);
            if (ch == '\n' && waktu == 3){
                studio++;
                waktu = 0;
                kursi = 0;
            }
            else if (ch == '\n' && waktu < 3){
                waktu++;
                kursi = 0;
            }
            else{
                int temp = ch - '0';
                StudioChair[studio][waktu][kursi] = temp;
                kursi++;
            }
        }
        fclose(f);

        // DELETE COMBO BOX
        counter = 0;
        delpos = counter;
```

```
        gint x = 400, y = 260;
        for (i = 0; i < 4; i++){
            x = 400;
            for (j = 0; j < 4; j++){
                char copyCounter[3];
                strcpy(copyCounter, FilmChair[counter]);
                buttonKursi = gtk_button_new_with_label(copyCounter);
                if (StudioChair[(qFront->qStudioFilm - 1)][(qFront->qStudioTime
- 1)][counter] != 0){
                    gtk_widget_set_name(buttonKursi, "late");
                    // DELETE FROM COMBO BOX

gtk_combo_box_text_remove(GTK_COMBO_BOX_TEXT(comboSelectKursi), delpos);
                }
                else{
                    // add to stack
                    sChair = (Stack*)malloc(sizeof(Stack));
                    strcpy(sChair->sChairName, FilmChair[counter]);
                    sChair->sNext = NULL;
                    if (sTopChair == NULL){
                        sTopChair = sChair;
                    }
                    else{
                        Stack *prev = sTopChair;
                        while(prev->sNext != NULL){
                            prev = prev->sNext;
                        }
                        prev->sNext = sChair;
                    }
                    delpos++;
                }
                counter++;

                gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), buttonKursi, x,
y);
                gtk_widget_set_size_request(buttonKursi, 50,50);
                x += 100;
            }
            y += 100;
        }
        gtk_widget_show_all(window);
        gtk_window_close(GTK_WINDOW(window));
        display_dashboard_pemilihan_kursi();
    }
}

void add_kursi(GtkWidget *widget, GtkWidget *window){
    if (sTopChair == NULL){
        display_error_dialog("Kursi telah habis dipesan", window);
    }
    else{
        // GET INPUT
        char text[10];
        reset_string(text, 10);
        strcpy(text,
gtk_combo_box_text_get_active_text(GTK_COMBO_BOX_TEXT(comboSelectKursi)));

        // SEARCH POSITION TO REMOVE IN COMBOBOX
        Stack *curr;
        curr = sTopChair;
        int count = -1;
        while(curr != NULL){
            count++;
            if (strcmp(curr->sChairName, text) == 0){
                if (sTopChair->sNext == NULL){
                    sTopChair = NULL;
                    free(curr);
                    curr = NULL;
                }
                else{
```

```
                    if (curr == sTopChair){
                        sTopChair = sTopChair->sNext;
                        free(curr);
                        curr = NULL;
                    }
                    else{
                        Stack *prev;
                        prev = sTopChair;
                        while(prev->sNext != curr){
                            prev = prev->sNext;
                        }
                        prev->sNext = curr->sNext;
                        curr->sNext = NULL;
                        free(curr);
                        curr = NULL;
                    }
                }
                break;
            }
            curr = curr->sNext;
        }
        gtk_combo_box_text_remove(GTK_COMBO_BOX_TEXT(comboSelectKursi), count);

        // ADD TO STACK
        ChairSelected++;
        sChair = (Stack*)malloc(sizeof(Stack));
        strcpy(sChair->sChairName, text);
        sChair->sNext = NULL;
        if (sTop == NULL){
            sTop = sChair;
        }
        else{
            Stack *prev;
            prev = sTop;
            while(prev->sNext != NULL){
                prev = prev->sNext;
            }
            prev->sNext = sChair;
        }

        // DISPLAY STACK
        Stack *now;
        now = sTop;
        char textDisplay[100];
        reset_string(textDisplay, 100);
        while(now != NULL){
            strcat(textDisplay, now->sChairName);
            strcat(textDisplay, "\n");
            now = now->sNext;
        }
        gtk_widget_destroy(GTK_WIDGET(labelTiketDipilih));
        labelTiketDipilih = gtk_label_new(textDisplay);
        gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelTiketDipilih, 800,
200);
        gtk_widget_show_all(window);
    }
}

void undo_kursi(GtkWidget *widget, GtkWidget *window){
    char text[10];
    reset_string(text, 10);
    ChairSelected--;
    if (sTop == NULL){
        display_error_dialog("Error belum ada dipilih!", window);
    }
    else{
        // POP FROM STACK
        Stack *curr = sTop, *prev = sTop;
        if (sTop->sNext == NULL){
            strcpy(text, sTop->sChairName);
```

```
            free(sTop);
            sTop = NULL;
        }
        else{
            while(curr->sNext != NULL){
            curr = curr->sNext;
            }
            while(prev->sNext != curr){
                prev = prev->sNext;
            }
            prev->sNext = NULL;
            strcpy(text, curr->sChairName);
            free(curr);
            curr = NULL;
        }

        // DISPLAY STACK
        char textDisplay[100];
        reset_string(textDisplay, 100);
        Stack *now;
        now = sTop;
        while(now != NULL){
            strcat(textDisplay, now->sChairName);
            strcat(textDisplay, "\n");
            now = now->sNext;
        }
        gtk_widget_destroy(GTK_WIDGET(labelTiketDipilih));
        labelTiketDipilih = gtk_label_new(textDisplay);
        gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelTiketDipilih, 800,
200);
        gtk_widget_show_all(window);

        // ADD TO STACK CHAIR
        sChair = (Stack*)malloc(sizeof(Stack));
        strcpy(sChair->sChairName, text);
        sChair->sNext = NULL;

        if(sTopChair == NULL){
            sTopChair = sChair;
        }
        else{
            Stack *prev;
            prev = sTopChair;
            while(prev->sNext != NULL){
                prev = prev->sNext;
            }
            prev->sNext = sChair;
        }

        // SORT STACK CHAIR
        bool isSorted = false;
        while(!isSorted){
            isSorted = true;
            Stack *curr = sTopChair;
            while (curr->sNext != NULL){
                if (strcmp(curr->sChairName, curr->sNext->sChairName) > 0){
                    isSorted = false;
                    Stack *temp;
                    temp = (Stack*)malloc(sizeof(Stack));
                    strcpy(temp->sChairName, curr->sChairName);
                    strcpy(curr->sChairName, curr->sNext->sChairName);
                    strcpy(curr->sNext->sChairName, temp->sChairName);
                    free(temp);
                    temp = NULL;
                }
                curr = curr->sNext;
            }
        }

        // SEARCH POSITION
```

```
        int counter = -1;
        curr = sTopChair;
        while(curr != NULL){
            counter++;
            if(strcmp(curr->sChairName, text) == 0){
                break;
            }
            curr = curr->sNext;
        }

        // ADD TO COMBOBOX AT RIGHT POSITION
        gtk_combo_box_text_insert_text(GTK_COMBO_BOX_TEXT(comboSelectKursi),
counter, text);
    }
}

void display_dashboard_pemilihan_kursi(){
    get_user_ticket_from_file();

    sTopChair = NULL;
    sTop = NULL;
    int counter = 0;
    ChairSelected = 0;

    GtkWidget *windowPemilihanKursi;
    GtkWidget *bgPemilihanKursi;

    GtkWidget *labelTitle;
    GtkWidget *labelServePeople;
    GtkWidget *labelTicketAvail;
    GtkWidget *labelNoStudio;
    GtkWidget *labelTiketSementara;

    // BUTTON
    GtkWidget *buttonBackToDash;
    GtkWidget *buttonConfirmPilih;
    GtkWidget *buttonScreenArea;

    // ICON
    GdkPixbuf *iconPemilihanKursi;

    gtk_init(NULL, NULL);

    // WINDOW
    windowPemilihanKursi = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(windowPemilihanKursi), "Pemilihan Kursi");
    gtk_window_set_default_size(GTK_WINDOW(windowPemilihanKursi), 1280, 720);
    gtk_window_set_position(GTK_WINDOW(windowPemilihanKursi),
GTK_WIN_POS_CENTER);

    // LAYOUT
    layoutPemilihanKursi = gtk_layout_new(NULL, NULL);
    gtk_container_add(GTK_CONTAINER (windowPemilihanKursi),
layoutPemilihanKursi);

    // ICON
    iconPemilihanKursi= gdk_pixbuf_new_from_file("src/image/icon.png", NULL);
    gtk_window_set_icon(GTK_WINDOW(windowPemilihanKursi), iconPemilihanKursi);

    // BACKGROUND
    bgPemilihanKursi = gtk_image_new_from_file("src/image/Pembelian.png");
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), bgPemilihanKursi, 0, 0);

    labelTiketDipilih = gtk_label_new("");

    labelTiketSementara = gtk_label_new("Tiket yang Sudah Dipilih:");
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelTiketSementara, 800,
150);

    labelTitle = gtk_label_new("Menu Pemilihan Kursi");
```

```
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelTitle, 500, 75);
    gtk_widget_set_name(labelTitle, "labelTitle");

    labelServePeople = gtk_label_new("Pelanggan:");
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelServePeople, 150,
200);

    if(qFront == NULL){
        labelEmptyData = gtk_label_new("Data kosong");
        servePerson = gtk_label_new("Data kosong");
        labelCountTicket = gtk_label_new("Data kosong");
        labelStudioName = gtk_label_new("Data kosong");
        labelFilmTime = gtk_label_new("Data kosong");
    }
    else{
        labelEmptyData = gtk_label_new("");
        char temp[3];
        reset_string(temp, 3);

        sprintf(temp, "%d", qFront->qTiketDibeli);
        labelCountTicket = gtk_label_new(temp);
        servePerson = gtk_label_new(qFront->qUsername);

        char tempStudioFilm[5];
        reset_string(tempStudioFilm, 5);
        sprintf(tempStudioFilm, "%d", qFront->qStudioFilm);
        labelStudioName = gtk_label_new(tempStudioFilm);
        labelFilmTime = gtk_label_new(qFront->qFilmTime);

        // BUTTON
        buttonAddKursi = gtk_button_new_with_label("Tambahkan");
        gtk_widget_set_name(buttonAddKursi, "buttonAction");
        gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), buttonAddKursi, 150,
500);

        buttonUndoKursi = gtk_button_new_with_label("Batalkan");
        gtk_widget_set_name(buttonUndoKursi, "buttonAction");
        gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), buttonUndoKursi, 300,
500);

        buttonScreenArea = gtk_button_new_with_label("Screen Area");
        gtk_widget_set_name(buttonScreenArea, "screenArea");
        gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), buttonScreenArea, 500,
500);
        gtk_widget_set_size_request(buttonScreenArea, 300, 10);

        int i, j, delpos;
        // COMBO BOX
        comboSelectKursi = gtk_combo_box_text_new();
        for (i = 0; i < 16; i++){
            char temp[5];
            reset_string(temp, 5);
            strcpy(temp, FilmChair[i]);
            gtk_combo_box_text_append_text(GTK_COMBO_BOX_TEXT(comboSelectKursi),
temp);
        }
        gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), comboSelectKursi, 150,
400);
        gtk_widget_set_name(comboSelectKursi, "comboKursi");
        gtk_widget_set_size_request(comboSelectKursi, 80, 30);

        // READ DATA FROM STUDIO CHAIR DATA
        FILE *f;
        f = fopen("data/studio_chair_data.txt", "r");
        char ch;
        int studio = 0, waktu = 0, kursi = 0;
        while(!feof(f)){
            ch = fgetc(f);
            if (ch == '\n' && waktu == 3){
                studio++;
```

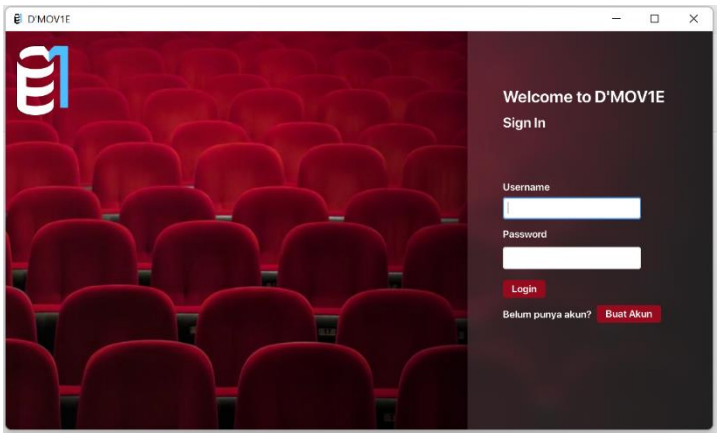
```
                waktu = 0;
                kursi = 0;
            }
            else if (ch == '\n' && waktu < 3){
                waktu++;
                kursi = 0;
            }
            else{
                int temp = ch - '0';
                StudioChair[studio][waktu][kursi] = temp;
                kursi++;
            }
        }
        fclose(f);
        // END OF READ DATA FROM STUDIO CHAIR DATA

        // COMBOBOX CREATION
        counter = 0;
        delpos = counter;
        gint x = 500, y = 210;
        for (i = 0; i < 4; i++){
            x = 500;
            for (j = 0; j < 4; j++){
                char copyCounter[3];
                strcpy(copyCounter, FilmChair[counter]);
                buttonKursi = gtk_button_new_with_label(copyCounter);
                if (StudioChair[(qFront->qStudioFilm - 1)][(qFront->qStudioTime
- 1)][counter] != 0){
                    gtk_widget_set_name(buttonKursi, "late");
                    // DON'T ADD TO COMBOBOX

gtk_combo_box_text_remove(GTK_COMBO_BOX_TEXT(comboSelectKursi), delpos);
                }
                else{
                    gtk_widget_set_name(buttonKursi, "avail");
                    // ADD TO STACK CHAIR
                    sChair = (Stack*)malloc(sizeof(Stack));
                    strcpy(sChair->sChairName, FilmChair[counter]);
                    sChair->sNext = NULL;
                    if (sTopChair == NULL){
                        sTopChair = sChair;
                    }
                    else{
                        Stack *prev = sTopChair;
                        while(prev->sNext != NULL){
                            prev = prev->sNext;
                        }
                        prev->sNext = sChair;
                    }
                    delpos++;
                }
                counter++;
                gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), buttonKursi, x,
y);
                gtk_widget_set_size_request(buttonKursi, 50,50);
                x += 70;
            }
            y += 70;
        }
        // ADD AND REMOVE CHAIR SIGNAL
        g_signal_connect(buttonAddKursi, "clicked", G_CALLBACK(add_kursi),
windowPemilihanKursi);
        g_signal_connect(buttonUndoKursi, "clicked", G_CALLBACK(undo_kursi),
windowPemilihanKursi);
    }
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), servePerson,300, 200);
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelCountTicket, 300,
150);
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelStudioName, 230, 250);
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelFilmTime, 350, 250);
```

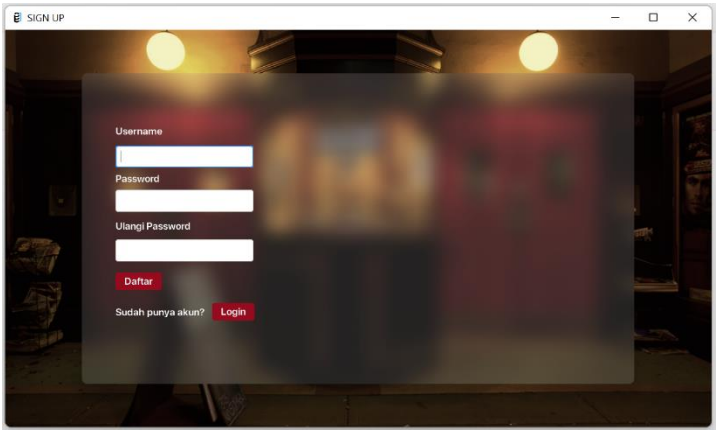
```
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelEmptyData, 800, 200);

    // LABEL
    labelTicketAvail = gtk_label_new("Jumlah Tiket: ");
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelTicketAvail, 150,
150);

    labelNoStudio = gtk_label_new("Studio: ");
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), labelNoStudio, 150 ,250);

    // BUTTON
    buttonBackToDash = gtk_button_new_with_label("Dashboard");
    gtk_widget_set_name(buttonBackToDash, "buttonAction");
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi  ), buttonBackToDash, 150,
600);

    buttonConfirmPilih = gtk_button_new_with_label("Pilih Kursi");
    gtk_widget_set_name(buttonConfirmPilih, "buttonAction");
    gtk_layout_put(GTK_LAYOUT(layoutPemilihanKursi), buttonConfirmPilih, 1000,
600);

    // CSS
    GdkDisplay *display;
    display = gdk_display_get_default();
    GdkScreen *screen;
    screen = gdk_display_get_default_screen(display);
    GtkCssProvider *css = gtk_css_provider_new();
    gtk_css_provider_load_from_path(css, "src/css/dashboard_pemilihan.css",
NULL);
    gtk_style_context_add_provider_for_screen(screen, GTK_STYLE_PROVIDER(css),
GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);

    // DISPLAY
    gtk_widget_show_all(windowPemilihanKursi);

    // SIGNAL
    g_signal_connect(windowPemilihanKursi, "destroy", G_CALLBACK(gtk_main_quit),
NULL);
    g_signal_connect(buttonBackToDash, "clicked",
G_CALLBACK(handle_display_dashboard), windowPemilihanKursi);
    g_signal_connect(buttonConfirmPilih, "clicked",
G_CALLBACK(display_warn_pemilihan_kursi), windowPemilihanKursi);

    gtk_main();
}
void handle_display_dashboard_pemilihan_kursi(GtkWidget *widget,  GtkWidget
*window){
    gtk_window_close(GTK_WINDOW(window));
    display_dashboard_pemilihan_kursi();
}
```

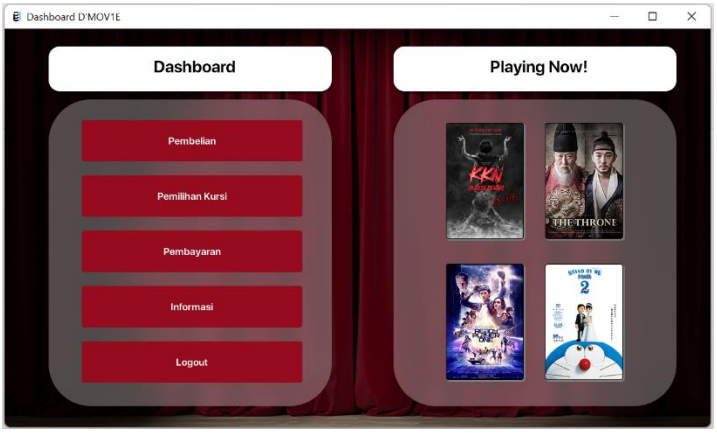
Informasi lengkap mengenai aplikasi dan source codenya dapat dilihat melalui tautan: https://github.com/putuwaw/d-mov1e

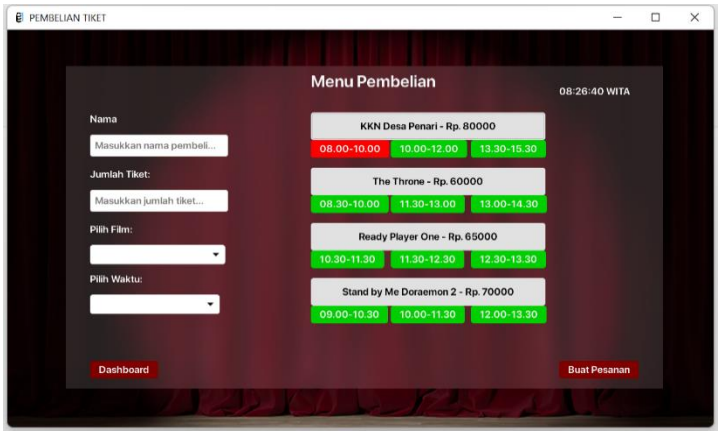
3.2 Hasil Capture

Berikut adalah hasil *capture* atau *screenshot* dari program dengan beberapa uji coba inputan:

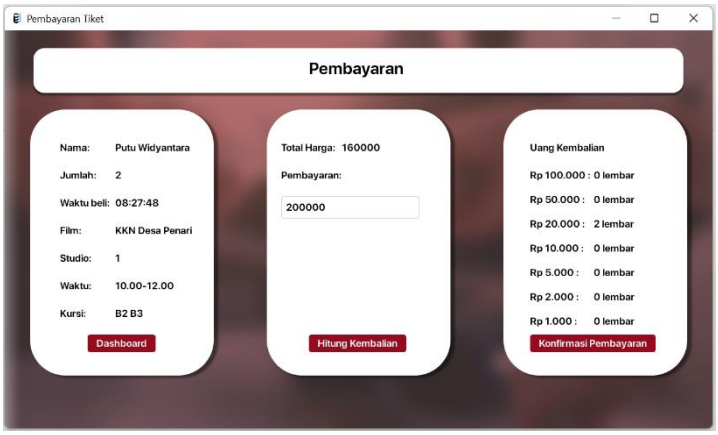


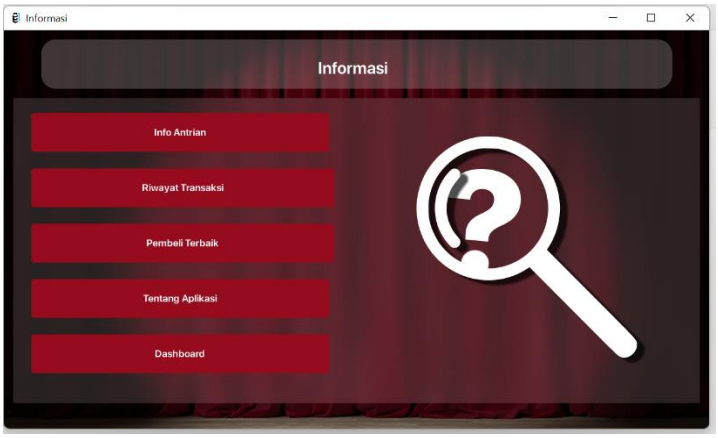


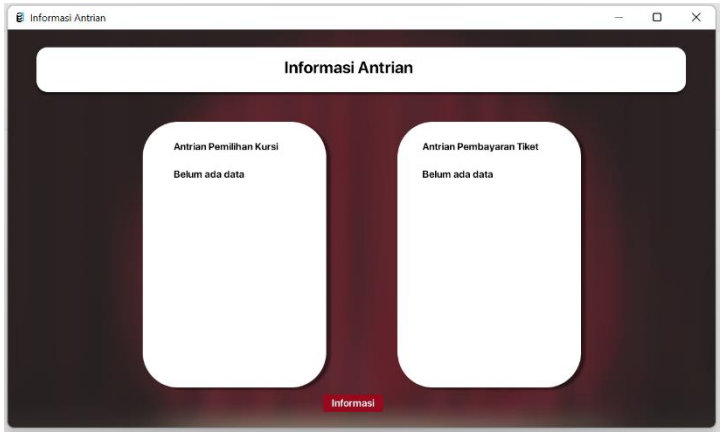


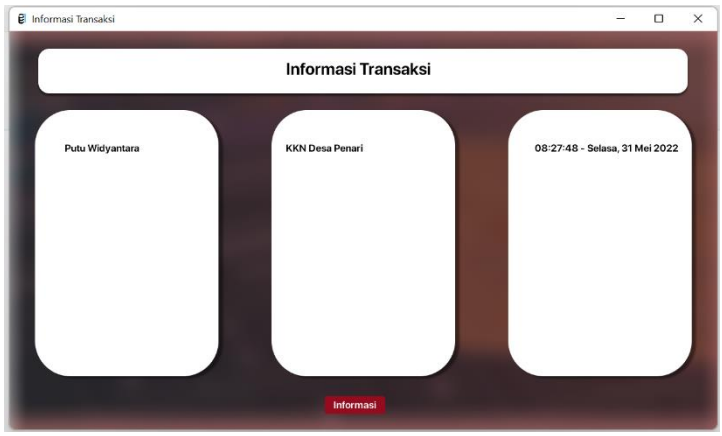

PEMBELIAN TIKET
Menu Pembelian
08:26:40 WITA
Nama
Masukkan nama pembeli...
Jumlah Tiket:
Masukkan jumlah tiket...
Pilih Film:
Pilih Waktu:
KKN Desa Penari - Rp. 80000
08.00-10.00
10.00-12.00
13.30-15.30
The Throne - Rp. 60000
08.30-10.00
11.30-13.00
13.00-14.30
Ready Player One - Rp. 65000
10.30-11.30
11.30-12.30
12.30-13.30
Stand by Me Doraemon 2 - Rp. 70000
09.00-10.30
10.00-11.30
12.00-13.30
Dashboard
Buat Pesanan

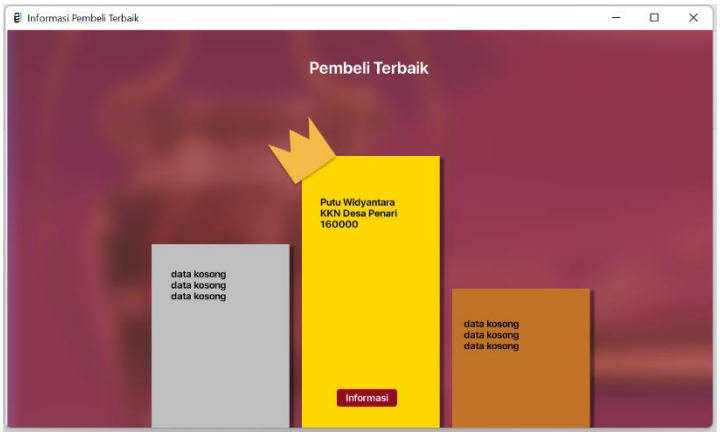

Pemilihan Kursi
Menu Pemilihan Kursi
Jumlah Tiket: 2
Pelanggan: Putu Widyantara
Studio: 1
10.00-12.00
Tiket yang Sudah Dipilih:
B2
B3
A1
A2
A3
A4
B1
B2
B3
B4
C1
C2
C3
C4
D1
D2
D3
D4
Tambahkan
Batalkan
Screen Area
Dashboard
Pilih Kursi


Pembayaran Tiket
Pembayaran
Nama: Putu Widyantara
Jumlah: 2
Waktu beli: 08:27:48
Film: KKN Desa Penari
Studio: 1
Waktu: 10.00-12.00
Kursi: B2 B3
Dashboard
Total Harga: 160000
Pembayaran:
200000
Hitung Kembalian
Uang Kembalian
Rp 100.000 : 0 lembar
Rp 50.000 : 0 lembar
Rp 20.000 : 2 lembar
Rp 10.000 : 0 lembar
Rp 5.000 : 0 lembar
Rp 2.000 : 0 lembar
Rp 1.000 : 0 lembar
Konfirmasi Pembayaran


Informasi
Informasi
Info Antrian
Riwayat Transaksi
Pembeli Terbaik
Tentang Aplikasi
Dashboard


Informasi Antrian
Informasi Antrian
Antrian Pemilihan Kursi
Belum ada data
Antrian Pembayaran Tiket
Belum ada data
Informasi


Informasi Transaksi
Informasi Transaksi
Putu Widyantara
KKN Desa Penari
08:27:48 - Selasa, 31 Mei 2022
Informasi


Informasi Pembeli Terbaik
Pembeli Terbaik
Putu Widyantara
KKN Desa Penari
160000
data kosong
data kosong
data kosong
data kosong
data kosong
data kosong
Informasi


Informasi Aplikasi
D'MOV1E adalah aplikasi manajemen tiket bioskop berbasis desktop yang mampu menangani pembelian, pemilihan, dan juga pembayaran tiket. Selain itu juga memiliki fitur untuk melihat antrian pembeli. D'MOV1E juga dapat melihat pembeli terbaik dan melihat riwayat transaksi. D'MOV1E dibuat dalam bahasa pemrograman C dengan UX library GTK.
Informasi
Putu Widyantara Artanta Wibawa
(2108561005)
Gede Krisnawa Sandhya Wandhana
(2108561017)
Putu Putri Pratiwi
(2108561010)

BAB IV

PENUTUP

4.1 Kesimpulan

Dari hasil praktikum tugas akhir yang sudah dikerjakan dan dibuat bersama-sama dengan anggota kelompok. Kami dapat mengambil kesimpulan bahwa pembuatan tugas akhir ini menggunakan hampir seluruh materi yang digunakan disaat perkuliahan dan contoh implementasinya seperti ketika membuat sign up page dimana kita memerlukan username dan password, tentunya password tersebut harus tetap menjadi rahasia sehingga menggunakan metode hashing untuk melakukan enkripsi pada password tersebut dan ketika membuat sign in page, kita diharuskan untuk membuat source code yang berfungsi untuk mengecek kecocokan masukan username dan password dari user dengan username dan password yang telah terdaftar di program tersebut atau pada database program. selain itu, queue juga berperan penting pada program ini yang digunakan untuk eksekusi pada user yang masuk terlebih dahulu jadi dengan algoritma "*first in first out*" ketika akan dilakukannya pemilihan kursi, sedangkan untuk pemilihan kursi ini menggunakan materi stack, dimana ketika akan melakukan penghapusan pilihan yang pertama kali dihapus adalah top of stack tersebut.