Table of Contents

[**INTRODUCTION** 2](#_Toc511631949)

[**PART 1: SOURCE CODE** 2](#_Toc511631950)

[**PART II: SYSTEM DEVELOPMENT PROCESS** 30](#_Toc511631951)

[**A.** **Problem Specification** 30](#_Toc511631952)

[**B.** **Analysis** 30](#_Toc511631953)

[**C.** **Design** 30](#_Toc511631954)

[**D.** **Implementation** 30](#_Toc511631955)

[**E.** **Testing and Verification** 30](#_Toc511631956)

[**PART III: PROGRAM/USER’S MANUAL** 31](#_Toc511631957)

[**A.** **Starting the program** 31](#_Toc511631958)

[**B.** **Branch manager section** 32](#_Toc511631959)

[**C.** **Registration desk** 35](#_Toc511631960)

[**D.** **Rental desk** 37](#_Toc511631961)

Table of Figures

[Figure 1: Main menu 31](#_Toc511632363)

[Figure 2: Branch Manager Section 32](#_Toc511632364)

[Figure 3: Bug for entering a spaced string 33](#_Toc511632365)

[Figure 4: Adding a new video record 33](#_Toc511632366)

[Figure 5:Updating a video record 34](#_Toc511632367)

[Figure 6: Updating a video record 34](#_Toc511632368)

[Figure 7: Deleting a video record 35](#_Toc511632369)

[Figure 8: Registration desk section 35](#_Toc511632370)

[Figure 9: Adding a new customer record 36](#_Toc511632371)

[Figure 10: Updating an existing customer record 36](#_Toc511632372)

[Figure 11: Deleting a customer record 37](#_Toc511632373)

[Figure 12: Rental desk section 37](#_Toc511632374)

[Figure 13: Renting out a video 38](#_Toc511632375)

[Figure 14: Clearance after video return 39](#_Toc511632376)

[Figure 15: Showing all video records 39](#_Toc511632377)

[Figure 16: Showing all rented videos 39](#_Toc511632378)

# **INTRODUCTION**

The following is a report on a video rental information system developed in C language. The system is used for storing video and customer records and renting of videos to the customers.

MySQL database is used to store the data permanently whereas C has been used for retrieving and displaying the data.

# **PART 1: SOURCE CODE**

main.c file

Two global variables have been used i.e. MYSQL \*db1\_con, \*db2\_con. These are connection streams that are being used by other functions, hence need to be accessible by all of them. They are only initiliazed and modified in the main method but are available for the other methods to access them.

#include <stdio.h>

#include <stdlib.h>

#include "mysql-connector/include/mysql.h"

#include "db\_manipulation.h"

/\*

The main.c file contains the function invocations for creating databases and tables. It also has menus for assisting users in

navigation.

\*/

MYSQL \*db1\_con,\*db2\_con; //global variables for connections to databases videorental and videorentalcustomers respecively.

int main(int argc, char \*\*argv)

{

createDatabase();

db1\_con = mysql\_init(NULL);

db2\_con = mysql\_init(NULL);

createConnectionStream(db1\_con,db2\_con); //making connection to the database

createTableCustomers(db1\_con); //creates a table which holds the customer records

createTableVideos(db1\_con); //creates a table which holds the video records

createTableRented(db1\_con); //creates a table which holds the rented videos' records

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* VIDEO RENTAL INFORMATION SYSTEM \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" Welcome \n");

printf("---------------------------------------------------------------\n\n");

mainUsersMenu();

closeConnections(db1\_con,db2\_con); //closes connection to the databases

exit(0);

}

/\*

function which provides a menu for the three main users of the system i.e. branch manager, registration desk and rental desk.

Each user group has their own functions that they perform hence the need to be separated.

\*/

void mainUsersMenu()

{

int choice;

printf("Please choose which user section you would like to access\n");

printf("1. Branch Manager\n");

printf("2. Registration desk\n");

printf("3. Rental desk\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

branchManager(); //displays the menu for the operations that branch manager can perform

break;

case 2:

registrationDesk(); //displays the menu for the operations that registration desk can perform

break;

case 3:

rentalDesk(); //displays the menu for the operations that rental desk can perform

break;

default:

printf("\nPlease make a selection from the given options\n");

mainUsersMenu(); //recurse until user makes an acceptable pick

break;

}

}

/\*This function provides a menu for all operations that the branch manager performs.

The BM can add a new video, alter/change the details of existing video records or delete a video record.

\*/

void branchManager()

{

system("cls");

printf("-----------------------------------------------\n");

printf("- Branch manager -\n");

printf("-----------------------------------------------\n");

printf("Please choose an operation to perform\n");

printf("1. Add new video title\n");

printf("2. Update video record\n");

printf("3. Delete video record\n");

int choice;

scanf("%d",&choice);

switch(choice)

{

case 1:

newVideo(db1\_con); //adding a new video record

break;

case 2:

updateVideoDetails(db1\_con); //updating an existing video record

break;

case 3:

deleteVideo(db1\_con); //deleting a video record

break;

default:

printf("\nPlease make a selection from the given options\n");

branchManager(); //recurses until user makes an acceptable pick

break;

}

}

/\*This function provides customer record handling services done by the registration desk. They include

adding a new customer, updating details of an existing customer and deleting details of a customer.

\*/

void registrationDesk()

{

system("cls");

printf("-----------------------------------------------\n");

printf("- Registration desk -\n");

printf("-----------------------------------------------\n");

printf("Please choose an operation to perform\n");

printf("1. Add new customer\n");

printf("2. Update customer record\n");

printf("3. Delete customer record\n");

int choice;

scanf("%d",&choice);

switch(choice)

{

case 1:

newCustomer(db1\_con,db2\_con); //adding a new customer

break;

case 2:

updateCustomerDetails(db1\_con); //updating an existing customer record

break;

case 3:

deleteCustomer(db1\_con); //deleting a customer record

break;

default:

printf("\nPlease make a selection from the given options\n");

registrationDesk(); //recurse until user makes acceptable pick

break;

}

}

/\*This function provides for video rental services done by the rental desk. They include renting out video, returning of a video by

customer, checking a customer's rental history, showing all available video records and showing all rented video records.

The system uses a subsystem of search which is invoked during most of these operations

\*/

void rentalDesk()

{

system("cls");

printf("-----------------------------------------------\n");

printf("- Rental desk -\n");

printf("-----------------------------------------------\n");

printf("Please choose an operation to perform\n");

printf("1. Rent out video\n");

printf("2. Video return\n");

printf("3. Check rental history\n");

printf("4. Show all video records\n");

printf("5. Show all rented videos\n");

int choice;

scanf("%d",&choice);

switch(choice)

{

case 1:

rentVideo(db1\_con,db2\_con); //renting out video to customer

break;

case 2:

returnVideo(db1\_con,db2\_con); //registering return of a video rented by customer

break;

case 3:

//showRentalHistory(db2\_con); //showing a customer's rental history

break;

case 4:

showAllVideos(db1\_con); //displaying all available videos to be rented

break;

case 5:

showAllRentedVideos(db1\_con); //displaying all videos that have been rented

break;

default:

printf("\nPlease make a selection from the given options\n");

rentalDesk(); //recurse until user makes acceptable pick

break;

}

}

Create\_db.c file

#include <stdio.h>

#include <stdlib.h>

#include "mysql-connector/include/mysql.h"

#include "db\_manipulation.h"

//The first connection is for the videorental databse whereas the second one is for the videorentalcustomers database

//Failure of connection will provide an error and an immediate termination of the program execution.

void createConnectionStream(MYSQL \*db1\_con , MYSQL \*db2\_con)

{

if(mysql\_real\_connect(db1\_con, "localhost", "vicente", "vincent96", "VideoRental", 0, NULL, 0) == NULL)

{

display\_error(db1\_con);

}

if (mysql\_real\_connect(db2\_con, "localhost", "vicente", "vincent96", "VideoRentalCustomers", 0, NULL, 0) == NULL)

{

display\_error(db2\_con);

}

}

/\*function for creating two databases: videorental and videorentalcustomers.

The connection function mysql\_real\_connect is provided by mysql.h which helps in connecting to mysql server

\*/

void createDatabase()

{

MYSQL \*create\_db\_con = mysql\_init(NULL);

if(create\_db\_con == NULL)

{

fprintf(stderr, "%s\n", mysql\_error(create\_db\_con));

exit(1);

}

/\*parameters server connection stream, server host (in this case localhost),user,password, database,port,

unix socket,client flag (e.g. if executing multiple queries at once)\*/

if (mysql\_real\_connect(create\_db\_con, "localhost", "vicente", "vincent96", NULL, 0, NULL, 0) == NULL)

{

display\_error(create\_db\_con);

}

if(mysql\_query(create\_db\_con, "CREATE DATABASE IF NOT EXISTS VideoRental"))

{

display\_error(create\_db\_con);

}

if(mysql\_query(create\_db\_con, "CREATE DATABASE IF NOT EXISTS VideoRentalCustomers"))

{

display\_error(create\_db\_con);

}

mysql\_close(create\_db\_con);

}

//creates a table customers which is used to store customer records

//parameter - a connection to videorental database

void createTableCustomers(MYSQL \*con)

{

if (mysql\_query(con, "CREATE TABLE IF NOT EXISTS Customers (No INT NOT NULL AUTO\_INCREMENT , CustomerID VARCHAR(20) NOT NULL , Surname VARCHAR(25) NOT NULL, OtherNames VARCHAR(70) NOT NULL,Telephone INT(15) NOT NULL , Address VARCHAR(20) NOT NULL , PRIMARY KEY (CustomerID), UNIQUE(No)) ENGINE = InnoDB")) {

display\_error(con);

}

}

//creates a table videos which is used to store video records

//parameter - a connection to videorental database

void createTableVideos(MYSQL \*con)

{

if (mysql\_query(con, "CREATE TABLE IF NOT EXISTS Videos (No INT NOT NULL AUTO\_INCREMENT, VideoID VARCHAR(20) NOT NULL,Title VARCHAR(70) NOT NULL, Copies INT(20) NOT NULL,Type VARCHAR(10) NOT NULL , Director VARCHAR(70) NOT NULL,Actor VARCHAR(70) NOT NULL, Genre VARCHAR(20) NOT NULL, Charges INT(10) NOT NULL,PRIMARY KEY (VideoID),UNIQUE(No)) ENGINE = InnoDB")) {

display\_error(con);

}

}

//creates a table rented which is used to store rented video records

//parameter - a connection to videorental database

void createTableRented(MYSQL \*con)

{

if (mysql\_query(con, "CREATE TABLE IF NOT EXISTS Rented (No INT NOT NULL AUTO\_INCREMENT, VideoID VARCHAR(20) NOT NULL,CustomerID VARCHAR(20) NOT NULL, DateRented DATE NOT NULL , Fee INT(20) NOT NULL, Deadline DATE NOT NULL , Penalty INT(10) NOT NULL DEFAULT '0' ,PRIMARY KEY(No)) ENGINE = InnoDB")) {

display\_error(con);

}

}

//creates a table for each new customer with table name being customer id. The table then stores the rental history of the customer

//parameters - a connection to videorentalcustomers database, customer id

void createPerCustomerTable(MYSQL \*con,char \*cust\_id)

{

char s1[510] = "CREATE TABLE IF NOT EXISTS ";

char s3[410] = " (No INT NOT NULL AUTO\_INCREMENT, VideoID VARCHAR(20) NOT NULL, DateRented DATE NOT NULL, DateReturned DATE NULL, Penalty INT(10) NOT NULL DEFAULT '0', PRIMARY KEY(No)) ENGINE = InnoDB";

strcat(s1,cust\_id);

char \*query = strcat(s1,s3); //concatenates the string arrays to form a query executable by mysql.

if (mysql\_query(con, query)) {

display\_error(con);

}

}

//displays a mysql error to the terminal or console and terminates the program immediately.

void display\_error(MYSQL \*con)

{

fprintf(stderr, "%s\n", mysql\_error(con));

mysql\_close(con);

exit(1);

}

//closes connection stream to the databases

//parameters - conncetion stream to videorental database, conncetion stream to videorentalcustomers database

void closeConnections(MYSQL \*db1\_con , MYSQL \*db2\_con)

{

mysql\_close(db1\_con);

mysql\_close(db2\_con);

}

Manipulate\_db.c

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "mysql-connector/include/mysql.h"

#include "db\_manipulation.h"

//a structure to hold the customer details when adding a new customer record

typedef struct Customer

{

char surname[30];

char otherNames[70];

long telephone;

char \*customerID;

long address;

};

//a structure to hold the video details when adding a new video record

typedef struct Video

{

char title[50];

char type[15];

char director[50];

char actor[50];

char genre[25];

int copies;

int charges;

char \*videoID;

};

//a structure to hold the rented video details when adding a new rented video record

typedef struct RentedVideo

{

char videoID[15];

char customerID[15];

char charges[7];

char dateDue[20];

};

//a structure to hold the video details for the field(s) being updated

typedef struct UpdateVideo

{

char title[30];

char charges[7];

int copies;

char videoID[15];

};

//a structure to hold the customer details for the field(s) being updated

typedef struct UpdateCustomer

{

char surname[30];

char otherNames[40];

char telephone[15];

char address[15];

int copies;

char customerID[15];

};

//a structure to hold details when making a return recording of a rented video

typedef struct VideoReturn

{

char penalty[10];

char customerID[15];

char videoID[15];

char dateRented[15];

};

/\*function for adding a new customer record. If successfully added, a new table for the customer is created for storing their

rental history records

parameters - connection to videorental db, connection to videorentalcustomers db

\*/

void newCustomer(MYSQL \*db1\_con,MYSQL \*db2\_con)

{

printf("\n New Customer Record ");

printf("\n---------------------------------------\n");

struct Customer \*custPtr,cust1;

custPtr = &cust1; //initialize pointer to structure which will hold the details for the new customer during the registration process

printf("Please enter the following fields. (All fields must be filled)\n");

printf("Surname: ");

scanf ("%s",custPtr->surname);

printf("\nOtherNames: ");

scanf ("%s",custPtr->otherNames);

printf("\nTelephone: ");

scanf ("%ld",&custPtr->telephone);

printf("\nAddress: ");

scanf ("%ld",&custPtr->address);

custPtr->customerID = generateCustomerID();

printf("\nYour Customer ID = %s\n",custPtr->customerID);

//this section provides for concatenating the details and parts of sql query to form a full sql query that can be executed.

char s1[500] = "INSERT INTO Customers (CustomerID,Surname,OtherNames,Telephone,Address) VALUES('";

strcat(s1,custPtr->customerID);

strcat(s1,"', '");

strcat(s1,custPtr->surname);

strcat(s1,"', '");

strcat(s1,custPtr->otherNames);

strcat(s1,"',");

char t\_tel[15];

itoa(custPtr->telephone, t\_tel, 10);

strcat(s1,t\_tel);

strcat(s1,", ");

char t\_address[15];

itoa(custPtr->address, t\_address, 10);

strcat(s1,t\_address);

char \*query = strcat(s1,")");

if (mysql\_query(db1\_con, query)) {

display\_error(db1\_con);

}

else

{

printf("\nSucessfully added!!\n");

}

createPerCustomerTable(db2\_con,custPtr->customerID); //creates a new table with the customer id as its name

}

/\*function for adding a new video record.

parameters - connection to videorental db

\*/

void newVideo(MYSQL \*db1\_con)

{

struct Video \*vidPtr,vid1;

vidPtr = &vid1; //initialize pointer to structure which will hold the details for the new customer during the registration process

printf("\n New Video Record ");

printf("\n-------------------------------------\n");

printf("Please enter the following fields. (All fields must be filled)\n");

printf("Title: ");

scanf ("%s",vidPtr->title);

printf("Type: ");

scanf ("%s",vidPtr->type);

printf("Director: ");

scanf ("%s",vidPtr->director);

printf("Actor: ");

scanf ("%s",vidPtr->actor);

printf("Genre: ");

scanf ("%s",vidPtr->genre);

printf("Copies: ");

scanf ("%d",&vidPtr->copies);

printf("Charges: ");

scanf ("%d",&vidPtr->charges);

vidPtr->videoID = generateVideoID();

printf("\nVideo ID = %s\n",vidPtr->videoID);

//this section provides for concatenating the details and parts of sql query to form a full sql query that can be executed.

char s1[500] = "INSERT INTO Videos (VideoID,Title,Type,Copies,Director,Genre,Actor,Charges) VALUES('";

strcat(s1,vidPtr->videoID);

strcat(s1,"', '");

strcat(s1,vidPtr->title);

strcat(s1,"', '");

strcat(s1,vidPtr->type);

strcat(s1,"',");

char t\_copies[15];

itoa(vidPtr->copies, t\_copies, 10);

strcat(s1,t\_copies);

strcat(s1,", '");

strcat(s1,vidPtr->director);

strcat(s1,"', '");

strcat(s1,vidPtr->genre);

strcat(s1,"', '");

strcat(s1,vidPtr->actor);

strcat(s1,"',");

char t\_charges[15];

itoa(vidPtr->charges, t\_charges, 10);

strcat(s1,t\_charges);

char \*query = strcat(s1,")");

if (mysql\_query(db1\_con, query)) {

display\_error(db1\_con);

}

else

{

printf("\nSucessfully added!!\n");

}

}

/\*function for updating existing video record details.

parameters - connection to videorental db

\*/

void updateVideoDetails(MYSQL \*con)

{

struct UpdateVideo \*updPtr,update;

updPtr = &update;

printf("\n Update Video Record ");

printf("\n-------------------------------------\n");

searchForVideo(con);

printf("\nPlease re-enter the video ID\n");

scanf("%s",updPtr->videoID);

printf("\nPlease choose the field to update\n");

printf("1. Title\n");

printf("2. Copies\n");

printf("3. Charges\n");

int choice;

scanf("%d",&choice);

char s1[250] = "UPDATE Videos SET ";

switch(choice)

{

case 1:

printf("Please enter new title\n");

scanf("%s",updPtr->title);

strcat(s1,"Title = '");

strcat(s1,updPtr->title);

strcat(s1,"'");

break;

case 2:

printf("Please enter new charges\n");

scanf("%s",updPtr->charges);

strcat(s1,"Charges = ");

strcat(s1,updPtr->charges);

break;

case 3:

printf("Please update number of copies\n");

scanf("%d",&updPtr->copies);

strcat(s1,"Copies = ");

char t\_copies[15];

itoa(updPtr->copies, t\_copies, 10);

strcat(s1,t\_copies);

break;

default:

printf("Please choose a selection from the options given\n");

updateVideoDetails(con);

break;

}

strcat(s1," WHERE VideoID = '");

strcat(s1,updPtr->videoID);

char \*query = strcat(s1,"'");

if (mysql\_query(con, query))

{

display\_error(con);

}

else

{

printf("Video record successfully updated\n");

}

}

/\*function for updating existing customer record details.

parameters - connection to videorental db

\*/

void updateCustomerDetails(MYSQL \*con)

{

printf("\n Update Customer Record ");

printf("\n---------------------------------------\n");

struct UpdateCustomer \*updPtr,update;

updPtr = &update;

printf("Please enter the customerID\n");

scanf("%s",updPtr->customerID);

printf("Please choose the field to update\n");

printf("1. Surname\n");

printf("2. Other Names\n");

printf("3. Telephone\n");

printf("4. Address\n");

int choice;

scanf("%d",&choice);

char s1[200] = "UPDATE Customers SET ";

switch(choice)

{

case 1:

printf("Please enter new surname\n");

scanf("%s",updPtr->surname);

strcat(s1,"Surname = '");

strcat(s1,updPtr->surname);

strcat(s1,"'");

break;

case 2:

printf("Please change other name\n");

scanf("%s",updPtr->otherNames);

strcat(s1,"OtherNames = '");

strcat(s1,updPtr->surname);

strcat(s1,"'");

break;

case 3:

printf("New Telephone\n");

scanf("%s",updPtr->telephone);

strcat(s1,"Telephone = ");

strcat(s1,updPtr->telephone);

break;

case 4:

printf("New Address\n");

scanf("%s",updPtr->address);

strcat(s1,"Address = ");

strcat(s1,updPtr->address);

break;

default:

printf("Please choose a selection from the options given\n");

updateCustomerDetails(con);

break;

}

strcat(s1," WHERE CustomerID = '");

strcat(s1,updPtr->customerID);

char \*query = strcat(s1,"'");

if (mysql\_query(con, query))

{

display\_error(con);

}

else

{

printf("Customer record successfully updated\n");

}

}

void searchForVideo(MYSQL \*con)

{

char movie\_val[100];

printf("Please search using either Title, Director or Actor\n");

scanf("%s",movie\_val);

char s1[500] = "SELECT \* FROM Videos WHERE Title LIKE UPPER ('%";

strcat(s1,movie\_val);

strcat(s1,"%') OR Director LIKE UPPER ('%");

strcat(s1,movie\_val);

strcat(s1,"%') OR Actor LIKE UPPER ('%");

strcat(s1,movie\_val);

char \*query = strcat(s1,"%') ORDER BY No ASC");

if (mysql\_query(con, query))

{

display\_error(con);

}

MYSQL\_RES \*result = mysql\_store\_result(con);

if (result == NULL)

{

display\_error(con);

}

int num\_fields = mysql\_num\_fields(result);

MYSQL\_ROW row;

printf("\n----------------------------------------------------------------------------------------------------------------------------------------\n");

printf("%-15s%-15s%-15s%-15s%-15s%-15s%-15s%-15s%-15s\n","No","Video ID","Title","Copies","Type","Director","Actor","Genre","Charges");

printf("\n----------------------------------------------------------------------------------------------------------------------------------------\n");

while ((row = mysql\_fetch\_row(result)))

{

int i=0;

for(i = 0; i < num\_fields; i++)

{

printf("%-15s", row[i] ? row[i] : "NULL");

}

printf("\n");

}

printf("\n----------------------------------------------------------------------------------------------------------------------------------------\n");

mysql\_free\_result(result);

}

void searchForRentedVideo(MYSQL \*con,char \*customerID)

{

char movie\_val[100];

printf("Please search using either Title, Director or Actor\n");

scanf("%s",movie\_val);

char s1[500] = "SELECT VideoID,Title FROM Videos WHERE Title LIKE UPPER ('%";

strcat(s1,movie\_val);

strcat(s1,"%') OR Director LIKE UPPER ('%");

strcat(s1,movie\_val);

strcat(s1,"%') OR Actor LIKE UPPER ('%");

strcat(s1,movie\_val);

char \*query = strcat(s1,"%') ORDER BY No ASC");

if (mysql\_query(con, query))

{

display\_error(con);

}

MYSQL\_RES \*result = mysql\_store\_result(con);

if (result == NULL)

{

display\_error(con);

}

MYSQL\_ROW row = mysql\_fetch\_row(result);

char \*title = row[1];

char s2[200] = "SELECT VideoID, DateRented,Penalty FROM Rented WHERE CustomerID = '";

strcat(s2,customerID);

strcat(s2,"' AND VideoID = '");

strcat(s2,row[0]);

char \*query2 = strcat(s2,"'");

if (mysql\_query(con, query2))

{

display\_error(con);

}

MYSQL\_RES \*result2 = mysql\_store\_result(con);

if (result2 == NULL)

{

display\_error(con);

}

int num\_fields2 = mysql\_num\_fields(result2);

MYSQL\_ROW row2 = mysql\_fetch\_row(result2);

printf("\n-------------------------------------------------------------------\n");

printf("%-15s%-15s%-15s%-15s","Video ID","Title","Date Rented","Penalty");

printf("\n-------------------------------------------------------------------\n");

printf("%-15s%-15s%-15s%-15s",row2[0],title,row2[1],row2[2]);

printf("\n-------------------------------------------------------------------\n");

mysql\_free\_result(result);

mysql\_free\_result(result2);

}

void deleteVideo(MYSQL \*con)

{

printf("\n Delete Video Record ");

printf("\n-------------------------------------\n");

char vid\_id[15];

printf("Please use the video ID to do the deletion\n");

searchForVideo(con);

printf("Please enter the video ID for video to be deleted\n");

scanf("%s",vid\_id);

char s1[150] = "DELETE FROM Videos WHERE VideoID = '";

strcat(s1,vid\_id);

char \*query = strcat(s1,"'");

if (mysql\_query(con, query))

{

display\_error(con);

}

else

{

printf("Video successfully deleted\n");

}

}

void deleteCustomer(MYSQL \*con)

{

printf("\n Delete Customer Record ");

printf("\n---------------------------------------\n");

char cust\_id[15];

printf("Please enter the customer ID\n");

scanf("%s",cust\_id);

char s1[150] = "DELETE FROM Customers WHERE CustomerID = '";

strcat(s1,cust\_id);

char \*query = strcat(s1,"'");

if (mysql\_query(con, query))

{

display\_error(con);

}

else

{

printf("Video successfully deleted\n");

}

}

void rentVideo(MYSQL \*con,MYSQL \*con2)

{

printf("\n Rent Video ");

printf("\n-------------------------------------\n");

struct RentedVideo \*rentedPtr,rented1;

rentedPtr = &rented1;

printf("Please enter customer id \n");

scanf("%s",rentedPtr->customerID);

char s1[100] = "SELECT SUM(Penalty) AS total\_penalty FROM Rented WHERE CustomerID = '";

strcat(s1,rentedPtr->customerID);

char \*query = strcat(s1,"'");

if (mysql\_query(con, query))

{

display\_error(con);

}

MYSQL\_RES \*result = mysql\_store\_result(con);

if (result == NULL)

{

display\_error(con);

}

int num\_fields = mysql\_num\_fields(result);

MYSQL\_ROW row = mysql\_fetch\_row(result);

int j = atoi(row[0]);

if(j>0)

{

printf("You have a pending penalty of: %d\n",j);

printf("Do you still want to proceed with the rental?[y/n]\n");

char cho;

scanf("%s",&cho);

switch(cho)

{

case 'y':

case 'Y':

break;

case 'n':

case 'N':

printf("Thank you.\n");

exit(0);

break;

default:

system("cls");

mainUsersMenu();

break;

}

}

mysql\_free\_result(result);

searchForVideo(con);

printf("Please enter the following details:\n");

printf("Video ID\n");

scanf("%s",rentedPtr->videoID);

printf("Fee: ");

scanf ("%s",rentedPtr->charges);

printf("Deadline (yyyy-mm-dd)\n");

scanf("%s",rentedPtr->dateDue);

char s2[500] = "INSERT INTO Rented (VideoID,CustomerID,DateRented,Fee,Deadline) VALUES('";

strcat(s2,rentedPtr->videoID);

strcat(s2,"','");

strcat(s2,rentedPtr->customerID);

strcat(s2,"',NOW(),");

strcat(s2,rentedPtr->charges);

strcat(s2,",'");

strcat(s2,rentedPtr->dateDue);

char \*query2 = strcat(s2,"')");

//printf("%s",query2);

if (mysql\_query(con, query2))

{

display\_error(con);

}

char s3[100] = "UPDATE Videos SET Copies = Copies - 1 WHERE VideoID = '";

strcat(s3,rentedPtr->videoID);

char \*query3 = strcat(s3,"'");

if (mysql\_query(con, query3))

{

display\_error(con);

}

char s4[200] = "INSERT INTO ";

strcat(s4,rentedPtr->customerID);

strcat(s4," (VideoID,DateRented) VALUES('");

strcat(s4,rentedPtr->videoID);

char \*query4 = strcat(s4,"',NOW())");

if (mysql\_query(con2, query4))

{

display\_error(con2);

}

}

void returnVideo(MYSQL \*con,MYSQL \*con2)

{

printf("\n Return Rented Video ");

printf("\n-------------------------------------\n");

int amountPayed;

struct VideoReturn \*penlPtr,penalty1;

penlPtr = &penalty1;

printf("Please enter the customer ID\n");

scanf("%s",penlPtr->customerID);

searchForRentedVideo(con, penlPtr->customerID);

printf("\nPlease enter the video ID and the date rented (yyyy-mm-dd)\n");

scanf("%s%s",penlPtr->videoID,penlPtr->dateRented);

char s1[100] = "SELECT Penalty FROM Rented WHERE CustomerID = '";

strcat(s1,penlPtr->customerID);

char \*query = strcat(s1,"'");

if (mysql\_query(con, query))

{

display\_error(con);

}

MYSQL\_RES \*result = mysql\_store\_result(con);

if (result == NULL)

{

display\_error(con);

}

int num\_fields = mysql\_num\_fields(result);

MYSQL\_ROW row = mysql\_fetch\_row(result);

int j = atoi(row[0]);

if(j>0)

{

printf("There is a penalty of %d\n",j);

doPenalization(j);

}

printf("You have been cleared\n");

char s2[300] = "UPDATE ";

strcat(s2,penlPtr->customerID);

strcat(s2," SET DateReturned = NOW(), Penalty = ");

strcat(s2,row[0]);

strcat(s2," WHERE VideoID = '");

strcat(s2,penlPtr->videoID);

strcat(s2,"' AND DateRented = '");

strcat(s2,penlPtr->dateRented);

char \*query2 = strcat(s2,"'");

if (mysql\_query(con2, query2))

{

display\_error(con2);

}

char s3[200] = "UPDATE Videos SET Copies = Copies + 1 WHERE VideoID = '";

strcat(s3,penlPtr->videoID);

char \*query3 = strcat(s3,"'");

if (mysql\_query(con, query3))

{

display\_error(con);

}

char s4[200] = "DELETE FROM Rented WHERE CustomerID = '";

strcat(s4,penlPtr->customerID);

strcat(s4,"' AND VideoID = '");

strcat(s4,penlPtr->videoID);

char \*query4 = strcat(s4,"'");

if (mysql\_query(con, query4))

{

display\_error(con);

}

else

{

printf("\nCLEARANCE SUCCSSFUL\n");

}

}

void doPenalization(int penalty)

{

int amountPayed;

printf("Please enter payment: \n");

scanf("%d",&amountPayed);

int balance = amountPayed - penalty;

if(balance>0)

{

printf("Your balance is %d\n",balance);

}

else if(balance<0)

{

int left = penalty-amountPayed;

printf("Please add %d more.\n",left);

doPenalization(left);

}

}

void showAllVideos(MYSQL \*con)

{

printf("\n Display all Video Records ");

printf("\n---------------------------------------\n");

if (mysql\_query(con, "SELECT \* FROM Videos ORDER BY No ASC"))

{

display\_error(con);

}

MYSQL\_RES \*result = mysql\_store\_result(con);

if (result == NULL)

{

display\_error(con);

}

int num\_fields = mysql\_num\_fields(result);

MYSQL\_ROW row;

printf("\n----------------------------------------------------------------------------------------------------------------------------------------\n");

printf("%-15s%-15s%-15s%-15s%-15s%-15s%-15s%-15s%-15s\n","No","Video ID","Title","Copies","Type","Director","Actor","Genre","Charges");

printf("\n----------------------------------------------------------------------------------------------------------------------------------------\n");

while ((row = mysql\_fetch\_row(result)))

{

int i=0;

for(i = 0; i < num\_fields; i++)

{

printf("%-15s", row[i] ? row[i] : "NULL");

}

printf("\n");

}

printf("\n----------------------------------------------------------------------------------------------------------------------------------------\n");

mysql\_free\_result(result);

}

//function for displaying all the videos that have been rented

void showAllRentedVideos(MYSQL \*con)

{

printf("\n Display Rented Video Records ");

printf("\n-----------------------------------------\n");

if (mysql\_query(con, "SELECT v.Title,c.Surname, r.DateRented,r.Fee,r.Deadline,r.Penalty FROM Rented r LEFT JOIN Videos v ON r.VideoId = v.VideoId LEFT JOIN Customers c ON r.CustomerID = c.CustomerID"))

{

display\_error(con);

}

MYSQL\_RES \*result = mysql\_store\_result(con); //gets result from the query and stores them as a queryset

if (result == NULL)

{

display\_error(con);

}

int num\_fields = mysql\_num\_fields(result); //counts the number of columns for the queryset

MYSQL\_ROW row;

printf("\n-----------------------------------------------------------------------------------------------------------\n");

printf("%-15s%-15s%-15s%-15s%-15s%-15s%-15s\n","No","Title","Surname","Date Rented","Charges","Deadline","Penalty");

printf("\n-----------------------------------------------------------------------------------------------------------\n");

int j=1;

//loops to get data per row and display it on the console in the specified format

while ((row = mysql\_fetch\_row(result)))

{

int i=0;

printf("%-15d",j);

for(i = 0; i < num\_fields; i++)

{

printf("%-15s", row[i] ? row[i] : "NULL");

}

printf("\n");

j++;

}

printf("\n-----------------------------------------------------------------------------------------------------------\n");

mysql\_free\_result(result); //frees the result set to avoid memory leaks

}

/\*void showRentalHistory(MYSQL \*con)

{

char customerID[15];

printf("Please enter customer ID\n");

scanf("%s",customerID);

char s1[100] = "SELECT v.Title,c.DateRented, c.DateReturned, c.Penalty FROM ";

strcat(s1,customerID);

char \*query = strcat(s1," c LEFT JOIN Videos v ON v.VideoID = c.VideoID");

if (mysql\_query(con, query))

{

display\_error(con);

}

MYSQL\_RES \*result = mysql\_store\_result(con);

if (result == NULL)

{

display\_error(con);

}

int num\_fields = mysql\_num\_fields(result);

MYSQL\_ROW row;

printf("\n-----------------------------------------------------------------------------------------------------------\n");

printf("%-15s%-15s%-15s%-15s%-15s\n","No","Title","Date Rented","Date Returned","Penalty Paid");

printf("\n-----------------------------------------------------------------------------------------------------------\n");

int j=1;

while ((row = mysql\_fetch\_row(result)))

{

int i=0;

printf("%-15d",j);

for(i = 0; i < num\_fields; i++)

{

printf("%-15s", row[i] ? row[i] : "NULL");

}

printf("\n");

j++;

}

printf("\n-----------------------------------------------------------------------------------------------------------\n");

mysql\_free\_result(result);

}\*/

//function for randomly generating indices for the array whose corresponding elements are combined to form a video id

char \*generateCustomerID()

{

//using time as seed for random number generator

srand(time(0));

//array contains some characters used to develop the customer id

char charArray[54] = {'A','x','1','u','B','2','C','a','7','D','E','b','3','F','G','h','H','4','5','I','l','J','K',

'c','L','5','f','M','9','N','z','O','P','w','Q','0','m','R','S','j','1','T','U','4','8',

'V','d','W','e','X','2','Y','4','Z'};

char newId[7];

int i;

//generates a random index between 0 and 54 and retrieve the corresponding element. Then the element is copied in a new array

for (i = 0; i < 7; i++) {

int num = (rand() + 0) % (54 + 1);

newId[i] = charArray[num];

}

static char id[12];

memset(id, '\0', sizeof(id));

strncpy(id, newId, 7);

//returns an array which contains the customer id

return id;

}

//function for randomly generating indices for the array whose corresponding elements are combined to form a video id

char \*generateVideoID()

{

//using time as seed for random number generator

srand(time(0));

//array contains some characters used to develop the video id

char charArray[54] = {'A','x','1','u','B','2','C','a','7','D','E','b','3','F','G','h','H','4','5','I','l','J','K',

'c','L','5','f','M','9','N','z','O','P','w','Q','0','m','R','S','j','1','T','U','4','8',

'V','d','W','e','X','2','Y','4','Z'};

char newId[8];

int i;

//generates a random index between 0 and 54 and retrieve the corresponding element. Then the element is copied in a new array

for (i = 1; i < 7; i++) {

int num = (rand() + 0) % (54 + 1);

newId[i] = charArray[num];

}

newId[0] = 'V';

static char id[12];

memset(id, '\0', sizeof(id));

strncpy(id, newId, 7);

//returns an array which contains the video id

return id;

}

Db\_manipulation.h file

#ifndef DB\_MANIPULATION\_H\_INCLUDED

#define DB\_MANIPULATION\_H\_INCLUDED

#include "mysql-connector/include/mysql.h"

//provides for external invocations of the functions in different .c files

void createDatabase();

void createTableCustomers(MYSQL \*con);

void display\_error(MYSQL \*con);

void createConnectionStream(MYSQL \*db1\_con , MYSQL \*db2\_con);

void closeConnections(MYSQL \*db1\_con , MYSQL \*db2\_con);

char \*generateCustomerID();

char \*generateVideoID();

void newCustomer(MYSQL \*db1\_con,MYSQL \*db2\_con);

void newVideo(MYSQL \*db1\_con);

void searchForVideo(MYSQL \*con);

void rentVideo(MYSQL \*con,MYSQL \*con2);

void updateVideoDetails(MYSQL \*con);

void deleteVideo(MYSQL \*con);

void updateCustomerDetails(MYSQL \*con);

void deleteCustomer(MYSQL \*con);

void doPenalization(int penalty);

void returnVideo(MYSQL \*con,MYSQL \*con2);

void searchForRentedVideo(MYSQL \*con,char \*customerID);

void showAllVideos(MYSQL \*con);

//void showRentalHistory(MYSQL \*con);

#endif // DB\_MANIPULATION\_H\_INCLUDED

# **PART II: SYSTEM DEVELOPMENT PROCESS**

## **Problem Specification**

## **Analysis**

## **Design**

## **Implementation**

## **Testing and Verification**

# **PART III: PROGRAM/USER’S MANUAL**

The following is a simple user manual of the video rental information system.

## **Starting the program**

Run the ‘video rental information system.exe’ file by either right clicking on it and choosing ‘Open’ from the context menu or by double clicking it. A screen which looks like this will appear.

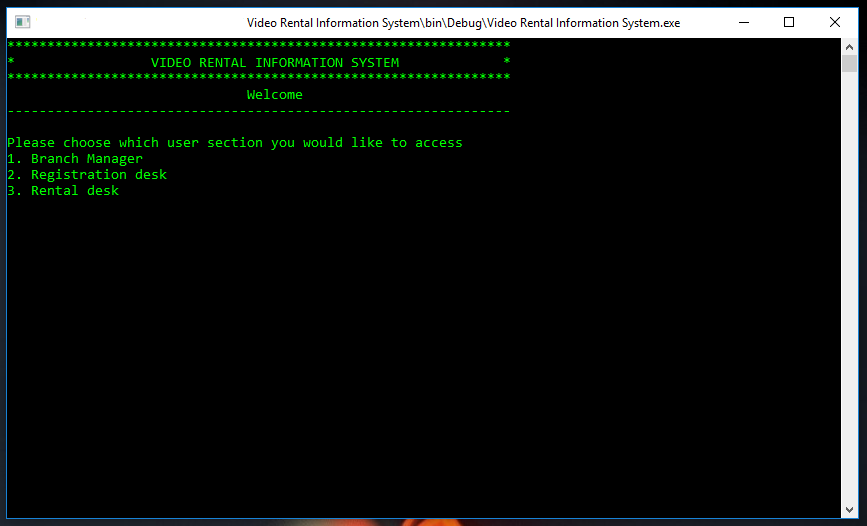


Figure 1: Main menu

A menu also appears which contains three sections i.e.

1. **Branch manager** – can do the following:
2. Add a new video title
3. Update the current video title by doing alterations
4. Delete a video record
5. **Registration desk** – can do the following:
6. Add a new customer
7. Update the record details of an existing customer
8. Delete records of a customer
9. **Rental desk** – can do the following:
10. Rent a video to a customer
11. Register a return of a rented video from a customer (also does penalization)
12. Check a customer’s rental history
13. Display all the available video records
14. Display all the rented video records

A cursor appears at the bottom which will awaits for input of your choice e.g. you can input 1 to enter the branch manager section.

Note: Entering any value that is not given as an option will prompt the system to re-display the menu asking you to enter the acceptable pick.

## **Branch manager section**

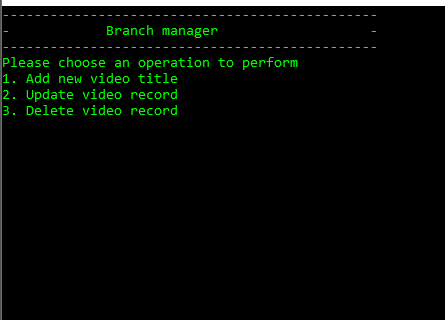


Figure 2: Branch Manager Section

1. **Add new video title**

This section is for adding a new video record. Once you are here, you are required to enter all the fields as indicated. Be careful not to put any spaced strings e.g. ‘The moon’. This causes the program to skip entry of some values e.g.

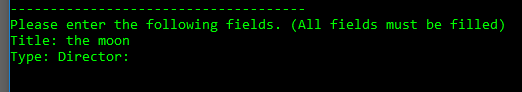


Figure 3: Bug for entering a spaced string

From here it is seen that the field Type has been skipped by the program.

After each entry, press Enter key to go to the next field and once completed press Enter key to submit the data to the database e.g.

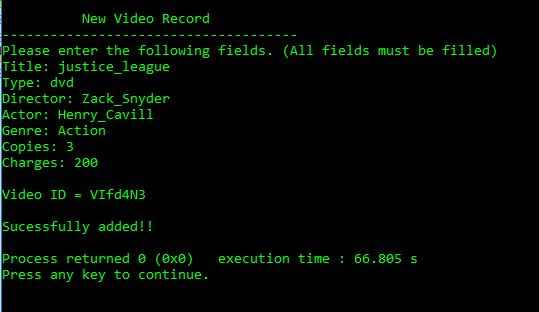


Figure 4: Adding a new video record

1. **Update video record**

This section provides for making changes to an existing video record. To do an update, the video id is required. However, since one knows the title of the video, one can search for the video using the title or part of it to get the video id e.g.

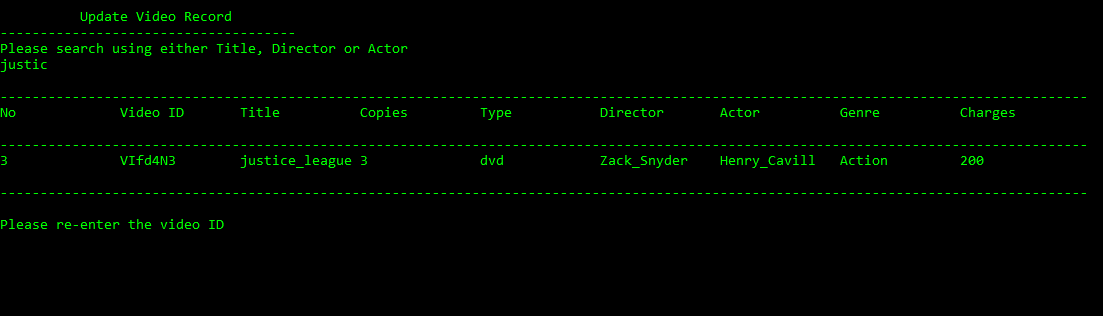


Figure 5:Updating a video record

Once you know the video id, you are required to re-enter it and then a menu for the field you wish to update will appear as follows

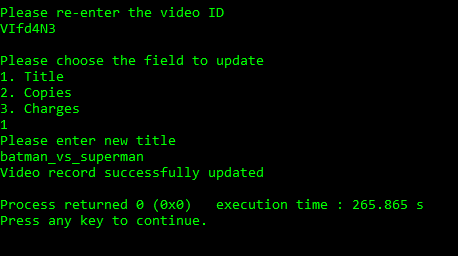


Figure 6: Updating a video record

1. **Delete video record**

This section is for deleting a video record from the database.

Also here, a search is done in order to obtain the video id which is then used in the deletion e.g.



Figure 7: Deleting a video record

## **Registration desk**

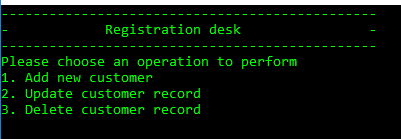


Figure 8: Registration desk section

1. **Add a new customer**

The following section provides for adding a new customer record. Simply enter the required fields while pressing the Enter key to move to the next key and then press Enter key to submit e.g.

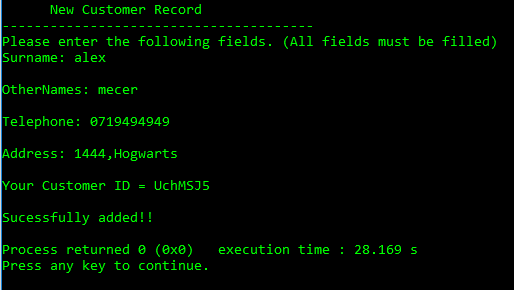


Figure 9: Adding a new customer record

Note: Like in ‘add new video title’ section, please do not input spaced strings

1. **Update customer record**

This section provides for editing and making changes to existing customer records.

Enter the customer id and then choose the field you wish to update.

Then fill the field will the new value and press Enter key to submit to the database e.g.

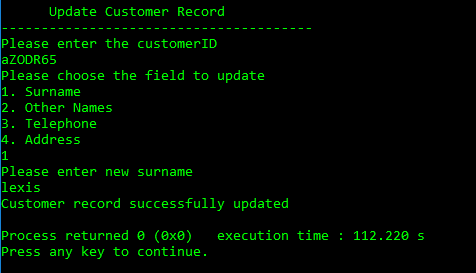


Figure 10: Updating an existing customer record

1. **Delete customer record**

This section provides for deletion of an existing customer record.

Enter the customer id number and then press Enter key to finish.

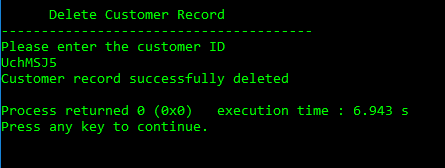


Figure 11: Deleting a customer record

## **Rental desk**

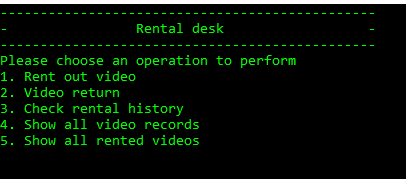


Figure 12: Rental desk section

1. **Rent out video**

This section provides for renting out a video title to a customer.

First, enter the customer id.

Then enter the video title or part of it to retrieve its records.

Enter the video id, fee and due date.

Press Enter key to do submission.

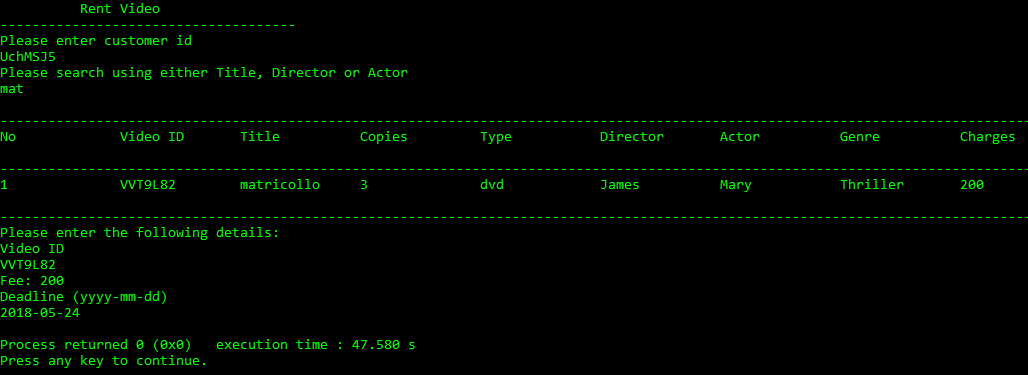


Figure 13: Renting out a video

1. **Video return**

This section provides for clearing a customer when they return a video they had rented before. If a penalty is available due to late submission, they will be required to pay it first before being cleared.

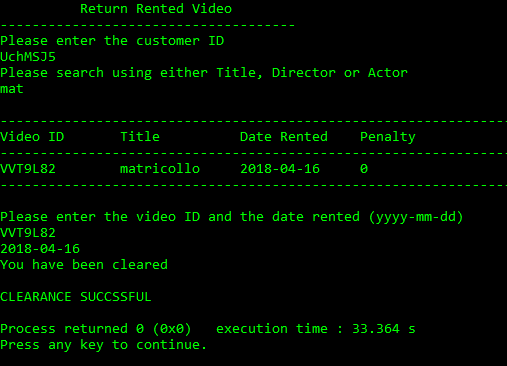


Figure 14: Clearance after video return

1. **Check rental history**

This section is for checking the rental history of a customer. All that is required is to enter the customer id.

1. **Show all video records**

This section provides for viewing all the video records in the system e.g.



Figure 15: Showing all video records

1. **Show all rented videos**

This section provides for viewing all rented video records e.g.

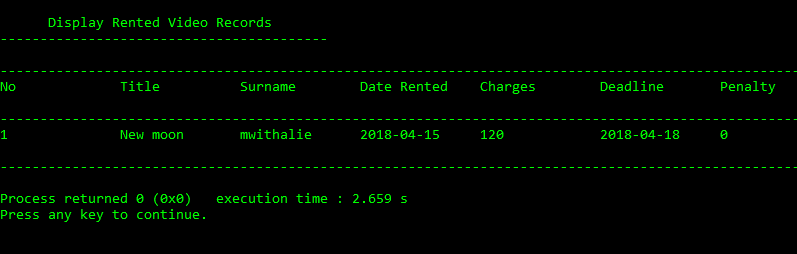


Figure 16: Showing all rented videos