*Team Cinema*

**Project Design**

Miles Bell, Char Williams, Tremaine Powell, Malcolm Gilbert, Ling Lan, Wilson Ebo

CMSC 495 (7980)

4/12/23

**Revision History**

| Revision Number | Date | Description | Name |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

**Design**

ADMINS: Create an application that keeps track of movie inventory. Keeps track of customer information and their rented movies, fees, and any overdue movies. Additionally new movies can be added to the current inventory. Users and Movies can be deleted and modified by Admin accounts.

USERS: Users are able to create an account, browse the movie inventory, rent and return rented films. Users can update their personal information (Email, Card info, phone number).

**Description**

The video rental system will be coded with Java and UI functionality will be created through Swing. The movie inventory will be stored within a SQL database that is connected to the Java code. Upon starting the application a User will be prompted to login to their account. They may either login, or register for an account. After a successful login or registration the user is sent to the homepage. Here there are the following options: View Movies, Rent Movie, Return Movie, Checkout, and Exit.

View Movies allows a user to browse all movie in the database

Rent Movies allows a user to rent an available movie within the database

Return Movie allows a user to return a rented movie associated with their account

Checkout allows a user to checkout their cart of movies to be rented

Exit allows a user to exit the application

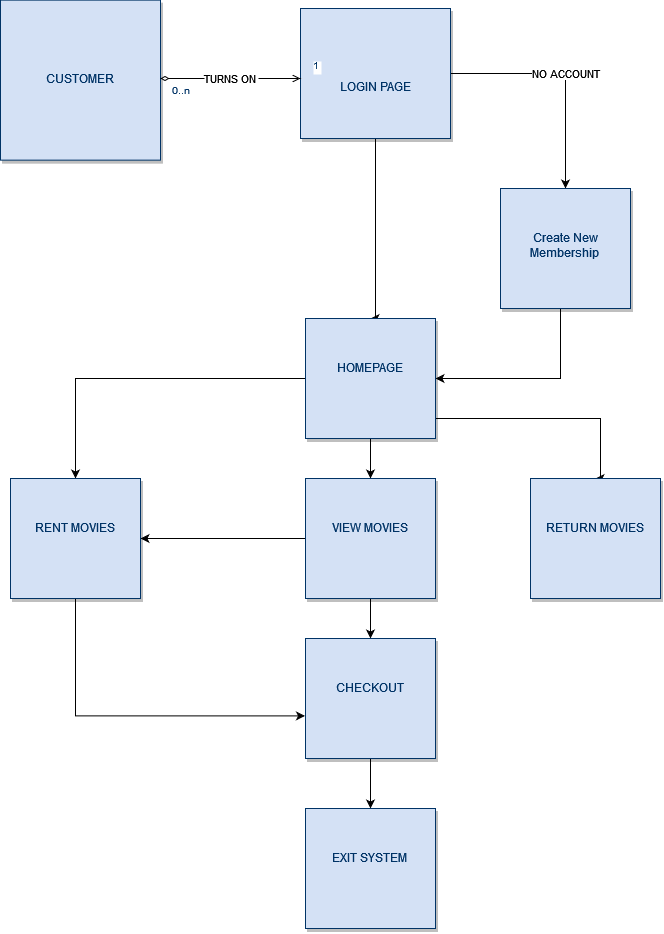
Inputs and Outputs will be handled through the UI.

Text fields will be used for logging in and out as well as inputting user information.

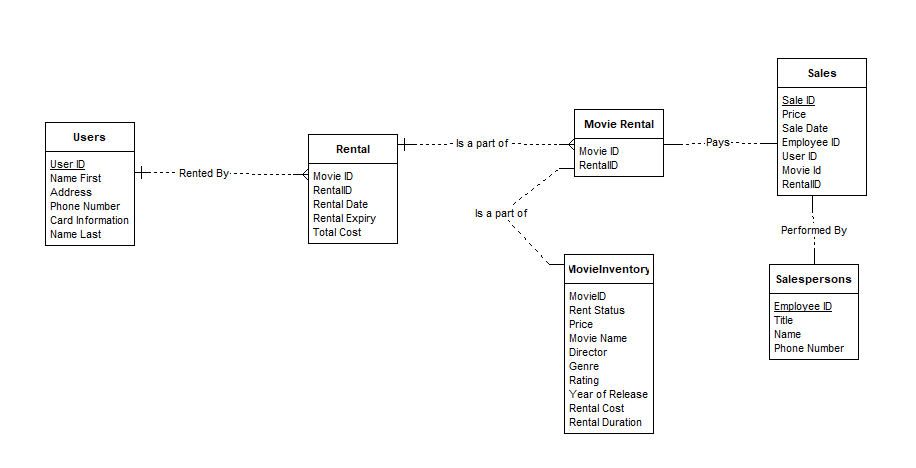
Buttons will be used for selecting the various windows and selecting movies

**Project Design Diagram**

Project Flowchart 1



Rental System Database EID

****

**Pseudocode**

ConnectionProvider

Public Class ConnectionProvider;

Public static Connection getcon;

Class.forName(“Insert JDBC Driver”);

Connection con=DriverManager.getconnection(“Insert Host Information”);

Creating Tables

>Create database movieDB

>use movieDB

>Create Table Users (Insert user variables)

User Registering

actionListener.Register(){

login.Display();

GUIRegister register = new GUIRegister ();

register.display();

actionListener.register{

int dataValidate = validate(register.data);

if (dataValidate==0){

User newUser = new User(register.data);

myUsers.add(newUser);

JOptionPane.("New User Added");

else if (dataValidate ==1){

JOptionPane.("Invalid inputs");

}

else {

JOptionPane.("user already in database");

}

};

actionListener.cancel{

System.exit(0);

};

}

}

}

**Deliverables**

JAVA Classes

ConnectionProvider.Java

-provides connection to the SQL database used

Login.Java

-launches the login window. Inputs are handled by JTextfields

Homepage.Java

-launches the homepage window.

NewUser.Java

-Launches registration window. Inputs are handled by JTextfields

ViewMovies.Java

-Launches the movie list from the database

ReturnMovie.Java

-Launches the return movie window

Checkout.Java

-Launches Checkout window. Inputs are handled by JTextField

SQL Tables

Users

(userID,namefirst,namelast,email,phonenumber)

Movies

(movieID, title, director, genre, rating, rentPrice

Rented Movies

(movieID, rentalID, rentalDate, rentalPrice)

**Test Cases**

| TEST CASES | How Tested | Expected Results | Actual Results | Pass |
| --- | --- | --- | --- | --- |
| 1:Check Results on validating User IDs and Passwords | Login with incorrect user and password credentials |  |  |  |
| 2: Validate proper creation of new Users | Create a new user. |  |  |  |
| 3: Adding new movies to the inventory (AS ADMIN) | (As Admin) add new film to the database |  |  |  |
| 4: Updating a movies info within the DB | (As Admin) edit an existing movies information inside the DB |  |  |  |
| 5: Validate payment is processed properly | Rent a movie and pay at the checkout window |  |  |  |