MOVIE RENTAL MANAGEMENT SYSTEM

CPSC 6119

Boyapati Nithisha Reddy

Small Project

Objective:

The objective of this project is to develop a movie rental management system using the MVC compound pattern. The project can be implemented in either Java, C++ or C#. I have chosen Java with NetBeans IDE. The system is made of the following entities

- Clients: They can rent movies in the system and are stored in a file called clients.txt
- Movies: These are the objects to be rented and are stored in a file called movies.txt
- Rental_Info: This contains information about who rented which movie, date of return and date of rental. Rental_Info is stored in a file called rental_info.txt

Requirements:

The system should be able to handle the following transactions

For Clients:

- Create a client.
- Delete a client. A client does not get physically deleted from the system. A flag in the record in file just states if the client is “logically” deleted.
- Search a client by last name, first name and show its information
- Show all clients

For Movies:

- Create a movie.
- Rent a movie.
- Return a movie.
- Show all movies, showing current rented movies first.
- Show current rented movies.
- Show who has historically rented a movie, sorting in descending order by date of rental.

Design Pattern used:

As per the requirement I used MVC compound pattern in developing this project. This pattern is used in separating the concern of an application. For this pattern the code is split into three packages named Model, View and controller.

Model:

The objects carrying the data is represented by Model. It contains four model classes and for each class in a model an interface is created. Each class of model contains array list of observers. Here the classes of model acts as source and each view uses removeObserver () and registerObserver () to unregister and register with the model. All the methods of removeObserver () and registerObserver () are present in the interfaces of a model and class of a model provides implementation for all those methods.

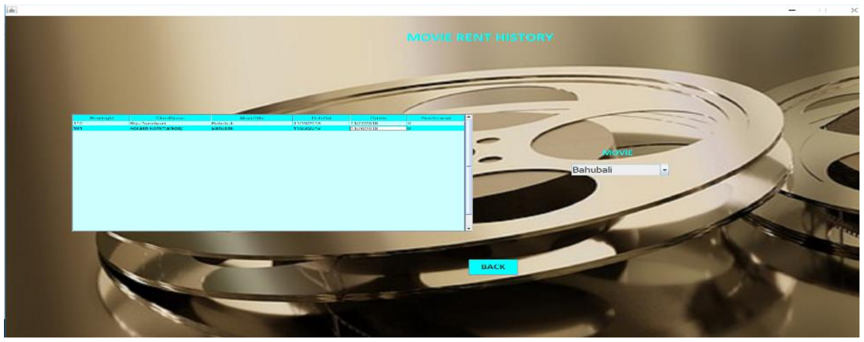
For registering a view, the classes of a model have got reference to the interfaces of view. Therefore, model contains has a relationship with the interface of view. As a part of MVC, observer patter has been used in this project development where view act as observer and model act as a source. View will be notified when there is a change in the model's state.

- ✓ Movie – This class is to maintain all the attributes related to movies, it implements IMovie and as well as provides implementation to each method contained in IMovie.
- ✓ Client – This class maintains all attributes related to clients, it implements IClient and as well as provides implementation to each method contained in IClient.
- ✓ MovieRentingInfo – This class maintains all attributes related to renting info, it implements IMovieRentingInfo and as well as provides implementation to all the methods contained in IMovieRentingInfo.
- ✓ FilePaths – This class maintains paths related to Movie, Client and Renting info, it implements IFilePaths and as well as provides implementation to each method contained in IFilePaths.

View:

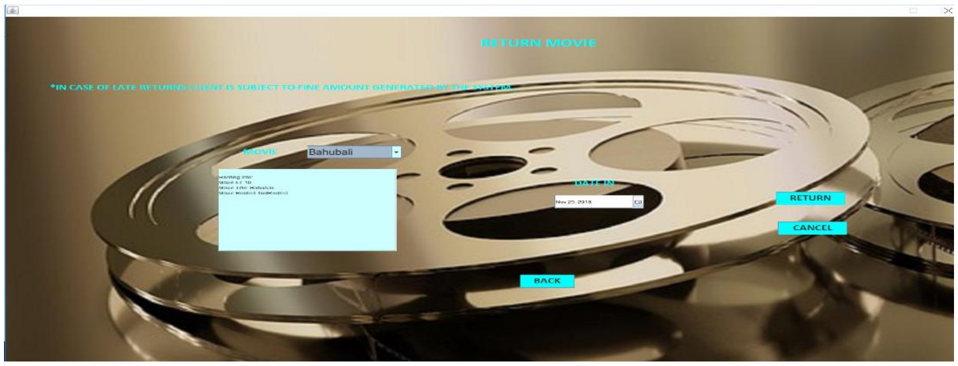
View gets registered with respective models by calling the removeObserver () and registerObserver () methods of a model and it acts as an observer as well. View contains reference to the model interface. For calling methods of a controller, it also contains interface to the controller where controller acts as an event handler. In this package user interfaces for view are created in this project.

- ✓ DisplayRentMovieHistoryPageGUI – This page is used in showing the history of the movie that is rented. Even if the client has been deleted from the system, the history of the deleted client can also be view if the client rented a movie in the past.



This page contains a dropdown button for selecting each movie and the history of that movie will be displayed in the table along with the fine amount. Implementation to all the methods contained in IDisplayRentMovieHistoryPageGUI is provided by this class.

- ✓ ReturnMoviePageGUI – This page is used to return a movie which was rented. Upon returning a movie the text file rental_info.txt will be updated. Once the movie is returned, based on the return date the fine will be calculated and the fine amount will be displayed in an alert. The fine amount will be $5 per day if the number of days is >8. Implementation to all the methods contained in IReturnMoviePageGUI is provided by this class.

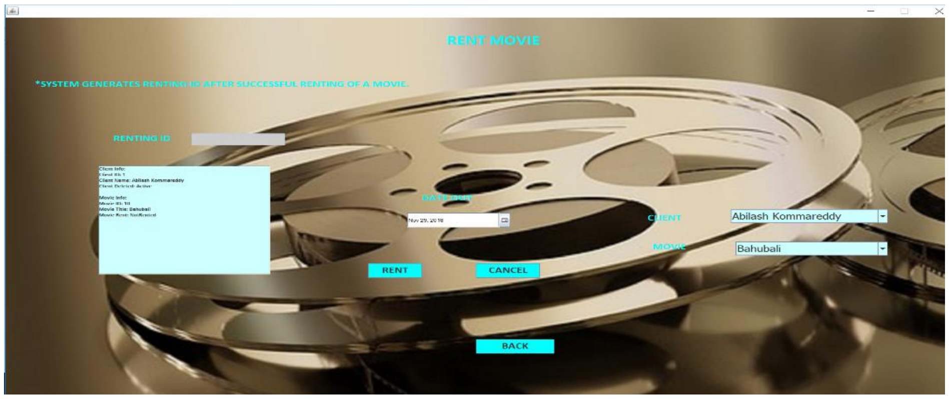

RETURN MOVIE
*IN CASE OF LATE RETURNS CLIENT IS SUBJECT TO FINE AMOUNT GENERATED BY THE SYSTEM.
MOVIE
DATE IN
RETURN
CANCEL
BACK

Bahubali

Renting Info:
Movie ID: 10
Movie Title: Bahubali
Movie Rented: NotRented

Nov 25, 2018

- ✓ RentMoviePageGUI – This page is used to rent a movie for a client. Once a movie is rented to a person the other person can not rent it unless the movie is returned. A popup appears if the same client wants to rent more than 3 movies. Implementation to all methods contained in IRentMoviePageGUI is provided by this class.


RENT MOVIE
*SYSTEM GENERATES RENTING ID AFTER SUCCESSFUL RENTING OF A MOVIE.
RENTING ID
DATE OUT
CLIENT
MOVIE
RENT
CANCEL
BACK

Client Info:
Client ID: 1
Client Name: Abilash Kommareddy
Client Deleted: Active

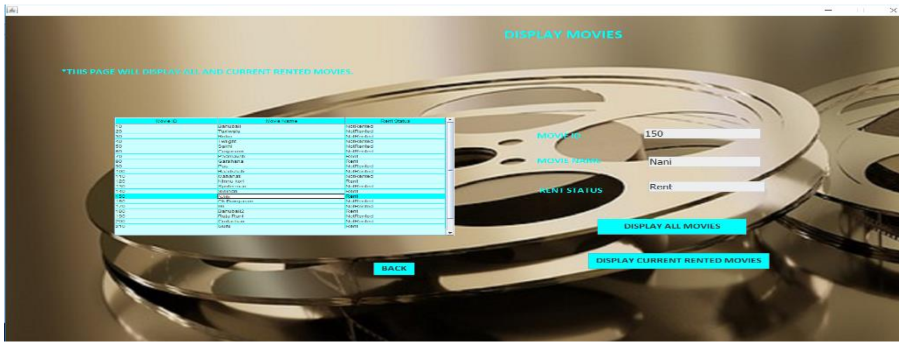
Movie Info:
Movie ID: 10
Movie Title: Bahubali
Movie Rent: NotRented

Nov 29, 2018

Abilash Kommareddy

Bahubali

- ✓ DisplayMoviesPageGUI – This page displays the list of all the movies. It contains two buttons one of which is display current rented history by clicking that button the table display only the list of rented movies. Implementation to all methods contained in IDisplayMoviesPageGUI is provided by this class.


DISPLAY MOVIES
*THIS PAGE WILL DISPLAY ALL AND CURRENT RENTED MOVIES.
MOVIE ID
MOVIE NAME
RENT STATUS
DISPLAY ALL MOVIES
DISPLAY CURRENT RENTED MOVIES
BACK

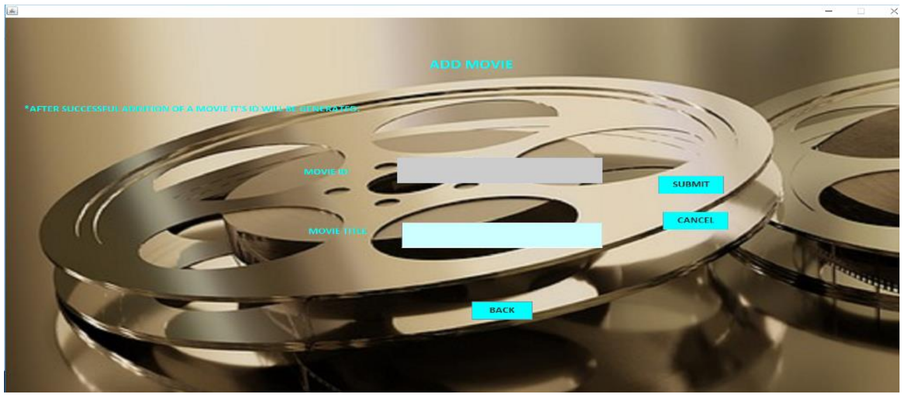
Movie ID	Movie Name	Rent Status
10	Bahubali	NotRented
20	[illegible]	NotRented
30	[illegible]	NotRented
40	Twilight	NotRented
50	Sakhi	NotRented
60	[illegible]	NotRented
70	[illegible]	Rent
80	Garshana	Rent
90	[illegible]	NotRented
100	[illegible]	NotRented
110	Mahanati	NotRented
120	Ninnu kori	Rent
130	[illegible]	NotRented
140	[illegible]	Rent
150	Nani	Rent
160	[illegible]	NotRented
170	[illegible]	NotRented
180	Bahubali2	Rent
190	Raja Rani	NotRented
200	[illegible]	NotRented
210	Guru	Rent

150

Nani

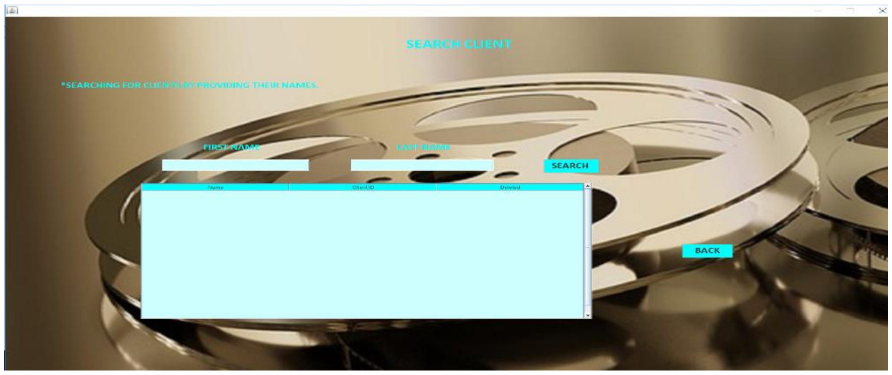
Rent

- ✓ AddMoviePageGUI – This page is used for adding a movie in the system. Once the movie is added the system generates an ID for the movie added. Implementation to all the methods contained in IAddMoviePageGUI is provided by this class.


ADD MOVIE
*AFTER SUCCESSFUL ADDITION OF A MOVIE IT'S ID WILL BE GENERATED.
MOVIE ID
MOVIE TITLE
SUBMIT
CANCEL
BACK

  This page is used in adding a new movie. A popup appears if the same movie is added saying that movie is already present in the system.

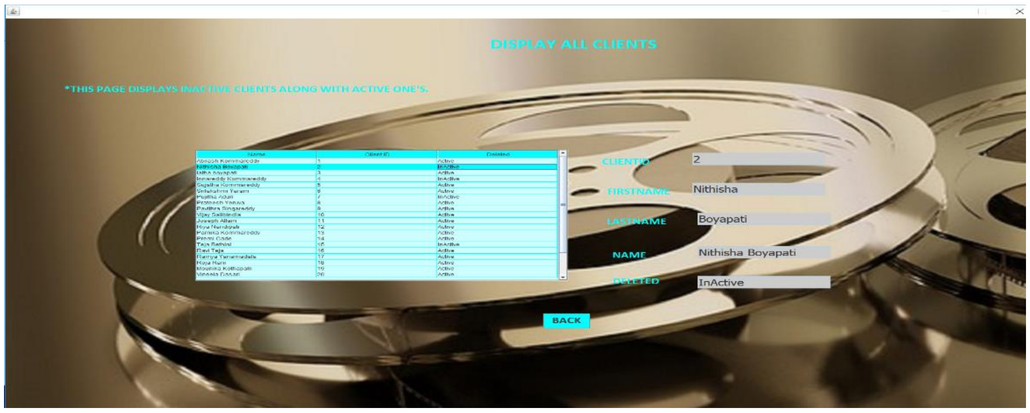
- ✓ SearchClientPageGUI – This page is used in search of a client with their names. Once the name is found it will be displayed in the table if not a popup displays saying results not found. Implementation to all the methods contained in ISearchClientPageGUI is provided by this class.



Name	ClientID	Deleted

  This page helps in finding the client easily with either their full name or first or last names.

- ✓ DisplayClientsPageGUI – This page is used for displaying the list of all active and logically deleted clients. Implementation to all methods in IDisplayClientsPageGUI is provided by this class.

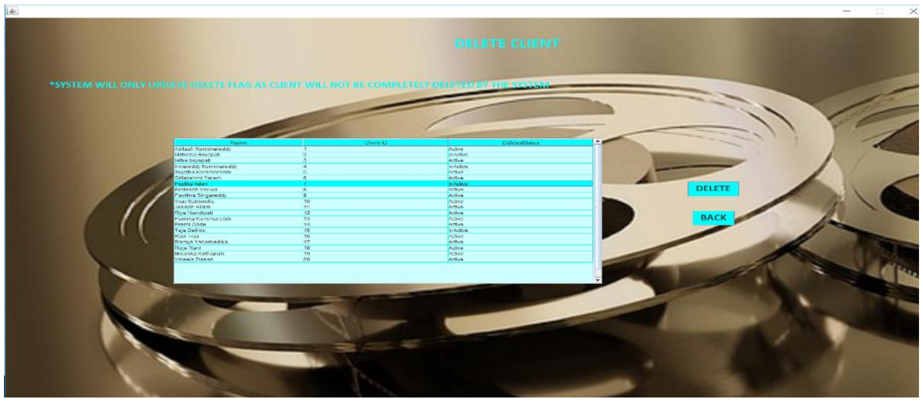

DISPLAY ALL CLIENTS
*THIS PAGE DISPLAYS INACTIVE CLIENTS ALONG WITH ACTIVE ONE'S.
BACK

Name	Client ID	Deleted
Abiesh Kommareddy	1	Active
Nithisha Boyapati	2	InActive
latha boyapati	3	Active
Innareddy Kommareddy	4	InActive
Sujatha Kommareddy	5	Active
Srilakshmi Yarram	6	Active
Pujitha Adari	7	InActive
Prateesh Yenuva	8	Active
Pavithra Singareddy	9	Active
Vijay Sallbindla	10	Active
Joseph Allam	11	Active
Riya Nandipati	12	Active
Pamika Kommareddy	13	Active
Premi Gade	14	Active
Teja Bathini	15	InActive
Ravi Teja	16	Active
Ramya Yarramadala	17	Active
Roja Nani	18	Active
Mounika Kothapalli	19	Active
Vineela Dasari	20	Active

CLIENTID: 2
FIRSTNAME: Nithisha
LASTNAME: Boyapati
NAME: Nithisha Boyapati
DELETED: InActive

This page displays all the clients irrespective of their status. Once the cursor is spotted on a name all the details will be displayed on the name fields.

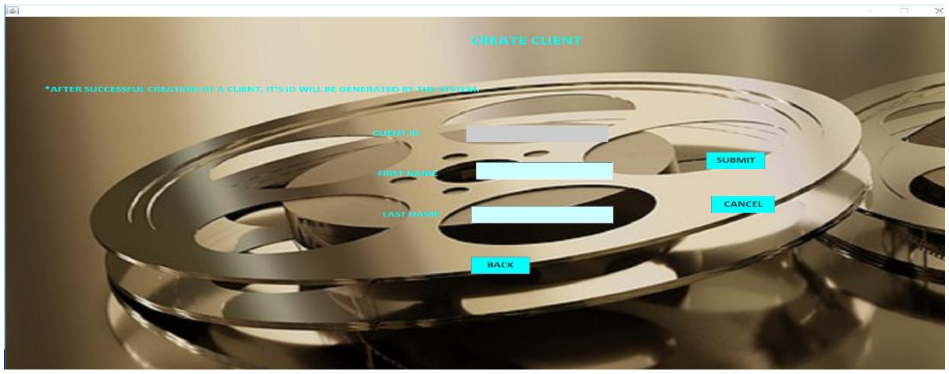
- ✓ DeleteClientPageGUI – This page is used for deleting a client in the system. Client will be logically deleted by changing the flag to True in the system


DELETE CLIENT
*SYSTEM WILL ONLY UPDATE DELETE FLAG AS CLIENT WILL NOT BE COMPLETELY DELETED BY THE SYSTEM
DELETE
BACK

Name	Client ID	DeletedStatus
Abilash Kommareddy	1	Active
Nithisha Bayyapu	2	InActive
Isitha bayyapu	3	Active
Innareddy Kommareddy	4	InActive
Sujatha Kommareddy	5	Active
Srilakshmi Yaram	6	Active
Pujitha Adari	7	InActive
Prakash Yenuva	8	Active
Pavithra Singareddy	9	Active
Vijay Subbendla	10	Active
Joseph Allam	11	Active
Riya Nandipati	12	Active
Pamika Kommareddy	13	Active
Premi Gada	14	Active
Teja Dethiri	15	InActive
Ravi Teja	16	Active
Ramya Yanamadala	17	Active
Roja Rani	18	Active
Mounika Kothapalli	19	Active
Vineela Dasari	20	Active

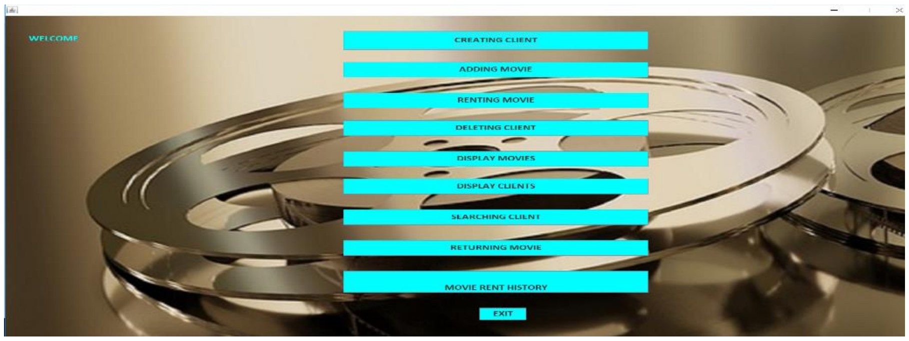
Implementation to all methods contained in IDeleteClientPageGUI is provided by this class. This page can logically delete clients in the system. By selecting the client on the table and clicking on delete button, the clients will become inactive.

- ✓ CreateClientPageGUI – This page is used for creating a client in the system upon successful addition of clients in the system an ID will be generated. Implementation to all methods contained in ICreateClientPageGUI is provided by this class.


CREATE CLIENT
*AFTER SUCCESSFUL CREATION OF A CLIENT, IT'S ID WILL BE GENERATED BY THE SYSTEM
CLIENT ID
FIRST NAME
LAST NAME
SUBMIT
CANCEL
BACK

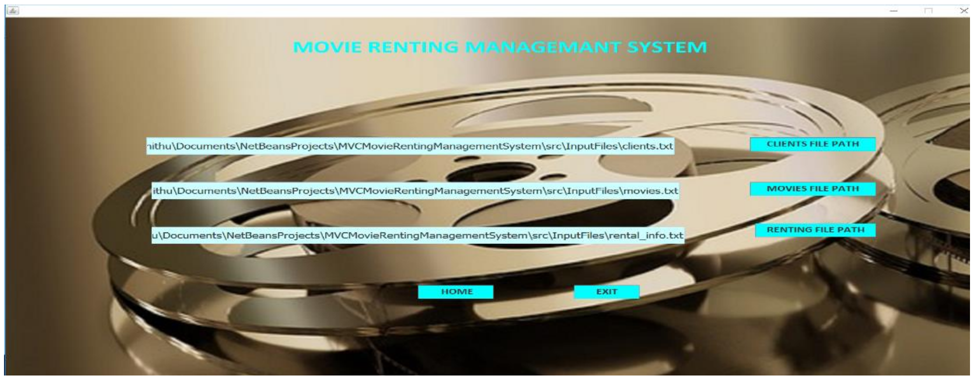
The system navigates to this page by clicking on Creating a client button in the home page where a client will be created. After successful creation of a client his/her ID will be generated by the system each time a new client is created.

- ✓ HomePageGUI – This page contains all the buttons which navigates to every other page. Implementation to all methods in IHomePageGUI is provided by this class.



After clicking home on SelectFilesPageGUI screen the system navigates to this page which contains all the icons used in this project. By clicking on exit button, one can come out of the system.

- ✓ SelectFilesPageGUI – This page is used for selecting paths for clients, movies and renting info. Implementation to all methods in ISelectFilesPageGUI is provided by this class.



```
ithu\Documents\NetBeansProjects\MVCMovieRentingManagementSystem\src\InputFiles\clients.txt
ithu\Documents\NetBeansProjects\MVCMovieRentingManagementSystem\src\InputFiles\movies.txt
u\Documents\NetBeansProjects\MVCMovieRentingManagementSystem\src\InputFiles\rental_info.txt
```

This page opens as soon as the application starts and the path where each file used for storing data is in this page.

Controller:

The package of controller classes ends with the name controller where an interface has been declared for each of the classes in controller package. Controller has reference to the interfaces of the classes in model as well as view classes. As controllers are responsible for event handling and views for holding UI both follows strategy pattern. Therefore, strategy pattern can be displayed in MVC by separately encapsulating the behaviors of event handling and UI's.

- ✓ SelectFilesController – This class holds reference to the IFilePaths which is a model class interface. Events generated by SelectFilesPageGUI view are handled by this class. All the methods of ISelectFilesController are implemented by this class.
- ✓ MovieRentingController – This class holds reference to IMovieRentingInfo, IMovie, IClient, IFilePaths which are model classes interfaces. Events generated by DisplayRentMovieHistoryPageGUI, ReturnMoviePageGUI, RentMoviePageGUI are handled by this class. All the methods of IMovieRentingController are implemented by this class.
- ✓ ClientController – This class holds interface to IClients and IFilePaths which are model class interfaces. Events generated by SearchClientPageGUI, DisplayClientsPageGUI, DeleteCleintPageGUI and CreateClientPageGUI are handled by this class. All methods of IClientController are implemented by this class.
- ✓ MovieController – This class holds interface to IMovie and IFilePaths which are model class interfaces. Events generated by DisplayMoviesPageGUI and AddMoviePageGUI are generated by this class. All methods of IMovieController are implemented by this class.

- ✓ HomeController – This class holds interface to IMovieRentingInfo, IMovie, IClient and IFilePaths which are model class interfaces. Events generated by HomePageGUI view are handled by this class. All methods of IHomeController are implemented by this class.

Software Used:

- NetBeans IDE
- JDK 1.7 or above
- Jcalendar.jar

File Structure:

As per the requirement three text files are used for storing the data entered into the system.

Clients.txt

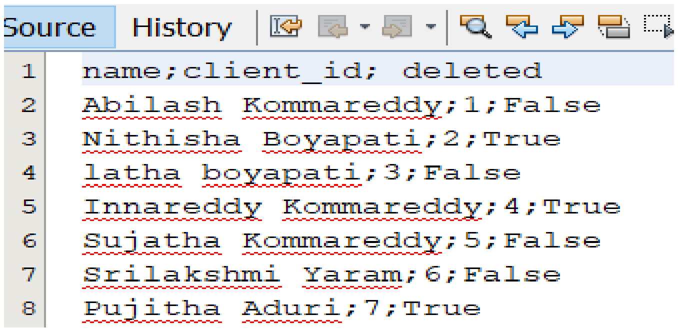
This file contains all the information related to the clients their name, ID and the status whether they are active or logically deleted from the system.

name; client_id; deleted

if deleted=False, then client is active in the system; if deleted=True then the client is logically deleted.

Example:

Thomas Jefferson; 3; True



```
name;client_id; deleted
Abilash Kommareddy;1;False
Nithisha Boyapati;2;True
latha boyapati;3;False
Innareddy Kommareddy;4;True
Sujatha Kommareddy;5;False
Srilakshmi Yaram;6;False
Pujitha Aduri;7;True
```

In the above screen shot it states that the clients under the deleted with False are active and with the flag True are logically deleted (Inactive) from the system.

Movies.txt

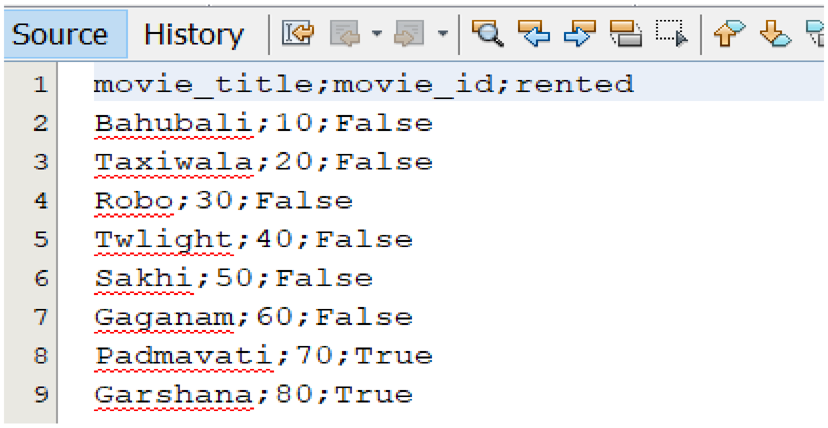
This file contains all the information related to movies like the title, ID and its rental status.

movie_title; movie_id; rented

if rented=False then is not rented currently; if rented=True is rented.

Example:

Altitude; 10; False



```
movie_title;movie_id;rented
Bahubali;10;False
Taxiwala;20;False
Robo;30;False
Twlight;40;False
Sakhi;50;False
Gaganam;60;False
Padmavati;70;True
Garshana;80;True
```

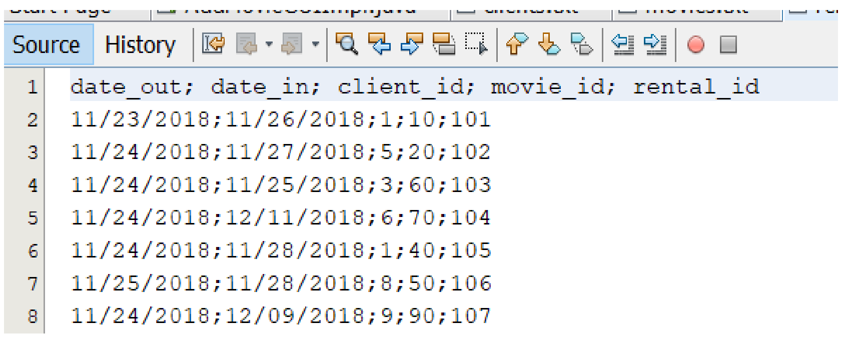
As available in the above screen shot the ID of a movie starts with 10. The status under rented as False indicated that the movie is not rented or returned and if the status if True that movie is current unavaible to rent as is already rented by some other person for a duration.

Rental_info.txt:

date_out; date_in; client_id; movie_id; rental_id;

date_out and date_in should be stored in format mm/dd/yyyy

if the movie has not been returned, then date_in is empty.



```
date_out; date_in; client_id; movie_id; rental_id
11/23/2018;11/26/2018;1;10;101
11/24/2018;11/27/2018;5;20;102
11/24/2018;11/25/2018;3;60;103
11/24/2018;12/11/2018;6;70;104
11/24/2018;11/28/2018;1;40;105
11/25/2018;11/28/2018;8;50;106
11/24/2018;12/09/2018;9;90;107
```

A record is be placed in this file every time a movie is rented, and the record is updated every time when a movie is returned.

Conclusion:

In conclusion, all the requirements for the project in developing Movie Rental Management system have been met. Here in this project each of the model class, view class and controller class contain an interface as a part of compound MVC pattern. The concrete Model class contains a reference to the view interface so that the model and view can interact with each other. The concrete view class is having reference to the controller interface and model interface. The concrete controller class is also containing a reference to the model interface as well as view class. As per the requirement each time the application starts it will ask the system to choose the file location. The system keeps only one copy of a movie and if the user tries to rent a movie which is currently rented then system tells that movie is not available.