*Team Cinema*

**Final Report**

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

CMSC 495 (7980)

5/9/23

**Project Plan**

**Table of Contents**

Introduction……3

Statement of Work……3

Resource List……3

Roles……4

Milestones……5

Risks ……5

Schedule……6

1. **Introduction**

This project plan will provide information on the goals of our project and how its development will be planned out. Included within are a Statement of Work, Our resources used, Roles, and Risks associated with development.

1. **Statement of Work**

For this assignment we have been tasked with developing a computer science project of our choice. We will develop a video rental system with three main functions. First a user will be able to browse the movies available for rent in the store. Secondly a user will be able to rent a movie from the store's database. Finally users will be able to return rented movies back to the store.

1. **Resources List**

To complete our project we will need the following resources:

1. Programming tool (Java)
2. A database software used to hold movie information (MySQL,Oracle)
3. A database server to hold DB info (Glassfish)

System Specification: Any PC with Java Functionality. PC with internet connection.

Operating Platform: Java SE 8, MySQL

1. **Roles**

| Role | Back End Development |
| --- | --- |
| Members | Miles, Ling |
| Description | Responsible for writing the backend code for the project |

| Role | Frontend Dev |
| --- | --- |
| Members | Malcolm, Wilson |
| Description | Responsible for creating the UI used for the application. |

| Role | Database Dev |
| --- | --- |
| Members | Tremaine, Char |
| Description | Responsible for programming the database functions |

**Milestones**

1. Create Project Design
2. Create Test Plan
3. Develop functioning backend code via Java
4. Develop functioning Database code via MySql
5. Develop frontend UI via Java
6. Create functionality between Java and SQL
7. Test Code
8. Finish Documentation

**Risks**

This section contains any risks involved with creating a project. Identifying these risks early on will prevent problems within the code from arising further down the development cycle.

| Risk | Impact | Action |
| --- | --- | --- |
| Team Availability | Being unable to meet as a team would cause us to fall behind in our schedule | Start communicating early and stay up to date on what we’ve accomplished |
| Unfamiliarity With Programming Languages | Not knowing the languages will make it difficult to cooperate on code | Make sure everyone is familiar with the language being used |
| Difficulty with Database Setup | Our project relies on proper database functionality | Ensure the SQL script works with the Java code and test it on different databases |

**Schedule**

| Task | Start Date | End Date |
| --- | --- | --- |
| Develop Project Design |  |  |
| Create Test Plan | 3/28/23 | 4/4/23 |
| Phase 1 | Start Date | End Date |
| Code java functionality | 4/5/23 | 4/11/23 |
| Code database functionality and queries | 4/5/23 | 4/11/23 |
| Test Java and SQL functionality | 4/5/23 | 4/11/23 |
| Phase 2 | Start Date | End Date |
| Create UI | 4/12/23 | 4/18/23 |
| Implement and Test UI Functionality | 4/12/23 | 4/18/23 |
| Phase 3 | Start Date | End Date |
| Testing | 4/19/23 | 4/25/23 |
| Documentation | 4/19/23 | 4/25/23 |
| Organize deliverables for the application | 4/26/23 | 5/9/23 |

**Project Members**

1. Miles Bell
2. Wilson Ebo
3. Tremaine Powell
4. Malcolm Gilbert
5. Ling Lan
6. Charlotte Williams

**Requirements Specification**

* The ability for users to create accounts
* View, Rent, and Return movie functionality

**System Specification**

* PC with the ability to run Java files
* Database functionality
* Mouse and Keyboard for User Input

**Database Design**

* Design the database schema based on the requirements specification
* Define relationships between entities and attributes for each entity
* Configure the database to handle data input and output

**User Interface Design**

* Develop a user interface for the video rental system
* Ensure that the interface is user-friendly and easy to navigate
* Implement functionality for browsing and selecting movies

**Application Logic**

* Implement application logic for rental and return transactions
* Develop functionality for browsing available movies
* Implement a search function for finding specific movies

**Testing**

* Test the application to ensure that all features work as expected
* Identify and fix any bugs that are discovered
* Perform user acceptance testing to ensure that the application meets user needs

**Deployment**

* Deploy the application to a production environment
* Ensure that the application is secure and stable
* Provide ongoing maintenance and support as needed

**User Guide**

This guide is intended for any users of the Team Cinema Video Rental System. This guide covers System Requirements, Installation, and Application Usage.

**Introduction**

*Key Features*:

An interactive User Interface that allows customers to browse, rent and return media from the store.

*Scope*:

This guide will cover the functions of the User Interface associated with our Video Rental System. It does not cover any external hardware the application may be compatible with

**System Description**

*System Requirements*:

Windows 10

Internet Connection

JDK 8 or above

*Installing the application:*

The .jar file created will be an executable and does not require an installation. Simply double click the exe to run it.

(JDBC can be used to connect different databases to the application)

It is recommended to place the .Jar file into a folder.

The application is intended to be used with a keyboard and mouse.

*Adding Database Functionality:*

The application contains functionality with MySQL. For functionality with alternate databases the specific driver must be added into the pom.xml file included with the deliverables.

These drivers are not included.

In order to configure the application to your local database the ConnectionProvider.java class must be edited. Once opened the DB\_URL, USER, and PASS credentials must be filled out with the proper information from your database. Without this the program will be unable to connect properly.

**Customer Use**

*First Use*:

Upon starting the application the user will be asked to create an account.

To create an account the following information will be required: First Name, Last Name, Username, Password and Email.

Alternatively the user may continue as a Guest Account.

*Homepage:*

Upon successful login the homepage window will pop-up. From here a user can View Movie inventory, Rent a Movie, or Return a movie.

*View Movies*:

Selecting “View Movies” takes you to a page containing all the currently available movies for rent. On this table you can see the movies Title, genre, Director, Rating, and if it is available for rent or not.

*Renting a Movie:*

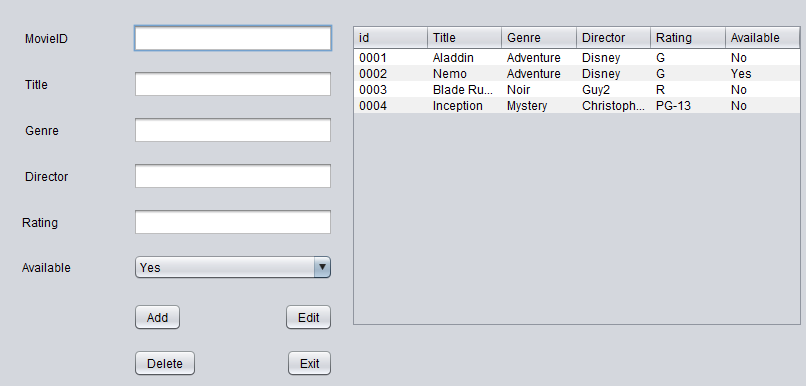
Selecting Rent Movie takes you to the Rent Movies window. The dropdown menu at the top allows you to select all movies currently in the inventory. To the right of the dropdown menu is the movie availability. Upon selecting a movie the availability box will let you know if the selected film is available for rental. If so, the user will be allowed to fill out the necessary credentials to rent the movie. If unavailable the text fields remain blank.

*Returning a Movie:*

The Return Movie page allows a user to return a film. Entering the Movie Title, UserID, and current Date. Then pressing enter will see if the film is past due and calculate any necessary fees. Pressing return will return the film to the inventory and change its availability from No to Yes.

**Admin Use**

*Adding a Movie:*

**

With the Add Movie menu admin users are able to add a new film to the database.

The movies ID,Title,Genre,Director,Rating and Availability can be added.

*Deleting a Movie:*

Clicking on a movie on the table to the right will highlight said film. Pressing the delete button will remove it from the database

*Editing a Movie:*

Clicking on a movie on the table to the right will highlight said film. From there you can change the movie information in the text fields provided. Pressing edit will update the movie information.

*Managing Users:*

Admins will be able to manage the information of customers' accounts.

A Username,Password,First and Last name, and email will be needed.

**Test Plan**

Our test plan will be used to identify high priority scenarios, high use cases, and areas that are prone to failure. Testing will be done to ensure that these vulnerabilities will not be present within the final deliverable application.

**Testing Scope**

Our test plan will cover both the front and backend of the application as well as the database compatibility and the UI.

**TESTING TABLE**

| Test Case | Item to test | How Tested | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| 1 | Menu Functionality | Transition through each window of the application. | App allows users to transition through the different menus on the homepage | Each button on the homepage takes the user to a new window | Pass |
| 2 | Menu Looping | Menu stays active until user decides to exit application | Application stays running and menus all connect with each other | Application stays running and menus all connect with each other till termination | Pass |
| 3 | Menu Closing | Test that the menus all return to the homepage | The close buttons should all return the user to the homepage | The close buttons for View Movies, Rent Movies, and Return Movie all take the user back to the homepage | Pass |
| 4 | Exiting the program | Click the exit button and see if application is terminated | App successfully closes upon exit button click | App successfully closes upon exit button click | Pass |
| 5 | Logging in | Use hard coded “admin” credentials to log in | Login is successful and takes user to homepage |  | Pass |
| 6 | Incorrect Login | Use incorrect credentials to log in | Login is unsuccessful and popup window informs user of this | The login does not work and a window saying “Incorrect username or password” is shown | Pass |
| 7 | Add User Class | Add a new user to the application | User is added and info is reflected in the “Users” table | User is added and info is reflected in the “Users” table | Pass |
| 8 | Add Film Class | Add a film to the database inventory | Film information is added and reflected in the “Movies” table | Film information is added and reflected in the “Movies” table | Pass |
| 9 | Edit Film | Film information is edited from the Add Film menu | Change the name of Finding Nemo to The Incredibles | Finding Nemo is renamed to The Incredibles and reflected in the “Movies” table | Pass |
| 10 | Rent Film Function | “Rent” Aladdin from the Rent Movie menu | Movie goes into the “rentals” table and availability changes from Yes to No | Movie goes into the “rentals” table and availability changes from Yes to No | Pass |
| 11 | Calculate Rental Fee | Rent a movie and check the rental fee calculation | The fee should be calculated based on the number of days rented | Rental fee is calculated but is not reflected in the Fee text field | Fail |
| 12 | Delete Film Function | Delete John Wick with the delete button | John Wick is removed from the Movies database | John Wick is removed from the Movies database and the table is refreshed to show this | Pass |
| 13 | Javas connection to the database server | Test the .getConnection function is working properly | Test database connection class | Database connection is successful as long as the proper pass is used | Pass |
| 14 | SQL Queries | Test various SQL queries are run properly (Select,Update,Delete) | Check the SQL queries for different functions | Queries for all classes but Rent Movie are successful 100% of the time | Fail |
| 15 | Moving SQL data from one table to another | “Rent” an object and move it from the available movies table to the rented movies table | Film should be moved from rentals to movies and shown as available | Change is reflected in the Movies table and availability but not the Rentals | Fail |
| 16 | User Input via UI | Test button clicks | Actionlistener for clicks works properly | Clicks work for buttons and textfields. However highlighting movie info is not alway working | Fail |
| 17 | User Input via input boxes | Test that user input is properly read/sent via text fields | Text fields can be typed in and auto populated as needed | Text fields are typed in and allow auto population as needed | Pass |
| 18 | Test Combo Box Yes/No | Check that the combo box for movie availability properly reflects the movies status | Box changes from Yes to No when a movie is rented | Box changes from Yes to No when a movie is rented | Pass |
| 19 | Create a Jar file for the executable | Create a Jar file that runs the Login.class | Upon executing the Jar class the program should run | Despite the Jar file being created and containing the files it does not run upon execution | Fail |

**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

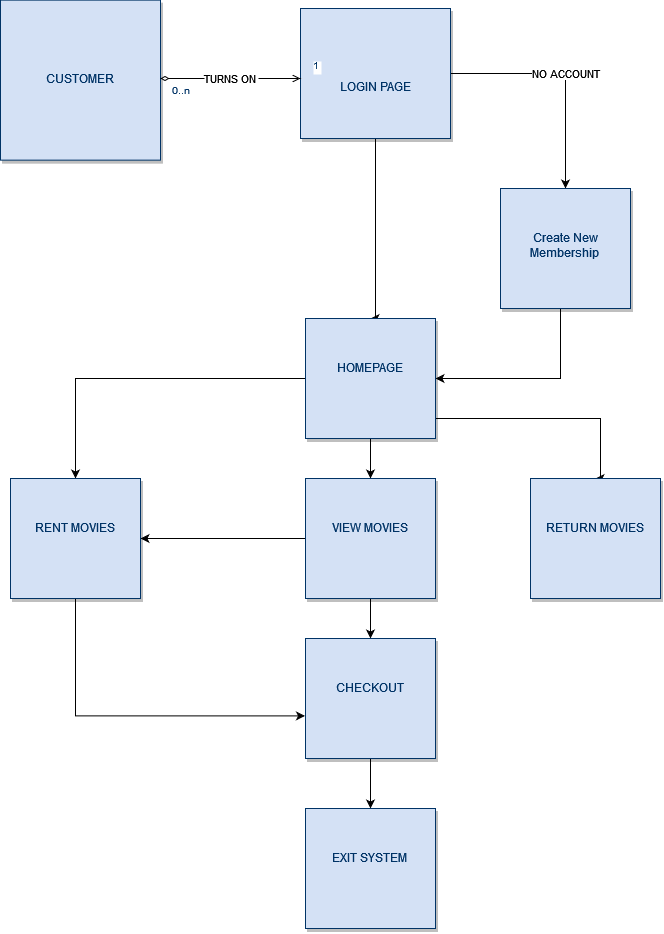
Inputs and Outputs will be handled through the UI.

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

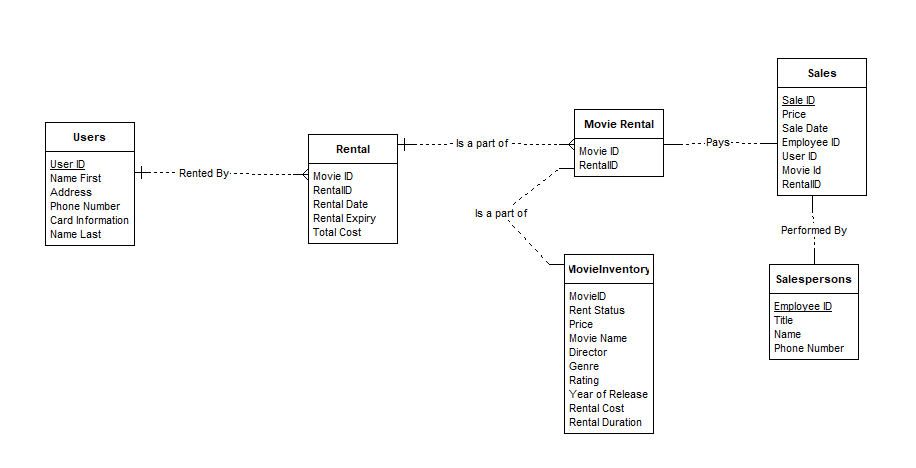
Buttons will be used for selecting the various windows and selecting movies

**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

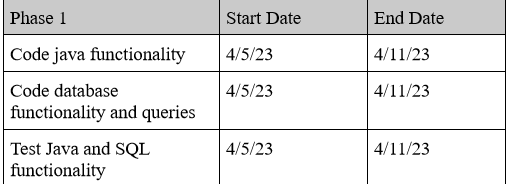
(movieID, title, director, genre, rating, rentPrice

Rented Movies

(movieID, rentalID, rentalDate, rentalPrice)

**Development History**

Phase 1 Report



Originally the milestone of Phase 1 was to have the java and database functionality running. However upon starting the coding processes it was found that the UI needed to be functioning first in order to test the database functionality. As a result the Phase 2 milestones (UI Creation and Testing) were done instead so that a testable product could be utilized.

Current Phase of Application

Login, Homepage, and Add user functionality has been implemented. It functions with the database but requires further coding and testing. Going between the different menus and exiting the program functions properly as well.

Problems

Main issues encountered were with database functionality. Each type of database requires its own driver to work with java. This required time to be spent on researching database drivers and dependencies instead of working on the application itself.

Additionally remote database functionality has not been implemented but may not be required.

**Phase 2**

The milestones for this phase of the application revolved around getting SQL statements working within the application. The basic skeleton of the application has been finished and things like logging in/out, and going from the different windows are functioning properly. So for phase two it was time to actually implement the database functionality of the application.

Current Phase of Application

With database functionality and most of the UI finished we could now work on interactions with the database itself. SQL statements to retrieve and save data have been implemented for the NewUser and NewMovie classes. These allow an admin user to add new users and films to the application. ViewMovies,RentMovie, and ReturnMovie classes are on pace to be finished on time.

Additionally a section on how to utilize a database has been added to the UserGuide. The application has compatibility with MySql databases but still requires the database url, the host name, and the password.

Problems and Risks

Having the application interact with the multiple tables of the database has caused some issues, as well as transferring data between said tables. For example, for a user to rent a movie their UserID needs to be a part of the rental table along with the MovieID. This requires taking data from the Users table (The UserID), and the Movies table (The MovieID), and putting them into the Rental table while keeping the integrity of the data.

Additionally, deciding how to present certain types of data needed to be figured out. For example a ViewMovies screen could be done in a multitude of ways. The movies table itself could be shown which provides all the information of a film (title,genre,director, etc.), but then implementing the rental functionality within the table would depend on how the table is shown. If it is done with a label there would need to be additional UI used to perform the rental.

**Phase 3 Report**

We are currently on track with the plan as the application is now finished for the most part. All the functionality with regards to Renting and Returning movies has now been implemented. The checkout function has been removed since the ReturnMovie class can calculate the return fee. Most stores would have their own Point of Sales system so there is no need for our application to contain that as well. A rentals table is now used to store information when a user rents a movie. Doing this changes the movie's availability from available to not available. When a move is unavailable users are unable to rent them from the store. This boolean yes/no function acts as the main function of the rentmovie class.

Current Phase of Application

The homepage UI has been redone to a better layout.

ViewMovies, RentMovie and ReturnMovie are now functional.

Add Movies now has Add,Update and Delete functions.

ViewMovies properly reflects all films in the “Movies” database and shows if they are available or not

RentMovie also shows if a film is available or not.

Documentation has been added to the deliverables for Creating Tables and Connecting the Application to a Database

Prepared Statements are now used instead of regular statements for adding SQL security and ease of use.

Problems and Risks

The SQL queries used for the different functions of the application are extremely case sensitive. One misspelling or incorrect table name will break certain functions. There are Try Catch statements in place to ensure that these do not break the program entirely, only the functions.

There are some UI issues occurring with auto filling data in the Add and Rent movie classes. Clicking on the film in the table should autofill the information into the text fields, and while it *does* work sometimes it requires multiple clicks.

Like in the last report setting up database functionality is up to the user of the application. There is instruction on how this should be done within the user guide but the admin will need to have an understanding of where to find the information needed to connect to the DB.

To Do

Functions of the code need to be tested for consistency.

Add additional database functionality (MS SQL Server, IBM DB2, Oracle, Microsoft Access)

Allow access to rent movies page from view movies page

Implement user specific logins

**Conclusion**

The application is very quick to run and seamlessly goes through the different menus. The Try,Catch statements used throughout the classes do a good job of giving user feedback when something is done incorrectly. If login credentials are put in incorrectly a window pops up to inform the user and this is done for other functions in the app as well. The Add Movies function works well and allows the admin to manage several aspects of the move inventory from one screen. Here films can be Added, Edited or Deleted and this is all reflected with the table shown on screen. The Rent Movies window clearly shows users whether or not a film is available and does not even allow an unavailable movie to be rented.

During the development of this application quite a bit was learned on how to use GUI’s within java. Almost all of the functionality is coded through the different actionListeners used with JFrame. As a result it was imperative that there was an understanding of how the different ActionEvents worked and needed to be coded. There was some deliberation on how to best connect the application to the database. Originally each class would have the code for connecting however that was found to be a bit redundant as the classes all needed to connect and disconnect. Instead a “ConnectionProvider” package was created and called on each of the classes allowing for an easier experience.

Most of the difficulties found with developing the application stemmed from integrating third party systems into the app itself. The java aspect of the programming process did not cause much trouble but integrating things like SQL, Queries, Creating executables, and managing plugins were the main issues. Adding database functionality required third party plugins to be added in the POM files which required a bit of googling to find out. Certain JFrame functions were not available in netbeans originally but were available in other IDE’s. This also required googling and downloading of additional plugins for functions thought to be included already.

Additionally since the application is dependent on a database it was difficult to have the program work universally. Although the program was configured to work with MySQL every database required a different password unique to each user's version of MySQL. Additionally the SQL queries used for accessing the tables were very case sensitive. If the program had the movies table as “Movies” and another group member had the movies table as “Movie” the program would not connect properly. This could be remedied by having a java class that creates the table but that also causes the issue of constantly building and dropping tables. In retrospect adding a window that prompted the user for the database credentials might have been a good idea. This would make it so one does not need to actually go into the java file to configure this before the app runs properly.

There are some limitations due to the primitive nature of JFrame. The GUI could not look as appealing as say a Netflix or Redbox movie rental GUI. We simply did not have the resources to create such a thing. Additionally since the app runs on a local database there is no way to remotely access parts of the inventory. Everything must be done locally. This also means users would not be able to rent a movie digitally or from a website. The program is intended for in-store use. Finally, while the program does calculate the rental fee it does not actually act as a Point of Sales system. Therefore it would have to work in tandem with a store's own POS to perform things like checking out or late fee payment.