Project Documentation

ISTE-330 Database Connectivity and Access

Movie Rental

Document: Movie Rental Documentation

Author(s): Tomislav Kulusic, Nikola Lenert, Ivan Kovacevic, Redzep Kolar, Frano Nola, Isabella Sturm

Date: 24 April 2017

Version: 4

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Description of Change* | *Author(s)* | *Date* |
| 1 | Deliverable 1 – Initial documentation | all | 2017-04-07 |
| 2 | Deliverable 2 – Fixes/ revision of initial documentation | Isabella Sturm, Frano Nola | 2017-04-13 |
| 3 | Deliverable 3 – User Manual | Frano Nola | 2017-04-21 |
| 4 | Deliverable 4 – Start Sections 5 & 6 |  | 2017-04-28 |

Table of contents

[1 Introduction 3](#_Toc481136188)

[1.1 Overview 3](#_Toc481136189)

[1.2 Purpose and Scope 3](#_Toc481136190)

[1.3 Background 3](#_Toc481136191)

[1.4 References 3](#_Toc481136192)

[1.5 Document Overview 3](#_Toc481136193)

[2 Problem Description and Solution Architecture 5](#_Toc481136194)

[2.1 Problem Description 5](#_Toc481136195)

[2.2 Architectural Design 5](#_Toc481136196)

[2.3 Database Layer and Database Abstraction (Connectivity) Layer Description 6](#_Toc481136197)

[2.4 Business Layer Description 9](#_Toc481136198)

[2.5 Presentation Layer Description 9](#_Toc481136199)

[2.6 Areas of particular concern 10](#_Toc481136200)

[3 Requirements 11](#_Toc481136201)

[3.1 Context 11](#_Toc481136202)

[3.2 Functional Requirements 11](#_Toc481136203)

[3.3 Other (Non-Functional) Requirements 12](#_Toc481136204)

[4 User Documentation 13](#_Toc481136205)

[4.1 Graphical User Interface Design 13](#_Toc481136206)

[4.2 User Manual 13](#_Toc481136207)

[5 Installation, Configuration and Acceptance Testing 14](#_Toc481136208)

[5.1 Installation 14](#_Toc481136209)

[5.2 Configuration 14](#_Toc481136210)

[5.3 Acceptance Testing 14](#_Toc481136211)

[6 Final Remarks and Conclusion 15](#_Toc481136212)

# Introduction

## Overview

Our team will be working on application that will provide an online movie rental service.  The users will be able to search for movies they would like to see, go through the list, pick few movies and rent them. Also, they will be able to see what movies they have for rent in their “shopping-cart” and all the information about that movie (the cost and due date).

## Purpose and Scope

The main purpose of the document is to provide an overview of the application as well as all of its features for potential users.

The intended audience of our web application is anyone who is old enough to rent movies. This includes teenagers, young adults, parents and anyone who is both able to and allowed to use the internet for renting movies.

For this project, we will create a website that has a clean user interface that will accommodate a large demographic of users. The website will allow all users to browse movies, but for renting movies, users will have to sign up or login. There is no subscription necessary for our service, but there is a one-time charge each time the user rents a movie. Along with renting movies, users will also be able to write reviews of the movies for other users to see.

To accomplish the project goals, each week the team will need to deliver on project deadlines.

* Milestone 1: Design Documentation and Architecture
* Milestone 2: Database Abstraction and Data Access Layer
* Milestone 3: User Documentation and Code
* Milestone 4: Presentation Layer and Business Layer
* Milestone 5: Final Code and Documentation

## Background

The document is created by a team of students who are in the process of developing their first database oriented application. The purpose of the document is for all team members to get very familiar with the functionalities of the application, how it should look and how the functionalities will be implemented.

## References

* Link from MyCourses to the project guide
* W3Schools - <https://www.w3schools.com/>
* MySQL - <https://dev.mysql.com/doc/connectors/en/apis-php.html>
* Bootstrap - <http://getbootstrap.com/getting-started/>
* jQuery - <http://api.jquery.com/>

## Document Overview

The first section of the document provides and introduction and overview of the project. The overview of the project is a brief description of what to expect followed by the purpose and the scope of the project which will explain the problem we are trying to solve. The background is a short description of the group as a whole and our purpose or goal behind the project. After the background, there is a list of references which includes documents such as documentation, standards, deliverables and meeting summaries. The last part of section one is the document overview, which lays out the structure and outline of this document.

The second chapter of the document includes a problem description that will be solved by this project. It also includes descriptions of the design of this document and descriptions of each layer including the database layer and database abstraction layer, the business layer, and the presentation layer as well as a description to how one layer connects to the next layer. To end this section of the document, we will address our areas of concern for the project.

In chapter three, the first part is a description of the context of the project with explanation of how to application connects and works with the environment. Then, there is a list of functional requirements followed by a list of non-function and other requirements of the project. Functional requirements include all the key functionalities (what the application must do) and user requirements which includes users, roles, and privileges. The non-functional requirements include hardware and software requirements and other general information such as performance and error handling.

Chapter four is user documentation. The purpose of this chapter is to describe the graphic user interface design and experience considerations as well as provide a user manual, which will list and describe each of the functions available for the user.

In chapter five, there will be an explanation of the prerequisites and requirements for installing or opening our application for use followed by the configuration of the application. Then, there is a description of the acceptance testing which describes what testing needs to be done to see if all of the requirements have been met for our application.

The last section of the documents is for final remarks and the conclusion. In this chapter, there will be a description of our experience working on the project, a summary of the hours spent working on the project, and a conclusion of how our resulting project compares to the original plan for it. The conclusion will include whether or not we met all of our requirements and if not, which ones we were unable to do and why.

# Problem Description and Solution Architecture

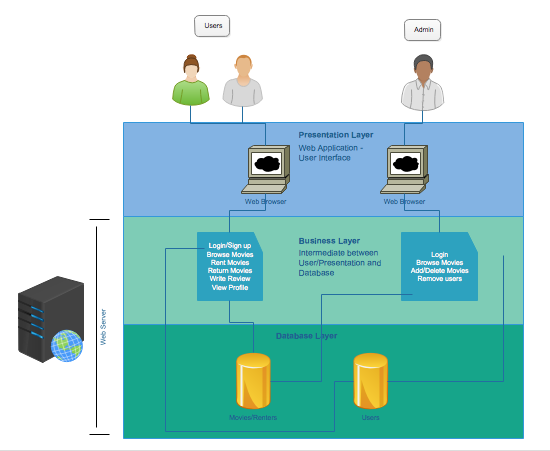
## Problem Description

The problem is that there are not enough places to rent movies online. Most sites that allow for movie rental require subscriptions. Subscriptions can be good if users watch a lot of movies per month or per year, but even then, subscriptions can be a hassle to manage and hinder users from signing up for the static monthly or yearly rate.

For users who don’t buy into subscriptions, they are left with a few inconvenient options: watch the movie when it comes out in theaters, buy the movie when it comes out on DVD, or drive to a movie store and rent it from there. Sometimes, desperate movie-lovers search for the pirated movie to stream online, which is not only illegal but also takes away from the experience of watching a good-quality movie.

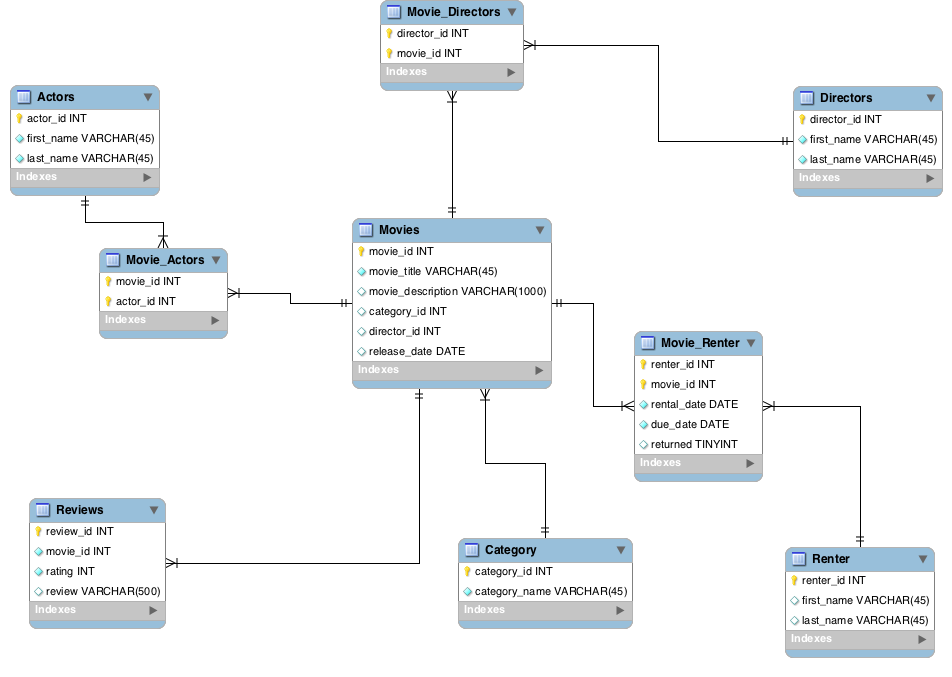
Our service combines the old-school movie stores with an online site to allow customers to watch the movies they love, in the comfort of their homes, without the hassle of a monthly or yearly fee.

## Architectural Design

**

For this project we chose to use PHP, JavaScript, HTML and CSS because we are most familiar with them. CSS and HTML will allow us to display the application to the user the way we imagined it, and PHP and JavaScript will allow us to make all the functionalities for the site that we want.

## Database Layer and Database Abstraction (Connectivity) Layer Description



|  |  |  |
| --- | --- | --- |
| **Column** | **Data type** | **Description** |
| **Movies** | | |
| movie\_id | INT | Primary key, autoincrement; each movie has its own movie id to connect Movies table with other tables |
| movie\_title | VARCHAR | Not null; title of movie in database that can be rented |
| movie\_description | VARCHAR | Description of movie (plot) |
| category\_id | INT | Foreign key that relates Movies to Category |
| release\_date | DATE | Date when the movie was released in theaters |
| **Movie\_Actors** | | |
| movie\_id | INT | Primary & foreign key; relates Movies to Actors - each movie can have more than one actor |
| actor\_id | INT | Primary & foreign key; relates Actors to Movies - each actor can be in more than one movie |
| **Actors** | | |
| actor\_id | INT | Primary key, autoincrement; each actor has a unique id to relate Actors table to other tables |
| first\_name | VARCHAR | Not null; first name of actor |
| last\_name | VARCHAR | Not null; last name of actor |
| **Movie\_Directors** | | |
| movie\_id | INT | Primary & foreign key; relates Movies to Director - each movie can have more than one director |
| director\_id | INT | Primary & foreign key; related Director to Movies - each director can direct more than one movie |
| **Directors** | | |
| director\_id | INT | Primary key, autoincrement; each director has a unique id to relate Directors table to other tables |
| first\_name | VARCHAR | Not null; first name of director |
| last\_name | VARCHAR | Not null; last name of director |
| **Category** | | |
| category\_id | INT | Primary key, autoincrement; each category has its own id to use to relate to other tables |
| category\_name | VARCHAR | Not null; name of category (Horror, Comedy, Romance, etc.) |
| **Movie\_Renter** | | |
| renter\_id | INT | Primary & foreign key; relates Renter to Movies; each renter can rent many movies |
| movie\_id | INT | Primary & foreign key; relates Movies to Renter; each movie can be rented by many renters |
| rental\_date | DATE | Not null; date when a movie is rented by renter |
| due\_date | DATE | Not null; date when movie needs to be returned by renter |
| returned | TINYINT | 0/1; if the movie is still with the renter, value is 0. When movie is returned, value is updated back to 1. |
| **Renter** | | |
| renter\_id | INT | Primary key, autoincrement; each renter has a unique id to allow Renter table to be related to other tables |
| username | VARCHAR | Unique key, not null; username of user which is unique to each user |
| first\_name | VARCHAR | Not null; first name of renter |
| last\_name | VARCHAR | Not null; last name of renter |
| email | VARCHAR | Not null; email address for renter, application use of email could be to notify user of overdue movie rental |
| card\_no | INT | credit card number to be used to renting movies |
| **Reviews** | | |
| review\_id | INT | Primary key, autoincrement; each review has its own id to be used to related Reviews table to other tables |
| movie\_id | INT | Unique & foreign key; relates Reviews to Movies- a review can only have one movie but each movie can have multiple reviews |
| rating | INT | Not null; Rating 1-5 of a movie, can be used to find average rating of movies or displayed along with review |
| review | VARCHAR | Written review of a movie that can be read by users |
| **Users** | | |
| username | VARCHAR | Primary key; each user has his/her own username and is used to connect users to Renters as well as to connect Users to Roles via the User\_Roles table |
| password | VARCHAR | Not null; password is used at login along with username to authenticate user for renting movies |
| email | VARCHAR | No null; email of user |
| **User\_Roles** | | |
| username | VARCHAR | Uses user’s username to relate Users to Roles |
| role\_id | INT | Uses role\_id to connect Roles to Users |
| **Roles** | | |
| role\_id | INT | Primary key, autoincrement; each role has its own role id to be used in related Roles to Users and Privileges |
| role\_name | VARCHAR | Roles include: Admin, User, Viewer |
| **Role\_Privileges** | | |
| role\_id | INT | Uses role\_id to relate Roles to Privileges |
| privilege\_id | INT | Uses privilege\_id to relate Privileges to Roles |
| **Privileges** | | |
| privilege\_id | INT | Primary key, autoincrement; each privilege has its own id to identify the privilege when related privileges to roles of users |
| privilege\_desc | VARCHAR | Describes what actions users are allowed to do; include: add movies, delete movies, modify user data, view movies, rent a movie, write a review, view profile |

## Business Layer Description

The business layer will include logic for both users and admin. The business layer acts a intermediate between the database layer and the presentation layer. When users (via the presentation layer) navigate around the site and perform the actions below, they are relying on the database to display information as well as updating the database. As for the admin, they are able to see and do anything a regular user can do, but they also have the privilege to add or remove from the database. But instead of directly accessing the database, all users (including admin) perform business logic operations that affect or use information from the database through a clean user interface.

Users:

* Login/ Sign up
  + User can login or sign up using email on landing page
* Browse and read about movies
  + Unregistered user won’t be able to watch the movies just browse
* Add movie to cart
  + Registered user will be able to add movies to cart in order to rent them
* Rent movie/ Check out
  + After user chooses what movie he wants to rent he can do check out
* Return movie
  + After time runs out the movie will be returned
* Write review
  + Only if the user rented the movie he can add a review to it

Admin:

* Add movie/ actor/ director
  + Admin can modify the information about the movie or add new movie
* Delete movie/ actor/ director
  + Just as he can add the movie he can delete them also
* Remove inactive user
  + If there is certain problem with user he can delete the user's account

## Presentation Layer Description

The Presentation Layer is a web application for renting movies. Using web technologies such as HTML, CSS, JavaScript and PHP, users will be able to browse a database of movies and rent them via a clean, intuitive user interface.

For the public, the presentation is simply a library of movies to browse through. Like many sites and service that require login for more options, public users have an option to sign up for an account. If they have an account but aren’t signed in, or if they don’t have an account, and they want to rent a movie, when they are browsing and select a movie there is a “Sign-Up or Login to Rent” option. General users, who have signed in, can browse the library of movies like the public. However, general users can also rent movies, return movies, and write reviews for movies they have seen. There is also a profile page, which tells the user information about their account, such as their name, email address, movies they have seen, and movies that they are currently renting.

As an admin, there is the same library of movies that the others users have, but on the same page, there is an option for the admin to add a new movie or delete a movie. When adding a movie, the admin can also add actors or directors or relate an existing actor or director to the movie. Instead of a profile page, the admin can look at information about users including name, email, movies each user is currently renting if any and related information about the rental (date rented, date due), the date when the account was created and the last login for that account.

## Areas of particular concern

The particular concern would be making the authentication for users and admins and then later on authorization.

# Requirements

## Context

As a user, the functionality associated with the database would include fetching data about the movies. Users should be able to browse the movies available for rent, and get information about the any perspective movie including description, actors in the movies, who directed the movies, when it was released, and any reviews written about the movies. If a user decides to rent any movies, although it is not necessarily a user’s role, the database will be updated to show that whatever movie(s) are being rented are no longer in available until they are returned by the user. This means renting and returning a movie will update the database based on actions taken by a user. Users can also insert into the ‘Reviews’ table because as a user, you should be able to write reviews, and that is one functionality in our application.

An admin on the other hand will mostly focus on inserting and deleting data, particularly in the ‘Movies’, ‘Actors’, Directors’ and related tables. As an admin, with authorization, he/she can insert new movies now available for rent and delete ones that should no longer be in stock. Also, an admin can fetch information about what movies are being rented, who rented them, and when they should be returned.

## Functional Requirements

There are key functionalities  for both users and ‘admin’. Both need to be able to communicate to the database of movie rentals. Overall, our application needs to be able to fetch, update, insert and delete data.

Admin

1. Add movies
   1. The admin is able to add movies to the site/database, as well as information related to the added movies. When adding movies, they can also add actors and directors.
2. Delete movies
   1. The admin is able to delete movies from the site/database, as well as information related to the deleted movies. When removing movies, they can also remove actors and directors, but it is not necessary - only the relation between the actor or director to the deleted movie will be removed.
3. View (and remove) users and data about users
   1. As an admin, the privilege to see information about the users and which movies they have rented or are renting is granted. Also, the admin can remove inactive users as well as users that abuse the service.
4. Log in/ password required for authorization
   1. In order to be granted the privileges as an admin, this user must provide the correct password to the admin account to proceed.

User

1. View movies for rent
   1. There will be  a web page that will display all the movies for rent. When the user clicks on the movie, there will be information about the movie including information such as movie description, ratings and reviews, and its availability.
2. Select a movie to rent
   1. When looking at a movie for rent, the user will be able to add movie to a ‘shopping cart’
3. Rent a movie
   1. Once the user selects a movie, he/she will be able to checkout and the movie will become unavailable until the user returns it.
4. Write a Review
   1. Upon return of a movie, the user will be able to write a review of the movie, which will be displayed to others who are looking to rent that movie.
   2. If a user is logged in, he/she can select to write a review of the movies
5. Sign up/ Log in
   1. Anyone can view the movies in the library, but in order to rent a movie, users must sign up or login to our service.
6. Profile
   1. After logging in, users will be able to view their profile which will have information including movies they have watched and movies that they currently have rented.

## Other (Non-Functional) Requirements

For this movie rental service, the necessary hardware would be a laptop or computer with access to the Internet (Wi-Fi). Ideally, we will have a web application that will run in a browser. For building the application we will need a text editor and will be using HTML, CSS and PHP (most important for database connectivity and access), as well as JavaScript and jQuery. Because we will be using PHP in a web application, we will also need a server.

The application will handle security issues through the use of Prepared Statements and using authentication when attempts are made to access admin privileges. While the application is running, any errors that may occur will be logged to a file or to the admin and a message will be reported to the user.

# User Documentation

## Graphical User Interface Design

When designing a graphical user interface, it is important for it to be simple and intuitive.

The simple design will allow users that are not familiar with this software to easily navigate through it and make actions without problems.

## User Manual

* Public:
  + By clicking on the movie poster you will be able to see more information about the movie
  + To register user needs to enter his email, unique username and password
  + To login user has to enter his username and password
* Registered User:
  + By clicking on the cart icon near the movie poster user can add movie to the cart
* Admin:
  + The user will be provided with options such as renting movie, checking what movies are available in the store. The user will be able to choose the movie he wants to rent, of course that movie has to be available in that time, and the time period of the rent.
  + For the admin panel user will be provided with additional functionalities like adding and removing movies.

# Installation, Configuration and Acceptance Testing

## Installation

Since we are building this application with java and SQL we need some basic things before we start building application. Firstly for java we need to install Java Development Kit (JDK). Then for writing code you need some IDE for in our case we used IntelliJ/Eclipse. For MySQL you need to download version of MySQL.

## Configuration

*For the configuration part we had to download MySQL driver and add it to the project library.*

*Without that we are not able to connect to database and work with it.*

## Acceptance Testing

[Testing done to determine if the requirements are met – describe typical usage and tests (deliverable 5).]

# Final Remarks and Conclusion

[Summarize the experiences, both in terms of the produced results and work on the project. List project deliverables, as well as positive (and negative) experiences and concerns. Comment on missing functionalities and possibilities for improvement and extensions. Estimate project effort (person-hours) and how it was distributed in time and per team roles (deliverable 5).]