**Movieteca**

Software Design Document

Name (s): RadulescuAlexandru, Preda Radu-Alexandru,

SacuianuIonut-Alexandru, Simon Jozsef-Gabriel

Lab Section: Software Engineering

Date: (03/23/2020)

**TABLEOFCONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | 1.1 | **INTRODUCTION**  Purpose | **3**  3 |
|  | 1.2 | Scope | 3 |
|  | 1.3 | Overview | 3 |
|  | 1.4 | Reference Material | 4 |
|  | 1.5 | Definitions and Acronyms | 4 |
| **2.** |  | **SYSTEMOVERVIEW** | **6** |
| **3.** |  | **SYSTEMARCHITECTURE** | **7** |
|  | 3.1 | Architectural Design | 7 |
|  | 3.2 | Decomposition Description | 9 |
|  | 3.3 | Design Rationale | 9 |
| **4.** | 4.1 | **DATADESIGN**  Data Description | **11**  11 |
|  | 4.2 | Data Dictionary | 11 |
| **5.** |  | **COMPONENTDESIGN** | **13** |
| **6.** |  | **HUMANINTERFACEDESIGN** | **16** |
|  | 6.1 | Overview of User Interface | 16 |
|  | 6.2 | Screen Images | 16 |
|  | 6.3 | Screen Objects and Actions | 19 |
| **7.** |  | **REQUIREMENTSMATRIX** | **20** |

**1.INTRODUCTION**

**1.1 Purpose**

This Software design document describes the system overview and the system architecture of the “Movieteca”. This document will also provide a data design description and a data dictionary. There will also be presented the component design and the human interface design which will provide an overview of the user interface and some screen images, also there will be presented the requirements matrix and the appendices.

**1.2 Scope**

The project idea is to implement an online movie rental system, where the customers can find different movies to rent or to buy from. The movies will have different providers, who will also have the possibility to add movies for rent or for sale.

The advantages of this web application against other applications are: the to be implemented easy to use interface, which will help the user do the action desired with a single button click; the informations about movies collected from a reliable and well known website (iMDb); and the chat system, which will help the user communicate easily and in private.

**1.3 Overview**

This document contains several representations of the software design components such as the system architecture data design and the human interface design.

This document contains eight chapters and thirteen subsections. The organization of those is:

|  |
| --- |
| 1. **INTRODUCTION**    1. *Purpose* |
| * 1. *Scope* |
| * 1. *Overview* |
| * 1. *Reference Material* |
| * 1. *Definitions and Acronyms* |
| 1. **SYSTEM OVERVIEW** |
| 1. **SYSTEM ARCHITECTURE** |
| * 1. *Architectural Design* |
| * 1. *Decomposition Description* |
| * 1. *Design Rationale* |
| 1. **DATA DESIGN**    1. *Data Description* |
| * 1. *Data Dictionary* |
| 1. **COMPONENT DESIGN** |
| 1. **HUMAN INTERFACE DESIGN** |
| * 1. *Overview of User Interface* |
| * 1. *Screen Images* |
| * 1. *Screen Objects and Actions* |
| 1. **REQUIREMENTS MATRIX** |
|  |

**1.4 Reference Material**

|  |  |  |
| --- | --- | --- |
| Nr. | Link | Description |
| *1* | <https://en.wikipedia.org/> | General informations |
| *2* | <http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf> | IEEE 830-1998 template |
| *3* | <https://www.imdb.com/interfaces/> | Datasets for the database |
| *4* | <https://datasets.imdbws.com/> | Datasets for the database |

**1.5 Definitions and Acronyms**

**XAMPP** - a free and open-source cross-platform web server solution stack package developed by Apache Friends consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

**PHP** - a popular general-purpose scripting language that is especially suited to web development.

**MySQL** - an open-source relational database management system.

**HTML**- Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser.

**CSS**- Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML.

**2.SYSTEM OVERVIEW**

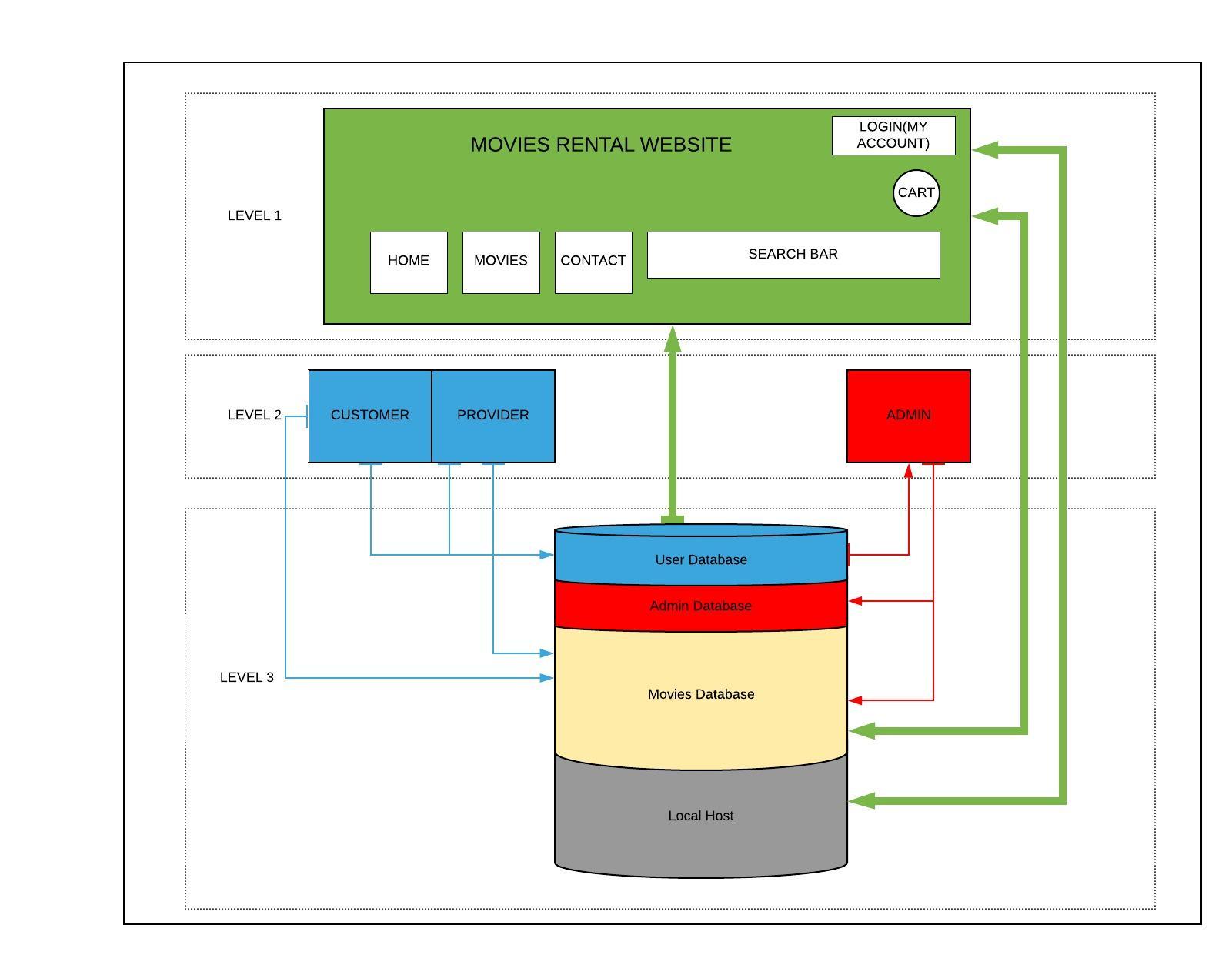
Movieteca is a website with a futuristic design, ready for everyday user to rent his favorite movies. This website will use the latest MsSQL Database in order to offer to our clients the best and the fastest experience possible on a website.

The website is designed to be compatible with services and infrastructure existing in the filming industry. We are negotiate with movie producers, filming studios, in order to have all the policies and security protocols in order.

The Movieteca will provide the following capabilities:

* User account management, which shows a detailed view of the client’s choices, rents status, rents return time and other options.
* Integration of all maintenance data which will allows the rental to be in real-time.
* Enhanced and additional user interface for a smooth navigation and easy to use method
* Feedback form which allows the user to leave a comment or a review to a movie, and also to tell how prompt were we in our provided services.

**3.SYSTEM ARCHITECTURE**

**3.1 Architectural Design**

Relations between levels:

1. Level 1 – Level 2:
2. Customer:

* Can login through the interface;
* Can visualize the website;
* Can rent/buy movies;
* Can customize their profile;
* Can see what movies he bought or rent, and also how much time he has left for a rented movie;
* Can chat with providers, or contact the administrator;
* Can add reviews for different providers and movies;
* Can add movies to cart;

1. Provider:

* Can login through the interface;
* Can visualize the website;
* Can request approval to add a movie for selling or renting;
* Can customize their profile;
* Can see the movies they added for rent or sell;
* Can chat with customers, or contact the administrator;

1. Administrator:

* Can login through the interface;
* Can visualize the website;
* By accessing “my account” section, he is able to respond to messages from users, approve or not requests, solve problems;
* Can add movies for sale or rent, when the owners require;

1. Level 2 – Level 3:
2. Customer:

* Through a query, this user will be able to login (the query will search for his login credentials in the database);
* By modifying the profile, the user will modify the informations from the database;
* By buying or renting a movie, this will be added to the database;

1. Provider:

* Through a query, this user will be able to login (the query will search for his login credentials in the database);
* By modifying the profile, the user will modify the informations from the database;
* If they add movies for sale or rent, the movies will be added to the database;

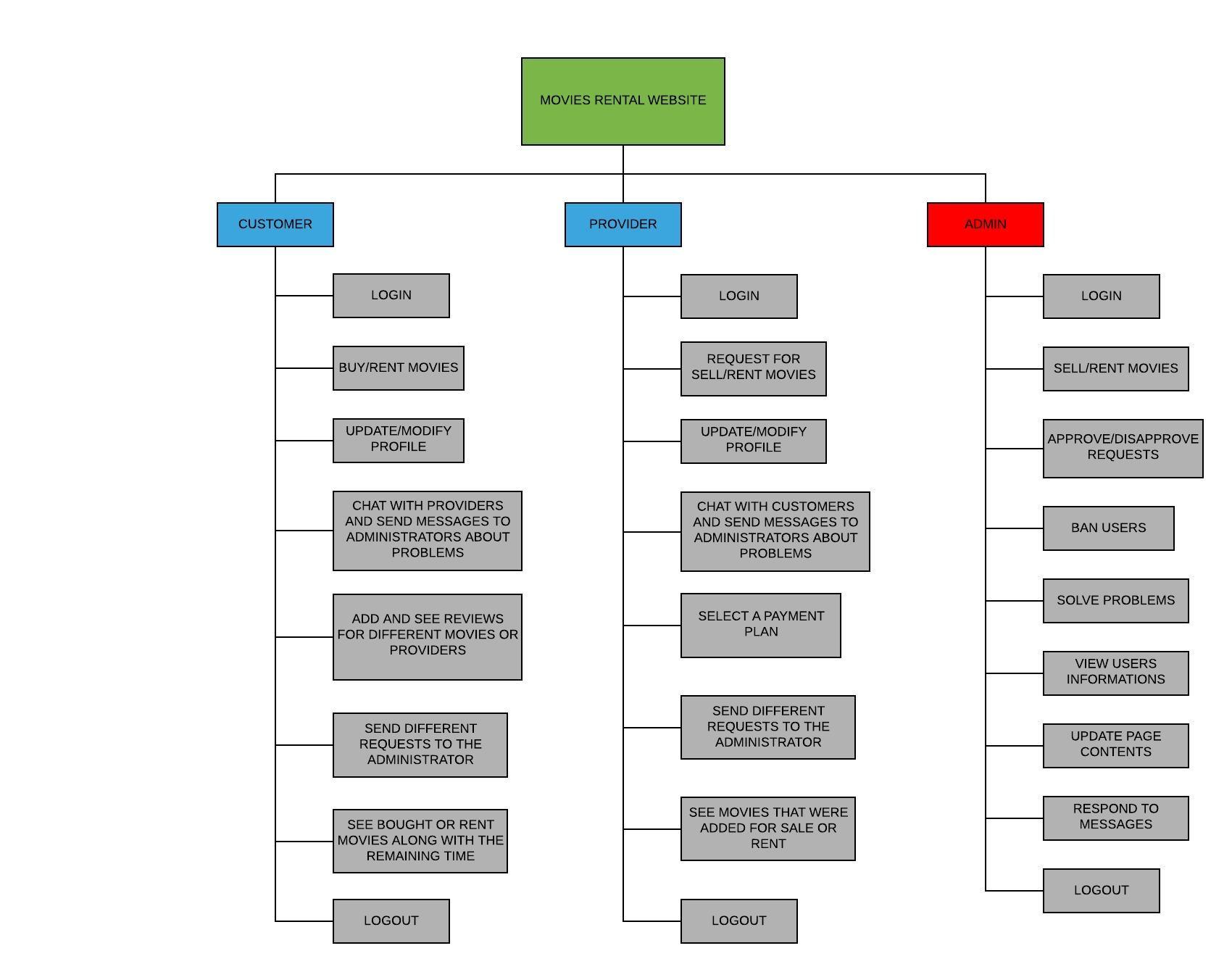
1. Administrator:

* Through a query, the administrator will be able to login (the query will search for his login credentials in the database);
* If they add movies for sale or rent, the movies will be added to the database;
* They are able to retrieve informations about different users from the database;

1. Level 1 – Level 3:

* The implementation of the functionalities are located in the local host, on a server where the web application will be running;
* If any data is introduced by a user or by an administrator, this will change the website appearance.

**3.2 Decomposition Description**



**3.3 Design Rationale**

As you can see from the diagram from subsection 3.1, our web application architecture is based on 3 levels (models), where the top layer is the user interface, and the bottom layer is the database. All the layers are built around the database with which they interconnect. We chose this architecture because is the most common one and the most used by all the software frameworks.

Each layer has a specific task, and depend on each other. The first layer, or the top one, the interface which is written in CSS, HTML and JavaScript, has the task to give the user the view of the web application. The second layer, the one from the middle, contains business logic, informations about data types, implementation of different functionalities and makes the connection between the top and the bottom layers, by transferring the data between them. This is done by using a controller. The bottom layer, and the last one, is the database which contains all the informations and data required.

The main advantages of this layered architecture are that the web application is easy to test as components belong to specific layers, and they can be tested separately, and it is easy and simple to implement and understand because naturally, most applications work in layers.

In conclusion, we thought that this architecture pattern is the best for our application, beside the reasons presented above, because we are unexperienced developers, so a simple pattern is perfect for us, this is a rudimentary application, where the user count will not exceed 100 for sure, and also the cost of the implementation is very low and related only on time consuming.

**4. DATA DESIGN**

**4.1 Data Description**

In our code files all the database tables are represented by models. These models are afterwards transformed into tables by a framework. We will use as data types: integer lists, integer variables, integer arrays, string lists and arrays, string variables. This will help us implement the functionalities for the queries, for example: log in, displaying the movies to users etc. The data structures are to be used in such ways as following tables at 4.2.

**4.2 Data Dictionary**

ADMIN

|  |  |  |
| --- | --- | --- |
|  | Type | NULL |
| Username | Varchar(30) | - |
| Password | Varchar(30) | - |
| Email | Varchar(30) | - |

USER

|  |  |  |
| --- | --- | --- |
|  | Type | NULL |
| User Type | Varchar(30) | - |
| Username | Varchar(30) | - |
| Password | Varchar(30) | - |
| Email | Varchar(30) | - |
| Sex | Varchar(10) | - |
| Birthdate | date | - |
| Country | Varchar(30) | - |
| City | Varchar(30) | - |
| Phone number | Varchar(30) | - |

SINGLE MOVIE

|  |  |  |
| --- | --- | --- |
|  | Type | NULL |
| Movie Name | Varchar(30) | - |
| Owned by | Varchar(30) | - |
| Sell price | Numeric | - |
| Rent price | Numeric | - |
| On stock | Numeric | - |

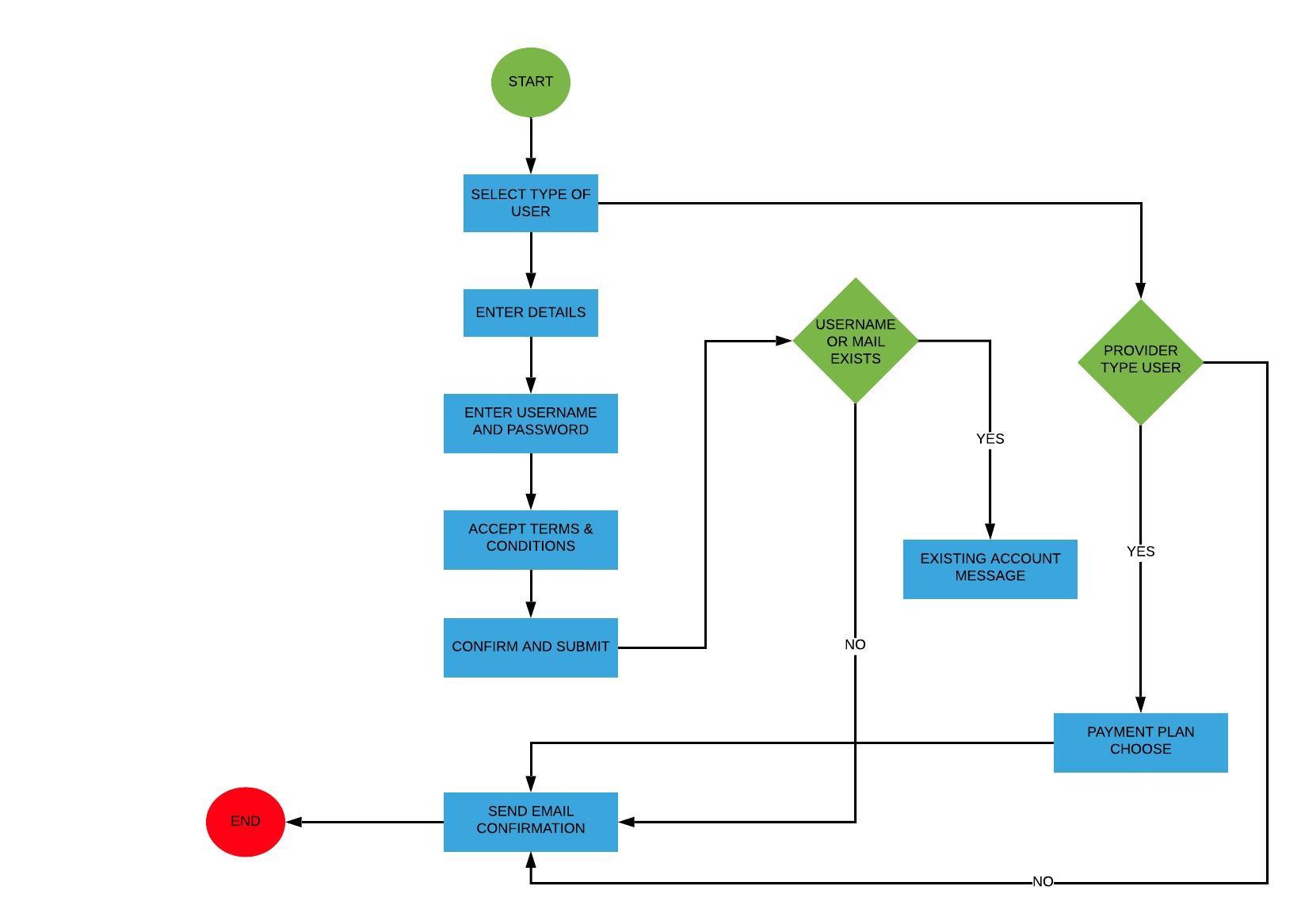
MOVIES LIST PROVIDER

|  |  |  |
| --- | --- | --- |
|  | Type | NULL |
| RENTED OR SOLD | Varhcar(30) | - |
| Moviename | Varchar(30) | - |
| Sold/Rented to | Varchar(30) | - |
| Price | Numeric | - |
| Date | Date | - |

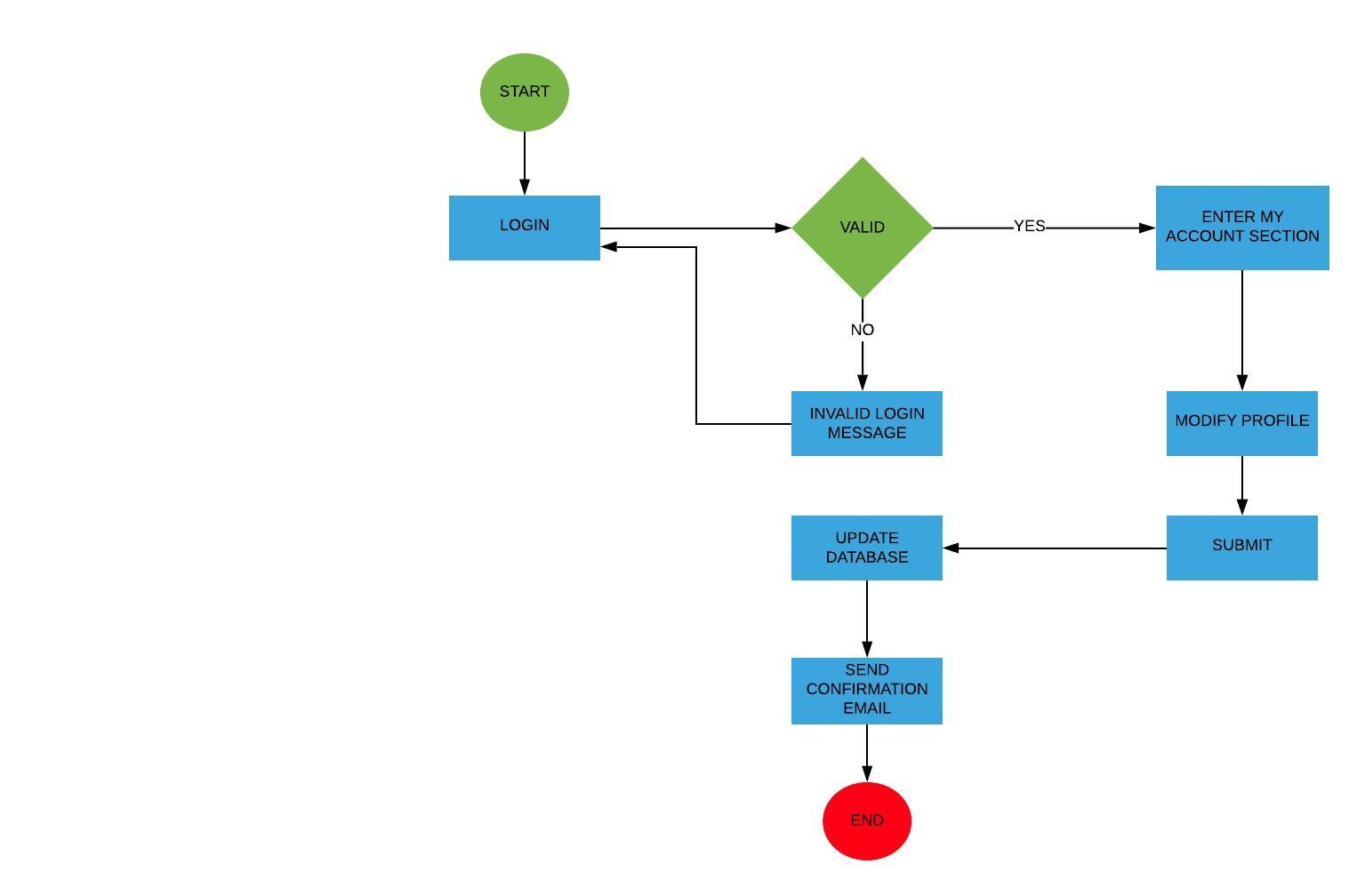
MOVIE LIST CUSTOMER

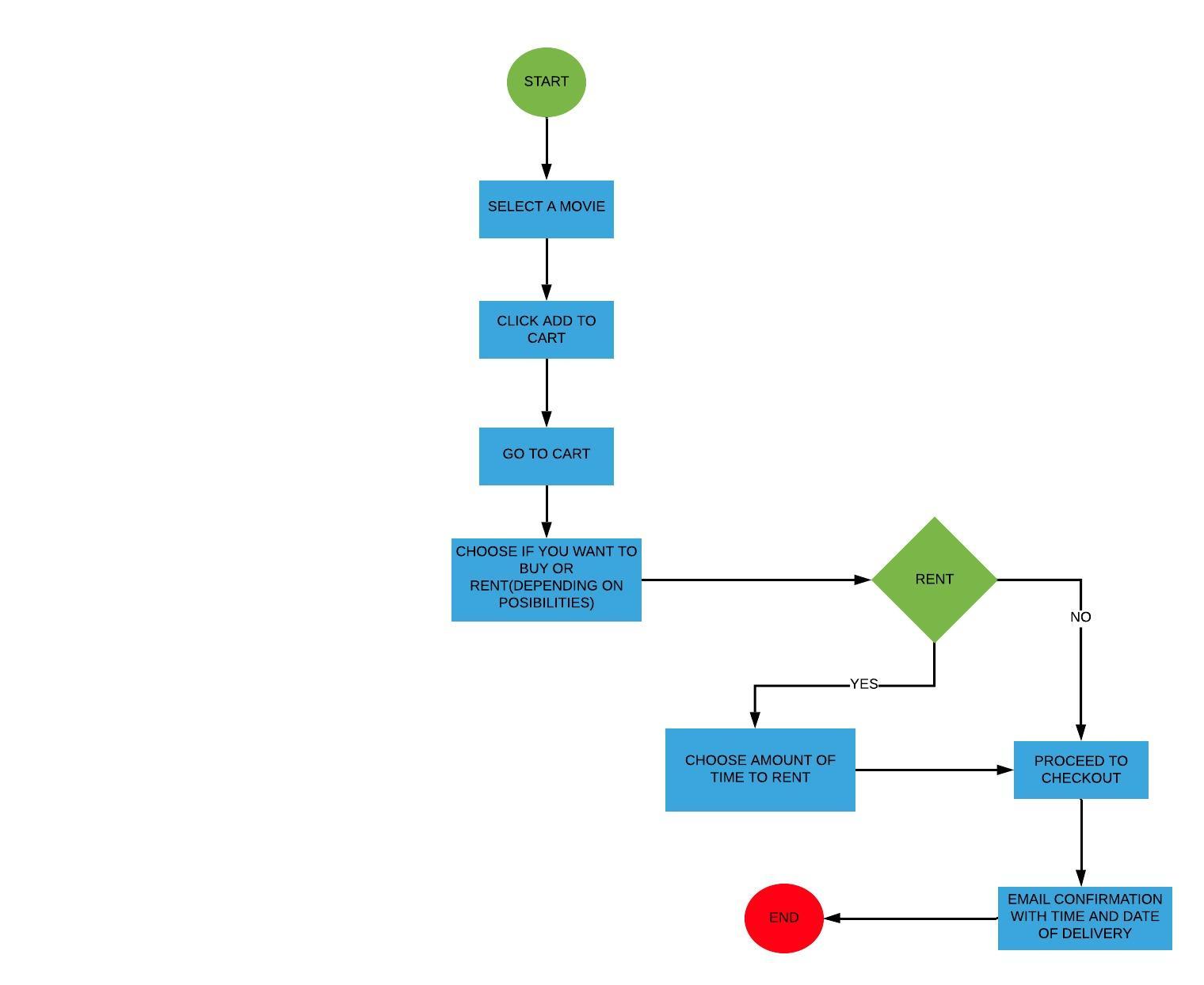
|  |  |  |
| --- | --- | --- |
|  | Type | NULL |
| RENTED OR SOLD | Varhcar(30) | - |
| Moviename | Varchar(30) | - |
| Sold/Rented FROM | Varchar(30) | - |
| Price | Numeric | - |
| Date | Date | - |

**5.COMPONENTDESIGN**

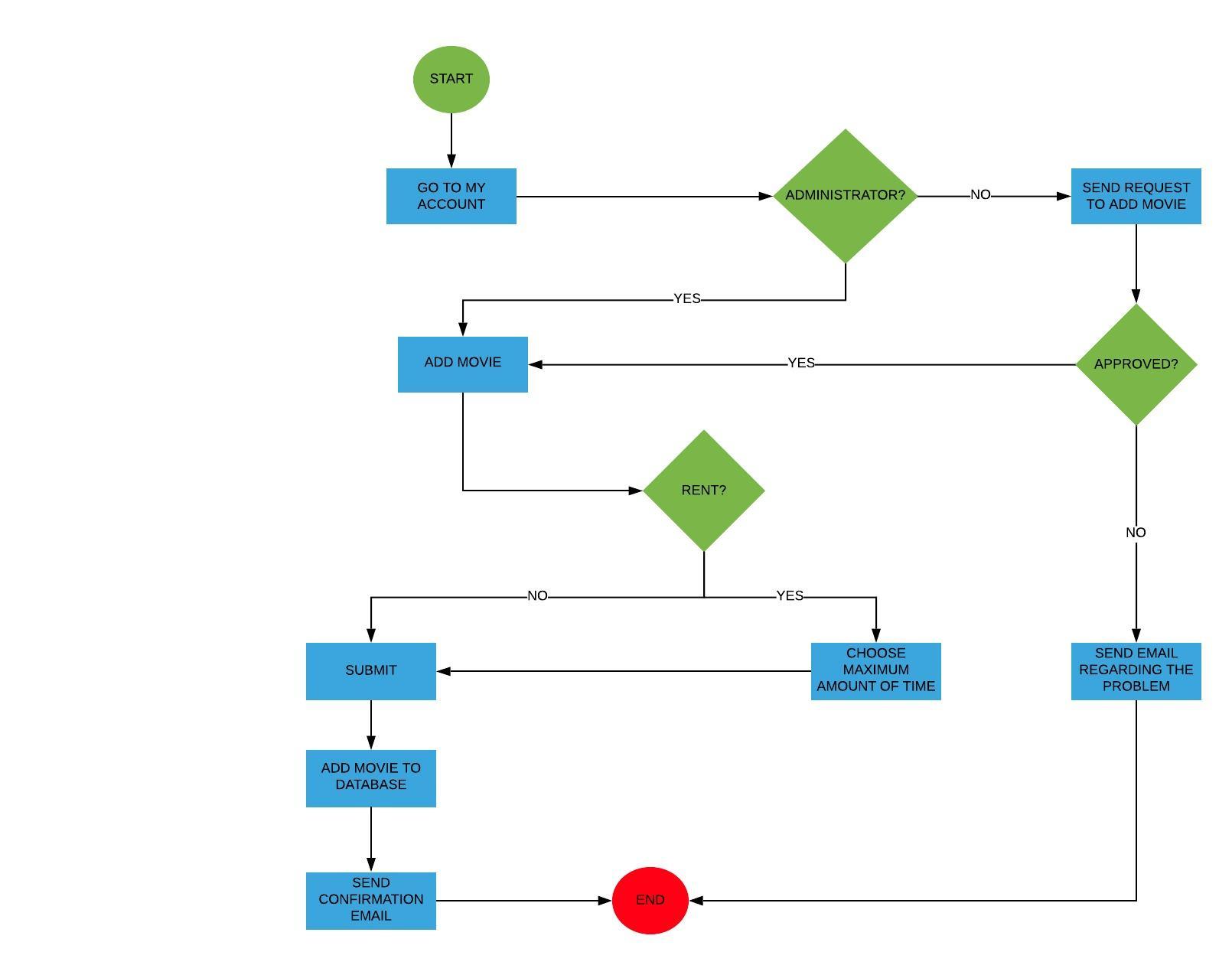
**Register process**:

**Profile modification process**:

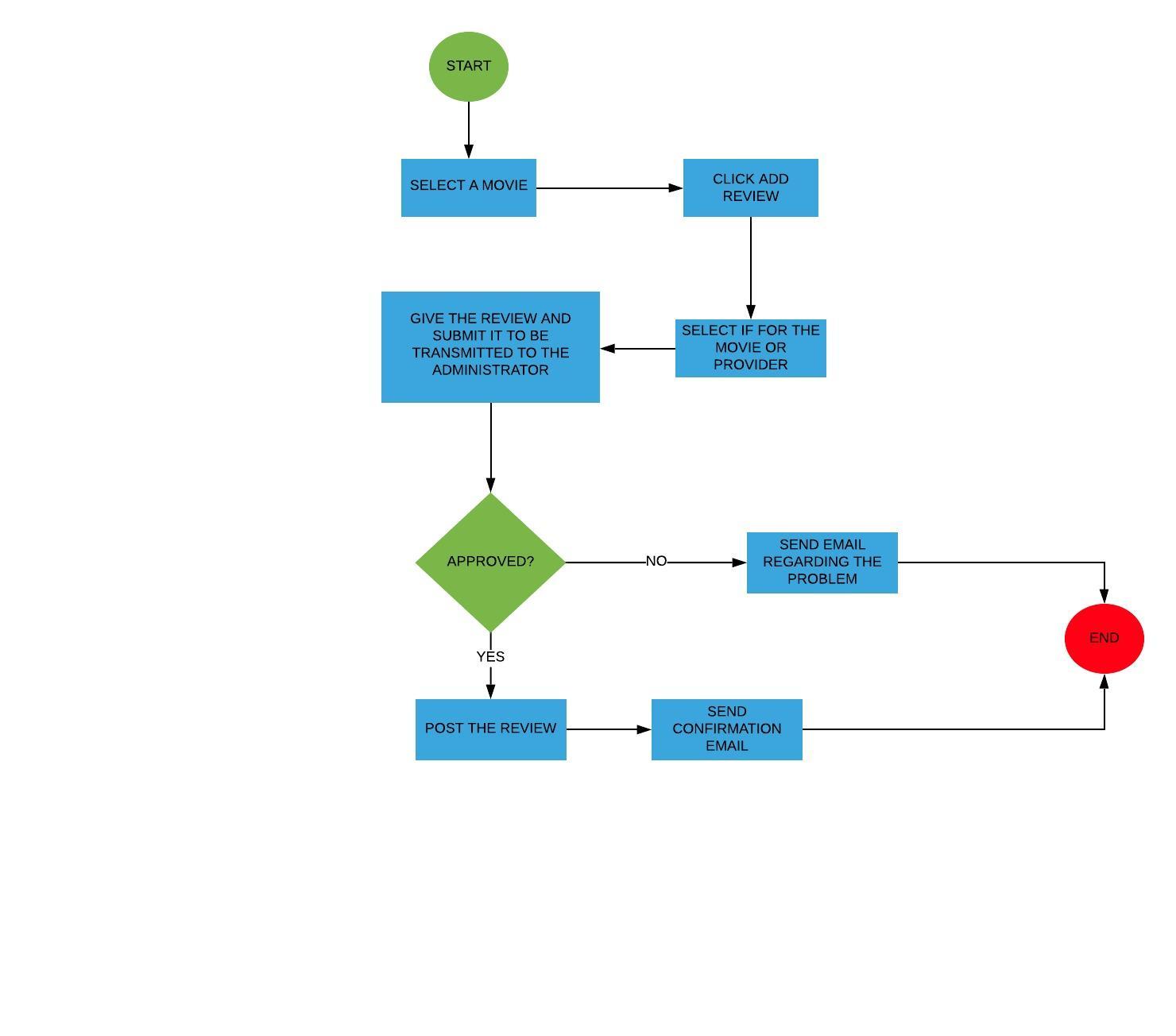


**Buy/Rent process**:

**Add movies for sale/rent process**:



**Post a review process**:



**6.HUMAN INTERFACE DESIGN**

**6.1 Overview of User Interface**

Based on the user reviews, upon accessing our products, you need to login, the login page is simple, all you have to do is to enter your **Username** and **Password**, and select what you are, an user or an admin. If the user does not have an account, he will be able to register. Unsuccessful registration/logins, will lead to a warning. Failing to complete the recommended fields, will lead to a warning. After the login process is done, the user will be able to browse: “Home”, “Movies”, “Contact” pages. Also for a quick search we have added search bar. After our user have decided what movie to pick, he can add it to his cart then proceed to checkout, also he can leave a to the movie that he have seen. If the user get something unclear, he can message us through live-chat functionality.

User interface share following qualities or characteristics:

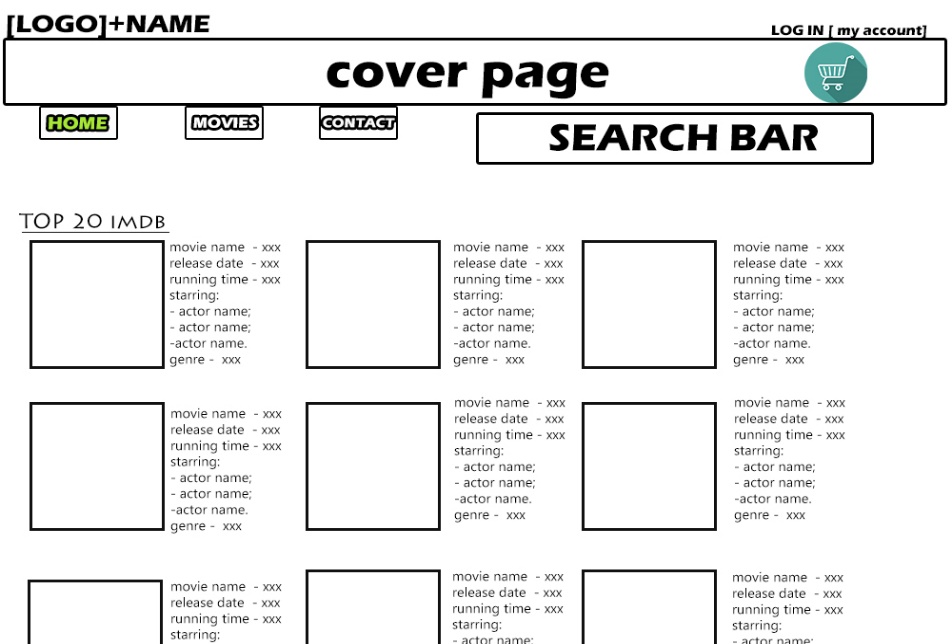
**Clarity**: The interface avoids ambiguity by making everything clear through language, flow, hierarchy and metaphors for visual elements.

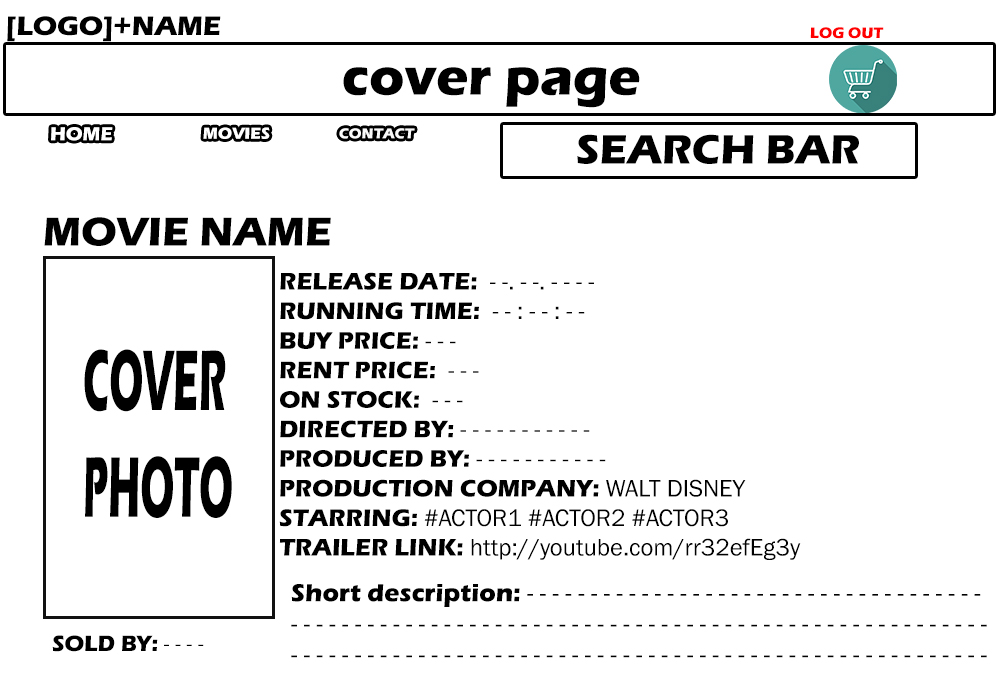
**Responsiveness**: This means a couple of things. First, responsiveness means speed: a good interface should not feel sluggish. Secondly, the interface should provide good feedback to the user about what’s happening and whether the user’s input is being successfully processed.

**Aesthetics**: While you don’t need to make an interface attractive for it to do its job, making something look good will make the time your users spend using your application more enjoyable; and happier users can only be a good thing.

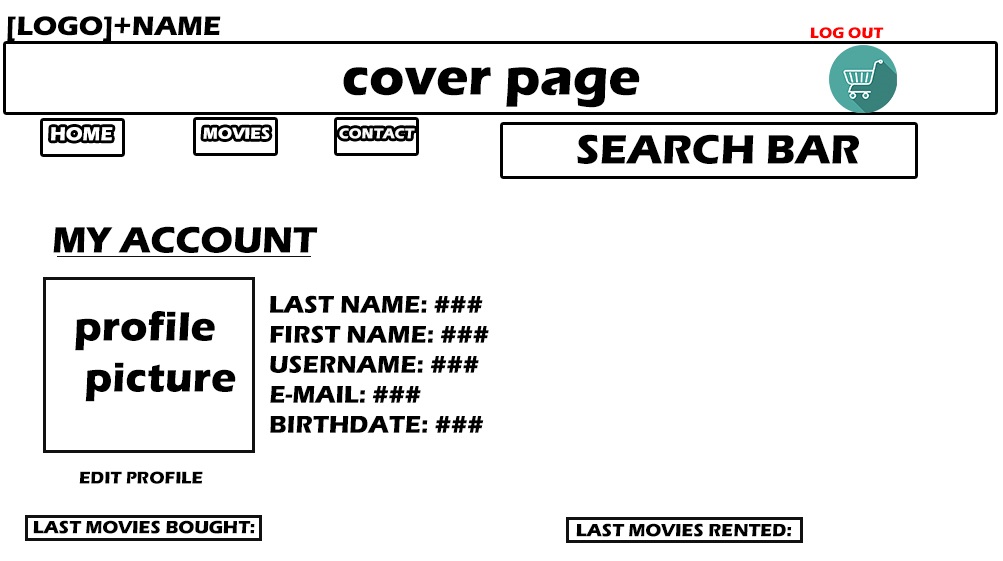
**Efficiency**: Time is money, and a great interface should make the user more productive through shortcuts and good design.

**6.2 Screen Images**

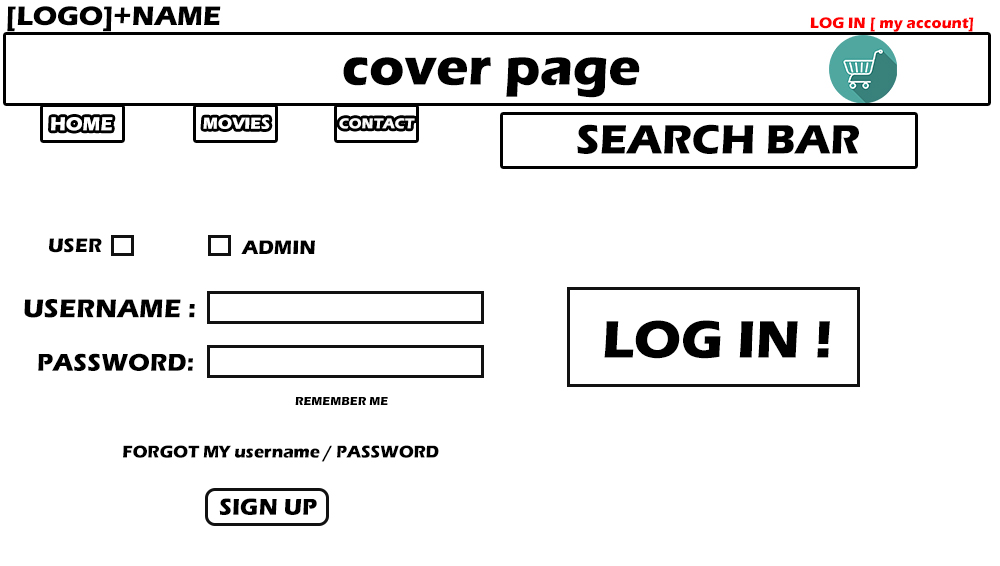
**HOME PAGE**

**MOVIE PAGE**

**MY ACCOUNT**



**LOG IN**



**CONTACT**



**6.3 Screen Objects and Actions**

Our web application consists in a few interface objects and each of them has a meaningful purpose. For example we have:

* Input Objects: Text field (Search bar, Log In, Register, Radio buttons, Add to Cart Button), live chat;
* Information Objects: Photos, Descriptions, Account details, live chat;
* Navigation Objects: Movies List, Rent / Bought List, Sold / Rented List (slide down);
* Top buttons (HOME / MOVIES / CONTACT / MY ACCOUNT )

**7.REQUIREMENTS MATRIX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **REQUIREMENTS**  **TRACEABILITY MATRIX** | | | | | | |
| Project Name: Car Rental Website | | | | | | |
| User class ID# | User Class Name | Functional Requirement ID# | Functional Requirement Name/Use Case | System Component | Priority | Test Case ID# |
| UC\_1 | Customer User | 1.1 | Website access | Link | High | #TC001 |
|  |  | 1.2 | Update notification | Newsletter | Medium | #TC002 |
|  |  | 1.3 | User Registration | Login/Register | High | #TC003 |
|  |  | 1.4 | User Login | Login/Register | High | #TC004 |
|  |  | 1.5 | Password retrieval | Login/Register | High | #TC005 |
|  |  | 1.6 | Rental search | Search module | High | #TC006 |
|  |  | 1.7 | Rental selection & checkout | Checkout module | High | #TC007 |
|  |  | 1.8 | Profile update | Profile module | High | #TC008 |
|  |  | 1.9 | Check Rented Movies | Profile module | High | #TC009 |
|  |  | 1.10 | Post feedback | Profile module | High | #TC010 |
|  |  | 1.11 | Sign out and Log back in | Login/Register | High | #TC011 |
| UC\_2 | Administrator User | 2.1 | Admin Login | Super Login module | High | #TC012 |
|  |  | 2.2 | Create/Delete movies listing | Admin Panel  module | High | #TC013 |
|  |  | 2.3 | Post movies listing | Admin Panel  module | High | #TC014 |
|  |  | 2.4 | Confirm/Cancel customer checkout | Admin Panel  module | High | #TC015 |
|  |  | 2.5 | Manage feedback | Admin Panel  module | High | #TC016 |
|  |  | 2.6 | Contact us live-chat management | Admin Panel  module | High | #TC017 |
|  |  | 2.7 | Check User Details | Admin Panel  module | High | #TC018 |
|  |  | 2.8 | Update public details | Admin Panel  module | High | #TC019 |
|  |  | 2.9 | Manage newsletter subscribers | Admin Panel  module | High | #TC020 |
|  |  | 2.10 | Dashboard menu statistics | Admin Panel  module | High | #TC021 |
|  |  | 2.11 | Change admin password | Admin Panel  module | High | #TC022 |
|  |  | 2.12 | Log out of Admin Panel | Admin Panel  module | High | #TC023 |
| UC\_3 | Guest User | 3.1 | Explore website | Link | High | #TC024 |
|  |  | 3.2 | Needs to create an account | Login/Register  prompt | High | #TC025 |