# Design

Design pattern is a well-defined solution to a common problem. It’s a template, not a solution. That’s why it is language independent. We can use in different other language platform. We need design pattern to properly create a class, instantiate object, interact between objects and write reusable code. Structural, creational and behavioral are its types.

## Structural design

It is made of variables and functions. So, when design pattern’s deal with the class structure then it comes in the structure of design patterns. It simplifies the structure by identifying the relationships. It focuses on, how classes inherit from each other.

1. **Class diagram(final)**

They are used to show the different objects in a system, their attributes, operations and relationships among them. It is static view of an application. Class notation consists of class name, class attributes and class operations.

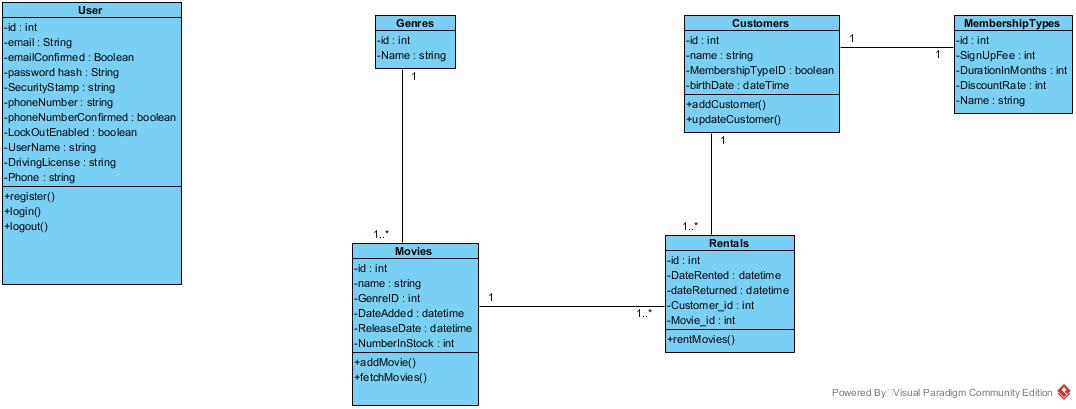


Fig: final class diagram

In above class diagram, genres have one to many relationships with movies, movies have one to many relationships with rentals, customers have one to many relationships with rentals and also one to one relationship with memberships types. User have no relationships.

1. **Data flow diagram**

A data flow diagram is a graphical representation of flow of information of any system.it can be divided into logical and physical. Logical flow means flow of a system to perform certain functionality whereas physical flow means implementation of logical flow.

Building blocks of DFD

**Process**: it receives input and provides different output. Every process has its own name that performs some function. A rounded rectangle represents a process.

**Data flow:** a data flow is a path for data to move from one part to another. Straight lines incoming arrow are input data flow and outgoing arrow are output data flow. Process will be in middle of the flows.

**Data store:** a data store is connected to a data-flow. Each data store must have an input data flow and an output

**External entity:** a rectangle represents an external entity. They either supply data or receive data. They do not process data. It must be connected to a data-flow.

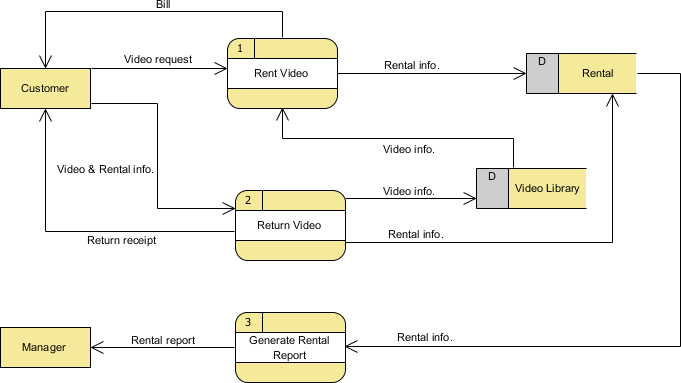


Fig: data flow diagram of online movie rental system

In the diagram the DFD shows how the system works when it is started. First the customer rent a video, in return customer gets the bill. Also, customer can return video, also its receipt gets returned to the customer. Returned video gets stored in video library database. From where customer can again rent a video. Rented video and returned video’s information is stored in rental database. A process of rental report is generated and reported to the manager.

## Behavioral design

When how does one class interact with other, then it comes under behavioral design pattern. It should be loosely coupled. it deals with communication of objects.

1. **Activity diagram**

It is basically a representation of flow from one activity to another. This activity can be described as operation of the system. It have pre-condition, actor input, system step, basic flow, alternative flow, returning alternative flow, parallel activities and post-conditions.

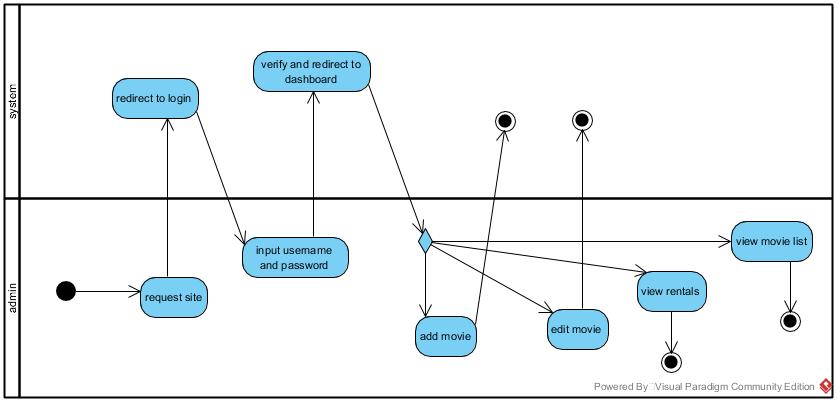


Fig: admin’s activity diagram

In the diagram, admin’s activity is pictured. First the admin requests a site, admin’s gets redirected to login in system. Admin inputs username and password. System verifies the entered username and redirects to dashboard and admin gets to add, edit, view rentals and movie.

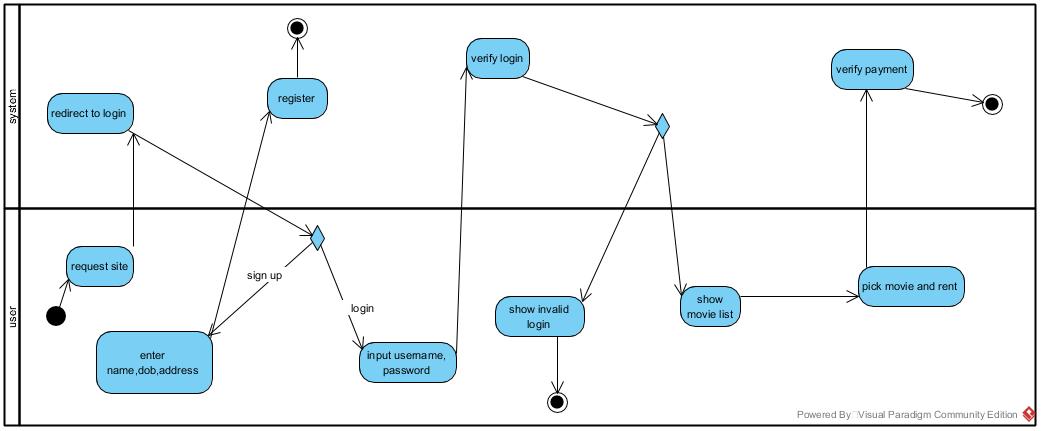


Fig: user’s activity diagram

First user requests a site and gets redirected to system’s login page. If user haven’t registered, they get redirected to register page, if they have already registered, they can login using valid username and password. If the login is successful they can enter into system’s where they can pick and rent a movie else user are shown invalid login error.

1. **Sequence diagram**

It shows how operations are carried out. It shows object interactions arranged in time sequence. A lifeline where different object live horizontally and exchange message between then.

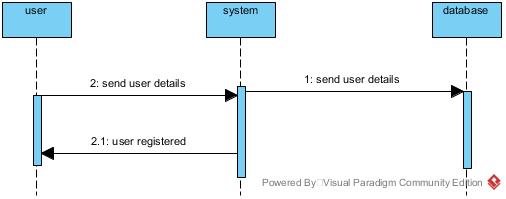


Fig: user’s registration sequence diagram

When user details are sent to a system, it gets saved into database and after that if valid information is provided, user gets registered successfully.

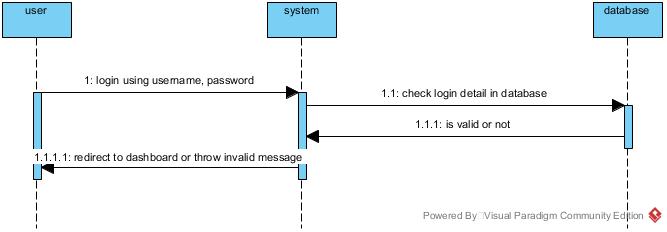


Fig: user’s login sequence diagram

During login valid username and password needs to be given to the system. Database checks login details, if given information is valid user is redirected to dashboard else it throws invalid message.

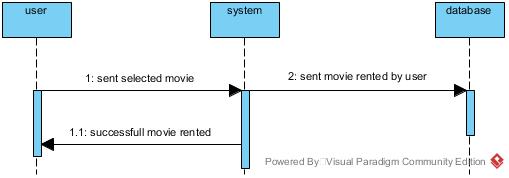


Fig: movie rent sequence diagram

During movie renting, user select a movie and rented movie information is sent to the database. After a movie is selected, it is rented successfully.

## Database design

A collection of process that helps in implementing data management system. It’s objective is to produce logical and physical design models of proposed database system.

1. **Data dictionary**

A data dictionary is a collection of descriptions of the data objects or items in a data model for the benefit of programmers and others who need to refer to them. A first step in analyzing a system of objects with which users interact is to identify each object and its relationship to other objects. This process is called data modeling and results in a picture of object relationships.

Here, I have created the data dictionary for the online Movie Rental System which is as given below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| Id | Nvarchar | 128 | No | Pri | ID number of Primary  Key |
| Name | Nvarchar | 256 | No |  | Name of the user |

**UserClaims**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| Id | Int | - | No | Pri | ID number of Primary  Key |
| UseId | Nvarchar | 128 | No |  | Unique Id of the User |
| ClaimType | Nvarchar | - | Yes |  |  |
| ClaimValue | Nvarchar | - | Yes |  |  |

**UserLogins**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| LoginProvider | Nvarchar | 128 | No | Pri | Login provider user |
| ProviderKey | Nvarchar | 128 | No | Pri | Login provider unique ID |
| UserId | Nvarchar | 128 | No | Pri | Unique Id of the user |

**UserRoles**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| UserId | Nvarchar | 128 | No | Pri | Unique Id of the user |
| RoleId | Nvarchar | 128 | No | Pri | Unique Id of the user to provide role |

**Users**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| Id | Nvarchar | 128 | No | Pri | Unique Id of the user |
| Email | Nvarchar | 256 | Yes |  | Email address of the user |
| EmailConfirmed | Bit | - | No |  | Email confirmation |
| PasswordHash | Nvarchar | - | Yes |  | Password hash of the user |
| SecurityStamp | Nvarchar | - | Yes |  | Security stamp for the password |
| PhoneNumber | Nvarchar | - | Yes |  | Phone number of the user |
| PhoneNumberConfirmed | Bit | - | No |  | Phone confirmation |
| Username | Nvarchar | 256 | No |  | Username of the user |
| DrivingLicense | Nvarchar | 255 | No |  | Driving License number  of the user |
| Phone | Nvarchar | 50 | No |  | Phone number of the user |

**Customers**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| Id | Int | 128 | No | Pri | Unique Id of the customer |
| Name | Nvarchar | 255 | No |  | Name of the customer |
| MembershipTypeId | Tinyint | - | No |  | Membership type of the customer |
| Bithdate | Datetime | - | Yes |  | Birthdate of the customer |

**Genres**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| Id | Tinyint | - | No | Pri | Id number for the genre |
| Name | Nvarchar | 255 |  |  | Name of the genre |

**MembershipTypes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| Id | Tinyint | - | No | Pri | ID number for the membershiptype |
| SignUpFee | Smallint | - | No |  | Sign up fee for the membership |
| DurationInMonths | Tinyint | - | No |  | Duration for the membership |
| DiscountRate | Tinyint | - | No |  | Discount Rate for the membership |
| Name | Nvarchar | - | No |  | Name of the membership |

**Movies**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| Id | Int | - | No | Pri | ID number for the movies |
| Name | Nvarchar | 255 | No |  | Name of the movie |
| GenreId | Tinyint | - | No | For | Genre id from the genre table |
| DateAdded | Datetime | - | No |  | Movie added date |
| ReleaseDate | Datetime | - | No |  | Movie released date |
| NumberInStock | Tinyint | - | No |  | Movie Number in stock |

**Rentals**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| Id | Int | - | No | Pri | ID number for the rental |
| DateRented | Datetime | - | No |  | Movie rented date |
| DateReturned | Datetime | - | No |  | Movie returned date |
| Customer\_Id | Int | - | No | For | Customer id from customer table |
| Movie\_id | Int | - | No | For | Movie id from movie table |

1. **ER diagram**

it is used to sketch out the design of the database. It shows the logical structure of databases by defining their entities, attributes and relationship between them. They use different set of symbols such as rectangles, diamonds and ovals.

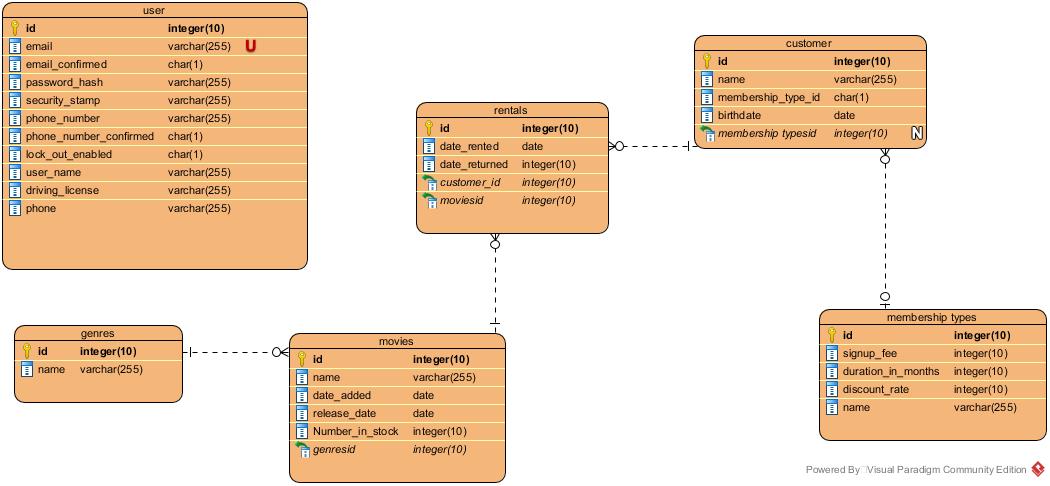


Fig: ER diagram for the project

In above ER diagram, id is primary key, genresid is foreign key in movies, customer\_id and moviesid are also foreign key in rentals and membership typesid is a foreign key for customer.

## Architecture

1. **UI**

UI or user interface is a design to make user interaction as simple as possible. It can be finished drawing through hands too. It should show technical functionality and visual elements. It should be adaptable to changing user needs.

