# CHAPTER 3: 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.

## **3.1 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.

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.

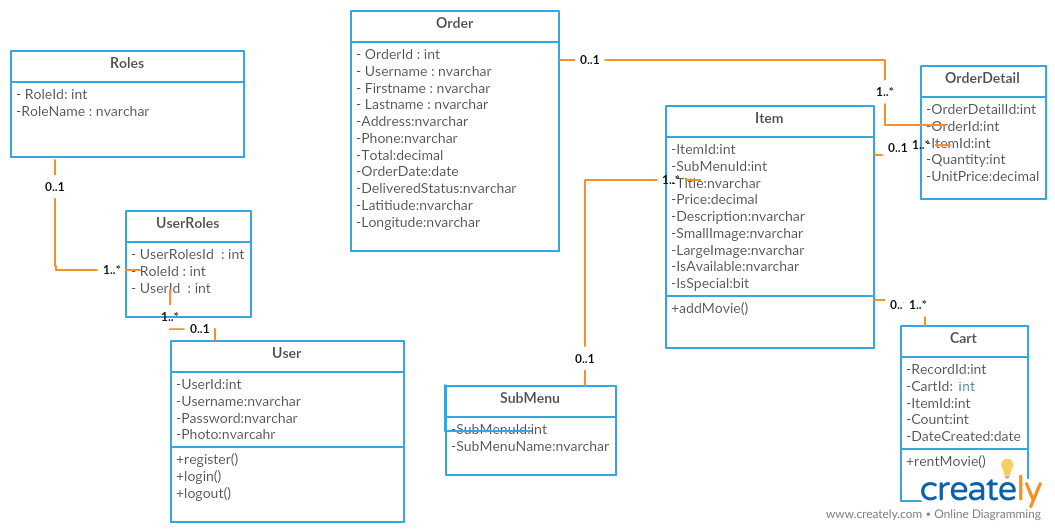


Fig: final class diagram

In above class diagram, user have one to many relationships with user roles also, roles have one to many relationships with user roles. Also, sub menu have one to many relationship with item, item have one to many relationship with cart and order detail and order have one to many relationship with order datail.

**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.

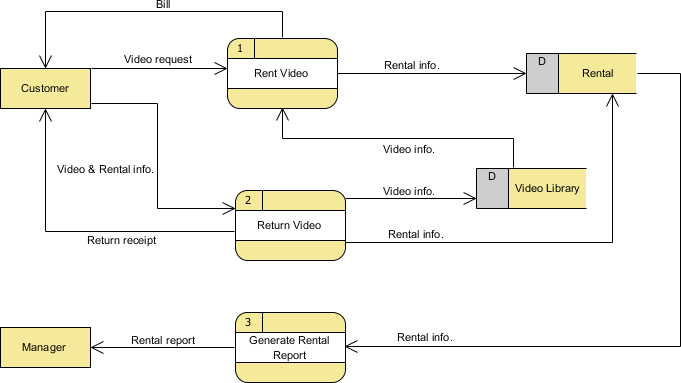


Fig: data flow diagram of online movie rental system

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.

## **3.2 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.

**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 has pre-condition, actor input, system step, basic flow, alternative flow, returning alternative flow, parallel activities and post-conditions.

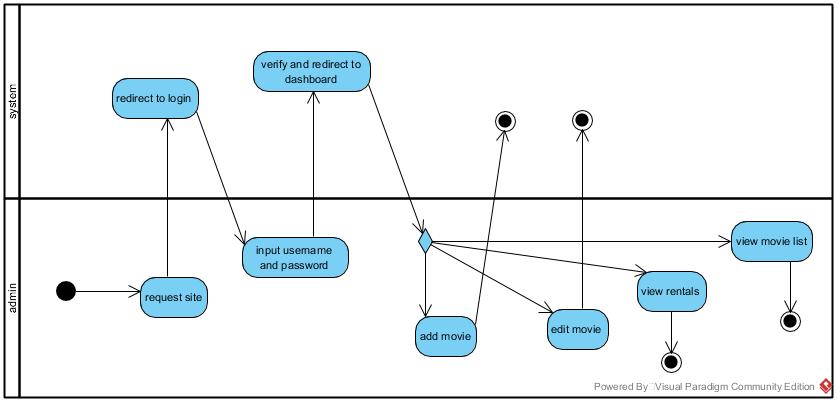


Fig: admin’s activity diagram

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.

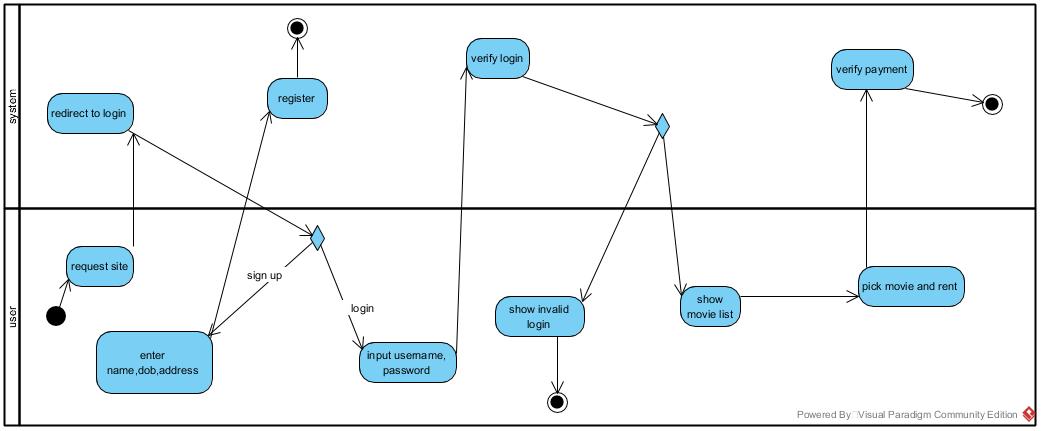


Fig: user’s activity diagram

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.

**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.

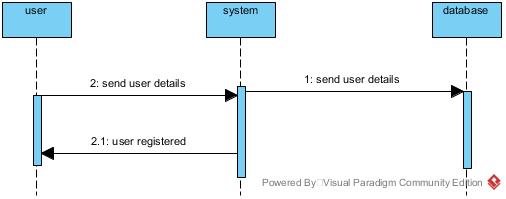


Fig: user’s registration sequence diagram

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.

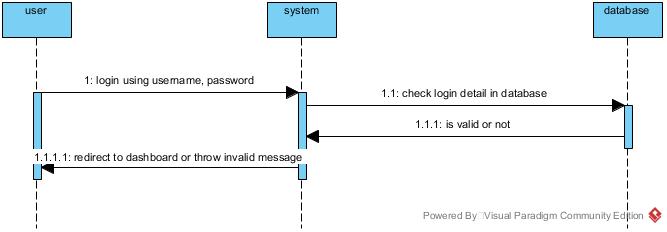


Fig: user’s login sequence diagram

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.

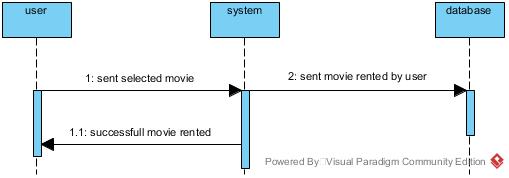


Fig: movie rent sequence diagram

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.

## **3.3 Database design**

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

**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:

**Roles**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| RoleId | int |  | no | Pri | ID number of Primary  Key |
| RoleName | Nvarchar | 50 | yes |  | Provided Role Name |

**UserRoles**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| UserRolesId | int | - | no | pri | Unique Id of the user |
| UserId | int | - | yes | for | Id from user table |
| RoleId | int | - | yes | for | Id from role table |

**Users**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| UserId | int | - | no | Pri | Unique Id of the user |
| Password | Nvarchar | -50 | Yes |  | password of the user |
| Username | Nvarchar | 50 | yes |  | Username of the user |
| Photo | Nvarchar | 50 | yes |  | Phone number of the user |

**Cart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| RecordId | Int |  | no | Pri | Unique Id of the cart table |
| CartId | Nvarchar | 50 | yes | for | Id provided by cart |
| ItemId | int | - | yes | for | Id provided by item |
| Count | int | - | yes |  | No. of movies |
| DateCreated | Date |  | yes |  | Date of movie rented |

**SubMenu**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| SubMenuId | int | - | no | Pri | Id number for the SubMenu |
| SubMenuName | Nvarchar | 50 | yes |  | Name of the SubMenu |

**OrderDetail**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| OrderDetailId | int | - | no | Pri | Unique id of orderdetail |
| OrderId | int | - | yes | for | Id provided by order table |
| ItemId | int | - | yes | for | Id provided by item table |
| Quantity | int | - | yes |  | Number of movie purchase |
| UnitPrice | decimal | (18,0) | yes |  | Price of payment |

**Item**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| ItemId | Int | - | no | Pri | ID number for the item |
| SubMenuId | int | - | yes | For | submenu id from the submenu table |
| Title | nvarchar | 50 | yes |  | Movie name |
| Price | decimal | (18,0) | yes |  | Movie price |
| Description | nvarchar | -50 | yes |  | Movie description |
| Small Image | nvarchar | 50 | yes |  | Movie small image |
| Large Image | nvarchar | 50 | yes |  | Movie large image |
| IsAvailable | nvarchar | 50 | yes |  | Checking movie available |
| Is Special | bit |  | yes |  | Checking movie special |

**Order**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Constrains | | Description |
| Null | Key |
| OrderId | Int | - | no | Pri | ID number for the order |
| Username | nvarchar | 50 | yes |  | User registered name |
| First name | nvarchar | 50 | yes |  | Users first name |
| Last name | nvarchar | 50 | yes |  | Users last name |
| address | nvarchar | 50 | yes |  | Users address |
| Phone | nvarchar | 50 | yes |  | Users Phone |
| Total | nvarchar | (18,0) | yes |  | Users payment |
| Order Date | date | 50 |  |  | Date ordered |
| Delivered status | nvarchar |  |  |  | True or false |

**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.

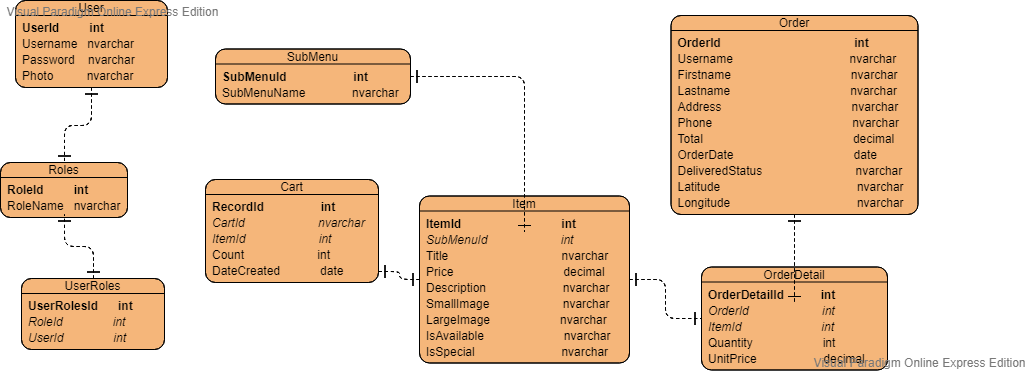


Fig: ER diagram for the project

In above ER diagram, userid is primary key for user table, rolesid is primary key for userroles table, Userroleid is primary key and roleid and userid are foreign key for userroles table. Submenuid is primary key for submenu table, recordid is primary key for cart table. Itemid id primary key for item table. Orderid is primary key for order table and orderdetailid is primary key for order detail table.

## **Architecture**

3.4.1 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.